Abstract

Graph Priors, Optimal Transport, and Deep Learning in Biomedical Discovery

Alexander Tong

2021

Recent advances in biomedical data collection allows the collection of massive datasets measuring thousands of features in thousands to millions of individual cells. This data has the potential to advance our understanding of biological mechanisms at a previously impossible resolution. However, there are few methods to understand data of this scale and type. While neural networks have made tremendous progress on supervised learning problems, there is still much work to be done in making them useful for discovery in data with more difficult to represent supervision. The flexibility and expressiveness of neural networks is sometimes a hindrance in these less supervised domains, as is the case when extracting knowledge from biomedical data. One type of prior knowledge that is more common in biological data comes in the form of geometric constraints.

In this thesis, we aim to leverage this geometric knowledge to create scalable and interpretable models to understand this data. Encoding geometric priors into neural network and graph models allows us to characterize the models' solutions as they relate to the fields of graph signal processing and optimal transport. These links allow us to understand and interpret this datatype. We divide this work into three sections. The first borrows concepts from graph signal processing to construct more interpretable and performant neural networks by constraining and structuring the architecture. The second borrows from the theory of optimal transport to perform anomaly detection and trajectory inference efficiently and with theoretical guarantees. The third examines how to compare distributions over an underlying manifold, which can be used to understand how different perturbations or conditions relate. For this we design an efficient approximation of optimal transport based on diffusion over a joint cell graph. Together, these works utilize our prior understanding of the data geometry to create more useful models of the data. We apply these methods to molecular graphs, images, single-cell sequencing, and health record data.

Graph Priors, Optimal Transport, and Deep Learning in Biomedical Discovery

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Alexander Tong

Dissertation Director: Smita Krishnaswamy

December 2021

Contents

Acknowledgements **xviii**

1 Introduction **1**

2 Preliminaries **5**

2.1 Graphs 5

2.1.1 Graphs and Random Walks on Data 5

2.1.2 Graph Signal Processing 6

2.1.3 Diffusion on a Graph 7

2.2 Deep Learning 8

2.2.1 Autoencoders 9

2.2.2 Lipschitz Neural Networks 9

2.2.3 Continuous Normalizing Flows 10

2.3 Optimal Transport 10

2.3.1 Integral Probability Measures and ϕ-divergences 11

2.3.2 Unbalanced Wasserstein Distances 12

2.3.3 Efficient Computation 13

I Deep Learning Using Graph Spectral Priors **15**

3 Graph Spectral Regularization **16**

3.1 Introduction 16

3.2 Related Work 18

3.3 Enforcing Graph Structure . . . 18

3.3.1 Learning and Reinforcing an Abstracted Feature-space Graph . . . 19

3.4 Experiments . . . 20

3.4.1 Fixed Structure . . . 20

3.4.2 Learning Graph Structure . . . 23

3.4.3 Computational Cost . . . 27

3.5 Conclusion . . . 28

Appendix **28**

4 Geometric Scattering **31**

4.1 Introduction . . . 31

4.2 Related Work . . . 33

4.3 Preliminaries: Geometric Scattering . . . 34

4.4 Adaptive Geometric Scattering Relaxation . . . 35

4.5 A Learnable Geometric Scattering Module . . . 38

4.6 Empirical Results . . . 42

4.6.1 Whole Graph Classification . . . 42

4.6.2 Graph Regression . . . 46

4.7 Conclusion . . . 48

Appendix **48**

4.A Proofs for Section 4.4 . . . 49

4.A.1 Proof of Lemma 4.4.1 . . . 49

4.A.2 Proof of Theorem 4.4.2 . . . 49

4.A.3 Proof of Theorem 4.4.3 . . . 51

4.B Gradients of the LEGS module . . . 52

4.C Datasets . . . 53

4.D Training Details . . . 55

4.D.1 Cross Validation Procedure . . . 55

4.E Ensembling Evaluation . . . 56

II Optimal Transport in Deep Learning 59

5 Fixing Bias in Reconstruction-based Anomaly Detection 60

5.1 Introduction . . . 60

5.2 Related Work . . . 63

5.3 Background . . . 65

5.4 The Lipschitz Anomaly Discriminator . . . 67

5.4.1 Choosing the Corrupted Distribution . . . 68

5.4.2 Combination with Other Models . . . 68

5.4.3 Theoretical Properties of Anomaly Discrimination . . . 69

5.5 Experimental Evaluation . . . 74

5.5.1 Unsupervised Anomaly Detection on Images . . . 75

5.5.2 Tabular Health Record Anomaly Detection . . . 78

5.5.3 Graph Anomaly Detection . . . 80

5.5.4 Network Architecture and Training Details . . . 81

5.6 Conclusion . . . 83

6 TrajectoryNet 84

6.1 Introduction . . . 84

6.2 Background and Related Work . . . 86

6.3 Preliminaries . . . 88

6.3.1 The Monge-Kantorovich Problem . . . 88

6.3.2 Dynamic Optimal Transport . . . 88

6.3.3 Continuous Normalizing Flows . . . 89

6.4 TrajectoryNet: Efficient Dynamic Optimal Transport . . . 90

6.4.1 Dynamic OT Approximation via Regularized CNF . . . 91

6.4.2 Further Adaptation for Single-Cell Trajectories . . . 93

6.4.3 Training . . . 96

6.5 Experiments . . . 97

6.5.1 Artificial Data . . . 97

6.5.2 Single-Cell Data . . . 99

6.6 Conclusion 101

Appendix 102

6.A Technical Details 105

6.A.1 Proof of Theorem 6.4.1 105

6.B Growth Rate Model Training 108

6.C Scaling with Dimension 108

6.D Biological Considerations 109

6.E Reproducibility 111

6.E.1 2D Examples 111

6.E.2 Single Cell Datasets 112

6.E.3 Software Versioning 112

III Optimal Transport in Graphs 113

7 DiffusionEMD 114

7.1 Introduction 114

7.2 Preliminaries 116

7.3 EMD Through the L^1 Metric Between Multiscale Density Estimates 118

7.4 Diffusion Earth Mover's Distance 119

7.4.1 Data Graphs and Density Estimates on Graphs 119

7.4.2 Diffusion Earth Mover's Distance Formulation 120

7.4.3 Theoretical Relation to EMD on Manifolds 120

7.4.4 Efficient Computation of Dyadic Scales of the Diffusion Operator . . 123

7.4.5 Subsampling Density Estimates 125

7.4.6 Diffusion EMD Based Embeddings of Samples 126

7.4.7 Gradients of the Earth Mover's Distance 126

7.5 Results 127

7.6 Conclusion 133

Appendix **133**

7.A General framework and proofs 134

7.A.1 General framework 134

7.A.2 Proofs of section 4.2 138

7.B Related Work 142

7.B.1 Multiscale Methods for Earth Mover's Distance 143

7.C Algorithm Details 144

7.C.1 Gradients of the Diffusion EMD 147

7.D Experiment Details 150

7.D.1 Metrics used 150

7.D.2 Line Example 151

7.D.3 Swiss Roll 151

7.D.4 MNIST 152

7.D.5 Single-cell COVID-19 Dataset 153

8 Unbalanced Transport on Graphs and Geometric Domains **155**

8.1 Introduction 155

8.2 Preliminaries 157

8.3 Problem Setup 160

8.4 Unbalanced Diffusion Earth Mover's Distance 161

8.4.1 Equivalence to (Unbalanced) Wasserstein Distance 162

8.4.2 An Efficient UDEMD Algorithm 164

8.5 Results 165

8.6 Conclusion 170

Appendix **170**

8.A Theoretical Results 171

8.A.1 Robustness to Corruption 173

8.B Unbalanced Transport 174

8.C Experiment Details 174

8.C.1 Metrics 175

8.C.2 Dataset Details 175

8.C.3 Method Details 177

8.C.4 Compute Details 178

9 Discussion and Future Work **179**

9.1 Future Work 179

9.1.1 Further Applications of Algorithms and Methods 180

9.1.2 Theoretical Generalization 181

9.1.3 Casual Learning 182

List of Figures

3.1 Shows average activation by digit over an (8x8) 2D grid using graph spectral regularization and convolutions following the regularization layer. Next, we segment the embedding space by class to localize portions of the embedding associated with each class. Notice that the digit 4 here serves as the null case and does not show up in the segmentation. Finally, we show the top 10% activation on the embedding of some sample images. For two digits (9 and 3) we show a normal input, a correctly classified but transitional input, and a misclassified input. The highlighted regions of the embedding space correlate with the semantic description of the input. 21

3.2 (a) shows the regularization structure between capsules. (b-c) Show reconstruction when one of the 16 dimensions in the DigitCaps representation is tweaked by $0.05 \in [-0.25, 0.25]$. (b) Without GSR each digit responds differently to perturbation of the same dimension. With GSR (c) a single dimension represents line thickness across all digits. 22

3.3 We show the structure of the training data and snapshots of the learned graph for (a) three modules and (b) eight modules. (c) shows we have the mean and 95% CI of the number of connected components in the trained graph for over 50 trials. 23

3.4 Shows (a) graph structure over training iterations (b) feature activations of parts of the trajectory. PHATE [146] embedding plots colored by (c) branch number and (b) inferred trajectory location showing the branching structure of the data. 24

3.5 Graph architecture, PCA plot, activation heatmaps of a standard autoencoder, β-VAE [90] and a graph regularized autoencoder. With relu activations normalized to $[0, 1]$ for comparison. In the model with graph spectral we are able to clearly decipher the hierarchical structure of the data, whereas with the standard autoencoder or the β-VAE the structure of the data is not clear. 25

3.6 Shows correlation between a set of marker genes for specific cell types and embedding layer activations. First with the standard autoencoder, then our autoencoder with graph spectral regularization. The left heatmap is biclustered, the right heatmap is grouped by connected components in the learned graph. We can see progression especially in the largest connected component where features on the right of the component correspond to less developed neurons. 26

4.1 LEGS module learns to select the appropriate scattering scales from the data. 38

4.2 Enzyme class exchange preferences empirically observed in Cuesta et al. [49], and estimated from LEGS and GCN embeddings. 45

4.3 CASP dataset LEGS-FCN % improvement over GCN in MSE of GDT prediction vs. Average GDT score. 47

5.1 (a,c) training data (b,e) autoencoder reconstruction score (c,f) LAD anomaly score (red = more normal, blue = more anomalous). Reconstruction is a poor proxy for data density and interpolates too well close to the data. 63

5.2 LAD trains a Lipschitz neural network f to discriminate between the data and a corrupted version of the data. Our trained network f^* is then used to score anomalies. Darker = more anomalous. 66

5.3 Unsupervised semantic anomaly detection setup for CIFAR10. (a) standard setup and (b) with training set corruption where a small percentage of images from anomalous classes are treated as part of the nominal training data. . . 75

5.4 Top 100 nominal images in test set of LAD (a) and DCAE (b) trained on the automobile class. (c) Mean over examples of pixel values for each class. Many car images have white background and/or bright colors that are far from the mean image. LAD does better at modeling such images. 79

5.5 (a) Shows input data embedding with PHATE [148] colored by Creatinine in mg/dL. (b) Shows the AUC of various anomaly detection models over levels of training set corruption on the VACS data. 80

6.1 TrajectoryNet learns trajectories of particles from distributions sampled over time. We use a Neural ODE to learn the derivative of the dynamics function. To find the output at time t_1 for a given input at time t_0 we integrate T times letting the ODE solver choose the integration timepoints. 86

6.2 Transporting a Gaussian (a) to an S-curve (b) via (c) static optimal transport, (d) Base TrajectoryNet without regularization follows density (e) TrajectoryNet with energy regularization demonstrates more straight paths similar to OT. 91

6.3 Density regularization or velocity regularization can be used to follow a 1D manifold in 2D. 98

6.4 A 1D distribution of data over time embedded in two dimensions along a smooth manifold. On a single branch (left), with a tree structure (center), and circle (right). 99

6.5 Cell growth model learned on Embryoid Body Data [148] 100

6.6 Shows the first 2 PCs of the mouse cortex dataset. (a-c) show the distributions for the first three timepoints. (d) shows the distribution of cells over PC1. the interpolated points for E14.5 using (e) static OT, and (f) TrajectoryNet with density regularization. (g-i) shows expression of three markers of early (Pax6) mid (Eomes) and late (Tbr1) stage neurons. 103

6.7 Shows the Embryoid body dataset projected into 2D with PHATE [148] with paths and densities imputed using TrajectoryNet. 104

6.8 For curated endpoints, shows location on PHATE dimensions, TrajectoryNet paths projected into PCA space, and trajectories for 4 genes. 104

6.9 The computation per evaluation is roughly linear in terms of dimension. . . 109

6.10 Density regularization or velocity regularization can be used to follow a 1D manifold in 2D. 110

6.11 Shows the ratio of spliced, ambiguous, and unspliced RNA counts over the 5 timepoints in the Embryoid body dataset. Mean unspliced here is around 10%-20% of total counts, in other systems this is near 30% [113]. 110

7.1 Diffusion EMD first embeds datasets into a common data graph G, then takes multiscale diffusion KDEs for each of the datasets. These multiscale KDEs are then used to compute the Diffusion Earth Mover's Distance between the datasets that can be used in turn to create graphs and embeddings (PHATE [147] shown here) of the datasets. 118

7.2 Wasserstein distance of indicator distributions $\mathbf{1}_x, x \in [0, 1]$ from $\mathbf{1}_{0.5}$ computed using linear EMD methods L^2 distance: (a) ClusterTree (b) QuadTree and (c) Diffusion EMD. 128

7.3 Swiss roll dataset embeddings of $m = 1000$ distributions with $n = 10,000$ total points rotated into 10D colored by ground truth 2D sheet axes. Diffusion EMD recreates the manifold better in similar time. 129

7.4 Accuracy of methods measured via P@10 (left) and Spearman coefficient (right), against their (log scaled) computation time in seconds on the swiss roll dataset. Variations of methods are over Chebyshev approximation order for Diffusion EMD, # of trees for tree methods, and number of iterations for conv. Sinkhorn. Diffusion EMD is more accurate than tree methods and orders of magnitude faster than conv. Sinkhorn even with a single iteration. 129

7.5 Ablation study of major parameters for Chebyshev polynomial approximation on the swiss roll dataset. Mean and std. over 10 runs over (a) values of the maximum scale K, (b) the rank threshold in interpolative decomposition, (c) the total number of centers in the L^1 representation which drops with decomposition, (d-e) performance against the # of scales, and the order of the polynomial, both are very stable after a certain point, and (f) time vs. the Chebyshev order. 130

7.6 Embedding of 210 patients through different manifold constructions. Visualizing patient eventual mortality and cell types predictive of disease outcome on each manifold. Laplacian smoothness reported on each signal for each manifold. 132

8.1 (a) Illustrates effect of smoothing component of UDEMD on noise. (b) Illustrates balanced and unbalanced transport maps. 162

8.2 On a ring graph $n = 500$ compares the UDEMD to the thresholded ground distance, this suggests that UDEMD closely approximates the thresholded ground distance with $\lambda \approx 2^K$. 164

8.3 UDEMD is more scalable than Sinkhorn-OT and performs better than graph total variation. (left) Shows performance as measured by P@100, the fraction of the 100 nearest neighbors predicted correctly, against problem size. (middle) Shows time against problem size, and (right) shows performance vs. time on a problem size of $n = 2000$ for different choices of K. We find a good tradeoff at $K = 4$. 166

8.4 UDEMD achieves better clustering than Euclidean and total variation (TV) distances, and performs similarly well to Sinkhorn-OT but is much more scalable with similar scalability to Euclidean and TV distances. (a) performance in terms of Silhouette score, adjusted rand index (ARI), normalized mutual information (NMI), and adjusted mutual information (AMI). 167

8.5 (a) Visualization of gene graphs of 46 genes canonical for different cell types using UDEMD and Euclidean ground distances (blue for B cells, orange for monocytes and green for T cells), (b) heat map of gene distances (c) clustering performance (d) silhouette score vs. maximum diffusion scale K. 168

8.6 Embeddings of patients modeled as signals over the Snomed-CT graph using TV distance (a top) and using UDEMD distance (a bottom), colored by patient diagnosis. UDEMD better organizes the space as noted by selected terms in (b-c), difference of confusion matrices in (d) and k-nearest neighbors classification accuracy on the diagnosis in (e). In (b) note that the TV embedding (top) creates a spurious separation (due to noise in the signal) between subsets of patients who display intracranial bleeding that is not distinguished by diagnosis. On the other hand the UDEMD embedding (bottom) shows a continuum of patients with this diagnosis. The same holds for patients with brain mass or tumor shown in green. (c) UDEMD embedding organizes patients with acute coronary syndrome as a continuous trajectory with patients who are discharged (with milder cases) towards the bottom and more severe cases towards the top. The TV embedding again splits this trajectory. 169

8.7 Compares the thresholded ground distance with threshold level λ (orange) with the UDEMD for different maximum scales $K \in \{1 \dots 7\}$ (blue) on the sphere with $n = 10000$. Larger K corresponds to larger λ as shown in the bottom right figure in a roughly linear relationship. 174

List of Tables

3.1 MNIST classification training and test accuracies for coefficient selected using cross validation over regularization weights in $[10^{-7}, 10^{-6}, \ldots, 10^{-2}]$ for various regularization methods with standard deviation over 10 replicates. . 27

3.2 Structure of MNIST classifiers with Laplacian smoothing 30

4.1 Dataset statistics, diameter, nodes, edges, clustering coefficient (CC) averaged over all graphs. Split into bio-chemical and social network types. . . . 42

4.2 Mean ± std. over 10 test sets on bio-chemical (top) and social network (bottom) datasets. 43

4.3 Train and test set mean squared error on CASP GDT regression task over three seeds. 46

4.4 Mean ± std. over four runs of mean squared error over 19 targets for the QM9 dataset, lower is better. 47

4.5 Mean ± std. over test set selection on cross-validated LEGS-RBF Net with reduced training set size. 55

4.6 Test set mean squared error on CASP GDT regression task across targets over 3 non-overlapping test sets. 56

4.7 Quantified distance between the empirically observed enzyme class exchange preferences of [49] and the class exchange preferences inferred from LEGS-FIXED, LEGS-FCN, and a GCN. We measure the cosine distance between the graphs represented by the chord diagrams in Figure 4.2. As before, the self-affinities were discarded. LEGS-Fixed reproduces the exchange preferences the best, but LEGS-FCN still reproduces well and has significantly better classification accuracy. 56

4.8 Mean ± standard deviation test set accuracy on biochemical and social network datasets. 57

4.9 Mean ± std. over four runs of mean squared error over 19 targets for the QM9 dataset, lower is better. 58

5.1 Shows the mean AUC over digits over 3 seeds for training set corruption levels from 0% to 10% on MNIST. 77

5.2 AUC on representative CIFAR10 classes over 3 seeds with mean over all classes. 77

5.3 Shows the mean anomaly rank of the black image in the nominal test data. Scores are in $[0, 1]$, higher is better. Scores are measured on the MNIST test set of each digit trained on the uncorrupted training set for each digit over 3 seeds. Since the black image is inherently easy to reconstruct, it receives a low anomaly score in reconstruction based models. 78

5.4 Mean ± std. over three runs on the six classes of the ENZYMES graph dataset of the area under the receiver operator characteristic (AUC) and the average precision (AP). 81

5.5 (a) Convolutional Lipschitz network architecture. (b) Convolutional autoencoder architecture. 82

5.6 (a) Dense Lipschitz network architecture. (b) Dense autoencoder architecture. 83

6.1 Shows the Wasserstein distance EMD and MSE for artificial datasets between the left out timepoint and the predicted points for our two generated datasets. Mean over 3 seeds. 98

6.2 Shows the Wasserstein distance between the left out timepoint and the predicted distribution for various methods on a 4 timepoint mouse embryo cortex dataset. Mean and standard deviation over 3 seeds. 100

6.3 Shows the Wasserstein distance (EMD) between the left out timepoint and the predicted distribution for various methods on the 5 timepoint Embryoid body dataset. 102

7.1 Classification accuracy, P@10, Spearman ρ and runtime (in minutes) on 70,000 distributions from Spherical MNIST. 131

7.2 Comparison of Multiscale Methods for Earth Mover's Distance 144

Acknowledgements

I am extremely grateful for all the support I have received these last few years in pursuing my Ph.D. There are so many people without whom I would not have made it this far.

First and foremost, I would like to thank my advisor Dr. Smita Krishnaswamy who has given me so many opportunities to learn and grow over these last three and a half years. Thank you for giving me the freedom to explore new ideas and introducing me to so many amazing colleagues and collaborators. I first met Smita through the deep learning seminar course she co-taught with Dr. Guy Wolf in 2018. That course and the people I met there convinced me to change my research direction and pursue the research presented here.

I would like to thank Professor James Aspnes for guiding my early graduate research. I had little experience doing research from undergrad so many of my foundational research skills are from our time working together.

I am forever grateful for the support of Dr. Guy Wolf. One of my first interactions with Guy was at a conference in New Orleans, and went something like "You're one of Smita's students, right? Great you should talk to this person." After which he promptly walked away. Guy thanks for introducing me to many collaborators many of whom I now call friends. I have had many more experiences with Guy since then and he always brings the fun and humor.

To the graduate student and postdoc colleagues I have had a chance to work with and laugh with, Matt Amodio, Dan Burkhardt, Egbert Castro, Scott Gigante, Manik Kuchroo, Jay Stanley, Aarthi Venkat. David van Dijk, Jessie Huang, Feng Gao, and all the other collaborators I have had a chance to work with, thank you for your thoughts, time, and advice. You made my graduate experience so much more enjoyable and productive.

To the wonderful undergrads I have had a chance to meet and work with: Kincaid Macdonald, Katherine Du, Andrew Benz, Brandon Zhu. It has been a wonderful time working together. Thank you for your enthusiasm, hard work, and invaluable contributions to this and other work.

I did not get here on my own. Thank you to all my amazing teachers at Tufts University and Lakeside school for giving me the education and mentorship that got me here. Keith Klinger, Lauren Bricker, Todd Kresser, and Siva Sankrithi. Thank you for your excellent teaching which I can only aspire to live up to. Thank you to Lenore Cowen, Soha Hassoun, Greg Aloupis, and Ben Hescott at Tufts for teaching me the field I now love, without you I would not be here today.

Finally, I owe my success to my family. My parents Rich Tong and Connie Mao, and siblings Calvin and Grace, thank you for your support and for always being there for me.

Chapter 1

Introduction

The field of data analysis is constantly changing. At my friend's defense someone asked if there will ever come a time when we no longer need new algorithms, a time when the field of data analysis will be as simple as picking the right tool off the shelf and applying it to some new data. The answer is of course no, as long as the data, the computers, or the questions continue to change we will always need new algorithms and methods.

Advances in data generation, processing, and storage have give us the unprecedented ability to learn from this data. In some areas The field of geometric data analysis attempts to make sense of data using local relationships between points.

The field of single-cell data analysis has only recently become possible with the advent of cheap sequencing technologies, allowing us to measure many aspects of individual cells at once. This leads to very high dimensional and noisy datasets which come from an underlying continuous manifold. This assumption on the data generating process is a common theme throughout this work, and we will see other priors come into play that are best described using one literature or another.

This dissertation begins with an overview of graph signal processing, deep learning and optimal transport in Chapter 2. While these fields have developed from different literature, each tries to understand how points or distributions of points sampled from some underlying metric space relate to each other. In this dissertation we blend ideas from these fields to gain insight into biomedical data, with an emphasis on single-cell transcriptomic data but also other graph and image datasets. These datatypes have in common that they either have some intrinsic relationships between points as in graph datasets, or are assumed to be sampled from some underlying low dimensional manifold, which can be represented locally.

This dissertation is divided into three sections, Part I focuses on blending deep learning and graph signal processing by borrowing ideas from graph signal processing to make more interpretable deep learning components and borrowing ideas from deep learning to make more performant inherently interpretable models developed from graph signal processing. Part II blends deep learning and optimal transport to create deep learning models with interpretable properties. Finally, in Part III we bring it full circle, blending optimal transport and graph signal processing for a fast embedding-based approach to optimal transport on points sampled from an underlying manifold. This embedding based approach allows us to quickly approximate the Wasserstein distance between many distributions without having to solve pairwise optimization problems.

In Chapter 3 we begin by proposing a more interpretable deep learning architecture using ideas from graph signal processing. Neurons in a layer are unordered and equivalent under permutation, thus from run to run it may be difficult to find the neuron responsible for a specific function. We add structure to the layer by imposing a graph structure between neurons in a given neural network layer. This leads to more repeatable and interpretable layers, where a neuron will perform the same function from run to run based on the graph structure by breaking the symmetries of weights in standard architectures without reducing expressiveness.

In Chapter 4 we bring ideas from deep learning to geometric scattering. Previous work showed that geometric scattering is competitive with current graph neural network architectures developed from the deep learning literature in terms of performance [73], but is not as flexible to new data, relying on a number of fixed parameters. In this chapter we investigate the benefits of a more flexible geometric scattering network which we call *learnable geometric scattering* (LEGS) which allows us to learn the fixed scattering parameters. We show this is particularly useful in biomedical graphs, where there are a wide variety of graphs, from small densely connected graphs, to large and sparsely connected graphs. The additional flexibility of LEGS proves useful in adapting to these varied datatypes.

In Chapter 5 we bring ideas from optimal transport to bear in the unsupervised anomaly detection problem. Here the goal is to given a training sample of normal points build a model that can detect anomalous points. For example, in images given a set of images of dogs, detect the cats in a test set. Generally this problem is tackled using reconstruction-based approaches, train a limited capacity model to reconstruct the training set then score test images based on how well this model reconstructs them, the idea being that the model will not be able to reconstruct points far from the training set. We point out three issues with this approach and suggest a different approach based on a capacity restricted encoder that directly scores input points. We relate this to solving the dual of the Wasserstein distance, and apply this to theory to provide guarantees on model output.

In Chapter 6 we relate a particular deep learning model called a continuous normalizing flow (CNF) [38] to dynamic optimal transport. We then use this to model the trajectories of single cells over time. One problem with current single cell transcriptomic measurements are that they are destructive — measuring a cell's state destroys the cell. This makes it difficult to measure an individual cell as it progresses over time. Here we collect population level data at a number of timepoints and infer the individual cell trajectories using a regularized CNF. We show that we can more accurately model the cell state over time from single-cell time series data.

In Chapter 7 we relate optimal transport over a manifold to a series of multiscale diffusions over that graph. By comparing the diffusion behavior of distributions over the graph we can approximate the earth mover's distance rapidly, particularly when we are looking for nearest neighbor distributions in Wasserstein metric with a manifold geodesic ground distance. We show how this generalizes existing multiscale approaches to the earth mover's distance computation and improves speed and accuracy while extending this to graph domains.

In Chapter 8 we extend the work in Chapter 7 to unbalanced optimal transport, which blends the earth mover's distance and a metric-free total variation distance. Intuitively, this is allowing mass to be created and destroyed (at cost) instead of transported. We apply this new unbalanced transport to a medical concept knowledge graph to understand concepts extracted from patient notes.

The following chapters in this dissertation are based on the following publications and preprints. Links to full publications available at: `https://alextong.net/publications`

* † Denote equal contribution.

- Chapter 3: Tong, A.*, van Dijk, D.*, Stanley III, J. S., Amodio, M., Yim, K., Muhle, R., Noonan, J., Wolf, G.† & Krishnaswamy, S.† *Interpretable Neuron Structuring with Graph Spectral Regularization.* in Advances in Intelligent Data Analysis XVIII (2020).

- Chapter 4: Tong, A.*, Wenkel, F.*, MacDonald, K. Krishnaswamy S.† & Wolf, G.† *Data-driven Learning of Geometric Scattering Modules for GNNs.* in IEEE MLSP (2021).

- Chapter 5: Portions of this chapter are published in Tong, A., Wolf, G. & Krishnaswamy, S. *Fixing Bias in Reconstruction-based Anomaly Detection with Lipschitz Discriminators.* in IEEE MLSP (2020). **Best Student Paper Award**. Further work, particularly on the application to detection of anomalous graphs will appear in the Journal of Signal Processing Systems.

- Chapter 6: Tong, A., Huang, J., Wolf, G.†, van Dijk, D.† & Krishnaswamy, S.† *TrajectoryNet: A Dynamic Optimal Transport Network for Modeling Cellular Dynamics.* in Proceedings of the 37th International Conference on Machine Learning (2020).

- Chapter 7: Tong, A.*, Huguet, G.*, Natik, A.*, MacDonald, K., Kuchroo, M., Coifman, R., Wolf, G.† & Krishnaswamy, S.† *Diffusion Earth Mover's Distance and Distribution Embeddings.* in Proceedings of the 38th International Conference of Machine Learning (2021).

- Chapter 8: Tong A.*, Huguet G.*, Shung D.*, Natik A., Kuchroo M., Lajoie G., Wolf G. & Krishnaswamy S. *Embedding Signals on Knowledge Graphs with Unbalanced Diffusion Earth Mover's Distance.* ArXiv preprint (2021)

Chapter 2

Preliminaries

This chapter focuses on existing frameworks to understand the *geometry* of data. In Section 2.1 we will focus on graph methods, where the geometry is understood by the local neighborhoods of each point. Next in Section 2.2 we will explore deep learning methods. Finally in Section 2.3 we will explore the literature on optimal transport, a way of understanding collections of points and distributions within a geometry.

2.1 Graphs

Graphs are a useful way to capture the geometry between points. Graphs can either be built from data based on some affinity kernel, or defined based on some extrinsic information as is the case in social networks or knowledge graphs. Often we will take graphs as the discretization of an underlying probability generating function on some continuous (and low dimensional) underlying manifold.

2.1.1 Graphs and Random Walks on Data

A graph $G = (V, E)$ consists of a set of N nodes (generally the data points) connected by a set of (potentially weighted) edges connecting pairs of nodes $e = (v_i, v_j, w) \in E$. We will primarily use the adjacency matrix representation A which is a sparse matrix which contains the edge weights between nodes v_i and v_j in cell ij i.e. $A_{ij} = w(e_{ij})$. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ be a dataset of N points in $\mathbb{R}^d$. Given a positive semi-definite kernel

$k(x, y)$ then we can define the graph with adjacency matrix $A_{ij} = k(v_i, v_j)$. From this we can define the (diagonal) degree matrix $D = \text{diag}(d)$ where $d_i = \sum_j A_{ij}$. From this we will use the combinatorial Laplacian $L = D - A$, the normalized random walk Laplacian $L_{rw} = D^{-1}L = I - D^{-1}A$, and the symmetric normalized Laplacian $L_{sym} = D^{-1/2}LD^{-1/2}$.

2.1.2 Graph Signal Processing

The field of graph signal processing generalizes the theory of signal processing on one dimensional signals to the irregular signals on graphs. While some concepts translate well to graph signal processing, others do not as we will see in the following.

First we define a signal on a graph as a function defined on the nodes $f : V \to \mathbb{R}$. This can be generalized to higher dimensions or the complex domain, but we stick with the case of real signals for clarity. Given this signal f, we can define frequencies on the graph based on the spectrum of the graph Laplacian. We note that L is symmetric and positive semi-definite, thus has real, non-negative eigenvalues and can be decomposed into an orthogonal eigenvector matrix Ψ and diagonal matrix of eigenvalues Λ as $\Psi \Lambda \Psi^T = L$. We note that this also applies to the symmetric normalized Laplacian L_{sym}. We will always have $0 = \lambda_0 \leq \lambda_1 \leq \cdots \lambda_N \leq N$ for the combinatorial Laplacian. We take these as the frequencies of the graph. This analogy makes sense in the case of a circle graph where node i is connected to nodes $i + 1$ and $i - 1$ (modulo N). In this case the eigenvectors are the sine and cosine functions i.e. L has eigenvectors:

$$\sin(2\pi ki/N), \quad \cos(2\pi ki/N) \tag{2.1}$$

for $0 \leq k \leq N/2$, and has eigenvalues $2 - 2\cos(2\pi k/N)$. Increasing k corresponds to increasing frequency in both the graph and the one dimensional domains.

This implies for a signal f on G we can define a fourier transform of f based on the frequencies of Ψ of L as $\hat{f} = \Psi f$. These are also called the loadings of f. We can define filters in this spectral domain of $\hat{f}$ operating on the eigenvalues of L. We can define various filters $h(\lambda)$ which operate in the spectral domain of L. We will mostly consider low-pass filters which keep the low frequency components of the signal.

2.1.3 Diffusion on a Graph

Define the lazy random walk matrix $P = (I + D^{-1/2}A)/2$. Here the laziness refers to the addition of self loops (averaging the adjacency matrix with I) which makes analysis of some periodic graphs simpler, for example bipartite graphs. The t-step lazy random walk of a signal f is then $P^t f$. This simulates a weighted random walk over the nodes of the graph with walk probability proportional to the edge weights. Belkin and Niyogi [13], Coifman and Lafon [46] used this random walk to define Laplacian eigenmaps and diffusion maps respectively. We will discuss diffusion maps as this will be helpful in understanding diffusion wavelets for Chapter 4 and the multiscale basis used for embedding signals on graphs in Chapters 7 and 8.

The diffusion map embeds points into Euclidean space such that the distance in Euclidean space between points is equivalent to the geodesic distance of an underlying manifold. Suppose $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ is a dataset of N points in $\mathbb{R}^d$ sampled from some lower dimensional manifold $\mathcal{M} \subset \mathbb{R}^d$. Then as the number of points gets large, we would like to approximate the geodesics along $\mathcal{M}$.

Coifman and Lafon [46] proposed constructing a Gaussian kernel over the points, normalizing for density, then embedding t step random walks on this graph.

The Gaussian affinity between points is defined as

$$k(x, y) = e^{-\|x-y\|_2^2/2\sigma^2} \tag{2.2}$$

Let $A_{ij} = k(x_i, x_j)$, and define then define the density normalized affinity matrix as

$$A^{(\alpha)} = D^{-\alpha} A D^{-\alpha} \tag{2.3}$$

where $D = \mathrm{diag}(\sum_j A_{ij})$ and the density normalized random walk as

$$P = \left(D^{(\alpha)}\right)^{-1} A^{(\alpha)} \tag{2.4}$$

where $D^{(\alpha)} = \mathrm{diag}(\sum_j A_{ij}^{(\alpha)})$.

P can be eigendecomposed as

$$P_{ij} = \sum_k \lambda_k \psi_k(x_i) \phi_k(x_j) \tag{2.5}$$

Where λ_k is the k^{th} eigenvalue, and ψ_k, ϕ_k are the k^{th} left and right eigenvectors respectively.

The diffusion distance between two points at time t can be thought of as the similarity between the rows of the P^t. Specifically,

$$D(x_i, x_j)_t^2 = \sum_k (P_{ik}^t - P_{jk}^t)^2 / \phi_0(k) \tag{2.6}$$

This can be equivalently defined in terms of the eigenvectors as

$$D(x_i, x_j)_t^2 = \sum_k \lambda_k^{2t} (\psi_k(x_i) - \psi_k(x_j))^2 \tag{2.7}$$

The diffusion map is defined as

$$\Psi_t(x) = \big(\lambda_0^t \psi_1(x), \lambda_2^t \psi_2(x), \ldots \lambda_2^t \psi_l(x)\big) \tag{2.8}$$

Defined in this way, the euclidean distance between points in the diffusion map are equal to the diffusion distance i.e.

$$D(x_i, x_j)_t^2 = \|\Psi_t(x_i) - \Psi_t(x_j)\|_2^2 \tag{2.9}$$

We will use the diffusion operator in Chapters 4 however we will find use for the L^1 distance more helpful in Chapters 7 and 8.

2.2 Deep Learning

Over the past decade deep learning has become a growing field, particularly in the image and natural language domains, but also in other datatypes. Initialy used for supervised classification and regression tasks — in some ways the simplest task — now models are being applied to more difficult tasks with fewer "labels" and less structured supervision. Tackling these more difficult problems moves us from the setting of modeling what we already know, to the setting of modeling what we do not know. In this work we aim to take general priors about how the data is structured and encode them into deep learning frameworks for more interpretable and accurate models that adhere to our prior knowledge. Whether this is in the type of anomalies that might occur as we investigate in Chapter 5 or in the behavior of the transcriptome of cells as in Chapter 6, building in these priors to deep neural networks is a key step in more useful models for discovery.

2.2.1 Autoencoders

One of the simplest models for unsupervised learning is the autoencoder. The goal here is often to create a better representation of the data by restricting the model capacity. In general we take an encoder $f : \text{Input} \to \mathbb{R}^d$ and a decoder $g : \mathbb{R}^d \to \text{Input}$ parameratized by deep neural networks and train them such that $\|g(f(x)) - x\|_2^2$ is small.

2.2.2 Lipschitz Neural Networks

A function is Lipschitz informally if its output does not vary too much over small changes in its input. This is a fairly weak statement of regularity, but is nevertheless useful, particularly in the context of optimal transport which we will see later. More formally, a function $f : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}$ is equipped with a metric $d_\mathcal{X}$ and $\mathcal{Y}$ is equipped with a metric $d_\mathcal{Y}$ is k Lipschitz continuous if for all x, y:

$$\frac{d_\mathcal{Y}(f(x), f(y))}{d_\mathcal{X}(x, y)} \leq k \tag{2.10}$$

It would be useful to optimize over the space of all Lipschitz functions, unfortunately it is often not computationally feasible. Instead it is much easier to optimize over the space of Lipschitz functions parameterized by a neural network architecture. There are a few works on how to enforce this constraint, the first is through weight clipping [8]. If all of the weights are small and each layer is 1-Lipschitz, then the whole network will also be 1-Lipschitz. Later work uses gradient regularization [84], with the idea of regularizing the magnitude of the gradient with respect to the input, as it is easy to show if $\|\nabla f(x)\| \leq k$ for all x then a function is k Lipschitz.

2.2.3 Continuous Normalizing Flows

A continuous normalizing flow as defined in [38, 81] models an ODE. Since these two works there is a growing literature on modeling ODEs. The continuous normalizing flow is paramterized by a neural network $f : (x, t) \to dx$. This neural network models the derivate of with respect to time of x. The magic of a continuous normalizing flow is the efficient forward and backwards integration of the flow with a standard neural network architechture for f. The backpropogation is achieved through the adjoint method and has constant memory needs regardless of the number of integration steps. This allows us to disentangle the number of integration steps from the parameterization and think about the input flow. We will use this framework in Chapter 6 to model the movement of cells through transcript space.

2.3 Optimal Transport

Optimal transport is a framework for lifting distances between points to distances between measures. Given a transport cost between locations, optimal transport is the problem of the best transportation of one measure to another. Originally studied by Kantorovich in the context of transporting goods to factories [100], the field of optimal transport has grown considerably in the last decade [50, 8, 191, 165, 61] and is widely applied across machine learning for both the analysis of datasets that are naturally analyzed as distributions [181], and as an interpretation of Lipschitz regularized functions [8, 84]. We now define the Wasserstein distance, also known as the Kantorovich or Kantorovich-Rubenstein metric.

Let μ, ν be two probability distributions on a measurable space Ω with metric $d(\cdot, \cdot)$, $\Pi(\mu, \nu)$ be the set of joint probability distributions π on the space $\Omega \times \Omega$, where for any subset $\omega \subset \Omega$, $\pi(\omega \times \Omega) = \mu(\omega)$ and $\pi(\Omega \times \omega) = \nu(\omega)$. The α-Wasserstein distance W_d also known as the earth mover's distance (EMD) for $\alpha = 1$ is defined as:

$$W_d^{\alpha}(\mu, \nu) := \inf_{\pi \in \Pi(\mu,\nu)} \int_{\Omega \times \Omega} d(x,y)^{\alpha} \pi(dx, dy). \tag{2.11}$$

Define the Lipschitz constant of the function $f : \Omega \to \mathbb{R}$ with respect to the metric d as $\|f\|_d := \max_{x \in \Omega \neq y \in \Omega} \frac{|f(x) - f(y)|}{d(x,y)}$, then the dual of equation 2.11 is:

$$W_d(\mu, \nu) = \sup_{f : \|f\|_d \leq 1} \int_{\Omega} f(x)\mu(dx) - \int_{\Omega} f(y)\nu(dy). \tag{2.12}$$

This formulation is known as the Kantorovich-Rubenstein dual with the witness function f. Chapters 5, 7, and 8 will exploit this dual formulation in particular for its computational benefits and its usefulness as optimizing over the space of regularized functions.

2.3.1 Integral Probability Measures and ϕ-divergences

This dual form connects the Wasserstein distance to a family of metrics over probability measures known as the integral probability measures (IPMs). For a family of functions $\mathcal{F}$ IPMs have the form:

$$\text{IPM}(\mu, \nu) := \sup_{f \in \mathcal{F}} \left| \int f(x)\mu(dx) = \int f(y)\nu(dy) \right|. \tag{2.13}$$

A number of metrics can be formulated in this way:

1. **Wasserstein metric** with respect to the ground metric d is the IPM with the family $W_d := \{f \mid \|f\|_d \leq 1\}$.

2. **MMD** the maximum mean discrepency with respect to the kernel k is the IPM with the family $M_k := \{f \in \mathcal{H}_k \mid \|f\|_k \leq 1$, where $\|f\|_k$ denotes the norm of f in the reproducing kernel Hilbert space (RKHS) of k.

3. **TV** the total variation distance is the IPM with $TV := \{f \mid \max_{x \in \Omega} |f(x)| \leq 1\}$.

4. **Kolmogorov** when $\Omega := \mathbb{R}^D$ is the IPM with $\mathcal{K} := \{f \mid \mathbf{1}_{(-\infty,x)} : x \in \mathbb{R}^D\}$.

Another widely studied family of divergences are the ϕ-divergences defined as divergences of the following form.

$$D_\phi(\mu, \nu) := \begin{cases} \int_\Omega \phi(\frac{\mu}{\nu}) & \text{if } \mu \ll \nu \\ +\infty & \text{else} \end{cases}, \tag{2.14}$$

where $\phi : \Omega \to \mathbb{R}$ is a convex function, $\phi(1) = 0$, and $\mu \ll \nu$ denotes μ is absolutely continuous with respect to ν. A number of well known metrics fall into this family with $\Omega := \mathbb{R}$:

1. **KL** the Kullback-Liebler divergence is a ϕ-divergence with $\phi(x) = x \log x$.

2. **TV** the total variation distance is a ϕ-divergence with $\phi(x) = |x - 1|$.

3. **Hellinger** with $\phi(x) = (\sqrt{x} - 1)^2$.

4. χ^2**-divergence** with $\phi(x) = (x - 1)^2$.

Interestingly the Total Variation distance is the only (non-trivial) metric that is both a ϕ-divergence and an integral probability metric [193]. An intuitive interpretation of the differences between IPMs and ϕ-divergences is that ϕ-divergences are vertical distances between measures and IPMs are horizontal. ϕ-divergences are parameterized in terms of the ratio of $\frac{\mu}{\nu}$ over the space, this can be thought of as distances which depend only on the pointwise ratio of the two measures. In contrast, IPMs are horizontal distances motivated by *weak convergence* of measures, in that they compare measures based on a (potentially horizontally regularized) function. In particular, in the case of an IPM with a Lipschitz regularized family of functions, this is exactly a horizontal distance, as is clearly expressed in the primal form of the Wasserstein distance.

2.3.2 Unbalanced Wasserstein Distances

The unbalanced Wasserstein distance considers problems where the measures μ and ν are unbalanced and may have unequal mass. To deal with these unequal masses, they consider a Wasserstein distance with additional teleportation cost usually taken as a ϕ-divergence.

The Unbalanced Wasserstein distance (UW) can be generally defined as:

$$\mathrm{UW}_d(\mu, \nu) := \inf_{\mu', \nu'} \left\{ \mathrm{W}_d(\mu', \nu') + \lambda_\mu D_\phi(\mu, \mu') + \lambda_\nu D_\phi(\nu, \nu') \right\}. \tag{2.15}$$

In connection with the entropic regularized version of optimal transport [50] the KL-divergence is a common choice. In particular, as proved in [129]and [44] the Wasserstein-Fisher-Rao where $d(x, y) = -\log\cos(\min(d(x, y)/\delta, \pi/2))$, with some cutoff distance δ then

$$\mathrm{WFR}(\mu, \nu) := \mathrm{UW}_d(\mu, \nu)^{\frac{1}{2}} \tag{2.16}$$

Later we will investigate a similarly useful variant using the total variation unbalanced Wasserstein distance in Chapter 8.

2.3.3 Efficient Computation

When both μ and ν are discrete distributions over the a discrete measure space $(\mathcal{X}, d_{\mathcal{X}})$ then there are a variety of algorithms for computing the transport exactly and approximately. When μ, ν are discrete distributions over points in $\mathbb{R}^d$ with m, n sized supports respectively this can be equivalently expressed in matrix notation as

$$\begin{aligned} W_d(\mu, \nu) := \min_{\Pi \geq 0} & \sum_{i=1}^{m} \sum_{j=1}^{n} \Pi_{ij} d(x_i, x_j) \\ \text{subject to: } & \sum_{i=1}^{m} \Pi_{ij} = \nu_j, \quad \forall j \in \{1, \dots, n\} \\ & \sum_{j=1}^{n} \Pi_{ij} = \mu_j, \quad \forall i \in \{1, \dots, m\} \end{aligned} \tag{2.17}$$

This constrained optimization can be solved using a network-flow algorithm in $\tilde{O}(n^3)$ time and $O(n^2)$ space. This however, is much to slow to be useful in modern machine learning problems where linear or log-linear time is preferred.

Entropic Regularization Since formulated in 2013 by Cuturi [50], the entropy regularized version of the Wasserstein distance has received significant attention. Somewhat surprisingly, adding an entropy penalty to the transport matrix admits an extremely simple algorithm and potentially more generalizable performance in some situations. The entropy regularized version of the problem relies on a matrix entropy penalty $\mathcal{H}(\Pi) = -\sum_{ij} \Pi_{ij} \log \Pi_{ij}$. The entropy regularized optimization is now:

$$\begin{aligned} W_d(\mu, \nu) &:= \min_{\Pi \geq 0} \sum_{i=1}^{m} \sum_{j=1}^{n} \Pi_{ij} d(x_i, x_j) + \gamma \mathcal{H}(\Pi) \\ \text{subject to: } &\sum_{i=1}^{m} \Pi_{ij} = \nu_j, \quad \forall j \in \{1, \dots, n\} \\ &\sum_{j=1}^{n} \Pi_{ij} = \mu_j, \quad \forall i \in \{1, \dots, m\} \end{aligned} \tag{2.18}$$

By using the Sinkhorn scaling algorithm, we can solve this optimization by iterative matrix vector products of the exponentiated cost matrix $\exp(-\gamma d)$. In practice, when the costs are near 1, the γ regularization can be as low as 0.01 without running into numerical difficulties. The Sinkhorn method has a time complexity of $\tilde{O}(n^2)$ and space complexity of $O(n^2)$. There are more recent works that attempt to scale this to larger problems. Using nystrom extension, manifold approximation [192], multiscale approximation [96], or gradient descent [8, 76] can all reduce the computational expense.

Computation using the dual formulation Since it is in general difficult to optimize over the entire space of 1-Lipschitz functions, many works optimize the cost over a modified family of functions such as functions parameterized by clipped neural networks [8], functions defined over trees [115], or functions defined over Haar wavelet bases [75]. In Chapters 7 and 8 we will utilize wavelets defined over a graph to approximate the optimal Lipschitz witness function.

Part I

Deep Learning Using Graph Spectral Priors

Chapter 3

Graph Spectral Regularization

3.1 Introduction

Common intuitions and motivating explanations for the success of deep learning approaches rely on analogies between artificial and biological neural networks, and the mechanism they use for processing information. However, one aspect that is overlooked is the spatial organization of neurons in the brain. Indeed, the hierarchical spatial organization of neurons, determined via fMRI and other technologies [153, 134], is often leveraged in neuroscience works to explore, understand, and interpret various neural processing mechanisms and high-level brain functions. In artificial neural networks (ANN), on the other hand, hidden layers offer no organization that can be regarded as equivalent to the biological one. This lack of organization poses great difficulties in exploring and interpreting the internal data representations provided by hidden layers of ANNs and the information encoded by them. This challenge, in turn, gives rise to the common treatment of ANNs as black boxes whose operation and data processing mechanisms cannot be easily understood. To address this issue, we focus on the problem of modifying ANNs to learn more interpretable feature spaces without degrading their primary task performance.

While most neural networks are treated as black boxes, we note that there are methods in ANN literature for understanding the activations of filters in convolutional neural networks (CNNs) [117], either by examining trained networks [221], or by learning a better representation [126, 223, 172, 194, 176], but such methods rarely apply to other types of networks, in particular dense neural networks (DNNs) where a single activation is often not interpretable on its own. Furthermore, convolutions only apply to datatypes where we know the feature structure apriori, as in the case of images and natural language. In layers of a DNN, there is no enforced structure between neurons. The correspondence between neurons and concepts is only determined based on the random initialization of the network. In this work, we encourage *structure between neurons* in the same layer, creating more localized and interpretable layers in dense architectures.

More specifically we propose a *Graph Spectral Regularization* to encourage arbitrary graph structure between neurons within a layer. The internal layers of a neural network are constrained to take the structure of a graph, with graph neighbors activating on similar inputs. This allows us to map the activations of a given layer over the graph and interpret new input by examining the activations. We show that graph-structuring a hidden layer causes useful, interpretable features to emerge. For instance, we show that grid-structuring a layer of a classification network creates a structure over which convolution can be applied, and local receptive fields can be traced to understand classification decisions.

While a majority of the time imposing a known graph structure gives interpretable results, there are circumstances where we would like to learn the graph structure from data. In such cases we can learn and emphasize the natural graph structure of the feature space. We do this by an iterative process of encoding the data, and modifying the graph based on the feature co-activation patterns. This procedure reinforces existing patterns in the data. This allows us to learn an abstracted graph structure of features in high-dimensional domains such as single-cell RNA sequencing.

The main contributions of this work are as follows: (1) Demonstration of hierarchical, spatial, and smoothed feature maps for interpretability in dense networks. (2) A novel method for learning and reinforcing the natural graph structure for complex feature spaces. (3) Demonstration of graph learning and abstraction on single-cell RNA-sequencing data.

3.2 Related Work

Disentangled Representation Learning: While there is no precise definition of what makes for a disentangled representation, the aim is to learn a representation that axis aligns with the generative factors of the data [91, 4]. [90] suggest a way to disentangle the representation of variational autoencoders [104] with β-VAE. Subsequent work has generalized this to discrete representations [59], and simple hierarchical representations [65]. These works focus on learning a single vector representation of the data, where each element represents a single concept. In contrast, our work learns a representation where groups of neurons may be involved in representing a single concept. Moreover, disentangled representation learning can only be applied to unsupervised models and only the most compressed level of either an autoencoder [90] or generative adversarial network as in [40], whereas graph spectral regularization (GSR) can be applied to any or all layers of the network.

Graph Structure in ANNs: Graph based penalties have been used in the graph signal processing literature [14, 225, 189], but are rarely used in an ANN setting. In the biological data setting, [142] used a graph penalty in sparse logistic regression on gene expression data. Another way of utilizing graph structure is through graph convolutional networks (GCN). GCNs are a related body of work introduced by [80], and expanded on by [180], but focus on a different set of problems (For an overview see [216]). GCNs require a known graph structure. We focus on learning a graph representation of general data. This learned graph representation could be used as the input to a GCN similar to our MNIST example.

3.3 Enforcing Graph Structure

We consider the intra-layer relationships between neurons or larger structures such as capsules. For a given layer of neurons we construct a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_N\}$ the set of vertices and $E \subseteq V \times V$ the set of edges. Let W be the weighted symmetric adjacency matrix of size $N \times N$ with $W_{ij} = W_{ji} \geq 0$ representing the weight of the edge between v_i and v_j. The graph Laplacian L is then defined as $L = D - W$ where $D_{ii} = \sum_j W_{ij}$ and $D_{ij} = 0$ for $i \neq j$.

To enforce smoothing we use the Laplacian smoothing loss. On some activation vector z and fixed Laplacian L we formulate the graph spectral regularization function G as:

$$G(z, \mathbf{L}) = z^T \mathbf{L} z = \sum_{ij} W_{ij} ||z_i - z_j|| \tag{3.1}$$

Where $||\cdot||$ denotes the Frobenius norm. We add it to the reconstruction or classification loss with a weighting term α. This adds an additional objective that activations should be smooth along the graph defined by L. This optimization procedure applies to any multi-layer model and valid graph Laplacian. We apply this algorithm to grid, and hierarchical graph structures on both autoencoder and classification dense architectures.

Algorithm 1 Graph Learning

```
Input batches $x_i$, model $M$ with latent layer activations $z_i$, regularization weight $\alpha$.
Pre-train $M$ on $x_i$ with $\alpha = 0$
for $i = 1$ to $T$ do
    Create Graph Laplacian $L_i$ from activations $z_i$
    for $j = 1$ to $m$ do
        Train $M$ on $x_i$ with $\alpha = w$ and $L = L_i$ with MSE + loss in eq. 3.1
    end for
end for
```

3.3.1 Learning and Reinforcing an Abstracted Feature-space Graph

Instead of enforcing smoothness over a fixed graph, we can learn a feature graph from the data (See Algorithm 1) using neural network activations themselves to bootstrap the process. Note, that most graph and kernel-based methods are applied over the space of observations but not over the space of features. One of the reasons is because it is even more difficult to define a distance between features than it is between observations. To circumvent this problem, we propose to learn a feature graph in the latent space of a neural network using feature co-activations as a measure of similarity.

We proceed by creating a graph using feature activation similarity, then applying this graph using Laplacian smoothing for a number of iterations. This converges to a graph of a latent feature space at the level of granularity of the number of dimensions in the corresponding layer.

Our algorithm for learning the graph consists of two phases. First, a pretraining phase where the model is learned with no graph regularization. Second, we alternate between constructing the graph from the similarities of the embedding layer features and further training the network for reconstruction and smoothness on the graph. There are many ways to create a graph from the feature × datapoint activation matrix. We use an adaptive Gaussian kernel,

$$K(z_i, z_j) = \frac{1}{2}exp\bigg(-\frac{||z_i - z_j||_2^2}{\sigma_i^2}\bigg) + \frac{1}{2}exp\bigg(-\frac{||z_i - z_j||_2^2}{\sigma_j^2}\bigg)$$

where σ_i is the adaptive bandwidth for node i which we set as the distance to the k^{th} nearest neighbor of feature. An adaptive bandwidth Gaussian kernel is necessary for general architectures as the scale of the activations is not fixed. Batch normalization can also be used to limit the activation scale.

Since we are smoothing on the graph then constructing a new graph from the smoothed signal the learned graph converges to a steady state where the mean squared error acts as a repulsive force to stop the graph collapsing any further. We present the results of graph learning a biological dataset and show that the learned structure adds interpretability to the activations.

3.4 Experiments

Through examples, we show that visualizing the activations of data on the regularized layer highlights relationships in the data that are not easily visible without it. We establish this with two examples on fixed graphs, then move to graphs learned from the structure of the data with two examples of hierarchical structure and two with progression structure.

3.4.1 Fixed Structure

Enforcing fixed graph structure localizes activations for similar datapoints to a region of the graph. Here we show that enforcing a 8x8 grid graph on a layer of a dense MNIST classifier causes receptive fields to form, where each digit occupies a localized group of neurons on the grid. This can, in principle, be applied to any neural network layer to group neurons activating to similar features. Like in FMRI data or a convolutional neural network, we can examine the activation patterns for each localized group of neurons. For a second example, we show the usefulness in encouraging localized structure on a capsulenet architecture [176]. Where we are able to create globally consistent structure for better alignment of features between capsules.

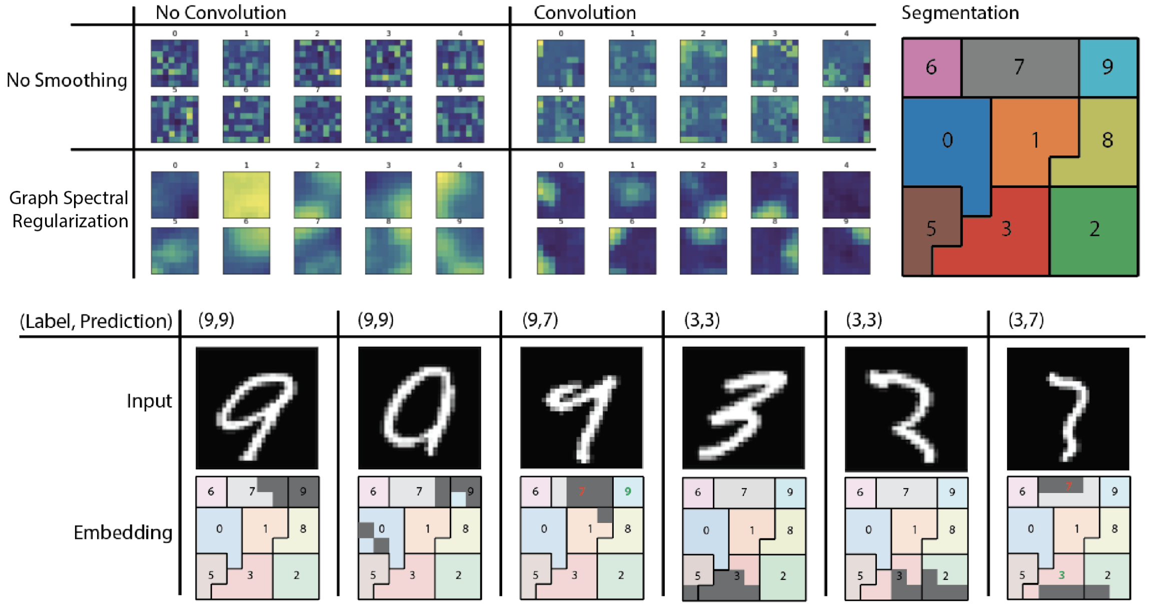


Figure 3.1: Shows average activation by digit over an (8x8) 2D grid using graph spectral regularization and convolutions following the regularization layer. Next, we segment the embedding space by class to localize portions of the embedding associated with each class. Notice that the digit 4 here serves as the null case and does not show up in the segmentation. Finally, we show the top 10% activation on the embedding of some sample images. For two digits (9 and 3) we show a normal input, a correctly classified but transitional input, and a misclassified input. The highlighted regions of the embedding space correlate with the semantic description of the input.

Enforcing Grid Structure on MNIST

Without GSR, activations are unstructured and as a result are difficult to interpret, in that it is difficult to visually identify even which class a digit comes from based on the activation pattern (See Fig. 3.1). With GSR we can organize the activations making this representation more visually distinguishable. Since we can now take this embedding as an image, it is possible to use a standard convolutional architecture in subsequent layers in order to further filter the encodings. When we add 3 layers of 3x3 2D convolutions with 2x2 max pooling we see that representations for each digit are compressed into specific areas of the image. This leads to the formation of receptive fields over the network pertaining to similar datapoints. Using these receptive fields, we can now extract the features responsible for digit classification. For example, features that contribute to the activation of the top right of our grid we can associate with those features that contribute to being the digit 9.

The activation patterns on the embedding layer correspond well to a human perception of the digit type. The 9 that is misclassified as 7 both has significant activation in the 7 region of the embedding layer, and looks visually close to a 7. We can now interpret the embedding layer as a sort of brain map, where the map can map regions of activations, to types of inputs. This is not possible in a standard neural network, where activations are not spatially organized.

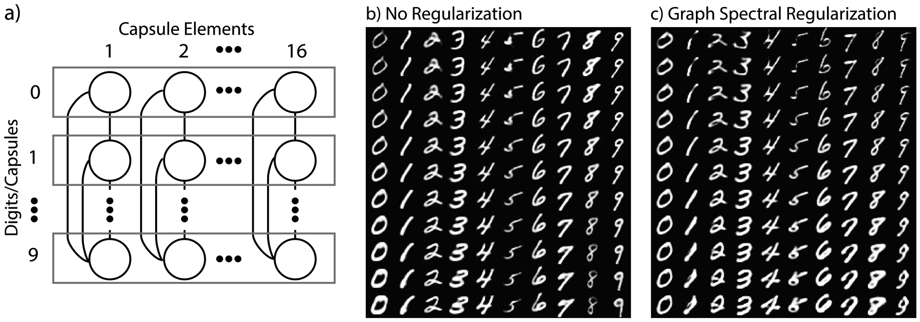


Figure 3.2: (a) shows the regularization structure between capsules. (b-c) Show reconstruction when one of the 16 dimensions in the DigitCaps representation is tweaked by $0.05 \in [-0.25, 0.25]$. (b) Without GSR each digit responds differently to perturbation of the same dimension. With GSR (c) a single dimension represents line thickness across all digits.

Enforcing Node Consistency on Capsule Networks

Capsule networks [176] represent the input as a set of vectors where norm denotes activation and each component corresponds to some abstract feature. These elements are generally unordered. Here we use GSR to order these features consistently between digits. We train a capsule net on MNIST with GSR on 16 fully connected graphs between the 10 digit capsules. In the standard capsule network, each capsule orders features randomly based on initialization. However, with GSR we obtain a *consistent feature ordering*, e.g. node 1 corresponds to line thickness across all digits. GSR enforces a more ordered and interpretable encoding where localized regions are similarly organized, and the global line thickness feature is consistently learned between digits. More generally, GSR can be used to order nodes such that features common across capsules appear together. Finally, GSR does not degrade performance much, as can be seen by the digit reconstructions in Fig. 3.2.

In these examples the goal was to enforce a specified structure on unstructured features, but next we will examine the case where the goal is to learn the structure of the reduced feature space.

3.4.2 Learning Graph Structure

Using the procedure defined in Sec. 3.3.1, we can learn a graph structure. We first show that depending on the data, the learned graph exhibits either cluster or trajectory structure. We then show that our framework can learn structures that are hierarchical, i.e. subclusters within clusters or trajectories within clusters. Hierarchies are a difficult structure for other interpretability methods to learn [65]. However, our method naturally captures this by allowing for arbitrary graph structure among neurons in a layer.

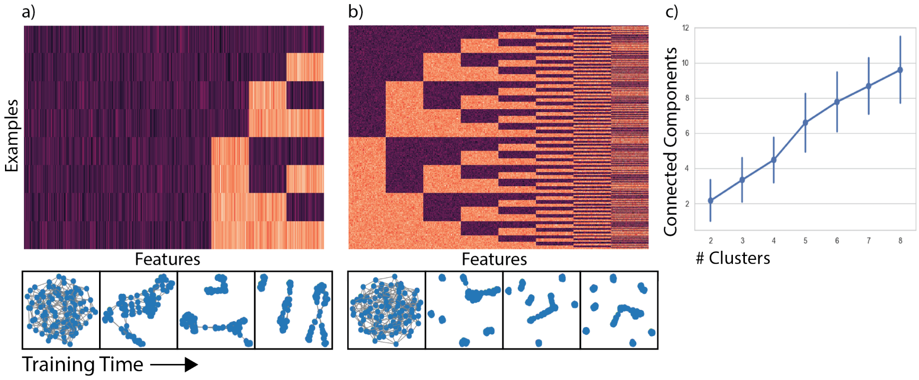


Figure 3.3: We show the structure of the training data and snapshots of the learned graph for (a) three modules and (b) eight modules. (c) shows we have the mean and 95% CI of the number of connected components in the trained graph for over 50 trials.

Cluster Structure on Generated Data

We structure our n^{th} dataset to have exactly n feature clusters. We generate the data with n clusters by first creating 2^n data points representing the binary numbers from 0 to $2^n - 1$, then added Gaussian noise $N(0, 0.1)$. This creates a dataset with a ground truth number of feature clusters. In the n^{th} dataset the learned graph should have n connected components for n independent features. In Fig. 3.3 (a-b) we can see how this graph evolves over time for 3 and 8 modules. (c) shows how the learned graph learns the correct number of connected components for each ground truth number of clusters.

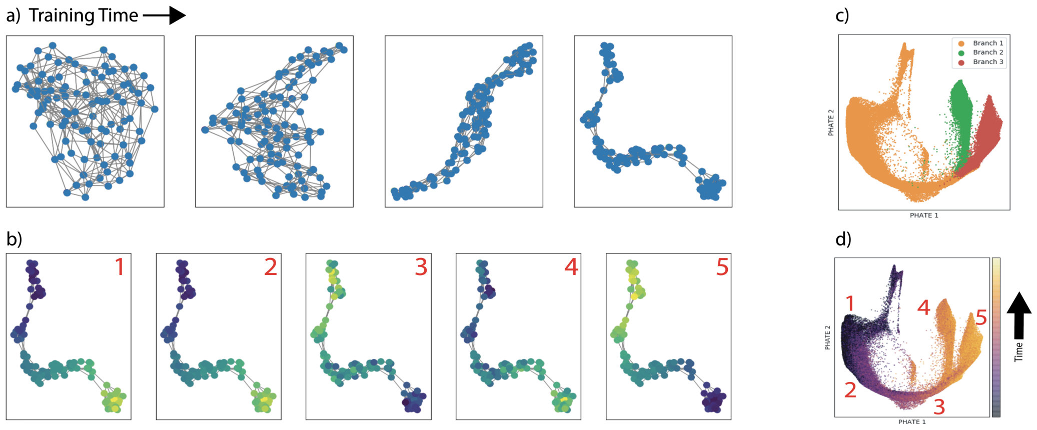


Figure 3.4: Shows (a) graph structure over training iterations (b) feature activations of parts of the trajectory. PHATE [146] embedding plots colored by (c) branch number and (b) inferred trajectory location showing the branching structure of the data.

Trajectory Structure on T cell Development Data

Next, we test graph learning on biological mass cytometry data, which is a high dimensional, single-cell protein dataset, measured on differentiating T cells from the Thymus [185]. The T cells lie along a bifurcating progression where the cells eventually diverge into two lineages (CD4+ and CD8+). Here, the structure of the data is a trajectory (as opposed to a pattern of clusters). We can see in Fig. 3.4 how the activated nodes in the graph embedding layer correspond to locations along the data trajectory, and importantly, the learned graph is a single connected component. The activated nodes (yellow) move from the bottom of the embedding to the top as T-cells develop into CD8+ cells. The CD4+ lineage is also CD8-

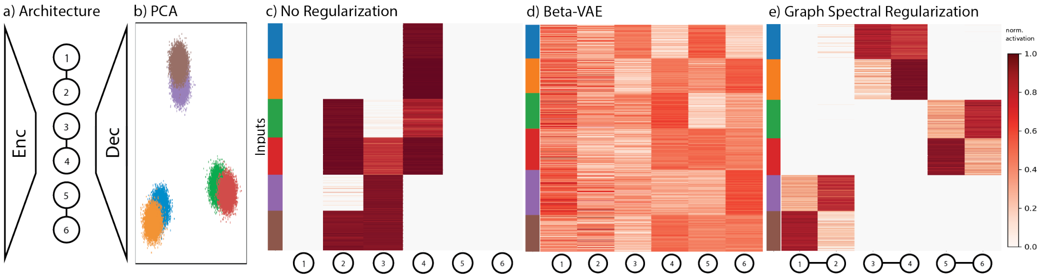


Figure 3.5: Graph architecture, PCA plot, activation heatmaps of a standard autoencoder, β-VAE [90] and a graph regularized autoencoder. With relu activations normalized to $[0, 1]$ for comparison. In the model with graph spectral we are able to clearly decipher the hierarchical structure of the data, whereas with the standard autoencoder or the β-VAE the structure of the data is not clear.

and thus looks like a mixture between the CD8+ branch and the naive T cells. The learned graph structure here has captured the transitioning structure of the underlying data.

Clusters within Clusters on Generated Data

We demonstrate graph spectral regularization on data that is generated with a structure containing sub-clusters. Our data contains three large-scale structures, each comprising two Gaussian sub clusters generated in 15 dimensions (See Fig. 3.5). We use this dataset as it has both global and local structure. We demonstrate that our graph spectral regularized model is able to pick up on both the global and local structure of this dataset where disentangling methods such as β-VAE cannot. We use a graph-structure layer with six nodes with three connected node pairs and employ the graph spectral regularization. After training, we find that each node pair acts as a "super node" that detects each large-scale cluster. Within each super node, each of the two nodes encodes one of each of the two Gaussian substructures. Thus, this specific graph topology is able to extract the hierarchical topology of the data.

Hierarchical Cluster and Trajectory Structure on Developing Mouse Cortex Data.

In Fig. 3.6 we learn a graph on a single-cell RNA-sequencing dataset of over 4000 cells and over 8000 genes. The data contains a set of cells in the process of developing from neural

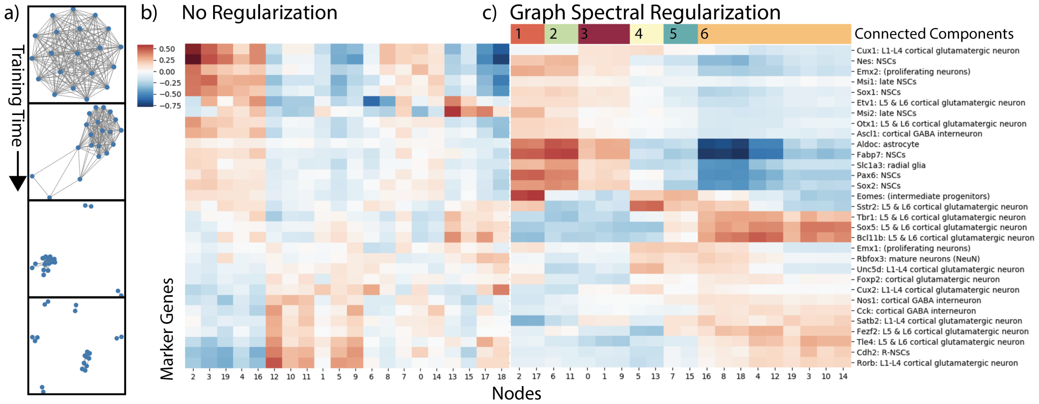


Figure 3.6: Shows correlation between a set of marker genes for specific cell types and embedding layer activations. First with the standard autoencoder, then our autoencoder with graph spectral regularization. The left heatmap is biclustered, the right heatmap is grouped by connected components in the learned graph. We can see progression especially in the largest connected component where features on the right of the component correspond to less developed neurons.

stem cells to full neurons in the mouse brain. While there are many gene modules that contribute to the neuronal development, there are some states that have been studied. We use a list of cell type marker genes to validate our method. We use 1000 PCA components of the data in an autoencoder with a 20-dimensional embedding space. We learn the graph using an adaptive bandwidth gaussian kernel with the bandwidth for each feature set to the Euclidean distance to the nearest neighboring feature.

Our graph learns six components that represent meta features over the gene space. We can identify each with a specific type of cell or related types of cells. For example, the light green component (cluster 2) represents the very early stage neural stem cells as it is highly correlated with increased Aldoc, Pax6 and Sox2 gene expression. Most interesting to examine is cluster 6, the largest component, which represents development into mature neurons. Within this component we can see a progression from just after intermediate progenitors on the left (showing Eomes expression) to more mature neurons with higher expression of Tbr1 and Sox5. With a standard autoencoder we cannot see progression structure of this dataset. While some of the more global structure is captured, we fail to see the data progression from intermediate progenitors to mature neurons. Learning a graph allows us to create receptive fields e.g. clusters of neurons that correspond to specific structures within the data, in this case cell types. Within these neighborhoods, we can pick up on the substructure within a single cell type, i.e. their developmental trajectory.

3.4.3 Computational Cost

Our method can be used to increase interpretability without much loss in representation power. At low levels, GSR can be thought of as rearranging the activations so that they become spatially coherent. As with other interpretability methods, GSR is not meant to increase representation power, but create useful representations with low cost in power. Since GSR does not require an information bottleneck such as in β-VAE, a GSR layer can be very wide, while still being interpretable. In comparing loss of representation power, GSR should be compared to other regularization methods, namely L1 and L2 penalties (See Table 3.1). In all three cases we can see that a higher penalty reduces the model capacity. GSR affects performance in approximately the same way as L1 and L2 regularizations do. To confirm this, we ran a MNIST classifier and measured train and test accuracy with 10 replicates. Graph spectral regularization adds a bit more overhead than elementwise activation penalties. However, the added cost can be seen as containing one matrix vector operation per pass. Empirically, GSR shows similar computational cost as other simple regularizations such as L1 and L2. To compare costs, we used a Keras model with Tensorflow backend [1] on a Nvidia Titan X GPU and a dual Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz, and with batchsize 256. we observed during training 233 milliseconds (ms) per step with no regularization, 266ms for GSR, and 265ms for L2 penalties.

Regularization	Training accuracy	Test Accuracy	Coefficient
None	99.1 ± 0.3	97.5 ± 0.3	N/A
L1	98.9 ± 0.3	97.4 ± 0.4	10^{-4}
L2	98.3 ± 0.3	98.0 ± 0.2	10^{-4}
GSR (ours)	99.3 ± 0.3	98.0 ± 0.3	10^{-3}

Table 3.1: MNIST classification training and test accuracies for coefficient selected using cross validation over regularization weights in $[10^{-7}, 10^{-6}, \dots, 10^{-2}]$ for various regularization methods with standard deviation over 10 replicates.

3.5 Conclusion

We have introduced a novel biologically inspired method for regularizing features of the internal layers of dense neural networks to take the shape of a graph. We show that coherent features emerge and can be used to interpret the underlying structure of the dataset. Furthermore, when the intended graph is not known apriori, we have presented a method for learning the graph structure, which learns a graph relevant to the data. This regularization framework takes a step towards more interpretable neural networks, and has applicability for future work seeking to reveal important structure in real-world biological datasets as we have demonstrated here.

Appendix

We use Leaky relus with a coefficient of 0.2 [137] for all layers except for the embedding and output layers unless otherwise specified. We use the ADAM optimizer with default parameters [102].

Laplacian Smoothing on an Autoencoder We use an autoencoder with five fully connected layers for the hierarchical example. The layers have widths [50,50,6,50,50]. To perform Laplacian smoothing on this autoencoder we add a term to the loss function. Let ν be the activation vector on the embedding layer, then we add a penalty term $\alpha\nu^T L\nu$ where α is a weighting hyperparameter to the standard mean squared error loss. For the biological example the layers have widths [50,50,20,50,50]. When not otherwise mentioned we use a graph spectral regularization strength of $\alpha = 0.001$.

MNIST Classifier Architecture The basic classifier that we use consists of two convolution and max pooling layers followed by the dense layer where we apply Laplacian smoothing. We use the cross entropy loss to train the classification network in this case. Note that while we use convolutions before this layer for the MNIST example, in principle, techniques applied here could be applied to non image data by using only dense layers until the Laplacian smoothing layer which constructs an image for each datapoint. Table a shows the architecture when no convolutions are used. Table b exhibits the architecture when convolution and max pooling layers are used after the Laplacian smoothing layer constructs a 2D image.

#	type	patch/stride	depth	output size
1	convolution	5x5/1	32	28x28x32
2	max pool	2x2/2		14x14x32
3	convolution	5x5/1	64	14x14x64
4	max pool	2x2/2		7x7x64
5	dense		64	8x8
6	dense		10	1x10

(a) Basic MNIST classifier used with and without Laplacian smoothing on layer 5.

#	type	patch/stride	depth	output size
1	convolution	5x5/1	32	28x28x32
2	max pool	2x2/2		14x14x32
3	convolution	5x5/1	64	14x14x64
4	max pool	2x2/2		7x7x64
5	dense		64	8x8
6	convolution	3x3/1	16	8x8x16
7	max pool	2x2/2		4x4x16
8	convolution	3x3/1	16	4x4x16
9	max pool	2x2/2		2x2x16
10	convolution	3x3/1	16	2x2x16
11	dense		10	1x10

(b) MNIST classifier structure with convolutions following the Laplacian smoothing layer (layer 6).

Table 3.2: Structure of MNIST classifiers with Laplacian smoothing

Chapter 4

Geometric Scattering

4.1 Introduction

Geometric deep learning has recently emerged as an increasingly prominent branch of machine learning in general, and deep learning in particular [28]. At the core of geometric deep learning is the use of graph neural networks (GNNs) in general, and graph convolutional networks (GCNs) in particular, which ensure neuron activations follow the geometric organization of input data by propagating information across graph neighborhoods [30, 51, 106, 86, 217, 3]. However, recent work has shown the difficulty in generalizing these methods to more complex structures, identifying common problems and phrasing them in terms of oversmoothing [122], oversquashing [6] or underreaching [12].

Using graph signal processing terminology from Kipf and Welling [106], these issues can be partly attributed to the limited construction of convolutional filters in many commonly used GCN architectures. Inspired by the filters learned in convolutional neural networks, GCNs consider node features as graph signals and aim to aggregate information from neighboring nodes. For example, Kipf and Welling [106] presented a typical implementation of a GCN with a cascade of averaging (essentially low pass) filters. We note that more general variations of GCN architectures exist [51, 86, 217], which are capable of representing other filters, but as investigated in Alon and Yahav [6], they too often have difficulty in learning long-range connections.

Recently, an alternative approach was presented to provide deep geometric representation learning by generalizing Mallat's scattering transform [139], originally proposed to provide a mathematical framework for understanding convolutional neural networks, to graphs [72, 70, 228] and manifolds [163]. The geometric scattering transform can represent nodes or graphs based on the *scattering* or multi-scale diffusions, and differences between scales of diffusions of graph signals (i.e., node features). Similar to traditional scattering, which can be seen as a convolutional network with non-learned wavelet filters, geometric scattering is defined as a GNN with handcrafted graph filters, constructed as diffusion wavelets over the input graph [47], which are then cascaded with pointwise absolute-value nonlinearities. This wavelet cascade results in permutation equivariant node features that are typically aggregated via statistical moments over the graph nodes, as explained in detail in Sec. 4.3, to provide a permutation invariant graph-level representation. The efficacy of geometric scattering features in graph processing tasks was demonstrated in Gao et al. [72], with both supervised learning and data exploration applications. Moreover, their handcrafted design enables rigorous study of their properties, such as stability to deformations and perturbations, and provides a clear understanding of the information extracted by them, which by design (e.g., the cascaded band-pass filters) goes beyond low frequencies to consider richer notions of regularity [71, 164].

However, while geometric scattering transforms provide effective universal feature extractors, their handcrafted design does not allow for the automatic task-driven representation learning that is so successful in traditional GNNs and neural networks more generally. Here we focus on bridging the two frameworks, by incorporating the richer, multi-scale and multi-frequency band features of geometric scattering into neural networks while allowing them to be flexible and trainable. In essence, we introduce a geometric scattering module which can be used within a larger neural network, while still allowing for fully connected or other layers to be present in the network. Our module is called a *learnable geometric scattering (LEGS) module* and has properties inherited from the scattering transform while allowing the scales of the diffusion, and the laziness of the diffusion to be learned. Moreover, we show that our framework is differentiable, allowing for backpropogation through it in a standard reverse mode auto differentiation library.

The benefits of our construction over standard GNNs, as well as pure geometric scattering, are discussed and demonstrated on graph classification and regression tasks in Sec. 4.6. In particular, we find that our network maintains the robustness to small training sets present in geometric scattering while improving classification on biological graph classification and regression tasks, in particular, in tasks where the graphs have a large diameter relative to their size, learnable scattering features improve performance over competing methods. We show that our construction performs better on tasks that require whole-graph representations with an emphasis on biochemical molecular graphs, where relatively large diameters and non-planar structures usually limit the effectiveness of traditional GNNs. We also show that our network maintains performance in social network and other node classification tasks where state-of-the-art GNNs perform well.

4.2 Related Work

We note that efforts to improve the capture of long-range connections in graph representation learning have recently yielded several spectral approaches based on using the Lancoz algorithm to approximate graph spectra [127], or based on learning in block Krylov subspaces [135]. Such methods are complementary to the work presented here, in that their spectral approximation can also be applied in the computation of geometric scattering when considering very long range scales (e.g., via spectral formulation of graph wavelet filters). However, we find that such approximations are not necessary in the datasets considered here and in other recent work focusing on whole-graph tasks, where direct computation of polynomials of the Laplacian is sufficient. Furthermore, recent attempts have also considered ensemble approaches with hybrid architectures that combine GCN and scattering channels [143], albeit particularly focused on node-level tasks, considered on a single graph at a time, rather than whole-graph tasks considered here on datasets comparing multiple graphs. Such ensemble approaches are also complimentary to the proposed approach in that hybrid architectures can also be applied in conjunction with the proposed LEGS module here. We discuss and demonstrate this in Sec. 4.E of the Appendix.

4.3 Preliminaries: Geometric Scattering

Let $\mathcal{G} = (V, E, w)$ be a weighted graph with $V := \{v_1, \ldots, v_n\}$ the set of nodes, $E \subset \{\{v_i, v_j\} \in V \times V, i \neq j\}$ the set of (undirected) edges and $w : E \to (0, \infty)$ assigning (positive) edge weights to the graph edges. Note that w can equivalently be considered as a function of $V \times V$, where we set the weights of non-adjacent node pairs to zero. We define a *graph signal* as a function $x : V \to \mathbb{R}$ on the nodes of $\mathcal{G}$ and aggregate them in a signal vector $\boldsymbol{x} \in \mathbb{R}^n$ with the i^{th} entry being $x[v_i]$.

We define the *weighted adjacency matrix* $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ of the graph $\mathcal{G}$ as

$$\boldsymbol{W}[v_i, v_j] := \begin{cases} w(v_i, v_j) & \text{if } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise,} \end{cases}$$

and the *degree matrix* $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ of $\mathcal{G}$ as $\boldsymbol{D} := \operatorname{diag}(d_1, \ldots, d_n)$ with $d_i := \deg(v_i) := \sum_{j=1}^{n} \boldsymbol{W}[v_i, v_j]$ being the *degree* of the node v_i.

The geometric scattering transform [72] relies on a cascade of graph filters constructed from a left stochastic diffusion matrix $\boldsymbol{P} := \frac{1}{2}\big(\boldsymbol{I}_n + \boldsymbol{W}\boldsymbol{D}^{-1}\big)$, which corresponds to transition probabilities of a lazy random walk Markov process. The laziness of the process signifies that at each step it has equal probability of either staying at the current node or transitioning to a neighbor, where transition probabilities in the latter case are determined by (normalized) edge weights. Scattering filters are then defined via the graph-wavelet matrices $\boldsymbol{\Psi}_j \in \mathbb{R}^{n \times n}$ of order $j \in \mathbb{N}_0$, as

$$\begin{aligned} \boldsymbol{\Psi}_0 &:= \boldsymbol{I}_n - \boldsymbol{P}, \\ \boldsymbol{\Psi}_j &:= \boldsymbol{P}^{2^{j-1}} - \boldsymbol{P}^{2^j} = \boldsymbol{P}^{2^{j-1}}\big(\boldsymbol{I}_n - \boldsymbol{P}^{2^{j-1}}\big), \quad j \geq 1. \end{aligned} \tag{4.1}$$

These diffusion wavelet operators partition the frequency spectrum into dyadic frequency bands, which are then organized into a full wavelet filter bank $\mathcal{W}_J := \{\boldsymbol{\Psi}_j, \boldsymbol{\Phi}_J\}_{0 \leq j \leq J}$, where $\boldsymbol{\Phi}_J := \boldsymbol{P}^{2^J}$ is a pure low-pass filter, similar to the one used in GCNs. It is easy to verify that the resulting wavelet transform is invertible, since a simple sum of filter matrices in $\mathcal{W}_J$ yields the identity. Moreover, as discussed in Perlmutter et al. [164], this filter bank forms a nonexpansive frame, which provides energy preservation guarantees as well as stability to perturbations, and can be generalized to a wider family of constructions that encompasses the variations of scattering transforms on graphs from Gama et al. [70, 71] and Zou and Lerman [228].

Given the wavelet filter bank $\mathcal{W}_J$, node-level scattering features are computed by stacking cascades of bandpass filters and element-wise absolute value nonlinearities to form

$$\boldsymbol{U}_p \boldsymbol{x} \coloneqq \boldsymbol{\Psi}_{j_m} |\boldsymbol{\Psi}_{j_{m-1}} \dots |\boldsymbol{\Psi}_{j_2} |\boldsymbol{\Psi}_{j_1} \boldsymbol{x}|| \dots |, \tag{4.2}$$

indexed (or parametrized) by the scattering path $p \coloneqq (j_1, \dots, j_m) \in \cup_{m \in \mathbb{N}} \mathbb{N}_0^m$ that determines the filter scales captured by each scattering coefficient. Then, a whole-graph scattering representation is obtained by aggregating together node-level features via statistical moments over the nodes of the graph [72]. This construction yields the geometric scattering features

$$\boldsymbol{S}_{p,q} \boldsymbol{x} \coloneqq \sum_{i=1}^{n} |\boldsymbol{U}_p \boldsymbol{x}[v_i]|^q, \tag{4.3}$$

indexed by the scattering path p and moment order q. Finally, we note that it can be shown that the graph-level scattering transform $\boldsymbol{S}_{p,q}$ guarantees node-permutation invariance, while $\boldsymbol{U}_p$ is permutation equivariant [164, 72].

4.4 Adaptive Geometric Scattering Relaxation

The geometric scattering construction, described in Sec. 4.3, can be seen as a particular GNN architecture with handcrafted layers, rather than learned ones. This provides a solid mathematical framework for understanding the encoding of geometric information in GNNs, as shown in Perlmutter et al. [164], while also providing effective unsupervised graph representation learning for data exploration, which also has some advantages even in supervised learning task, as shown in Gao et al. [72]. While the handcrafted design in Perlmutter et al. [164] and Gao et al. [72] is not a priori amenable to task-driven tuning provided by end-to-end GNN training, we note that the cascade in Eq. 4.3 does conform to a neural network architecture suitable for backpropagation. Therefore, in this section, we show how and under what conditions a relaxation of the laziness of the random walk and the selection of the scales preserves some of the useful mathematical properties established in Perlmutter et al. [164]. We then establish in Sec. 4.6 the empirical benefits of learning the diffusion scales over a purely handcrafted design.

We first note that the construction of the diffusion matrix $\boldsymbol{P}$ that forms the lowpass filter used in the fixed scattering construction can be relaxed to encode adaptive laziness by setting $\boldsymbol{P}_\alpha \coloneqq \alpha \boldsymbol{I}_n + (1-\alpha)\boldsymbol{W}\boldsymbol{D}^{-1}$. Where $\alpha \in [1/2, 1)$ controls the reluctance of the random walk to transition from one node to another. Setting $\alpha = 1/2$ gives an equal probability to stay in the same node as to transition to one of its neighbors. At this point, we note that one difference between the diffusion lowpass filter here and the one typically used in GCN and its variation is the symmetrization applied in Kipf and Welling [106]. However, Perlmutter et al. [164] established that for the original construction, this is only a technical difference since $\boldsymbol{P}$ can be regarded as self-adjoint under an appropriate measure which encodes degree variations in the graph. This is then used to generate a Hilbert space $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ of graph signals with inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{D}^{-1/2}} \coloneqq \langle \boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{y} \rangle$. The following lemma shows that a similar property is retained for our adaptive lowpass filter $\boldsymbol{P}_\alpha$.

Lemma 4.4.1. *The matrix $\boldsymbol{P}_\alpha$ is self-adjoint on the Hilbert space $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ from Perlmutter et al. [164].*

We note that the self-adjointness shown here is interesting, as it links models that use symmetric and asymmetric versions of the Laplacian or adjacency matrix. Namely, Lemma 4.4.1 shows that the diffusion matrix $\boldsymbol{P}$ (which is column normalized but not row normalized) is self-adjoint, as an operator, and can thus be considered as "symmetric" in a suitable inner product space, thus establishing a theoretical link between these design choices.

As a second relaxation, we propose to replace the handcrafted dyadic scales in Eq. 4.1 with an adaptive monotonic sequence of integer diffusion time scales $0 < t_1 < \cdots < t_J$, which can be selected or tuned via training. Then, an adaptive filter bank is constructed as $\mathcal{W}'_J \coloneqq \{\boldsymbol{\Psi}'_j, \boldsymbol{\Phi}'_J\}_{j=0}^{J-1}$, with

$$\begin{aligned}
&\boldsymbol{\Phi}'_J \coloneqq \boldsymbol{P}_\alpha^{t_J}, \\
&\boldsymbol{\Psi}'_0 \coloneqq \boldsymbol{I}_n - \boldsymbol{P}_\alpha^{t_1}, \\
&\boldsymbol{\Psi}'_j \coloneqq \boldsymbol{P}_\alpha^{t_j} - \boldsymbol{P}_\alpha^{t_{j+1}}, \quad 1 \leq j \leq J-1.
\end{aligned} \tag{4.4}$$

The following theorem shows that for any selection of scales, the relaxed construction of $\mathcal{W}'_J$ yields a nonexpansive frame, similar to the result from Perlmutter et al. [164] shown for the original handcrafted construction.

Theorem 4.4.2. *There exist a constant $C > 0$ that only depends on t_1 and t_J such that for all $\boldsymbol{x} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$,*

$$C\|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} \leqslant \|\boldsymbol{\Phi}'_J\boldsymbol{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} + \sum_{j=0}^{J} \|\boldsymbol{\Psi}'_j\boldsymbol{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}} \leqslant \|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-\frac{1}{2}}},$$

where the norm considered here is the one induced by the space $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$.

Intuitively, the upper (i.e., nonexpansive) frame bound implies stability in the sense that small perturbations in the input graph signal will only result in small perturbations in the representation extracted by the constructed filter bank. Further, the lower frame bound ensures certain energy preservation by the constructed filter bank, thus indicating the nonexpansiveness is not implemented in a trivial fashion (e.g., by constant features independent of input signal).

In the next section we leverage the two relaxations described here to design a neural network architecture for learning the configuration $\alpha, t_1, \ldots, t_J$ of this relaxed construction via backpropagation through the resulting scattering filter cascade. The following theorem establishes that for any such configuration, extracted from $\mathcal{W}'_J$ via Eqs. 4.2-4.3, is permutation equivariant at the node-level and permutation invariant at the graph level. This guarantees that the extracted (in this case learned) features indeed encode intrinsic graph geometry rather than a priori indexation.

Theorem 4.4.3. *Let $\boldsymbol{U}'_p$ and $\boldsymbol{S}'_{p,q}$ be defined as in Eq. 4.2 and 4.3 (correspondingly), with* *the filters from* $\mathcal{W}'_J$ *with an arbitrary configuration* $0 < \alpha < 1$, $0 < t_1 < \cdots < t_J$. *Then, for any permutation* Π *over the nodes of* $\mathcal{G}$, *and any graph signal* $\boldsymbol{x} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ *we have* $\boldsymbol{U}'_p \Pi \boldsymbol{x} = \Pi \boldsymbol{U}'_p \boldsymbol{x}$ *and* $\boldsymbol{S}'_{p,q} \Pi \boldsymbol{x} = \boldsymbol{S}'_{p,q} \boldsymbol{x}$, *for* $p \in \cup_{m \in \mathbb{N}} \mathbb{N}_0^m, q \in \mathbb{N}$, *where geometric scattering implicitly considers here the node ordering supporting its input signal.*

We note that the results in Lemma 4.4.1 and Theorems 4.4.2-4.4.3, as well as their proofs, closely follow the theoretical framework proposed by Perlmutter et al. [164]. We carefully account here for the relaxed learned configuration, which replaces the originally handcrafted configuration there. For completeness, the adjusted proofs appear in Sec. 4.A of the Appendix.

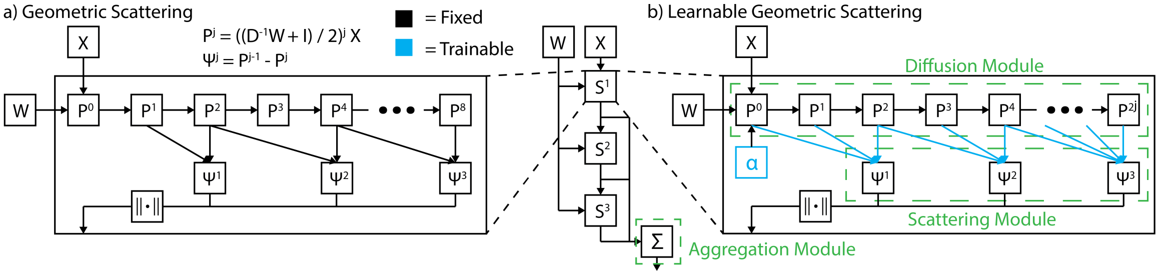


Figure 4.1: LEGS module learns to select the appropriate scattering scales from the data.

4.5 A Learnable Geometric Scattering Module

The adaptive geometric scattering construction presented in Sec. 4.4 is implemented here in a data-driven way via a backpropagation-trainable module. Throughout this section, we consider an input graph signal $\boldsymbol{x} \in \mathbb{R}^n$ or, equivalently, a collection of graph signals $\boldsymbol{X} \in \mathbb{R}^{n \times N_{\ell-1}}$. The forward propagation of these signals can be divided into three major submodules. First, a *diffusion submodule* implements the Markov process that forms the basis of the filter bank and transform, while allowing learning of the laziness parameter α. Then, a *scattering submodule* implements the filters and the corresponding cascade, while allowing the learning of the scales $t_1, \ldots, t_J$. Finally, the *aggregation module* collects the extracted features to provide a graph and produces the task-dependent output.

The diffusion submodule. We build a set of $m \in \mathbb{N}$ subsequent diffusion steps of the signal $\boldsymbol{x}$ by iteratively multiplying the diffusion matrix $\boldsymbol{P}_\alpha$ to the left of the signal, resulting in

$$\left[\boldsymbol{P}_\alpha \boldsymbol{x}, \boldsymbol{P}_\alpha^2 \boldsymbol{x}, \boldsymbol{P}_\alpha^3 \boldsymbol{x}, \ldots, \boldsymbol{P}_\alpha^m \boldsymbol{x}\right],$$

Since $\boldsymbol{P}_\alpha$ is often sparse, for efficiency reasons these filter responses are implemented via an RNN structure consisting of m RNN modules. Each module propagates the incoming hidden state $\boldsymbol{h}_{t-1}, t = 1, \ldots, m$ with $\boldsymbol{P}_\alpha$ with the readout $\boldsymbol{o}_t$ equal to the produced hidden state,

$$\boldsymbol{h}_t \coloneqq \boldsymbol{P}_\alpha \boldsymbol{h}_{t-1}, \quad \boldsymbol{o}_t \coloneqq \boldsymbol{h}_t.$$

Our architecture and theory enable the implementation of either trainable or nontrainable α, which we believe will be useful for future work as indicated, for example, in Gao and Ji [74].

The scattering submodule. Next, we consider the selection of $J \leq m$ diffusion scales for the flexible filter bank construction with wavelets defined according to Eq. 4.5. We found this was the most influential part of the architecture. We experimented with methods of increasing flexibility:

1. Selection of $\{t_j\}_{j=1}^{J-1}$ as dyadic scales (as in Sec. 4.3 and Eq. 4.1), fixed for all datasets (LEGS-FIXED),
2. Selection of each t_j using softmax and sorting by j, learnable per model (LEGS-FCN and LEGS-RBF, depending on output layer explained below).

For the scale selection, we use a selection matrix $\boldsymbol{F} \in \mathbb{R}^{J \times m}$, where each row $\boldsymbol{F}_{(j,\cdot)}, j = 1, \ldots, J$ is dedicated to identifying the diffusion scale of the wavelet $\boldsymbol{P}_\alpha^{t_j}$ via a one-hot encoding. This is achieved by setting

$$\boldsymbol{F} \coloneqq \sigma(\boldsymbol{\Theta}) = [\sigma(\boldsymbol{\theta}_1), \sigma(\boldsymbol{\theta}_2), \ldots, \sigma(\boldsymbol{\theta}_J)]^T,$$

where $\boldsymbol{\theta}_j \in \mathbb{R}^m$ constitute the rows of the trainable weight matrix $\boldsymbol{\Theta}$, and σ is the softmax function. While this construction may not strictly guarantee an exact one-hot encoding, we assume that the softmax activations yield a sufficient approximation. Further, without loss of generality, we assume that the rows of $\boldsymbol{F}$ are ordered according to the position of the leading "one" activated in every row. In practice, this can be easily enforced by reordering the rows. We now construct the filter bank $\widetilde{\mathcal{W}}_{\boldsymbol{F}} \coloneqq \{\widetilde{\boldsymbol{\Psi}}_j, \widetilde{\boldsymbol{\Phi}}_J\}_{j=0}^{J-1}$ with the filters

$$\begin{aligned}
\widetilde{\boldsymbol{\Phi}}_J\boldsymbol{x} &= \sum_{t=1}^{m} \boldsymbol{F}_{(J,t)}\boldsymbol{P}_\alpha^t\boldsymbol{x}, \\
\widetilde{\boldsymbol{\Psi}}_0\boldsymbol{x} &= \boldsymbol{x} - \sum_{t=1}^{m} \boldsymbol{F}_{(1,t)}\boldsymbol{P}_\alpha^t\boldsymbol{x}, \qquad\qquad (4.5)\\
\widetilde{\boldsymbol{\Psi}}_j\boldsymbol{x} &= \sum_{t=1}^{m} \left[\boldsymbol{F}_{(j,t)}\boldsymbol{P}_\alpha^t\boldsymbol{x} - \boldsymbol{F}_{(j+1,t)}\boldsymbol{P}_\alpha^t\boldsymbol{x}\right], \quad 1 \leq j \leq J-1,
\end{aligned}$$

matching and implementing the construction of $\mathcal{W}'_J$ (Eq. 4.4).

The aggregation submodule. While many approaches may be applied to aggregate node-level features into graph-level features such as max, mean, sum pooling, and the more powerful TopK [74] or attention pooling [205], we follow the statistical-moment aggregation explained in Secs. 4.3-4.4 motivated by Gao et al. [72], Perlmutter et al. [164] and leave exploration of other pooling methods to future work.

Incorporating LEGS into a larger neural network As shown in Gao et al. [72] on graph classification, this aggregation works particularly well in conjunction with support vector machines (SVMs) based on the radial basis function (RBF) kernel. Here, we consider two configurations for the task-dependent output layer of the network, either using two fully connected layers after the learnable scattering layers, which we denote LEGS-FCN, or using a modified RBF network [29], which we denote LEGS-RBF, to produce the final classification.

The latter configuration more accurately processes scattering features as shown in Table 4.2. Our RBF network works by first initializing a fixed number of movable anchor points. Then, for every point, new features are calculated based on the radial distances to these anchor points. In previous work on radial basis networks these anchor points were initialized independent of the data. We found that this led to training issues if the range of the data was not similar to the initialization of the centers. Instead, we first use a batch normalization layer to constrain the scale of the features and then pick anchors randomly from the initial features of the first pass through our data. This gives an RBF-kernel network with anchors that are always in the range of the data. Our RBF layer is then $\mathrm{RBF}(\boldsymbol{x}) = \phi(\|\mathrm{BatchNorm}(\boldsymbol{x}) - \boldsymbol{c}\|)$ with $\quad \phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$.

Backpropagation through the LEGS module The LEGS module is fully suitable for incorporation in any neural network architecture and can be backpropogated through. To show this we write the partial derivatives with respect to the LEGS module parameters α, $\boldsymbol{\Theta}$, and $\boldsymbol{W}$ here. The gradient of the filter $\widetilde{\boldsymbol{\Psi}}_j$ with respect to α can be written as

$$\begin{aligned}\frac{\partial \widetilde{\boldsymbol{\Psi}}_j}{\partial \alpha} = \sum_{t=1}^{m} \Bigg[& (\boldsymbol{F}_{j,t} - \boldsymbol{F}_{j+1,t}) \\ & \times \sum_{k=1}^{t} \boldsymbol{P}_\alpha^{k-1} (\boldsymbol{I}_n - \boldsymbol{W}\boldsymbol{D}^{-1}) \boldsymbol{P}_\alpha^{t-k} \Bigg]. \end{aligned} \tag{4.6}$$

Gradients with respect to scale weights θ_k, $k \in [1, \dots m]$, where σ is the softmax function, can be written as

$$\frac{\partial \widetilde{\boldsymbol{\Psi}}_j}{\partial \boldsymbol{\Theta}_{k,t}} = \begin{cases} \boldsymbol{P}_\alpha^t \sigma(\boldsymbol{\theta}_j)_t (1 - \sigma(\boldsymbol{\theta}_j))_t & \text{if } k = j, \\ \boldsymbol{P}_\alpha^t \sigma(\boldsymbol{\theta}_j)_t \sigma(\boldsymbol{\theta}_k)_t & \text{if } k = j+1, \\ \boldsymbol{0}_{n \times n} & \text{else.} \end{cases} \tag{4.7}$$

Finally, the gradient with respect to the adjacency matrix entry $\boldsymbol{W}_{ab}$, where we denote $\boldsymbol{J}^{ab}$ the matrix with $\boldsymbol{J}^{ab}_{(a,b)} = 1$ and all other entries zero, can be written as

$$\begin{aligned}\frac{\partial \widetilde{\boldsymbol{\Psi}}_j}{\partial \boldsymbol{W}_{ab}} = (1-\alpha) \sum_{t=1}^{m} \Bigg[& (\boldsymbol{F}_{j,t} - \boldsymbol{F}_{j+1,t}) \times \\ & \sum_{k=1}^{t} \boldsymbol{P}_\alpha^{k-1} (\boldsymbol{J}^{ab} \boldsymbol{D}^{-1}) \boldsymbol{P}_\alpha^{t-k} \Bigg]. \end{aligned} \tag{4.8}$$

Gradients of filters $\widetilde{\boldsymbol{\Phi}}_J$ and $\widetilde{\boldsymbol{\Psi}}_0$, which are simple modifications of the partial derivatives of $\widetilde{\boldsymbol{\Psi}}_j$ and derivations of these gradients can be found in Sec. 4.B of the Appendix.

4.6 Empirical Results

Here we investigate the LEGS module on whole graph classification and graph regression tasks, that arise in a variety of contexts, with emphasis on the more complex biochemical datasets. We focus on biochemical graph datasets as they represent a less prominent but equally important challenge in the field of graph learning as compared with social network datasets. Unlike other types of data, biochemical graphs do not exhibit the small-world structure of social graphs and may have large graph diameters for their size. Further, the connectivity patterns of biomolecules are very irregular due to 3D folding and long-range connections, and thus ordinary local node aggregation methods may miss such connectivity differences.

Table 4.1: Dataset statistics, diameter, nodes, edges, clustering coefficient (CC) averaged over all graphs. Split into bio-chemical and social network types.

	# Graphs	# Classes	Diameter	Nodes	Edges	CC
DD	1178	2	19.81	284.32	715.66	0.48
ENZYMES	600	6	10.92	32.63	62.14	0.45
MUTAG	188	2	8.22	17.93	19.79	0.00
NCI1	4110	2	13.33	29.87	32.30	0.00
NCI109	4127	2	13.14	29.68	32.13	0.00
PROTEINS	1113	2	11.62	39.06	72.82	0.51
PTC	344	2	7.52	14.29	14.69	0.01
COLLAB	5000	3	1.86	74.49	2457.22	0.89
IMDB-B	1000	2	1.86	19.77	96.53	0.95
IMDB-M	1500	3	1.47	13.00	65.94	0.97
REDDIT-B	2000	2	8.59	429.63	497.75	0.05
REDDIT-12K	11929	11	9.53	391.41	456.89	0.03
REDDIT-5K	4999	5	10.57	508.52	594.87	0.03

4.6.1 Whole Graph Classification

We perform whole graph classification by using eccentricity (max distance of a node to other nodes) and clustering coefficient (percentage of links between the neighbors of the node compared to a clique) as node features as are used in Gao et al. [72]. We compare against graph convolutional networks (GCN) [106], GraphSAGE [86], graph attention net-

Table 4.2: Mean ± std. over 10 test sets on bio-chemical (top) and social network (bottom) datasets.

	LEGS-RBF	LEGS-FCN	LEGS-FIXED	GCN	GraphSAGE	GAT	GIN	GS-SVM	Baseline
DD	72.58 ± 3.35	72.07 ± 2.37	69.09 ± 4.82	67.82 ± 3.81	66.37 ± 4.45	68.50 ± 3.62	42.37 ± 4.32	72.66 ± 4.94	**75.98 ± 2.81**
ENZYMES	36.33 ± 4.50	**38.50 ± 8.18**	32.33 ± 5.04	31.33 ± 6.89	15.83 ± 9.10	25.83 ± 4.73	36.83 ± 4.81	27.33 ± 5.10	20.50 ± 5.99
MUTAG	33.51 ± 4.34	82.98 ± 9.85	81.84 ± 11.24	79.30 ± 9.66	81.43 ± 11.64	79.85 ± 9.44	83.57 ± 9.68	**85.09 ± 7.44**	79.80 ± 9.92
NCI1	**74.26 ± 1.53**	70.83 ± 2.65	71.24 ± 1.63	60.80 ± 4.26	57.54 ± 3.33	62.19 ± 2.18	66.67 ± 2.90	69.68 ± 2.38	56.69 ± 3.07
NCI109	**72.47 ± 2.11**	70.17 ± 1.46	69.25 ± 1.75	61.30 ± 2.99	55.15 ± 2.58	61.28 ± 2.24	65.23 ± 1.82	68.55 ± 2.06	57.38 ± 2.20
PROTEINS	70.89 ± 3.91	71.06 ± 3.17	67.30 ± 2.94	74.03 ± 3.20	71.87 ± 3.50	73.22 ± 3.55	**75.02 ± 4.55**	70.98 ± 2.67	73.22 ± 3.76
PTC	**57.26 ± 5.54**	56.92 ± 9.36	54.31 ± 6.92	56.34 ± 10.29	55.22 ± 9.13	55.50 ± 6.90	55.82 ± 8.07	56.96 ± 7.09	56.71 ± 5.54
COLLAB	75.78 ± 1.95	75.40 ± 1.80	72.94 ± 1.70	73.80 ± 1.73	**76.12 ± 1.58**	72.88 ± 2.06	62.98 ± 3.92	74.54 ± 2.32	64.76 ± 2.63
IMDB-BINARY	64.90 ± 3.48	64.50 ± 3.50	64.30 ± 3.68	47.40 ± 6.24	46.40 ± 4.03	45.50 ± 3.14	64.20 ± 5.77	**66.70 ± 3.53**	47.20 ± 5.67
IMDB-MULTI	41.93 ± 3.01	40.13 ± 2.77	41.67 ± 3.19	39.33 ± 3.13	39.73 ± 3.45	39.73 ± 3.61	38.67 ± 3.93	**42.13 ± 2.53**	39.53 ± 3.63
REDDIT-BINARY	**86.10 ± 2.92**	78.15 ± 5.42	85.00 ± 1.93	81.60 ± 2.32	73.40 ± 4.38	73.35 ± 2.27	71.40 ± 6.98	85.15 ± 2.78	69.30 ± 5.08
REDDIT-MULTI-12K	38.47 ± 1.07	38.46 ± 1.31	39.74 ± 1.31	**42.57 ± 0.90**	32.17 ± 2.04	32.74 ± 0.75	24.45 ± 5.52	39.79 ± 1.11	22.07 ± 0.98
REDDIT-MULTI-5K	47.83 ± 2.61	46.97 ± 3.06	47.17 ± 2.93	**52.79 ± 2.11**	45.71 ± 2.88	44.03 ± 2.57	35.73 ± 8.35	48.79 ± 2.95	36.41 ± 1.80

work (GAT) [205], graph isomorphism network (GIN) [217], Snowball network [135], and fixed geometric scattering with a support vector machine classifier (GS-SVM) as in Gao et al. [72], and a baseline which is a 2-layer neural network on the features averaged across nodes (disregarding graph structure). These comparisons are meant to inform when including learnable graph scattering features are helpful in extracting whole graph features. Specifically, we are interested in the types of graph datasets where existing graph neural network performance can be improved upon with scattering features. We evaluate these methods across 7 biochemical datasets and 6 social network datasets where the goal is to classify between two or more classes of compounds with hundreds to thousands of graphs and tens to hundreds of nodes (See Table 4.1). For more specific information on individual datasets see Appendix 4.C. We use 10-fold cross validation on all models which is elaborated on in Appendix 4.D. For an ensemble of the LEGS module with other graph neural networks as first explored with a fixed geometric scattering transform in Scattering-GCN [143] see Appendix 4.E.

LEGS outperforms on biochemical datasets. Most work on graph neural networks has focused on social networks which have a well-studied structure. However, biochemical graphs that represent molecules and tend to be overall smaller and less connected than social networks (see Table 4.1). In particular, we find that LEGS outperforms other methods by a significant margin on biochemical datasets with relatively small but high diameter graphs (NCI1, NCI109, ENZYMES, PTC), as shown in Table 4.2. On extremely small graphs we find that GS-SVM performs best, which is expected as other methods with more parameters can easily overfit the data. We reason that the performance increase exhibited by LEGS module networks, and to a lesser extent GS-SVM, on these biochemical graphs is due the ability of geometric scattering to compute complex connectivity features via its multiscale diffusion wavelets. Thus, methods that rely on a scattering construction would in general perform better, with the flexibility and trainability of the LEGS module giving it an edge on most tasks.

LEGS performs well on social network datasets and considerably improves performance in ensemble models. In Table 4.2, we see that on the social network datasets LEGS performs well. Overlooking the fixed scattering transform GS-SVM, which was tuned in Gao et al. [72] with a focus on these particular social network datasets, a LEGS module architecture is best on three out of the six social datasets and second best on the other three. If we also consider combining LEGS module features with GCN features the LEGS module performs the best on five out of six of the social network datasets. Across all datasets, an ensemble model considerably increases accuracy over GCN (see Table 4.8 in Appendix 4.E). This underlines the capabilities of the LEGS module, not only as an isolated model, but also as a tool for powerful hybrid GNN architectures. Similar to Min et al. [143], this supports the claim that the LEGS module (due to geometric scattering) is sensitive to complementary regularity over graphs, compared to many traditional GNNs.

LEGS preserves enzyme exchange preferences while increasing performance. One advantage of geometric scattering over other graph embedding techniques lies in the rich information present within the scattering feature space. This was demonstrated in [72] where it was shown that the embeddings created through fixed geometric scattering can be used to accurately infer inter-graph relationships. Scattering features of enzyme graphs within the ENZYMES dataset [26] possessed sufficient global information to recreate the enzyme class exchange preferences observed empirically by Cuesta et al. [49], using only linear methods of analysis, and despite working with a much smaller and artificially balanced dataset. We demonstrate here that LEGS retains similar descriptive capabilities, as shown in Figure 4.2 via chord diagrams where each exchange preference between enzyme

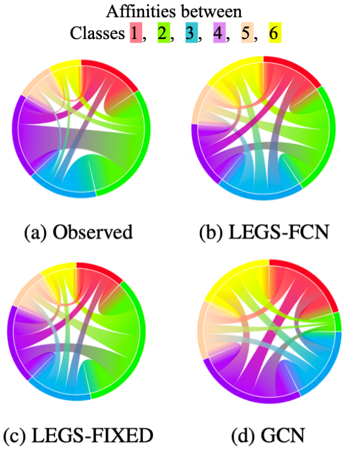


Figure 4.2: Enzyme class exchange preferences empirically observed in Cuesta et al. [49], and estimated from LEGS and GCN embeddings.

classes [estimated as suggested in 72] is represented as a ribbon of the corresponding size. Our results here (and in Table 4.7, which provides complementary comparison) show that, with relaxations on the scattering parameters, LEGS-FCN achieves better classification accuracy than both LEGS-FIXED and GCN (see Table 4.2) while also retaining a more descriptive embedding that maintains the global structure of relations between enzyme classes. We ran LEGS-FIXED and LEGS-FCN on the ENZYMES dataset. LEGS-FCN which allows the diffusion scales to be learned. For comparison, we also ran a standard GCN whose graph embeddings were obtained via mean pooling. To infer enzyme exchange preferences from their embeddings, we followed Gao et al. [72] in defining the distance from an enzyme e to the enzyme class EC_j as $\mathrm{dist}(e, \mathrm{EC}_j) := \|v_e - \mathrm{proj}_{C_j}(v_e)\|$, where v_i is the embedding of e, and C_j is the PCA subspace of the enzyme feature vectors within EC_j. The distance between the enzyme classes EC_i and EC_j is the average of the individual distances, $\mathrm{mean}\{\mathrm{dist}(e, \mathrm{EC}_j) : e \in \mathrm{EC}_i\}$. From here, the affinity between two enzyme classes is computed as $\mathrm{pref}(\mathrm{EC}_i, \mathrm{EC}_j) = w_i / \min(\frac{D_{i,i}}{D_{i,j}}, \frac{D_{j,j}}{D_{j,i}})$, where w_i is the percentage of enzymes in class i which are closer to another class than their own, and $D_{i,j}$ is the distance between EC_i and EC_j.

Robustness to reduced training set size. We remark that similar to the robustness shown in Gao et al. [72] for handcrafted scattering, LEGS-based networks are able to maintain accuracy even when the training set size is shrunk to as low as 20% of the dataset, with a median decrease of 4.7% accuracy as when 80% of the data is used for training, as discussed in the supplement (see Table 4.5).

4.6.2 Graph Regression

Table 4.3: Train and test set mean squared error on CASP GDT regression task over three seeds.

$(\mu \pm \sigma)$	Train MSE	Test MSE
LEGS-FCN	**134.34 ± 8.62**	**144.14 ± 15.48**
LEGS-RBF	140.46 ± 9.76	152.59 ± 14.56
LEGS-FIXED	136.84 ± 15.57	160.03 ± 1.81
GCN	289.33 ± 15.75	303.52 ± 18.90
GraphSAGE	221.14 ± 42.56	219.44 ± 34.84
GIN	221.14 ± 42.56	219.44 ± 34.84
Baseline	393.78 ± 4.02	402.21 ± 21.45

We next evaluate learnable scattering on two graph regression tasks, the QM9 [78, 215] graph regression dataset, and a new task from the critical assessment of structure prediction (CASP) challenge [149]. In the CASP task, the main objective is to score protein structure prediction/simulation models in terms of the discrepancy between their predicted structure and the actual structure of the protein (which is known a priori). The accuracy of such 3D structure predictions are evaluated using a variety of metrics, but we focus on the global distance test (GDT) score [144]. The GDT score measures the similarity between tertiary structures of two proteins with amino-acid correspondence. A higher score means two structures are more similar. For a set of predicted 3D structures for a protein, we would like to score their quality as quantified by the GDT score.

For this task we use the CASP12 dataset [149] and preprocess the data similarly to Ingraham et al. [94], creating a KNN graph between proteins based on the 3D coordinates of each amino acid. From this KNN graph we regress against the GDT score. We evaluate on 12 proteins from the CASP12 dataset and choose random (but consistent) splits with 80% train, 10% validation, and 10% test data out of 4000 total structures. We are interested in structure similarity and use no non-structural node features.

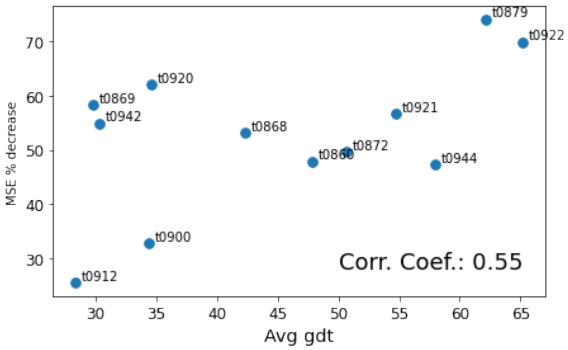


Figure 4.3: CASP dataset LEGS-FCN % improvement over GCN in MSE of GDT prediction vs. Average GDT score.

Table 4.4: Mean ± std. over four runs of mean squared error over 19 targets for the QM9 dataset, lower is better.

$(\mu \pm \sigma)$	Test MSE
LEGS-FCN	**0.216 ± 0.009**
LEGS-FIXED	0.228 ± 0.019
GraphSAGE	0.524 ± 0.224
GCN	0.417 ± 0.061
GIN	0.247 ± 0.037
Baseline	0.533 ± 0.041

LEGS outperforms on all CASP targets Across all CASP targets we find that LEGS-based architectures significantly outperforms GNN and baseline methods (See Table 4.6). This performance improvement is particularly stark on the easiest structures (measured by average GDT) but is consistent across all structures. In Figure 4.3 we show the relationship between percent improvement of LEGS over the GCN model and the average GDT score across the target structures. We draw attention to target t0879, where LEGS shows the greatest improvement over other methods. This target has long-range dependencies [154] as it exhibits metal coupling [123] creating long-range connections over the sequence. Since other methods are unable to model these long-range connections LEGS is particularly important on these more difficult to model targets.

LEGS outperforms on the QM9 dataset We evaluate the performance of LEGS-based networks on the quantum chemistry dataset QM9 [78, 215], which consists of 130,000 molecules with ~18 nodes per molecule. We use the node features from Gilmer et al. [78], with the addition of eccentricity and clustering coefficient features, and ignore the edge features. We whiten all targets to have zero mean and unit standard deviation. We train each network against all 19 targets and evaluate the mean squared error on the test set with mean and std. over four runs. We find that learning the scales improves the overall MSE, and particularly improves the results over difficult targets (see Table 4.4 for overall results and Table 4.9 for results by target). Indeed, on more difficult targets (i.e., those with large test error) LEGS-FCN is able to perform better, where on easy targets GIN is the best. Overall, scattering features offer a robust signal over many targets, and while perhaps less flexible (by construction), they achieve good average performance with significantly fewer parameters.

4.7 Conclusion

In this work we introduced a flexible geometric scattering module, that serves as an alternative to standard graph neural network architectures and is capable of learning rich multi-scale features. Our learnable geometric scattering module allows a task-dependent network to choose the appropriate scales of the multiscale graph diffusion wavelets that are part of the geometric scattering transform. We show that incorporation of this module yields improved performance on graph classification and regression tasks, particularly on biochemical datasets, while keeping strong guarantees on extracted features. This also opens the possibility to provide additional flexibility to the module to enable node-specific or graph-specific tuning via attention mechanisms, which are an exciting future direction, but out of scope for the current work.

Appendix

4.A Proofs for Section 4.4

4.A.1 Proof of Lemma 4.4.1

Let $\boldsymbol{M}_\alpha = \boldsymbol{D}^{-1/2}\boldsymbol{P}_\alpha\boldsymbol{D}^{1/2}$ then it can be verified that $\boldsymbol{M}_\alpha$ is a symmetric conjugate of $\boldsymbol{P}_\alpha$, and by construction is self-adjoint with respect to the standard inner product of $L^2(\mathcal{G})$. Let $\boldsymbol{x}, \boldsymbol{y} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ then we have

$$\begin{aligned}
\langle \boldsymbol{P}_\alpha\boldsymbol{x}, \boldsymbol{y}\rangle_{\boldsymbol{D}^{-1/2}} &= \langle \boldsymbol{D}^{-1/2}\boldsymbol{P}_\alpha\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{D}^{-1/2}\boldsymbol{D}^{1/2}\boldsymbol{M}_\alpha\boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{M}_\alpha\boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{M}_\alpha\boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{D}^{1/2}\boldsymbol{M}_\alpha\boldsymbol{D}^{-1/2}\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{D}^{-1/2}\boldsymbol{x}, \boldsymbol{D}^{-1/2}\boldsymbol{P}_\alpha\boldsymbol{y}\rangle \\
&= \langle \boldsymbol{x}, \boldsymbol{P}_\alpha\boldsymbol{y}\rangle_{\boldsymbol{D}^{-1/2}},
\end{aligned}$$

which gives the result of the lemma. □

4.A.2 Proof of Theorem 4.4.2

As shown in the previous proof (Sec. 4.A.1), $\boldsymbol{P}_\alpha$ has a symmetric conjugate $\boldsymbol{M}_\alpha$. Given the eigendecomposition $\boldsymbol{M}_\alpha = Q\Lambda Q^T$, we can write $\boldsymbol{P}_\alpha^t = \boldsymbol{D}^{1/2}Q\Lambda^t Q^T\boldsymbol{D}^{-1/2}$, giving the eigendecomposition of the propagated diffusion matrices. Furthermore, it can be verified that the eigenvalues on the diagonal of Λ are nonnegative. Briefly, this results from graph

Laplacian eigenvalues being within the range $[0,1]$, which means those of $\boldsymbol{W}\boldsymbol{D}^{-1}$ are in $[-1,1]$, which combined with $1/2 \leq \alpha \leq 1$ result in $\lambda_i := [\Lambda]_{ii} \in [0,1]$ for every j. Next, given this decomposition we can write

$$\begin{aligned}
&\boldsymbol{\Phi'}_J = \boldsymbol{D}^{1/2} Q \Lambda^{t_J} Q^T \boldsymbol{D}^{-1/2}, \\
&\boldsymbol{\Psi'}_j = \boldsymbol{D}^{1/2} Q (\Lambda^{t_j} - \Lambda^{t_{j+1}}) Q^T \boldsymbol{D}^{-1/2}, \quad 0 \leq j \leq J-1,
\end{aligned}$$

where we set $t_0 = 0$ to simplify notations. Then, we have

$$\begin{aligned}
\|\boldsymbol{\Phi'}_J \boldsymbol{x}\|^2_{D^{-1/2}} &= \langle \boldsymbol{\Phi'}_J \boldsymbol{x}, \boldsymbol{\Phi'}_J \boldsymbol{x} \rangle_{D^{-1/2}} \\
&= \langle D^{-1/2} \boldsymbol{D}^{1/2} Q \Lambda^{t_J} Q^T \boldsymbol{D}^{-1/2} \boldsymbol{x}, D^{-1/2} \boldsymbol{D}^{1/2} Q \Lambda^{t_J} Q^T \boldsymbol{D}^{-1/2} \boldsymbol{x} \rangle \\
&= \boldsymbol{x}^T \boldsymbol{D}^{-1/2} Q \Lambda^{t_J} Q^T Q \Lambda^{t_J} Q^T \boldsymbol{D}^{-1/2} \boldsymbol{x} = (\boldsymbol{x}^T \boldsymbol{D}^{-1/2} Q \Lambda^{t_J})(\Lambda^{t_J} Q^T \boldsymbol{D}^{-1/2} \boldsymbol{x}) \\
&= \|\Lambda^{t_J} Q^T \boldsymbol{D}^{-1/2} \boldsymbol{x}\|^2_2.
\end{aligned}$$

Further, since Q is orthogonal (as it is constructed from an eigenbasis of a symmetric matrix), if we consider a change of variable to $\boldsymbol{y} = Q^T \boldsymbol{D}^{-1/2} \boldsymbol{x}$, we have $\|\boldsymbol{x}\|^2_{\boldsymbol{D}^{-1/2}} = \|\boldsymbol{D}^{-1/2} \boldsymbol{x}\|^2_2 = \|\boldsymbol{y}\|^2_2$ while $\|\boldsymbol{\Phi'}_J \boldsymbol{x}\|^2_{D^{-1/2}} = \|\Lambda^{t_J} \boldsymbol{y}\|^2_2$. Similarly, we can also reformulate the operation of other filters in terms of diagonal matrices applied to $\boldsymbol{y}$ as $\mathcal{W}'_J$ as $\|\boldsymbol{\Psi'}_j \boldsymbol{x}\|^2_{D^{-1/2}} = \|(\Lambda^{t_j} - \Lambda^{t_{j+1}}) \boldsymbol{y}\|^2_2$.

Given the reformulation in terms of $\boldsymbol{y}$ and standard $L^2(\mathcal{G})$, we can now write

$$\|\Lambda^{t_J} \boldsymbol{y}\|^2_2 + \sum_{j=0}^{J-1} \|(\Lambda^{t_j} - \Lambda^{t_{j+1}}) \boldsymbol{y}\|^2_2 = \sum_{i=1}^{n} \boldsymbol{y}_i^2 \cdot \left(\lambda^{2t_J} + \textstyle\sum_{j=0}^{J-1} (\lambda_i^{t_j} - \lambda_i^{t_{j+1}})^2 \right).$$

Then, since $0 \leq \lambda_i \leq 1$ and $0 = t_0 < t_1 < \cdots < t_J$ we have

$$\lambda^{2t_J} + \sum_{j=0}^{J-1} (\lambda_i^{t_j} - \lambda_i^{t_{j+1}})^2 \leq \left(\lambda^{t_J} + \sum_{j=0}^{J-1} \lambda_i^{t_j} - \lambda_i^{t_{j+1}} \right)^2 = \left(\lambda^{t_J} + \lambda_i^{t_0} - \lambda_i^{t_J} \right)^2 = 1,$$

which yields the upper bound $\|\Lambda^{t_J} \boldsymbol{y}\|^2_2 + \sum_{j=0}^{J-1} \|(\Lambda^{t_j} - \Lambda^{t_{j+1}}) \boldsymbol{y}\|^2_2 \leq \|\boldsymbol{y}\|^2_2$. On the other hand, since $t_1 > 0 = t_0$, then we also have

$$\lambda^{2t_J} + \sum_{j=0}^{J-1} (\lambda_i^{t_j} - \lambda_i^{t_{j+1}})^2 \geq \lambda^{2t_J} + (1 - \lambda_i^{t_1})^2,$$

and therefore, by setting $C \coloneqq \min_{0 \leq \xi \leq 1}(\xi^{2t_J} + (1 - \xi^{t_1})^2) > 0$, whose positivity is not difficult to verify, we get the lower bound $\|\Lambda^{t_J}\boldsymbol{y}\|_2^2 + \sum_{j=0}^{J-1} \|(\Lambda^{t_j} - \Lambda^{t_{j+1}})\boldsymbol{y}\|_2^2 \geq C\|\boldsymbol{y}\|_2^2$. Finally, applying the reverse change of variable to $\boldsymbol{x}$ and $L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ yields the result of the theorem. □

4.A.3 Proof of Theorem 4.4.3

Denote the permutation group on n elements as S_n, then for a permutation $\Pi \in S_n$ we let $\overline{\mathcal{G}} = \Pi(\mathcal{G})$ be the graph obtained by permuting the vertices of $\mathcal{G}$ with Π. The corresponding permutation operation on a graph signal $\boldsymbol{x} \in L^2(\mathcal{G}, \boldsymbol{D}^{-1/2})$ gives a signal $\Pi\boldsymbol{x} \in L^2(\overline{\mathcal{G}}, \boldsymbol{D}^{-1/2})$, which we implicitly considered in the statement of the theorem, without specifying these notations for simplicity. Rewriting the statement of the theorem more rigorously with the introduced notations, we aim to show that $\overline{\boldsymbol{U}}'_p\Pi\boldsymbol{x} = \Pi\boldsymbol{U}'_p\boldsymbol{x}$ and $\overline{\boldsymbol{S}}'_{p,q}\Pi\boldsymbol{x} = \boldsymbol{S}'_{p,q}\boldsymbol{x}$ under suitable conditions, where the operation $\boldsymbol{U}'_p$ from $\mathcal{G}$ on the permuted graph $\overline{\mathcal{G}}$ is denoted here by $\overline{\boldsymbol{U}}'_p$ and likewise for $\boldsymbol{S}'_{p,q}$ we have $\overline{\boldsymbol{S}}'_{p,q}$.

We start by showing $\boldsymbol{U}'_p$ is permutation equivariant. First, we notice that for any Ψ_j, $0 < j < J$ we have that $\overline{\Psi}_j\Pi\boldsymbol{x} = \Pi\Psi_j\boldsymbol{x}$, as for $1 \leq j \leq J-1$,

$$\begin{aligned}
\overline{\Psi}_j\Pi\boldsymbol{x} &= (\Pi\boldsymbol{P}^{t_j}\Pi^T - \Pi\boldsymbol{P}^{t_{j+1}}\Pi^T)\Pi\boldsymbol{x} \\
&= \Pi(\boldsymbol{P}^{t_j} - \boldsymbol{P}^{t_{j+1}})\boldsymbol{x} \\
&= \Pi\Psi_j\boldsymbol{x}.
\end{aligned}$$

Similar reasoning also holds for $j \in \{0, J\}$. Further, notice that for the element-wise nature of the absolute value nonlinearity yields $|\Pi\boldsymbol{x}| = \Pi|\boldsymbol{x}|$ for any permutation matrix Π. Using these two observations, it follows inductively that

$$\begin{aligned}
\overline{\boldsymbol{U}}'_p \Pi \boldsymbol{x} :=& \boldsymbol{\Psi}'_{j_m} |\boldsymbol{\Psi}'_{j_{m-1}} \dots |\boldsymbol{\Psi}'_{j_2} |\boldsymbol{\Psi}'_{j_1} \Pi \boldsymbol{x}|| \dots | \\
=& \boldsymbol{\Psi}'_{j_m} |\boldsymbol{\Psi}'_{j_{m-1}} \dots |\boldsymbol{\Psi}'_{j_2} \Pi |\boldsymbol{\Psi}'_{j_1} \boldsymbol{x}|| \dots | \\
& \vdots \\
=& \Pi \boldsymbol{\Psi}'_{j_m} |\boldsymbol{\Psi}'_{j_{m-1}} \dots |\boldsymbol{\Psi}'_{j_2} |\boldsymbol{\Psi}'_{j_1} \boldsymbol{x}|| \dots | \\
=& \Pi \boldsymbol{U}'_p \boldsymbol{x}.
\end{aligned}$$

To show $\boldsymbol{S}'_{p,q}$ is permutation invariant, first notice that for any statistical moment $q > 0$, we have $|\Pi \boldsymbol{x}|^q = \Pi |\boldsymbol{x}|^q$ and further as sums are commutative, $\sum_j (\Pi \boldsymbol{x})_j = \sum_j \boldsymbol{x}_j$. We then have

$$\overline{\boldsymbol{S}}'_{p,q} \Pi \boldsymbol{x} = \sum_{i=1}^{n} |\overline{\boldsymbol{U}}'_p \Pi \boldsymbol{x}[v_i]|^q = \sum_{i=1}^{n} |\Pi \boldsymbol{U}'_p \boldsymbol{x}[v_i]|^q = \sum_{i=1}^{n} |\boldsymbol{U}'_p \boldsymbol{x}[v_i]|^q = \boldsymbol{S}'_{p,q} \boldsymbol{x},$$

which, together with the previous result, completes the proof of the theorem. □

4.B Gradients of the LEGS module

Here we analyze the gradients of the LEGS module with respect to its outputs. As depicted in Fig. 4.1, the LEGS module has 3 inputs, W, Θ, and α, and J permutation equivariant output matrices $\widetilde{W}_{\boldsymbol{F}} = \{\widetilde{\boldsymbol{\Psi}}_j, \widetilde{\boldsymbol{\Phi}}_J\}_{j=0}^{J-1}$. Here we compute partial derivatives of $\widetilde{\boldsymbol{\Psi}}_j$ for $1 \leq j \leq J-1$. The other related gradients of $\widetilde{\boldsymbol{\Psi}}_0$ and $\widetilde{\boldsymbol{\Phi}}_J$ are easily deducible from these. The following partial derivatives and an application of the train rule yield the gradients presented in the main paper:

$$\frac{\partial \widetilde{\boldsymbol{\Psi}}_j}{\partial \boldsymbol{F}_{k,t}} = \begin{cases} \boldsymbol{P}_\alpha^t & \text{if } k = j, \\ -\boldsymbol{P}_\alpha^t & \text{if } k = j+1, \\ \mathbf{0}_{n \times n} & \text{else,} \end{cases} \tag{4.9}$$

$$\frac{\partial \widetilde{\boldsymbol{\Psi}}_j}{\partial \boldsymbol{P}_\alpha^t} = (F_{j,t} - F_{j+1,t}), \tag{4.10}$$

$$\frac{\partial \boldsymbol{F}_{k,t}}{\boldsymbol{\Theta}_{k,t}} = \begin{cases} \sigma(\boldsymbol{\theta}_j)(1 - \sigma(\boldsymbol{\theta}_j)) & \text{if } k = j, \\ -\sigma(\boldsymbol{\theta}_j)\sigma(\boldsymbol{\theta}_k) & \text{else,} \end{cases} \tag{4.11}$$

$$\frac{\partial \boldsymbol{P}_\alpha^t}{\alpha} = \sum_{k=1}^{t} \boldsymbol{P}_\alpha^{k-1}(\boldsymbol{I}_n - \boldsymbol{W}\boldsymbol{D}^{-1})\boldsymbol{P}_\alpha^{t-k}, \tag{4.12}$$

$$\frac{\partial \boldsymbol{P}_\alpha^t}{\boldsymbol{W}_{ab}} = (1-\alpha)\sum_{k=1}^{t} \boldsymbol{P}_\alpha^{k-1}(\boldsymbol{J}^{ab}\boldsymbol{D}^{-1})\boldsymbol{P}_\alpha^{t-k}. \tag{4.13}$$

4.C Datasets

In this section we further analyze individual datasets. Relating composition of the dataset as shown in Table 4.1 to the relative performance of our models as shown in Table 4.2.

DD [55]: Is a dataset extracted from the protein data bank (PDB) of 1178 high resolution proteins. The task is to distinguish between enzymes and non-enzymes. Since these are high resolution structures, these graphs are significantly larger than those found in our other biochemical datasets with a mean graph size of 284 nodes with the next largest biochemical dataset with a mean size of 39 nodes.

ENZYMES [26]: Is a dataset of 600 enzymes divided into 6 balanced classes of 100 enzymes each. As we analyzed in the main text, scattering features are better able to preserve the structure between classes. LEGS-FCN slightly relaxes this structure but improves accuracy from 32 to 39% over LEGS-FIXED.

NCI1, NCI109 [211]: Contains slight variants of 4100 chemical compounds encoded as graphs. Each compound is separated into one of two classes based on its activity against non-small cell lung cancer and ovarian cancer cell lines. Graphs in this dataset are 30 nodes with a similar number of edges. This makes for long graphs with high diameter.

PROTEINS [26]: Contains 1178 protein structures with the goal of classifying enzymes vs. non enzymes. GCN outperforms all other models on this dataset, however the Baseline model, where no structure is used also performs very similarly. This suggests that the graph structure within this dataset does not add much information over the structure encoded in the eccentricity and clustering coefficient.

PTC [196]: Contains 344 chemical compound graphs divided into two classes based on whether or not they cause cancer in rats. This dataset is very difficult to classify without features however LEGS-RBF and LEGS-FCN are able to capture the long range connections slightly better than other methods.

COLLAB [218]: 5000 ego-networks of different researchers from high energy physics, condensed matter physics or astrophysics. The goal is to determine which field the research belongs to. The GraphSAGE model performs best on this dataset although the LEGS-RBF network performs nearly as well. Ego graphs have a very small average diameter. Thus shallow networks can perform quite well on them as is the case here.

IMDB [218]: For each graph, nodes represent either actresses or actors and there is an edge between them if they are in the same movie. These graphs are also ego graphs around specific actors. IMDB-BINARY classifies between action and romance genres. IMDB-MULTI classifies between 3 classes. Somewhat surprisingly GS-SVM performs the best with other LEGS networks close behind. This could be due to oversmoothing on the part of GCN and GraphSAGE when the graphs are so small.

REDDIT [218]: Graphs in REDDIT-BINARY/MULTI-5K/MULTI-12K datasets each graph represents a discussion thread where nodes correspond to users and there is an edge between two nodes if one replied to the other's comment. The task is to identify which subreddit a given graph came from. On these datasets GCN outperforms other models.

QM9 [78, 215]: Graphs in the QM9 dataset each represent chemicals with 18 atoms. Regression targets represent chemical properties of the molecules.

Table 4.5: Mean ± std. over test set selection on cross-validated LEGS-RBF Net with reduced training set size.

Train, Val, Test %	80%, 10%, 10%	70%, 10%, 20%	40%, 10%, 50%	20%, 10%, 70%
COLLAB	75.78 ± 1.95	75.00 ± 1.83	74.00 ± 0.51	72.73 ± 0.59
DD	72.58 ± 3.35	70.88 ± 2.83	69.95 ± 1.85	69.43 ± 1.24
ENZYMES	36.33 ± 4.50	34.17 ± 3.77	29.83 ± 3.54	23.98 ± 3.32
IMDB-BINARY	64.90 ± 3.48	63.00 ± 2.03	63.30 ± 1.27	57.67 ± 6.04
IMDB-MULTI	41.93 ± 3.01	40.80 ± 1.79	41.80 ± 1.23	36.83 ± 3.31
MUTAG	33.51 ± 4.34	33.51 ± 1.14	33.52 ± 1.26	33.51 ± 0.77
NCI1	74.26 ± 1.53	74.38 ± 1.38	72.07 ± 0.28	70.30 ± 0.72
NCI109	72.47 ± 2.11	72.21 ± 0.92	70.44 ± 0.78	68.46 ± 0.96
PROTIENS	70.89 ± 3.91	69.27 ± 1.95	69.72 ± 0.27	68.96 ± 1.63
PTC	57.26 ± 5.54	57.83 ± 4.39	54.62 ± 3.21	55.45 ± 2.35
REDDIT-BINARY	86.10 ± 2.92	86.05 ± 2.51	85.15 ± 1.77	83.71 ± 0.97
REDDIT-MULTI-12K	38.47 ± 1.07	38.60 ± 0.52	37.55 ± 0.05	36.65 ± 0.50
REDDIT-MULTI-5K	47.83 ± 2.61	47.81 ± 1.32	46.73 ± 1.46	44.59 ± 1.02

4.D Training Details

We train all models for a maximum of 1000 epochs with an initial learning rate of $1e^{-4}$ using the ADAM optimizer [103]. We terminate training if validation loss does not improve for 100 epochs testing every 10 epochs. Our models are implemented with Pytorch [158] and Pytorch geometric. Models were run on a variety of hardware resources. For all models we use $q = 4$ normalized statistical moments for the node to graph level feature extraction and $m = 16$ diffusion scales in line with choices in [72].

4.D.1 Cross Validation Procedure

For all datasets we use 10-fold cross validation with 80% training data 10% validation data and 10% test data for each model. We first split the data into 10 (roughly) equal partitions. For each model we take exactly one of the partitions to be the test set and one of the remaining nine to be the validation set. We then train the model on the remaining eight partitions using the cross-entropy loss on the validation for early stopping checking every ten epochs. For each test set, we use majority voting of the nine models trained with that

Table 4.6: Test set mean squared error on CASP GDT regression task across targets over 3 non-overlapping test sets.

	LEGS-RBF	LEGS-FCN	LEGS-FIXED	GCN	GraphSAGE	GIN	Baseline
t0860	197.68 ± 34.29	**164.22 ± 10.28**	206.20 ± 28.46	314.90 ± 29.66	230.45 ± 79.72	262.35 ± 66.88	414.41 ± 26.96
t0868	131.42 ± 8.12	**127.71 ± 14.26**	178.45 ± 5.64	272.14 ± 26.34	191.08 ± 21.96	170.05 ± 27.26	411.98 ± 57.39
t0869	106.69 ± 9.97	132.12 ± 31.37	**104.47 ± 14.16**	317.22 ± 12.75	244.38 ± 40.58	217.02 ± 57.01	393.12 ± 48.70
t0872	144.11 ± 24.88	148.20 ± 23.63	**134.48 ± 8.25**	293.96 ± 19.00	221.13 ± 28.74	240.89 ± 24.17	374.48 ± 33.70
t0879	89.00 ± 44.94	80.14 ± 16.21	**64.63 ± 15.92**	309.23 ± 69.40	172.41 ± 73.07	147.77 ± 15.72	364.79 ± 144.32
t0900	193.74 ± 10.78	171.05 ± 25.41	**158.56 ± 9.87**	254.11 ± 18.63	209.07 ± 11.90	265.77 ± 79.99	399.16 ± 83.48
t0912	**113.00 ± 22.31**	169.55 ± 27.35	150.70 ± 8.53	227.17 ± 22.11	192.28 ± 39.45	271.30 ± 28.89	406.25 ± 31.42
t0920	**80.46 ± 14.98**	136.94 ± 36.43	84.83 ± 19.70	361.19 ± 71.25	261.72 ± 59.67	191.86 ± 37.85	398.22 ± 25.60
t0921	187.89 ± 46.15	165.97 ± 42.39	**142.97 ± 27.09**	382.69 ± 20.27	260.49 ± 16.09	207.19 ± 24.84	363.92 ± 35.79
t0922	254.83 ± 91.28	**110.54 ± 43.99**	227.73 ± 26.41	366.72 ± 8.10	290.71 ± 7.22	130.46 ± 11.64	419.14 ± 45.49
t0942	188.55 ± 11.10	167.53 ± 22.01	**137.21 ± 7.43**	371.31 ± 9.90	233.78 ± 84.95	254.38 ± 47.21	393.03 ± 24.93
t0944	146.59 ± 8.41	**138.67 ± 50.36**	245.79 ± 58.16	263.03 ± 9.43	199.40 ± 51.11	157.90 ± 2.57	404.12 ± 40.82

Table 4.7: Quantified distance between the empirically observed enzyme class exchange preferences of [49] and the class exchange preferences inferred from LEGS-FIXED, LEGS-FCN, and a GCN. We measure the cosine distance between the graphs represented by the chord diagrams in Figure 4.2. As before, the self-affinities were discarded. LEGS-Fixed reproduces the exchange preferences the best, but LEGS-FCN still reproduces well and has significantly better classification accuracy.

LEGS-FIXED	LEGS-FCN	GCN
0.132	0.146	0.155

test set. We then take the mean and standard deviation across these test set scores to average out any variability in the particular split chosen. This results in 900 models trained on every dataset. With mean and standard deviation over 10 ensembled models each with a separate test set.

4.E Ensembling Evaluation

Recent work by Min et al. [143] combines the features from a fixed scattering transform with a GCN network, showing that this has empirical advantages in semi-supervised node classification, and theoretical representation advantages over a standard [106] style GCN. We ensemble the learned features from a learnable scattering network (LEGS-FCN) with those of GCN and compare this to ensembling fixed scattering features with GCN as in Min et al. [143], as well as the solo features. Our setting is slightly different in that we use the GCN features from pretrained networks, only training a small 2-layer ensembling network on the combined graph level features. This network consists of a batch norm layer, a 128 width fully connected layer, a leakyReLU activation, and a final classification layer down to the number of classes. In Table 4.8 we see that combining GCN features with fixed scattering features in LEGS-FIXED or learned scattering features in LEGS-FCN always helps classification. Learnable scattering features help more than fixed scattering features overall and particularly in the biochemical domain.

Table 4.8: Mean ± standard deviation test set accuracy on biochemical and social network datasets.

	GCN	GCN-LEGS-FIXED	GCN-LEGS-FCN
DD	67.82 ± 3.81	**74.02 ± 2.79**	73.34 ± 3.57
ENZYMES	31.33 ± 6.89	31.83 ± 6.78	**35.83 ± 5.57**
MUTAG	79.30 ± 9.66	82.46 ± 7.88	**83.54 ± 9.39**
NCI1	60.80 ± 4.26	70.80 ± 2.27	**72.21 ± 2.32**
NCI109	61.30 ± 2.99	68.82 ± 1.80	**69.52 ± 1.99**
PROTEINS	74.03 ± 3.20	73.94 ± 3.88	**74.30 ± 3.41**
PTC	56.34 ± 10.29	**58.11 ± 6.06**	56.64 ± 7.34
COLLAB	73.80 ± 1.73	**76.60 ± 1.75**	75.76 ± 1.83
IMDB-BINARY	47.40 ± 6.24	65.10 ± 3.75	**65.90 ± 4.33**
IMDB-MULTI	39.33 ± 3.13	**39.93 ± 2.69**	39.87 ± 2.24
REDDIT-BINARY	81.60 ± 2.32	86.90 ± 1.90	**87.00 ± 2.36**
REDDIT-MULTI-12K	42.57 ± 0.90	45.41 ± 1.24	**45.55 ± 1.00**
REDDIT-MULTI-5K	52.79 ± 2.11	**53.87 ± 2.75**	53.41 ± 3.07

Table 4.9: Mean ± std. over four runs of mean squared error over 19 targets for the QM9 dataset, lower is better.

	LEGS-FCN	LEGS-FIXED	GCN	GraphSAGE	GIN	Baseline
Target 0	**0.749 ± 0.025**	0.761 ± 0.026	0.776 ± 0.021	0.876 ± 0.083	0.786 ± 0.032	0.985 ± 0.020
Target 1	**0.158 ± 0.014**	0.164 ± 0.024	0.448 ± 0.007	0.555 ± 0.295	0.191 ± 0.060	0.593 ± 0.013
Target 2	**0.830 ± 0.016**	0.856 ± 0.026	0.899 ± 0.051	0.961 ± 0.057	0.903 ± 0.033	0.982 ± 0.027
Target 3	0.511 ± 0.012	**0.508 ± 0.005**	0.549 ± 0.010	0.688 ± 0.216	0.555 ± 0.006	0.805 ± 0.025
Target 4	**0.587 ± 0.007**	**0.587 ± 0.006**	0.609 ± 0.009	0.755 ± 0.177	0.613 ± 0.013	0.792 ± 0.010
Target 5	**0.646 ± 0.013**	0.674 ± 0.047	0.889 ± 0.014	0.882 ± 0.118	0.699 ± 0.033	0.833 ± 0.026
Target 6	0.018 ± 0.012	0.020 ± 0.011	0.099 ± 0.011	0.321 ± 0.454	**0.012 ± 0.006**	0.468 ± 0.005
Target 7	0.017 ± 0.005	0.024 ± 0.008	0.368 ± 0.015	0.532 ± 0.405	**0.015 ± 0.005**	0.379 ± 0.013
Target 8	0.017 ± 0.005	0.024 ± 0.008	0.368 ± 0.015	0.532 ± 0.404	**0.015 ± 0.005**	0.378 ± 0.013
Target 9	0.017 ± 0.005	0.024 ± 0.008	0.368 ± 0.015	0.532 ± 0.404	**0.015 ± 0.005**	0.378 ± 0.013
Target 10	0.017 ± 0.005	0.024 ± 0.008	0.368 ± 0.015	0.533 ± 0.404	**0.015 ± 0.005**	0.380 ± 0.014
Target 11	**0.254 ± 0.013**	0.279 ± 0.023	0.548 ± 0.023	0.617 ± 0.282	0.294 ± 0.003	0.631 ± 0.013
Target 12	0.034 ± 0.014	0.033 ± 0.010	0.215 ± 0.009	0.356 ± 0.437	**0.020 ± 0.002**	0.478 ± 0.014
Target 13	0.033 ± 0.014	0.033 ± 0.010	0.214 ± 0.009	0.356 ± 0.438	**0.020 ± 0.002**	0.478 ± 0.014
Target 14	0.033 ± 0.014	0.033 ± 0.010	0.213 ± 0.009	0.355 ± 0.438	**0.020 ± 0.002**	0.478 ± 0.014
Target 15	0.036 ± 0.014	0.036 ± 0.011	0.219 ± 0.009	0.359 ± 0.436	**0.023 ± 0.002**	0.479 ± 0.014
Target 16	0.002 ± 0.002	0.001 ± 0.001	0.017 ± 0.034	0.012 ± 0.022	**0.000 ± 0.000**	0.033 ± 0.013
Target 17	0.083 ± 0.047	**0.079 ± 0.033**	0.280 ± 0.354	0.264 ± 0.347	0.169 ± 0.206	0.205 ± 0.220
Target 18	**0.062 ± 0.005**	0.176 ± 0.231	0.482 ± 0.753	0.470 ± 0.740	0.321 ± 0.507	0.368 ± 0.525

Part II

Optimal Transport in Deep Learning

Chapter 5

Fixing Bias in Reconstruction-based Anomaly Detection

5.1 Introduction

A common problem in real-world data analysis is to identify outliers or anomalies in complex high-dimensional data without labels or annotations. At a high level we would like points "similar enough" to the training data to be scored as nominal and points that are not similar to be scored as anomalous. The real question is then how to describe similarity to the training set. We consider anomalous points as those that have a low likelihood of occurring in data generated from the nominal distribution and are likely generated through some other (anomalous) process. This formulation suggests a density estimation solution. However, in high-dimensional datasets direct density estimation suffers from the well-known curse of dimensionality [227], requiring a sample size exponential in the dimension. This motivates the use of deep networks, which have proven effective in high dimensional spaces and particularly in cases where there exists structure between dimensions such as in images or graphs. In such cases, convolutions can take advantage of feature locality.

Autoencoders are common in deep unsupervised anomaly detection with reconstruction error serving as a way to identify anomalies, as they give worse reconstruction points that are far from the training set [34, 155]. However, recent work studying overparameterized autoencoders suggests that MSE trained autoencoders tend to memorize their training set, essentially becoming one-nearest-neighbor density estimators [168]. While they may converge to the true density with infinite samples [224], this memorization can be detrimental for anomaly detection in realistic data. In particular, autoencoders are sensitive to outliers in the training data and biased towards easily reconstructed points, even if they are anomalous (e.g., in low-density regions of the convex hull of the data). In addition, many types of autoencoders such as convolutional autoencoders have built-in invariances (for example to shifts in an image) that make it difficult to detect certain types of anomalies. In particular we note that autoencoders are:

1. sensitive to outliers in the training data. When a few anomalous points contaminate the training data, their effectiveness rapidly deteriorates,
2. biased towards simple to reconstruct points, particularly in the span of the data, often easily reconstructing points in low density regions within the hull,
3. are only able to work well on domains that can be decoded,
4. and may contain built-in invariances (for example to noise or shifts) that make certain types of anomalies difficult to detect.

Some recent work (e.g., [35, 2, 175]) propose the addition of loss terms that constrain the autoencoder latent space to better limit the capacity, while other works propose leveraging adversarial losses to compute reconstruction error in the generator latent space [182]. However, these methods all still fundamentally rely on a reconstruction penalty (with its inherent deficiencies) for quantifying abnormality of evaluated points compared to training data.

Here, we propose a new framework for anomaly detection that falls outside standard reconstruction-based methods by considering transformations that *do* lead to anomalies. Our model, the Lipschitz anomaly discriminator (LAD) takes an input point and outputs its anomaly score directly. To train LAD we minimize the score of training points and maximize the score of “corrupted” points. Here corrupted points are training points that have been transformed to be more anomalous. We find that the addition of these corruption functions gives a number of benefits over reconstruction-based methods that cannot incorporate knowledge of corruption functions.

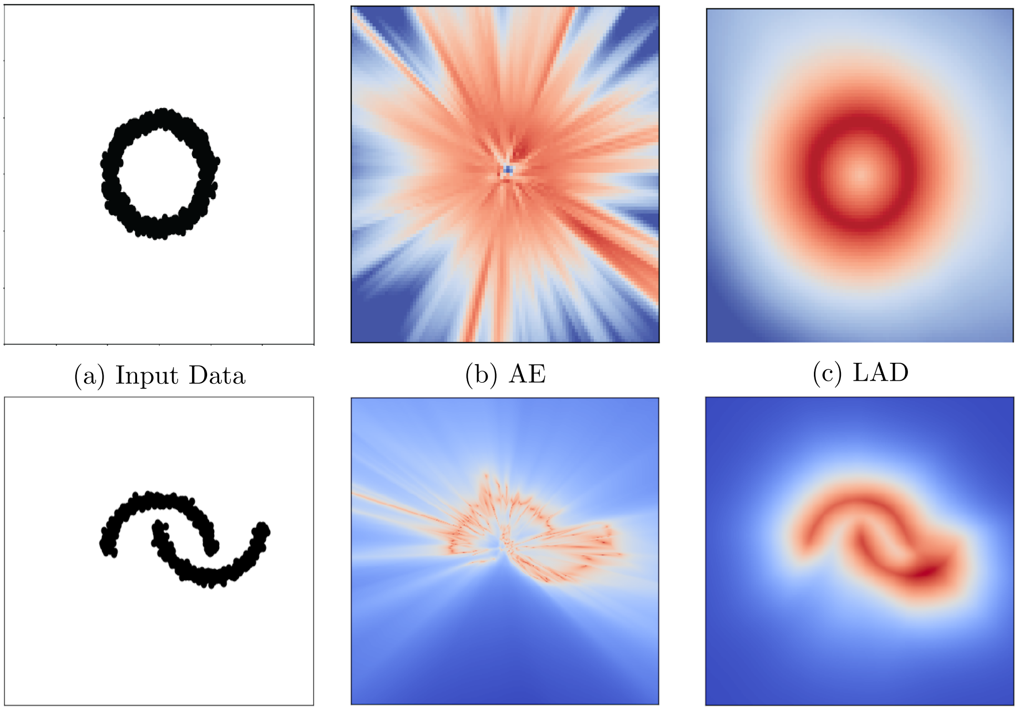
1. LAD is less sensitive to outliers in the training data, when a few anomalous points contaminate the training data the performance of reconstruction-based methods deteriorates more rapidly than that of LAD.
2. LAD does not require a specification of the bottleneck layer size, a key factor that has recently received significant attention [174].
3. LAD does not exhibit bias where some data points are intrinsically easier to reconstruct than others.
4. LAD does not require a decoder to the input space, this is critical in some applications where it is difficult to reconstruct inputs from the encoding space as in graph structured data.

Interestingly, LAD is connected to integral probability measures (IPMs), which provide an alternative probabilistic perspective to that of ϕ-divergences, which are the basis for variational and adversarial autoencoders used in anomaly detection. IPMs in some sense measure the difference between two distributions rather than the quotient [193]. Using the difference instead of the quotient leads to benefits in terms of robustness to a corrupted training set and to certain perturbations of the training set, which we explore in Section 5.4.3. We show the benefits of the LAD framework on health record data where the anomalies can be quite concentrated, and in graph datasets which are difficult to autoencode.

The remainder of the paper is organized as follows. In Section 5.2, we review the literature and discuss the strengths and weaknesses of existing methods. In Section 5.3, we present the preliminary formulations, and in Section 5.4, we develop our framework and study its theoretical properties. Section 5.5 includes our numerical experiments, and finally, Section 5.6 is our conclusion.

5.2 Related Work

Reconstruction-based methods Many works use an information constrained model trained with reconstruction error assuming that outliers will be poorly reconstructed. The first deep methods either use traditional outlier detection methods on the learned encoding [7, 62], or directly use the reconstruction error of test points [88, 178]. More recent works have additional losses but are still fundamentally based on reconstruction penalties. In [35, 2, 166, 162] the authors add losses to constrain the latent space of an autoencoder to better limit the capacity. In Section 5.5, we demonstrate how reconstruction-based methods are sensitive to contamination of the training set, exhibit bias where some points are intrinsically easier to reconstruct than others based on the architecture, and how the requirement of a decoder is detrimental in graph data, which is difficult to decode.



(a) Input Data (b) AE (c) LAD

(d) Input Data (e) AE (f) LAD

Figure 5.1: (a,c) training data (b,e) autoencoder reconstruction score (c,f) LAD anomaly score (red = more normal, blue = more anomalous). Reconstruction is a poor proxy for data density and interpolates too well close to the data.

GAN methods Generative adversarial networks seem like another promising direction for anomaly detection as they can model complex datatypes. There are a few ways GANs have been incorporated in existing work, however they are all ultimately similar to the autoencoder based models in that they rely on a model of the data and reconstruction error either in the latent space or in the data space. GANs have been used to find the nearest datapoint to any given point in order to find a type of reconstruction error. In particular the datapoint in question is traced back to a point in the latent space of the GAN (via backpropagation), and the nearest point in the data is generated by forward propagation of the latent point. The reconstruction error is then computed between the GAN-generated point and the potentially anomalous point [182]. In [222] this process is done by a learned inverse network. In [5], authors use an autoencoder to reconstruct points, but such that they are within the training data, as enforced by a GAN loss. Thus this network has both a GAN loss and an autoencoder loss. In general, these methods do not perform as well as autoencoders when scoring whole images [175, 53], which is the task we focus on here, and may suffer from the same type of problems as we note in this paper on autoencoder based reconstruction scoring. Other work attempts to constrain the autoencoder embedding space with an adversarial loss such that it only represents only in-class points [162] or conforms to a prior distribution [166]. This may remove some of the accurately reconstructed low-density interpolations but may make it even more sensitive to outliers by increasing the volume of embedded space assigned to each sample in the training set.

Support vector methods Support vector based methods attempt to separate the normal data from the anomalous space with either planes (OC-SVM) [183] or hyperspheres (SVDD) [195]. One-class neural networks (OC-NN) [36] and deep support vector data description (DSVDD) in [173] generalize these to use deep networks. These methods attempt to create an embedding space where standard OC-NN and SVDD can be applied. These generalizations do not perform as well as autoencoder based methods [166, 162].

Distance-based density estimation methods A relatively efficient way to estimate density is using nearest neighbor distances [108, 169, 224, 60]. We compare against the method in [27] as one of the best performing methods over a wide variety of datasets [227, 33]. However, these methods require a k-nearest neighbor query, which requires at least $O(n \log n)$ time and at least $O(n)$ space [21]. We instead use a model-based method that has complexity independent of dataset size. We adapt the distance-based framework to high dimensions using the dual formulation of the Wasserstein distance.

5.3 Background

Given samples from a nominal probability distribution P over $\mathcal{X}$, the *density level set* formulation can be seen as the following. Given some $\alpha > 0$ we wish to produce a decision function $c : \mathcal{X} \to \{0, 1\}$ on the nominal probability density $p(x)$, such that $c(x) = \mathbf{1}[(p(x) > \alpha]$ where $\mathbf{1}[\cdot]$ denotes the indicator function and α is predefined either using some absolute density level or in relation to some quantile of the training data controlling the false positive error [224].

Recently, there is renewed interest in the use of transportation metrics, such as the Wasserstein metric, to train neural networks. The Wasserstein GAN [8] and later works [84] demonstrated the advantages of training GANs with a Wasserstein based loss. The Wasserstein-1 metric, also known as Earth-mover distance, between two distributions P and Q defined over $\mathbb{R}^n$ with some distance metric d is:

$$W(P, Q) = \inf_{\pi \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\pi}\big[d(x, y)\big] \tag{5.1}$$

This can be thought of as transport plan π that incurs the minimum amount of work to move one pile of dirt (P) to another (Q). By the Kantorovich-Rubinstein duality [207] for the Euclidean distance metric, i.e., $d(x, y) = \|x - y\|$,

$$W(P, Q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q} f(x) \tag{5.2}$$

Here f is often called the witness function, as it "witnesses" the difference between P and Q maximally. By constraining the network to be 1-Lipschitz we can, by gradient descent, approximate the Wasserstein distance between the two distributions for which we have samples.

This second formulation can be more generally described in terms of integral probability measures (IPMs), which are a family measures that quantify the distance between distributions in terms of integrations over some witness constrained function f. Depending on the form of f, IPMs encompass the well-known Wasserstein distance, the total variation distance, Kolmogorov distance, maximum mean discrepancy (MMD) depending on the restrictions placed on the family of functions optimized over $\mathcal{F}$. In the case of the Wasserstein distance $\mathcal{F} = \|f\|_L \leq 1$, i.e. functions with Lipschitz constant at most 1. We focus on the Wasserstein distance and the family of Lipschitz functions as these are easy to approximate using neural networks, but we note that all of our theoretical results can be adapted more generally to IPMs. In anomaly detection, basing our method on IPMs is useful as IPMs are more robust to some sorts of noise and can have better rates of convergence than other divergences measured on a pointwise basis, which are most commonly ϕ-divergences [193].

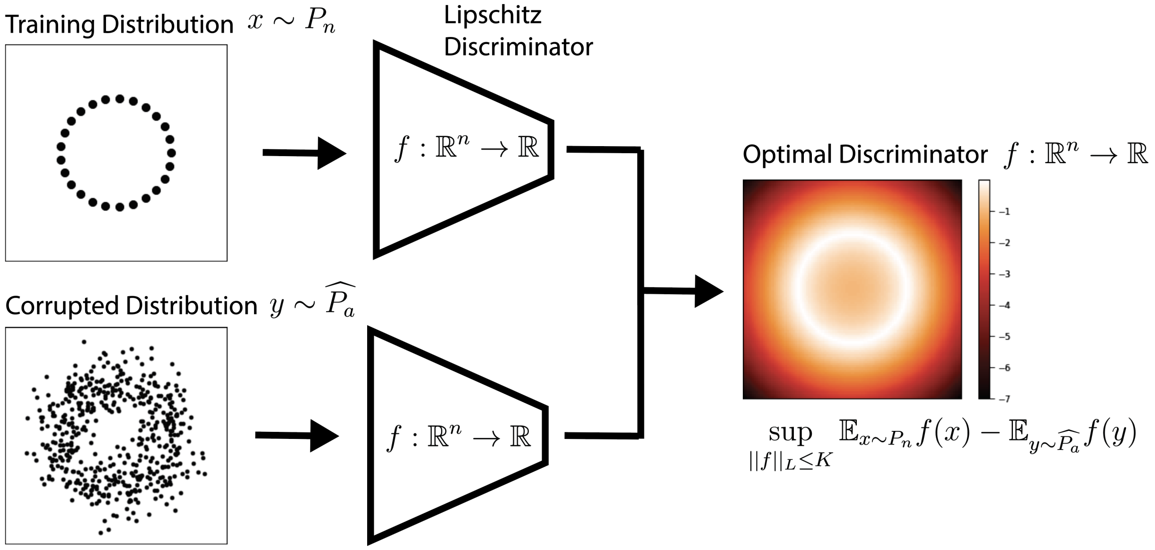

Training Distribution $x \sim P_n$
Lipschitz Discriminator
$f : \mathbb{R}^n \to \mathbb{R}$
Corrupted Distribution $y \sim \widehat{P_a}$
$f : \mathbb{R}^n \to \mathbb{R}$
Optimal Discriminator $f : \mathbb{R}^n \to \mathbb{R}$
−1
−2
−3
−4
−5
−6
−7
$\sup_{||f||_L \leq K} \mathbb{E}_{x \sim P_n} f(x) - \mathbb{E}_{y \sim \widehat{P_a}} f(y)$

Figure 5.2: LAD trains a Lipschitz neural network f to discriminate between the data and a corrupted version of the data. Our trained network f^* is then used to score anomalies. Darker = more anomalous.

5.4 The Lipschitz Anomaly Discriminator

We propose to learn an anomaly scoring function as the output of a neural network we call *Lipschitz anomaly discriminator* (LAD). Our neural network function f is trained to maximally discriminate between nominal training data P_n, and a corrupted version of it $\widehat{P_a}$. Since we are tackling the unsupervised anomaly detection problem, we use a corrupted version of the training data as a substitute for the true (unknown) distribution of anomalous points P_a (See Fig. 5.2). This corrupted version of the training data allows us to directly encode knowledge about what kinds of transformations might lead to anomalous points, which is not possible when using a reconstruction quality approach. To incorporate knowledge of what kinds of transformations and invariances do not lead to anomalies, we can take advantage of the neural network structure of f and, as in other deep anomaly detection models, augment the training data.

While general forms of f can be considered, we focus on f with global Lipschitz constraints. We focus on Lipschitz constrained networks for the following three reasons: (1) This constraint gives assurance that similar datapoints will be scored similarly, which is not true of a network with arbitrary gradients. (2) There exist previous work in the generative adversarial network literature that explore how to efficiently optimize a Lipschitz constrained network [8, 84], which we can borrow from. For other desirable constrained families of functions, for instance Hölder continuous functions, there do not exist efficient stochastic gradient descent algorithms for solving. (3) Optimizing over the family of Lipschitz functions is equivalent to solving for the Wasserstein distance, which is well studied and has useful theoretical and geometric interpretations.

To optimize our discriminator, we use the gradient penalty formulation in WGAN-GP [84], given by:

$$\begin{aligned} L = {} & \mathbb{E}_{x \sim P_n}\left[f(x)\right] - \mathbb{E}_{x \sim \widehat{P_a}}\left[f(x)\right] \\ & + \lambda \mathbb{E}_{x \sim P_x}\left[(\|\nabla_x f(x)\|_2 - 1)^2\right], \end{aligned} \tag{5.3}$$

where P_x is obtained by sampling uniformly along straight lines between pairs of points sampled from $\widehat{P_a}$ and P_n. We use $\lambda = 10$ as suggested in the original work. This enforces a gradient of unit norm at least in the interior of the data.

We note that while the GAN-based anomaly detection [182, 222, 5] methods also use discriminators (and generators), these methods are more similar to the autoencoder methods in that they use reconstruction error, effectively the difference between an image and its closest neighbor that can be generated by the GAN to determine anomalies. They do not use the discriminator to directly score anomalies.

5.4.1 Choosing the Corrupted Distribution

The choice of the corrupted distribution to train against is useful in building inductive bias into the model. In LAD, choice of the corruption process allows us to bias the model towards what kinds of anomalies to expect.

The difference in the average score on the true anomalies and the average score on the true nominal points is bounded above by the Wasserstein distance between the true anomaly distribution and the estimate of the anomaly distribution (see Eq. 5.8). This guarantees that the better the estimate of the anomaly distribution, the better the performance of LAD. Practically, we find the corruption process should be datatype dependent, but the exact amount of noise is not critical. In our experiments, we first corrupt the data with gaussian noise on all datatypes. When we have information on the structure of the data, we add additional structured corruption. For example, on images we shuffle image patches and on graphs we remove and add edges with some probability.

5.4.2 Combination with Other Models

Since LAD is quite different from the methods based on autoencoders, it is natural to consider what happens when we combine separately trained LAD and autoencoder models. We call this model LAD+CAE, our Lipschitz anomaly discriminator combined with a convolutional autoencoder. This model provides a state of performance on MNIST and near state of the art on CIFAR10 with a relatively small model (the same sized used on MNIST) without extensive tuning.

Since these two models have outputs on different scales, to combine them we divide each score by the training set standard deviation for that model before summing the two scores. This is a very simple ensemble method that could be improved with more investigation. However, the performance gain by combining these two models suggests that they have very different biases in which type of samples they score as anomalous. Where each model has an advantage is an interesting question which we investigate in Section 5.5.1.

5.4.3 Theoretical Properties of Anomaly Discrimination

Training set contamination

The standard anomaly detection task assumes access to training data sampled i.i.d. from some nominal distribution, which is unrealistic in a big data setting. Formally, we consider the problem where samples are drawn from a mixture of the nominal probability distribution P_n and the anomalous distribution P_a. That is, we are given n samples $\{x_i\}_{i=1}^n$ drawn i.i.d. from the probability distribution $(1-\gamma)P_n + \gamma P_a$, where $\gamma \in (0,1)$ and $\gamma \ll 1$ represents the anomaly contamination. We show that while existing deep anomaly detection methods perform well on clean training data (i.e., $\gamma = 0$), they are very sensitive to even a small amount of contamination in the training set.

Suppose our training set is corrupted with a fraction $\gamma > 0$ of anomalous datapoints, then our method should still effectively distinguish between nominal and anomalous data. Because we are training the discriminator f to maximize the difference between the nominal and corrupted points subject to a Lipschitz constraint, then the Kantorovich-Rubinstein duality holds. Implying that if our corruption process follows the distribution of anomalies (i.e., $\widehat{P_a} = P_a$) then the change in the difference between the nominal and anomalous points is bounded from below. This is summarized in the following proposition.

Proposition 1. Let $f^{(A,B)}$ denote the optimal solution to $\max_{\|f\|_L \leq 1} \mathbb{E}_{x\sim A} f(x) - \mathbb{E}_{x\sim B} f(x)$, and in particular let $f^* = f^{(P_n,P_a)}$ and $f^{**} = f^{((1-\gamma)P_n+\gamma P_a, P_a)}$. If P_n has a compact support $\mathcal{S}_n$ and P_a has compact support $\mathcal{S}_a$ then

$$\begin{aligned}&\left|\mathbb{E}_{x\sim P_n}[f^*(x) - f^{**}(x)] + \mathbb{E}_{x\sim P_a}[f^{**}(x) - f^*(x)]\right| \\ &\qquad\leq \frac{1}{1-\gamma} W(P_n, (1-\gamma)P_n + \gamma P_a).\end{aligned}$$

Proof. Let $\mathcal{A} = \big|\mathbb{E}_{x\sim P_n}[f^*(x) - f^{**}(x)] - \mathbb{E}_{x\sim P_a}[f^*(x) - f^{**}(x)]\big|$. Then, we can write

$$\begin{aligned}
\mathcal{A} &= \big|\mathbb{E}_{x\sim P_n}[f^*(x) - f^{**}(x)] - \mathbb{E}_{x\sim P_a}[f^*(x) - f^{**}(x)]\big| \\
&= \big|(1-\gamma)\mathbb{E}_{x\sim P_n}[f^*(x) - f^{**}(x)] \\
&\quad + \gamma\mathbb{E}_{x\sim P_n}[f^*(x) - f^{**}(x)] \\
&\quad - \mathbb{E}_{x\sim P_a}[f^*(x) - f^{**}(x)]\big| \\
&= \big|\mathbb{E}_{x\sim(1-\gamma)P_n+\gamma P_a}[f^*(x) - f^{**}(x)] \\
&\quad - \mathbb{E}_{x\sim P_a}[f^*(x) - f^{**}(x)] + \gamma\mathcal{A}\big|,
\end{aligned}$$

which forms a geometric series. Therefore, for $\gamma < 1$ we get

$$\begin{aligned}
\mathcal{A} = \frac{1}{1-\gamma}&\big|\mathbb{E}_{x\sim(1-\gamma)P_n+\gamma P_a}[f^*(x) - f^{**}(x)] \\
&- \mathbb{E}_{x\sim P_a}[f^*(x) - f^{**}(x)]\big|.
\end{aligned}$$

We now examine the f^* and f^{**} portions of $\mathcal{A}$ separately. By reorganizing terms, we can write $\mathcal{A} = \frac{1}{1-\gamma}\big|\mathcal{A}_{f^*} + \mathcal{A}_{f^{**}}\big|$ with

$$\begin{aligned}
\mathcal{A}_{f^*} &= \mathbb{E}_{x\sim(1-\gamma)P_n+\gamma P_a}[f^*(x)] - \mathbb{E}_{x\sim P_a}[f^*(x)] \\
&= (1-\gamma)\mathbb{E}_{x\sim P_n}[f^*(x)] + \gamma\mathbb{E}_{x\sim P_a}[f^*(x)] \\
&\quad - \mathbb{E}_{x\sim P_a}[f^*(x)] \\
&= (1-\gamma)\left(\mathbb{E}_{x\sim P_n}[f^*(x)] - \mathbb{E}_{x\sim P_a}[f^*(x)]\right) \\
&= (1-\gamma)W(P_n, P_a); \\
\mathcal{A}_{f^{**}} &= \mathbb{E}_{x\sim P_a}[f^{**}(x)] - \mathbb{E}_{x\sim(1-\gamma)P_n+\gamma P_a}[f^{**}(x)] \\
&= -W((1-\gamma)P_n + \gamma P_a, P_a).
\end{aligned}$$

Combining terms and applying the triangle inequality,

$$\begin{aligned}\mathcal{A} &= \frac{1}{1-\gamma}\big|\mathcal{A}_{f^*} + \mathcal{A}_{f^{**}}\big| \\ &= \frac{1}{1-\gamma}\big|(1-\gamma)W(P_n, P_a) - W((1-\gamma)P_n + \gamma P_a, P_a)\big| \\ &\leq \frac{1}{1-\gamma}W(P_n, (1-\gamma)P_n + \gamma P_a),\end{aligned}$$

which proves the proposition. □

This result can be thought of bounding the difference of score on between normal points and anomalous points when the training set is corrupted by the addition of anomalies $0 \leq \gamma \leq 1$. When the training set is corrupted by a small amount, the anomaly scoring function does not change too much in expectation. Compared to a score based on reconstruction error, a single point can affect the scoring function an arbitrary large amount. For example, take an anomaly distribution that is a Dirac at one point which is easily reconstructable, e.g. the black image in MNIST. Then, corrupting the training data with a single anomalous example can take the MSE to zero for the entire anomaly distribution. This is an adversarial case; however, we find that this also occurs in practice (See Sec. 5.5.1).

Robustness to distant points

As is apparent in Fig. 5.1, for reconstruction-based anomaly detection methods, we are not guaranteed to correctly predict anomalous points even for points that are very far from the support of the nominal data distribution. On the other hand, LAD does not suffer from this instability. We formalize this in our next proposition. Intuitively, points sufficiently far away from the data will have a higher anomaly score than any point in the data.

Proposition 2. Let f^* be the optimal solution of $\sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{x\sim P_n}[f(x)] - \mathbb{E}_{y\sim P_a}[f(y)]\right)$, and let π be the optimal coupling between P_n and P_a defined as the minimizer of $W(P_n, P_a) = \inf_{\pi\in\Pi(P_n,P_a)} \mathbb{E}_{(x,y)\sim\pi}[\|x-y\|]$, where $\Pi(P_n, P_a)$ is the set of joint distributions whose marginals are P_n and P_a, respectively. Then, under the same conditions as Prop. 1, there exists $C > 0$ such that $f^*(y) \leq C - \inf_{x\in\mathcal{S}_n}\{\|x-y\|\}$ for P_a-almost every y.

Proof. First, we recall from Theorem 5.10 (iii) of [207] that since P_n and P_a have compact support f^* exists, and from Theorem 5.10 (ii) of [207] that π-almost surely, $f^*(x) - f^*(y) = \|x - y\|$, and therefore also π-a.s., $f^*(y) = f^*(x) - \|x - y\|$. Now, let

$$A = \{y : \exists x \in \mathcal{S}_n \text{ s.t. } f^*(y) = f^*(x) - \|x - y\|\}, \tag{5.4}$$

and let A^c be its complement. Then, clearly by definition, for any $(x, y) \in \mathcal{S}_n \times A^c$ we must have $f^*(y) \neq f^*(x) - \|x - y\|$, and thus $\pi(\mathcal{S}_n \times A^c) = 0$. Therefore, by the equality of marginals (and since $\mathcal{S}_n$ is the support of P_n, we have $P_a(A^c) = \pi(\mathcal{S}_n \times A^c) = 0$, which yields $P_a(A) = 1$. Thus, for P_a-almost every y we can write $f^*(y) = f(x') - \|x' - y\|$ for some $x' \in \mathcal{S}_n$. Finally, since $\mathcal{S}_n$ is compact and f^* is continuous, we can set $C = \max_{x \in \mathcal{S}_n} f^*(x)$ and clearly have both $f^*(x') \leq C$ and $\|x' - y\| \geq \inf_{x \in \mathcal{S}_n}\{\|x - y\|\}$, which yields the result in the proposition. □

Corollary 5.4.1. *Under the conditions of Prop. 2, there exists $R > 0$ such that for P_a-almost every y, if $\inf_{x \in \mathcal{S}_n} \|y - x\| > R$ then $f^*(y) < f^*(x)$ for P_n-almost every x.*

Proof. Let R be the diameter of $\mathcal{S}_n$ defined as $\sup_{x,y \in \mathcal{S}_n}\{\|x - y\|\}$, which must be finite since $\mathcal{S}_n$ is compact, and let $x_0 \in \mathcal{S}_n$ be a point where f^* reaches its maximum value, which was chosen as C in the proof of Prop. 2. Then, since f^* is a Lipschitz-continuous function, then for every $x \in \mathcal{S}_n$ we have $0 \leq C - f(x) \leq \|x - x_0\| \leq R$, thus, $f(x) \geq C - R$, and together with Prop. 2 we get the result of the corollary. □

For a single training example, this can be understood as a peak at the training example, with f^* decaying with slope 1 in every direction from this point. If we zoom out enough, every set of bounded support is similar to this single training example.

Relation to classification

Previous work has shown that integral probability measures are related to the negative risk of an optimal classifier selected from the same class of functions. In the case of the Wasserstein distance, this is all 1-Lipschitz thus as shown in [193] our optimization can be interpreted as finding the optimal classifier between the training data and the corrupted data in terms of risk, where risk is defined as

$$\mathcal{R}_{\mathcal{F}}^{L} = \inf_{f \in \mathcal{F}} \int_{\mathcal{X}} L(y, f(x)) d\mu(x, y) \tag{5.5}$$

where $L(y, f(x)) = -yf(x)$ with $y \in \{\pm 1\}$.

Robustness to smooth perturbations

The Wasserstein distance, and consequently our model, behaves well under bounded smooth perturbations. Let $h : \mathcal{X} \to \mathcal{X}$ be a continuous function such that $\|h(x) - x\| \leq \epsilon$ and let ν be the image of h applied pointwise to distribution μ defined over $\mathbb{R}^d$, $\nu = \int \mu(h(x))dx$ then as shown in [119], and easily verified using the Kantorovich dual, the Wasserstein distance between μ and ν, $W(\mu, \nu) \leq \epsilon$. This fact applies to our work in the following way, supposed we have access to a smooth perturbation $Q_n = P_n(h(x))dx$ of the true training data distribution P_n, where the perturbation h is bounded pointwise by ϵ, then by the triangle inequality over the Wasserstein distance,

$$|W(Q_n, \widehat{P_a}) - W(P_n, \widehat{P_a})| \leq \epsilon \tag{5.6}$$

$$\left|\left(\mathbb{E}_{Q_n} f_Q - \mathbb{E}_{\widehat{P_a}} f_Q\right) - \left(\mathbb{E}_{P_n} f_P - \mathbb{E}_{\widehat{P_a}} f_P\right)\right| \leq \epsilon. \tag{5.7}$$

The average difference in score between the nominal and anomalous distributions of our discriminator f_Q in the perturbed case is very similar to the optimal discriminator f_P in the unperturbed case. We note that this sort of bound is only possible under integral probability metrics. For example, we examine a simple example under a standard ϕ divergence, such as the KL divergence, which is used in variational and adversarial autoencoders. Let the nominal distribution be supported on an interval of length less than ϵ on $\mathbb{R}$, then under a shift of size ϵ, $h(x) = x + \epsilon$, then $KL(P, Q) \to \infty$ since the supports of P and Q do not intersect, whereas $W(P, Q) \leq \epsilon$.

As a final note, a similar bound can be found between the optimal discriminator function and the one learned by our estimate $\widehat{P_a}$ of the unknown true P_a. Indeed, since both $W(P_n, P_a) \leq W(P_a, \widehat{P_a}) + W(\widehat{P_a}, P_n)$ and $W(P_n, \widehat{P_a}) \leq W(P_n, P_a) + W(P_a, \widehat{P_a})$ by the triangle inequality, then

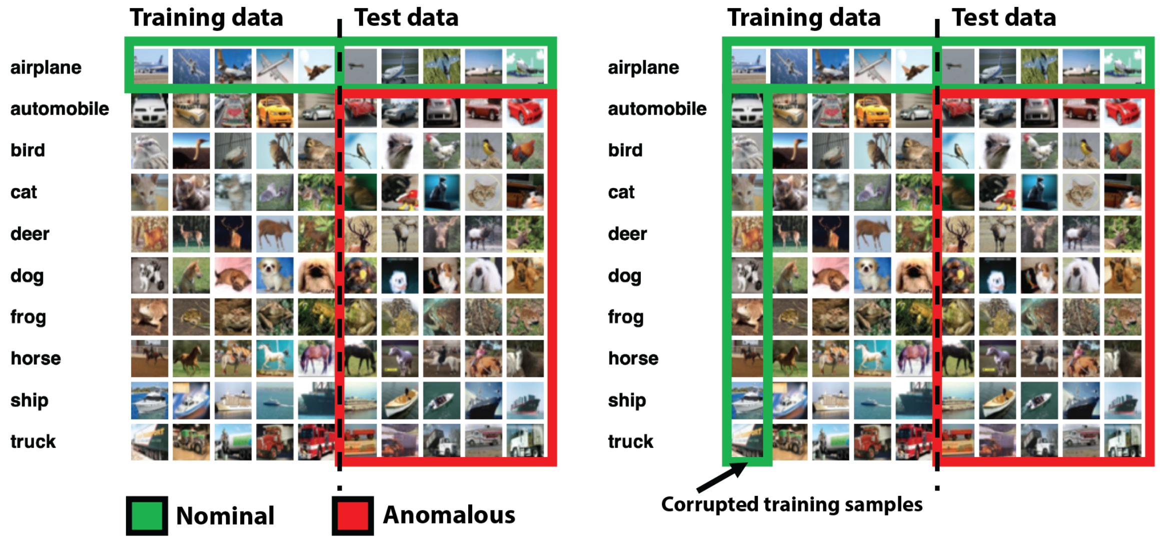
$$\left|W(P_n, P_a) - W(P_n, \widehat{P_a})\right| \leq W(P_a, \widehat{P_a})\,. \tag{5.8}$$

Our model, and in particular the difference in the average scores, is bounded close to the optimal discriminator depending on the choice of anomaly and corruption distributions.

5.5 Experimental Evaluation

We evaluate LAD in three domains, images in Section 5.5.1, health record (i.e. tabular data) in Section 5.5.2, and graphs in Section 5.5.3. In images, reconstruction-based methods are almost ubiquitous, and are the benchmark on unsupervised anomaly detection tasks for semantic anomaly detection [174] (See Figure 5.3 for the setup). Using the architecture of a convolutional autoencoder is key to this performance. We show that LAD performs quite well on these tasks and particularly well when either training set corruption is introduced or on specific biased points. On tabular data benefits from convolutions do not apply and the benefits of deep anomaly detection over more traditional distance-based methods is less clear, but LAD is still able to perform well as it is related as an integral probability measure to distance based methods. On graph structured data autoencoders fail because it is extremely difficult to "decode" from a node order equivariant vector representation of a graph back to the adjacency matrix.

In Figure 5.1, we compare LAD to an autoencoder on some illustrative 2D datasets. Autoencoder reconstruction error has no guarantees, especially on points far from the training data, which is apparent in the "streakiness" in (b,e). The autoencoder learns to reconstruct the data, but also many points outside of the data. Poor reconstruction outside of the data is only indirectly optimized and not guaranteed. LAD on the other hand optimizes a function that is explicitly minimized in low density regions around the data, so we are guaranteed that points far away from the data are scored as anomalies.



a) Unsupervised semantic anomaly detection b) with training set corruption

Figure 5.3: Unsupervised semantic anomaly detection setup for CIFAR10. (a) standard setup and (b) with training set corruption where a small percentage of images from anomalous classes are treated as part of the nominal training data.

5.5.1 Unsupervised Anomaly Detection on Images

We compare LAD against a selection of reconstruction-based methods (CAE, DCAE, RCAE, AND) [35, 88, 209], reconstruction-based methods with adversarial additions (ALOCC, AnoGAN) [182, 175], support vector methods (OCSVM, DSVDD) [183, 173], and more traditional methods (IF, LOF) [131, 27]. AND and AnoGAN models were not run for this paper so results are not available for training set contaminated instances (See Table 5.1). Our experiments are intended to compare LAD with existing reconstruction-based methods as well as traditional anomaly detection methods.

We show how reconstruction-based methods are sensitive to training set contamination (Sec. 5.5.1) and perform poorly on interpolated points (Sec. 5.5.1). Finally, we show how combined with a standard convolutional autoencoder, LAD can achieve excellent performance on MNIST and CIFAR10.

To evaluate our method, we apply our model to three datasets, MNIST, CIFAR10, and the veterans aging cohort study (VACS), an electronic health record dataset with lab values for 1.3 million visits of HIV-positive veterans. For an evaluation metric, we use the area under the curve (AUC) of the receiver operator characteristics (ROC) curve. The

AUC is both scale-invariant and threshold invariant; it measures how well predictions are ranked rather than the specific output values, and it measure the model quality over all classification thresholds, which are both desirable properties for anomaly detection models. The AUC for a model f can be thought of as the probability that a positive example (x^1) is scored higher than a negative example (x^0):

$$\texttt{AUC} = P[(f(x^1) > f(x^0)]; \tag{5.9}$$

All error bars are 95% confidence intervals and are computed over the test set. The test sets for the MNIST and CIFAR datasets are as originially prescribed, and the test sets for the VACS and graph datasets are determined using a random 10% split. All quantitative experiments were run over 3 initializations. All networks have depth 3 with LeakyReLU activations and widths are chosen to match the number of parameters within 1%. Convolutional layers were used for MNIST and CIFAR10 and dense layers were used for VACS.

Training set contamination

To demonstrate empirically the problem with contamination, we show what happens on a standard MNIST anomaly detection task as we add some corruption to the training set. Previous work considers the mean AUC over classes trained on each one of the 10 digits with pure training sets. We consider the same with one additional variable, training set contamination $\pi \in [0, 0.10]$, where up to 10% of the training set are other digits. To contaminate the training data for a given digit, we add random samples from the training data of the other 9 digits to the training data until we reach the correct fraction of training set contamination. To create the corrupted training distribution $\widehat{P_a}$, we take 4×4 patches of the image and shuffle these in random order. To evaluate each model, we use AUC over the entire test set containing 10,000 images, 1,000 of which are the nominal class. Examining the results in Table 5.1, most methods perform quite well on a training set that only contains nominal data, but performance decays rapidly with increased training set corruption. LAD performs competitively on the standard task, but importantly, as the contamination increases above

Train Corrupt.	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
LAD (ours)	0.940	0.937	0.933	0.933	0.926	0.923	0.921	0.917	0.911	0.905	0.901
LAD + CAE (ours)	**0.981**	**0.965**	**0.954**	**0.947**	**0.941**	**0.936**	**0.930**	**0.925**	**0.921**	**0.916**	**0.912**
ALOCC [175]	0.694	0.511	0.514	0.498	0.521	0.539	0.505	0.495	0.504	0.504	0.509
CAE [88]	0.965	0.925	0.898	0.887	0.876	0.868	0.859	0.851	0.844	0.837	0.832
DCAE [209]	0.967	0.925	0.899	0.886	0.874	0.865	0.857	0.850	0.842	0.834	0.829
DSVDD [173]	0.748	0.788	0.748	0.750	0.729	0.718	0.746	0.725	0.703	0.710	0.696
IF [131]	0.853	0.853	0.849	0.845	0.844	0.837	0.836	0.832	0.829	0.827	0.822
LOF [27]	0.973	0.958	0.918	0.873	0.830	0.789	0.762	0.745	0.730	0.718	0.709
OCSVM [183]	0.954	0.895	0.867	0.853	0.840	0.828	0.819	0.812	0.806	0.800	0.794
RCAE [35]	0.957	0.934	0.906	0.894	0.881	0.870	0.861	0.854	0.845	0.838	0.832

Table 5.1: Shows the mean AUC over digits over 3 seeds for training set corruption levels from 0% to 10% on MNIST.

1%, LAD continues to perform well, showing its robustness to training set contamination. This is expected in light of Prop. 1. When combined with a convolutional autoencoder as described in Sec. 5.4, LAD+CAE outperforms existing methods on this task.

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck	mean
LAD (ours)	0.597	**0.663**	0.411	0.531	0.362	0.561	0.409	0.592	**0.777**	**0.744**	0.565
LAD + CAE (ours)	**0.723**	0.497	0.652	0.538	0.725	0.544	0.714	0.565	0.763	0.529	**0.635**
ALOCC [175]	0.421	0.439	0.530	0.505	0.500	0.473	0.500	0.377	0.387	0.500	0.463
AND [2]	0.717	0.494	0.662	0.527	0.736	0.504	0.726	0.560	0.680	0.566	0.617
AnoGAN [182]	0.671	0.547	0.529	**0.545**	0.651	**0.603**	0.585	**0.625**	0.758	0.665	0.618
CAE [88]	0.683	0.454	0.677	0.524	**0.765**	0.525	**0.731**	0.541	0.706	0.435	0.604
DCAE [209]	0.689	0.447	**0.679**	0.534	**0.765**	0.526	0.729	0.539	0.717	0.426	0.605
DSVDD [173]	0.518	0.656	0.528	0.530	0.565	0.568	0.587	0.565	0.540	0.650	0.571
IF [131]	0.670	0.442	0.645	0.512	0.743	0.516	0.711	0.529	0.694	0.529	0.599
LOF [27]	0.661	0.440	0.649	0.506	0.688	0.511	0.692	0.518	0.685	0.405	0.575
OCSVM [42]	0.684	0.456	0.674	0.509	0.749	0.502	0.699	0.501	0.670	0.453	0.590
RCAE [35]	0.675	0.429	0.669	0.541	0.745	0.531	0.690	0.527	0.710	0.404	0.592

Table 5.2: AUC on representative CIFAR10 classes over 3 seeds with mean over all classes.

Bias towards interpolated points in the data convex hull

MNIST Digits vs. All-Black Image Further experiments to study the interpolation bias are shown on MNIST with an all-black image, which is close to the mean MNIST image. Anomaly scores of the black image are compared to anomaly scores of nominal test images by computing the rank of its score within the nominal test data. We train the model on all '0's in the training set, and for testing we compute the black-image score relative to that of the 1,000 test '0's, with higher rank being more anomalous. We would expect this image to be more anomalous than any of the images in the test set (i.e., have a rank of 1). Table 5.3 shows that models based on reconstruction error consider the all black image as

Digit	0	1	2	3	4	5	6	7	8	9	mean
ALOCC [175]	0.351	0.015	0.199	0.194	0.204	0.216	0.164	0.121	0.160	0.060	0.168
LAD (ours)	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CAE [88]	0.017	0.395	0.010	0.001	0.014	0.001	0.053	0.083	0.091	0.002	0.067
DCAE [209]	0.001	0.149	0.001	0.001	0.008	0.001	0.001	0.001	0.299	0.126	0.059
DSVDD [173]	0.757	0.480	0.136	0.368	0.917	0.273	0.847	0.571	0.627	0.736	0.571
LOF [27]	0.876	0.986	0.140	0.625	0.765	0.330	0.961	0.865	0.509	0.895	0.695
OCSVM [183]	0.916	0.977	0.355	0.498	0.668	0.259	0.803	0.863	0.553	0.873	0.677
RCAE [35]	0.023	0.418	0.001	0.001	0.001	0.001	0.005	0.001	0.007	0.033	0.049

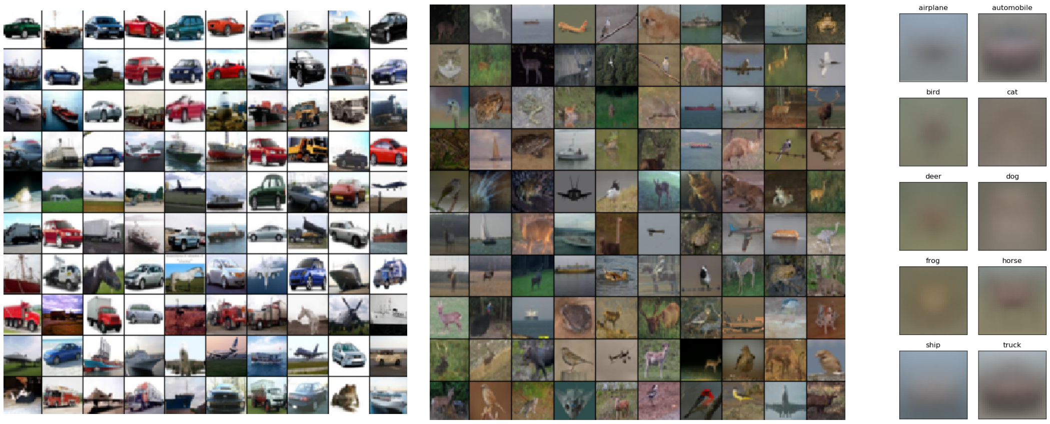
Table 5.3: Shows the mean anomaly rank of the black image in the nominal test data. Scores are in $[0, 1]$, higher is better. Scores are measured on the MNIST test set of each digit trained on the uncorrupted training set for each digit over 3 seeds. Since the black image is inherently easy to reconstruct, it receives a low anomaly score in reconstruction based models.

less anomalous than many of the test digits. LAD consistently ranks the black image as most anomalous.

CIFAR10 performance on high variability classes We perform the same training set corruption experiment on CIFAR10 to show performance on a more complicated dataset (see Table 5.2). We find there is a large variability in performance of each model between classes. In four of the classes, LAD outperforms all other models. Most methods perform poorly on the automobile and truck classes, which is consistent with [2, 173]. Additionally, reconstruction-based methods give low anomaly scores to test points that are close to the mean training image, as it is relatively easy to reconstruct. This is a poor inductive bias in cases where the nominal distribution has low density near the mean. In images of animals most of them are fairly close to the mean. However, in objects with the potential for bright colors (e.g., red cars on white backgrounds or black planes on white sky) these nominal images are very far from the mean training image for the class. This is shown in Fig. 5.4 where most nominal points predicted by a DCAE are relatively near to the mean image. In contrast, LAD gives more varied top nominal images often mistaking trucks for cars but not animals for cars.

5.5.2 Tabular Health Record Anomaly Detection

To establish usability in medical settings and on tabular data, we test our model on health records (see Fig. 5.5) that represent a large source of data that is error prone and difficult

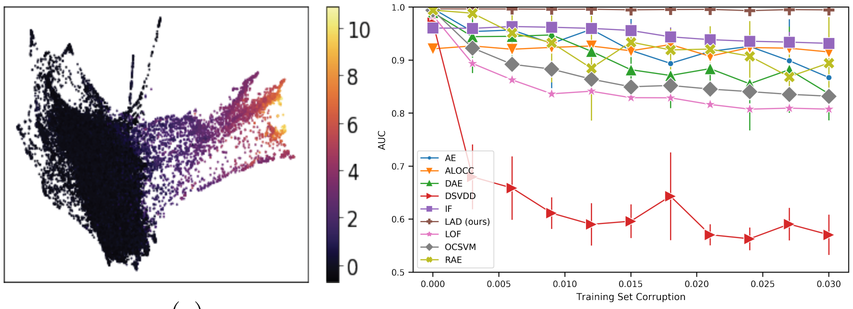


(a) (b) (c)

Figure 5.4: Top 100 nominal images in test set of LAD (a) and DCAE (b) trained on the automobile class. (c) Mean over examples of pixel values for each class. Many car images have white background and/or bright colors that are far from the mean image. LAD does better at modeling such images.

to cleanse of anomalies for training purposes. We use fully connected layers since there is no clear convolutional structure to the data. We also match the number of parameters in each model to within 1%. Here, we use the veterans aging cohort study (VACS) dataset[1] containing 1.3 million clinic visits by over 28,000 HIV-positive veterans. Ten HIV relevant lab values were chosen and standardized (zero mean, unit std). To establish a set of known nominal vs. anomalous points we use clinic visits with a Creatinine lab value $>$ 2 standard deviations away from the mean (in this case $>$ 4.30 mg/dL) as anomalous. This threshold is well above the normal patient reference range [95], indicating high risk for renal disease. We split our data into 80% training and validation set, and 20% test set. To vary the training set contamination, we add a percentage of the high Creatinine values to the nominal training set. Since these patients are all in a similar state, even adding 0.3% outliers to the training set drastically decreases performance. Fig. 5.5 shows the AUC performance of various deep models on this task. Adding even a small amount of high Creatinine lab values encourages an autoencoder to represent them, and thereby all other anomalous ones, reducing performance for reconstruction error based models. All models including more traditional anomaly detection methods have difficulty with these concentrated anomalies

[1] `https://medicine.yale.edu/intmed/vacs/`



(a)

(b)

Figure 5.5: (a) Shows input data embedding with PHATE [148] colored by Creatinine in mg/dL. (b) Shows the AUC of various anomaly detection models over levels of training set corruption on the VACS data.

under training set corruption. Our model robustly detects high Creatinine anomalies, with up to 3% training set contamination. This shows that LAD performs well in medical applications where data cleaning is difficult, and outliers are often concentrated around specific values.

5.5.3 Graph Anomaly Detection

There is a growing interesting in graph structured data. This type of data occurs naturally in social networks, molecular networks, protein interaction networks, and gene interaction networks. Graphs can be used to describe the relations between concepts or the physical structure between objects in a natural way. Most work has focused on learning supervised or semi-supervised node representations for a single large graph [106] (such as nodes in a citation network). Here we focus on learning graph level representations for datasets that contain many graphs, for instance graphs of protein molecules. Specifically, we focus on the problem of predicting anomalous or novel graph structures that are different from set of known graphs. For instance, in the critical assessment of functional annotation (CAFA) challenge [226], the goal is to predict the function of newly designed proteins, whose structure can be represented as graphs [79]. While graph convolutional networks [30, 78, 64] have been primarily applied to classify over known functional annotations, an interesting direction is to predict whether a protein will have a novel function, not described in the current annotation set. This can be particularly useful in drug discovery, where it is relatively easy to generate new protein structures with tools like ROSETTA [116], but we do not necessarily know the function of these newly generated structures. Since novel function is related to novel structure, it would be useful to find novel graph structures from some large potential set.

We compare LAD with the reconstruction error from graph autoencoder (GAE) and variational graph autoencoder (GVAE) [107] on this task. GAE and GVAE are models that learn how to reproduce the adjacency matrix from a node equivariant vector representation learned from the adjacency matrix and feature matrix. We split the training data in the same way as on the image classification datasets in Section 5.5.1 as shown in Figure 5.3. The ENZYMES graph dataset [26] contains 600 graphs split equally over six classes of enzymes. We split the dataset randomly into 80% train 10% validation and 10% test set randomly. To corrupt a graph, we added Gaussian noise of $\sigma = 1/5$ to each feature and dropped and added randomly 1/20 of the edges in each graph. While this is a difficult task, as the state of the art classifiers on this dataset get roughly 65% accuracy [64], we find that LAD is better able to classify these anomalous vs. nominal graphs on the test set as seen in Table 5.4.

$(\mu \pm \sigma)$	Test AUC	Test AP
LAD	**0.572 ± 0.127**	**0.288 ± 0.093**
GAE	0.528 ± 0.112	0.233 ± 0.095
VGAE	0.527 ± 0.109	0.233 ± 0.095

Table 5.4: Mean ± std. over three runs on the six classes of the ENZYMES graph dataset of the area under the receiver operator characteristic (AUC) and the average precision (AP).

5.5.4 Network Architecture and Training Details

In all experiments we use the Adam optimizer [103]. For our Lipschitz models we set $\beta_1 = 0$. For all others we use default optimizer parameters, $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$. As previously noted, network widths are chosen to match the number of parameters between autoencoder and encoder-only architectures. For all models we train with batchsize = 128, and 20,000 batches.

For our Lipschitz anomaly discriminator we use a simple convolutional discriminator architecture designed to approximately match the number of parameters used in the autoen-

Layer	Output Shape	Param #
InputLayer	(28, 28, 1)	0
Conv2D	(14, 14, 16)	272
LeakyReLU	(14, 14, 16)	0
Conv2D	(7, 7, 32)	8224
LeakyReLU	(7, 7, 32)	0
Conv2D	(4, 4, 64)	32832
LeakyReLU	(4, 4, 64)	0
Flatten	(1024)	0
Dense	(64)	65600
LeakyReLU	(64)	0
Dense	(64)	4160
LeakyReLU	(64)	0
Dense	(1)	65
	Total params:	111,153

(a)

Layer	Output Shape	Param #
InputLayer	(28, 28, 1)	0
Conv2D	(14, 14, 16)	272
LeakyReLU	(14, 14, 16)	0
Conv2D	(7, 7, 32)	8224
LeakyReLU	(7, 7, 32)	0
Conv2D	(4, 4, 64)	32832
LeakyReLU	(4, 4, 64)	0
Flatten	(1024)	0
Dense	(10)	10250
Dense	(256)	2816
LeakyReLU	(256)	0
Reshape	(4, 4, 16)	0
Conv2DT	(8, 8, 64)	16448
LeakyReLU	(8, 8, 64)	0
Conv2DT	(16, 16, 32)	32800
LeakyReLU	(16, 16, 32)	0
Conv2DT	(32, 32, 16)	8208
LeakyReLU	(32, 32, 16)	0
Conv2D	(32, 32, 1)	257
Cropping2D	(28, 28, 1)	0
	Total params:	112,107

(b)

Table 5.5: (a) Convolutional Lipschitz network architecture. (b) Convolutional autoencoder architecture.

coder model (Table 5.5a). For our image dataset comparisons, we use a simple convolutional autoencoder (Table 5.5b) with Leaky-ReLU activations. For the Deep-SVDD comparison, we use a similar model as in Table 5.5a, but without bias parameters (as required in [173]). For our ALOCC implementation we use the same architectures as below for the denoising autoencoder and discriminators. For the tabular data in the VACS dataset we use dense layers as shown in Tables 5.6(a,b) again to match the number of parameters.

Code reproducing these results is implemented in Keras and is available at `https://github.com/KrishnaswamyLab/LAD`.

Layer	Output Shape	Param #
Dense	256	2816
LeakyReLU	256	0
Dense	128	32896
LeakyReLU	128	0
Dense	64	8256
LeakyReLU	64	0
Dense	1	65
	Total params:	44,033

(a)

Layer	Output Shape	Param #
Flatten	10	0
Dense	128	1408
LeakyReLU	128	0
Dense	96	12384
LeakyReLU	96	0
Dense	64	6208
LeakyReLU	64	0
Dense	10	650
Dense	64	4160
LeakyReLU	64	0
Dense	96	6240
LeakyReLU	96	0
Dense	128	12416
LeakyReLU	128	0
Dense	10	1290
	Total params:	44,666

(b)

Table 5.6: (a) Dense Lipschitz network architecture. (b) Dense autoencoder architecture.

5.6 Conclusion

In this work, we introduce a Wasserstein-distance based deep anomaly detection framework. We show the advantage of our discriminator-based framework, which learns to reason about the probability density of the nominal data and produces an anomaly score based on distribution distances, as opposed to other deep methods that rely on reconstruction-error criteria. We show reconstruction methods may learn to reconstruct "averaged images" that are within the convex hull of the data. In contrast, LAD provides guarantees on points with low support in the training data. Furthermore, we showed that LAD significantly outperforms existing methods on slightly corrupted training data, which is more realistic than assuming anomaly-free training data in large datasets. We show the benefits of this discriminator in image, tabular health record, and graph domains. In particular on graphs, our encoder only architecture is extremely beneficial as graphs are difficult to decode from vector spaces.

Chapter 6

TrajectoryNet

6.1 Introduction

In data science we are often confronted with cross-sectional samples of time-varying phenomena, especially in biomedical data. Examples include health measurements of different age cohorts [152], or disease measurements at different stages of disease progression [210]. In these measurements we consider data that is sampled at multiple timepoints, but at each timepoint we have access only to a distribution (cross-section) of the population at that time. Extracting the longitudinal dynamics of development or disease from static snapshot measurements can be challenging as there are few methods of interpolation between distributions. Further exacerbating this problem is the fact that the same entities are often not measured at each time, resulting in a lack of point-to-point correspondences. Here, we propose to formulate this problem as one of unbalanced dynamic transport, where the goal is to transport entities from one cross sectional measurement to the next using efficient and smooth paths. Our main contribution is to establish a link between *continuous normalizing flows* (CNF) [81] and dynamic optimal transport [17], allowing us to efficiently solve the transport problem using a Neural ODE framework [38]. To our knowledge, TrajectoryNet[1] is the first method to consider the specific paths taken by a CNF between distributions.

The continuous normalizing flow formulation allows us to generalize optimal transport

[1] Code is available here: `https://github.com/KrishnaswamyLab/TrajectoryNet`

to a series of distributions as in recent work [41, 19]. These works focus on the theoretical aspects of the problem, here focus on the computational aspects. This link allows us to smooth flows over multiple and possibly unevenly spaced distributions in high dimensions. This matches the setting of time series data from single-cell RNA sequencing.

Single-cell RNA sequencing [138] is a relatively new technology that has made it possible for scientists to randomly sample the entire transcriptome, i.e., 20-30 thousand species of mRNA molecules representing transcribed genes of the cell. This technology can reveal detailed information about the identity of individual cells based on transcription factors, surface marker expression, cell cycle and many other facets of cellular behavior. In particular, this technology can be used to learn how cells differentiate from one state to another: for example, from embryonic stem cells to specified lineages such as neuronal or cardiac. However, hampering this understanding is the fact that scRNA-seq only offers static snapshots of data, since all cells are destroyed upon measurement. Thus it is impossible to monitor how an individual cell changes over time. Moreover, due to the expensive nature of this technology, generally only a handful of discrete timepoints are collected in measuring any transition process. TrajectoryNet is especially well suited to this data modality. Existing methods attempt to infer a trajectory within one timepoint [85, 177, 113], or interpolate linearly between two timepoints [219, 181], but TrajectoryNet can interpolate non-linearly using information from more than two timepoints. TrajectoryNet has advantages over existing methods in that it:

1. can interpolate by following the manifold of observed entities between measured timepoints, thereby solving the static-snapshot problem,

2. can create continuous-time trajectories of individual entities, giving researchers the ability to follow an entity in time,

3. forms a deep representational model of system dynamics, which can then be used to understand drivers of dynamics (gene logic in the cellular context), via perturbation of this deep model.

While our experiments apply this work specifically to cellular dynamics, these penalties can be used in many other situations where we would like to model dynamics based on cross-sectional population level data.

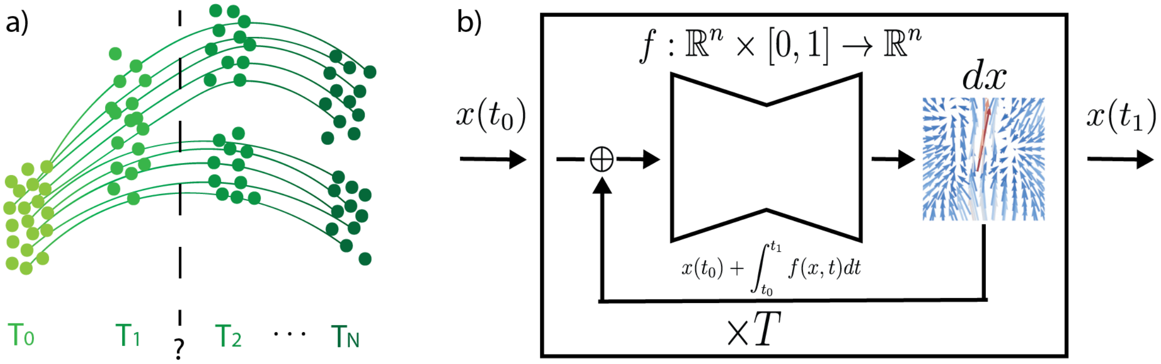

a)
T0
T1
?
T2
···
TN
b)
$f : \mathbb{R}^n \times [0,1] \to \mathbb{R}^n$
dx
$x(t_0)$
$x(t_1)$
$\oplus$
$x(t_0) + \int_{t_0}^{t_1} f(x,t)dt$
$\times T$

Figure 6.1: TrajectoryNet learns trajectories of particles from distributions sampled over time. We use a Neural ODE to learn the derivative of the dynamics function. To find the output at time t_1 for a given input at time t_0 we integrate T times letting the ODE solver choose the integration timepoints.

6.2 Background and Related Work

Optimal Transport. Introduced originally by [145] and in modern form by [100], the linear program formulation of static optimal transport (OT) has the relatively high cost of $O(n^3)$ for discrete measures. Recently, there have been a number of fast approximations using entropic regularization. Cuturi [50] presented a parallel algorithm for the discrete case as an application of Sinkhorn's algorithm [190]. Recent effort approximates OT on subspaces [151] or even a single dimension [110]. These efforts emphasis the importance to the field of obtaining fast OT algorithms. Another direction that has recently received increased attention is in unbalanced optimal transport where the goal is to relax the problem to add and remove mass [16, 44, 129, 181]. While many efficient static optimal transport algorithms exist, and recently for the unbalanced case [219], much less attention has focused on dynamic optimal transport, the focus of this work.

Dynamic Optimal Transport. Another formulation of optimal transport is known as *dynamic* optimal transport. Benamou and Brenier [17] showed how the addition of a natural time interpolation variable gives an alternative interpretation with links to fluid dynamics that surprisingly leads to a convex optimization problem. However, while solvers for the discretized dynamic OT problem are effective in low dimensions and for small problems they require a discretization of space into grids giving cost exponential in the dimension (See Peyré and Cuturi [165] Chap. 7 for a good overview of this problem). One of our main contributions is to provide an approximate solver for high dimensional smooth problems using a neural network.

Single-cell Trajectories from a Static Snapshot. Temporal interpolation in single-cell data started with solutions that attempt to infer an axis within one single time point of data cell "pseudotime" – used as a proxy for developmental progression – using known markers of development and the asynchronous nature of cell development [202, 20]. An extensive comparison of 45 methods for this type of analysis gives method recommendations based on prior assumptions on the general structure of the data [177]. However, these methods can be biased and fail in a number of circumstances [214, 118] and do not take into account experimental time.

Matching Populations from Multiple Time Points. Recent methods get around some of these challenges using multiple timepoints [87, 181, 219]. However, these methods generally resort to matching populations between coarse-grained timepoints, but do not give much insight into how they move between measured timepoints. Often paths are assumed to minimize total Euclidean cost, which is not realistic in this setting. In contrast, the methods that estimate dynamics from single timepoints [113, 22, 63, 89] have the potential to give relatively accurate estimation of local direction, but cannot give accurate global estimation of distributional shift. A recent line of work on generalizing splines to distributions [41, 19] investigates this problem from a theoretical perspective, but provides no efficient implementation.

With TrajectoryNet, we aim to unite these approaches into a single model combining in inferring continuous time trajectories from multiple timepoints, globally, while respecting local dynamics within a single timepoint.

6.3 Preliminaries

We provide an overview of static optimal transport, dynamic optimal transport [17], and continuous normalizing flows.

6.3.1 The Monge-Kantorovich Problem

We adopt notation from the standard text [206]. For two probability measures μ, ν defined on $\mathcal{X} \subset \mathbb{R}^n$, let $\Pi(\mu, \nu)$ denote the set of all joint probability measures on $\mathcal{X} \times \mathcal{X}$ whose marginals are μ and ν. Then the p-Kantorovich distance (or Wasserstein distance of order p)between μ and ν is

$$W(\mu, \nu)_p := \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X} \times \mathcal{Y}} d(x,y)^p d\pi(x,y) \right)^{1/p}, \tag{6.1}$$

where $p \in [1, \infty)$. This formulation has led to many useful interpretations both in GANs and biological networks. For the entropy regularized problem, the Sinkhorn algorithm [190] provides a fast and parallelizable numerical solution in the discrete case. Recent work tackles computationally efficient solutions to the exact problem [97] for the discrete case. However, for the continuous case solutions to the discrete problem in high dimensional spaces do not scale well. As the rate of convergence of the empirical Wasserstein metric between empirical measures $\hat{\mu}$ and $\hat{\nu}$ with bounded support is shown in [58] to be

$$\mathbb{E}[|W_p(\hat{\mu}_n, \hat{\nu}_n) - W_p(\mu, \nu)|] = O(n^{-\frac{1}{d}}) \tag{6.2}$$

where d is the ambient dimension. However, recent work shows that in high dimensions a more careful treatment that the rate depends on the intrinsic dimension not the ambient dimension [213]. As long as data lies in a low dimensional manifold in ambient space, then we can reasonably approximate the Wasserstein distance. In this work we approximate the support of this manifold using a neural network.

6.3.2 Dynamic Optimal Transport

Benamou and Brenier [17] defined and explored a dynamic version of Kantorovich distance.

Their work linked optimal transport distances with dynamics and partial differential equations (PDEs). For a fixed time interval $[t_0, t_1]$ with smooth enough, time dependent density and velocity fields, $P(x,t) \geq 0, f(x,t) \in \mathbb{R}^d$, subject to the continuity equation

$$\partial_t P + \nabla \cdot (Pf) = 0 \tag{6.3}$$

for $t_0 < t < t_1$ and $x \in \mathbb{R}^d$, and the conditions

$$P(\cdot, t_0) = \mu, \quad P(\cdot, t_1) = \nu \tag{6.4}$$

we can relate the squared L^2 Wasserstein distance to (P, f) in the following way

$$W(\mu,\nu)_2^2 = \inf_{(P,f)} (t_1 - t_0) \int_{\mathbb{R}^d} \int_{t_0}^{t_1} P(x,t)|f(x,t)|^2 dt dx \tag{6.5}$$

In other words, a velocity field $f(x,t)$ with minimum L^2 norm that transports mass at μ to mass at ν when integrated over the time interval is the optimal plan for an L^2 Wasserstein distance. The continuity equation is applied over all points of the field and asserts that no point is a source or sink for mass. The solution to this flow can be shown to follow a pressureless flow on a time-dependent potential function. Mass moves with constant velocity that linearly interpolates between the initial and final measures. For problems where interpolation of the density between two known densities is of interest, this formulation is very attractive. Existing computational methods for solving the dynamic formulation for continuous measures approximate the flow using a discretization of space-time [156]. This works well in low dimensions, but scales poorly to high dimensions as the complexity is exponential in the input dimension d. We next give background on continuous normalizing flows, which we show can provide a solution with computational complexity polynomial in d.

6.3.3 Continuous Normalizing Flows

A normalizing flow [170] transforms a parametric (usually simple) distribution to a more complicated one. Using an invertible transformation f applied to an initial latent random variable y with density P_y, We define $x = f(y)$ as the output of the flow. Then by the change of variables formula, we can compute the density of the output x:

$$\log P_x(\cdot) = \log P_y(\cdot) - \log \left| \det \frac{\partial f}{\partial y} \right| \tag{6.6}$$

A large effort has gone into creating architectures where the log determinant of the Jacobian is efficient to compute [170, 105, 157].

Now consider a continuous-time transformation, where the derivative of the transformation is parameterized by θ, thus at any timepoint t, $\frac{\partial x(t)}{\partial t} = f_\theta(x(t), t)$. At the initial time t_0, $x(t_0)$ is drawn from a distribution $P(x, t_0)$ which we also denote $P_{t_0}(x)$ for clarity, and it's continuously transformed to $x(t_1)$ by following the differential equation $f_\theta(x(t), t)$:

$$\begin{aligned} & x(t_1) = x(t_0) + \int_{t_0}^{t_1} f_\theta(x(t), t) dt, \quad x(t_0) \sim P_{t_0}(x), \\ & \log P_{t_1}(x(t_1)) = \\ & \qquad \log P_{t_0}(x(t_0)) - \int_{t_0}^{t_1} Tr \left(\frac{\partial f_\theta(x(t), t)}{\partial x(t)} \right) dt, \end{aligned} \tag{6.7}$$

where at any time t associated with every x through the flow can be found by following the inverse flow. This model is referred as continuous normalizing flows (CNFs) [38]. It can be likened to the dynamic version of optimal transport, where we model the measure over time rather than the mapping from P_{t_0} to P_{t_1}

Unsurprisingly, there is a deep connection between CNFs and dynamic optimal transport. In the next section we exploit this connection and show how CNFs can be used to provide a high dimensional solution to the dynamic optimal transport problem with TrajectoryNet.

6.4 TrajectoryNet: Efficient Dynamic Optimal Transport

In this section, we first describe how to adapt continuous normalizing flows to approximate dynamic optimal transport in (Section 6.4.1). We then describe further adaptations for analysis of single-cell data in (Section 6.4.2) and finally provide training details in (Section

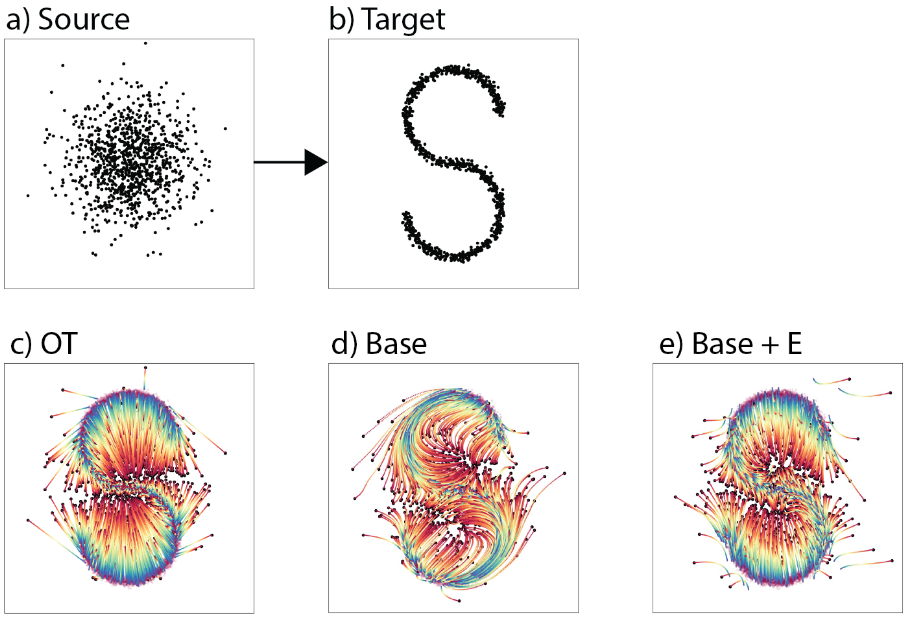


Figure 6.2: Transporting a Gaussian (a) to an S-curve (b) via (c) static optimal transport, (d) Base TrajectoryNet without regularization follows density (e) TrajectoryNet with energy regularization demonstrates more straight paths similar to OT.

6.4.3).

6.4.1 Dynamic OT Approximation via Regularized CNF

Continuous normalizing flows use a maximum likelihood objective which can be equivalently expressed as a KL divergence. In TrajectoryNet we add an energy regularization to approximate dynamic OT. Dynamic OT is expressed with an optimization over flows with constraints at t_0 and t_1 (see eq. equation 6.4). By relaxing this constraint to minimizing a divergence at t_1 CNFs can approximate dynamic OT.

For sufficiently large λ under constraint equation 6.3 this converges to the optimal solution in equation 6.5. This is encapsulated in the following theorem. See Appendix 6.A.1 for proof.

Theorem 6.4.1. *With time varying field* $f(x,t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ *and density* $P(x,t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^+$ *such that* $\int P(x,t)dx = 1$ *for all* $t_0 \leq t \leq t_1$ *and subject to the continuity* *equation 6.3. There exists a sufficiently large λ such that*

$$W(\mu,\nu)_2^2 = (t_1 - t_0) \inf_{(P,f)} \mathop{\mathbb{E}}_{x_0\sim\mu} \left[\int_{t_0}^{t_1} \|f(x(t),t)\|^2 \, dt \right]$$
$$+ \lambda KL(P(\cdot,t_1) \parallel \nu); \quad s.t.\ P(\cdot,t_0) = \mu$$

Intuitively, a continuous normalizing flow with a correctly scaled penalty on the squared norm of f approximates the W_2 transport between μ and ν. Dynamic optimal transport can be thought of as finding a distribution over paths such that the beginnings of the paths match the source distribution, end of the path matches the target distribution, and the cost of the transport is measured by expected path length. Continuous normalizing flows relax the target distribution match with a KL-divergence penalty. When the KL-divergence is small then the constraint is satisfied. However, CNFs usually do not enforce the path length constraint, which we add using a penalty on the norm of f as defined by the Neural ODE. When we impose this penalty over uniformly sampled data, this is equivalent to penalizing the expected path length.

We can approximate the first part of this continuous time equation using a Riemann sum as $\mathbb{E}_{x_0\sim\mu}\left[\int_{t_0}^{t_1} \|f_\theta(x,t)\|^2 \, dt\right] = \sum_{x\sim P_{t_0}} \sum_{i=t_0}^{t_1} \Delta t_i \|f_\theta(x,t)\|^2$. This requires a forward integration using a standard ODE solver to compute as shown in Chen et al. [38]. If we consider the case where the divergence is small, then this can be combined with the standard backwards pass for even less added computation. Instead of penalizing $\|f_\theta(x,t)\|^2$ on a forward pass, we penalize the same quantity on a backwards pass. Using the maximum likelihood and KL divergence equivalence, we obtain a loss

$$L(x) = -\log p(x) + \lambda_e \int_t \|f(x(t),t)\|^2 \tag{6.8}$$

Where the integral above is computed using an ODE solver. In practice, both a penalty on the Jacobian or additional training noise helped to get straight paths with a lower energy regularization λ_e. We found that a value of λ_e large enough to encourage straight paths, unsurprisingly also shortens the paths undershooting the target distribution. To counteract this, we add a penalty on the norm of the Jacobian of f as used in Vincent et al. [208], Rifai et al. [171]. Since f represents the derivative of the path, this discourages paths with high local curvature, and can be thought of as penalizing the second derivative (acceleration) of the flow. Our energy loss is then

$$L_{energy}(x) = \lambda_e \int_t \|f(\tilde{x}, t)\|^2 + \lambda_j \int_t \|J_f(\tilde{x})\|_F^2, \tag{6.9}$$

where $\|J_f(x)\|_F^2$ is the Frobenius norm of the Jacobian of f. Comparing Figure 6.2(e) to (d) we demonstrate the effect of this regularization. Without energy regularization TrajectoryNet paths follow the data. However, with energy regularization we approach the paths of the optimal map. TrajectoryNet solution biases towards undershooting the target distribution. Our energy loss gives control over how much to penalize indirect, high energy paths.

Optimal transport is traditionally performed between a source and target distribution. Extensions to a series of distributions is normally done by performing optimal transport between successive pairs of distributions as in Schiebinger et al. [181]. This creates flows that have discontinuities at the sampled times, which may be undesirable when the underlying system is smooth in time as in biological systems. The dynamic model approximates dynamic OT for two timepoints, but by using a single smooth function to model the whole series the flow becomes the minimal cost *smooth* flow over time.

6.4.2 Further Adaptation for Single-Cell Trajectories

Up to this point we have shown how to perform dynamic optimal transport in high dimensions with a regularized CNF. We now introduce priors needed to mimic cellular systems that are characterized by growth/death rather than just transport, endowed with a manifold structure, and knowledge of local velocity arrows. Similar priors may also be applicable to other data types. For example in studying the dynamics of a disease, people may be newly infected or cured, we may have knowledge on acceptable transition states, indicating a density penalty, or visits may be clustered such as in a hospital stay so we may have estimates of near term patient trends, indicating the use of velocity priors. To enforce these priors we add corresponding regularizations listed below:

1. A *growth rate regularization* that accommodates unbalanced transport, described in Section 6.4.2.

2. A *density-based penalty* which encourages interpolations that lie on dense regions of the data. Often data lies on a low-dimensional manifold, and it is desirable for paths to follow this manifold at the cost of higher energy (See Section 6.4.2 for details).

3. A *velocity regularization* where we enforce local estimates of velocity at measured datapoints to match the first time derivative of cell state change. (See Section 6.4.2 for details).

These regularizations are summarized in a single loss function defined as

$$\begin{aligned} L_T = &\underbrace{\overbrace{\sum_{i=1}^{k} - \log P_{t_i}(x_{t_i})}^{\text{Normalizing Flow}} + L_{energy}}_{\text{Dynamic OT}} \\ &+ \underbrace{L_{density} + L_{velocity} + L_{growth}}_{\text{Biological priors}} \end{aligned} \tag{6.10}$$

Allowing Unbalanced Optimal Transport

We use a simple and computationally efficient method that adapts discrete static unbalanced optimal transport to our framework in the continuous setting. This is a necessary extension but is by no means a focus of our work. While we could also apply an adversarial framework, we choose to avoid the instabilities of adversarial training and use a simple network trained from the solution to the discrete problem. We train a network $G(x, t) : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^+$, which takes as input a cell state and time pair and produces a growth rate of a cell at that time. This is trained to match the result from discrete optimal transport. For further specification see Appendix 6.B. We then fix weights of this network and modify the way we integrate mass over time to

$$\log M_{t_i}(x) = \log M_{t_{i-1}}(x) - \int_{t_{i-1}}^{t_i} Tr\left(\frac{\partial f_\theta(x(t), t)}{\partial x(t)}\right) dt \\ + \log G(x_{t_{i-1}}, t_{i-1}) \tag{6.11}$$

We note that adding growth rate regularization in this way does not guarantee conservation of mass. We could normalize $M(x)$ to be a probability distribution during training, e.g., as $P(x) = M(x)/\sum_{x \in \mathbb{R}^d} M(x)$. However, this now requires an integration over $\mathbb{R}^d$, which is too computationally costly. Instead, we use the equivalence of the maximum likelihood formulation over a fixed growth function g and normalize it after the network is trained.

Enforcing Transport on a Manifold

Methods that display or perform computations on the cellular manifold often include an implicit or explicit way of normalizing for density and model data geometry. PHATE [148] uses scatter plots and adaptive kernels to display the geometry. SUGAR [130] explicitly models the data geometry. We would like to constrain our flows to the manifold geometry but not to its density. We penalize the flow such that it is always close to at least a few measured points across all timepoints.

$$L_{density}(x, t_d) = \\ \sum_k \max(0, \text{min-k}\big(\{\|x(t_d) - z\| : z \in \mathcal{X}\}\big) - h) \tag{6.12}$$

This can be thought of as a loss that penalizes points until they are within h Euclidean distance of their k nearest neighbors. We use $h = 0.1$ and $k = 5$ in all of our experiments. We evaluate $L_{density}$ on an interpolated time $t_d \in (t_0, t_k)$ every batch.

Conforming to known Velocity

Often it is the case where it is easy to measure direction of change in a short time horizon, but not have good predictive power at the scale of measured timesteps. In health data, we can often collect data from a few visits over a short time horizon estimating the direction of a single patient in the near future. In single-cell data, RNA-velocity [113, 22] provides an estimates $\widehat{dx/dt}$ at every measured cell. We use these measurements to regularize the direction of flow at every measured point. Our regularization requires evaluating $f(x,t)$ periodically at every measured cell adding the regularization:

$$\begin{aligned} L_{velocity}(x, t, \widehat{dx/dt}) &= \text{cosine-similarity}(f(x,t), \widehat{dx/dt}) \\ &= \frac{f(x,t) \cdot \widehat{dx/dt}}{\|f(x,t)\| \left\|\widehat{dx/dt}\right\|} \end{aligned} \tag{6.13}$$

This encourages the direction of the flow at a measured point to be similar to the direction of local velocity. This ignores the magnitude of the estimate, and only heeds the direction. While RNA-velocity provides some estimate of relative speed, the vector length is considered not as informative, as it is unclear how to normalize these vectors in a system specific way [113, 22]. We note that while current estimates of velocity can only estimate direction, this does not preclude future methods that can give accurate magnitude estimates. $L_{velocity}$ can easily be adapted to take magnitudes into account by considering L_2 similarity for instance.

6.4.3 Training

For simplicity, the neural network architecture of TrajectoryNet consists of three fully connected layers of 64 nodes with leaky ReLU activations. It takes as input a cell state and time and outputs the derivative of state with respect to time at that point. To train a continuous normalizing flow we need access to the density function of the source distribution. Since this is not accessible for an empirical distribution we use an additional Gaussian at t_0, defining $P_{t_0}(\cdot) = \mathcal{N}(0,1)$, the standard Gaussian distribution, where $P_t(x)$ is the density function at time t.

For a training step we draw samples $x_{t_i} \sim \mathcal{X}_{t_i}$ for $i \in \{1, \dots, k\}$ and calculate the loss with a single backwards integration of the ODE. In the following sections we will explain how adding the individual penalty terms achieve regularized trajectories. While there are a number of ways to computationally approximate these quantities, we use a parallel method to iteratively calculate the $\log P_{t_i}$ based on $\log P_{t_{i-1}}$. To make a backward pass through all timepoints we start at the final timepoint, integrate the batch to the second to last timepoint, concatenate these points to the samples from the second to last timepoint, and continue till t_0, where the density is known for each sample. We note that this can compound the error especially for later timepoints if k is large or if the learned system is stiff, but gives significant speedup during training.

To sample from P_{t_i} we first sample $\hat{x}_{t_0} \sim P_{t_0}$ then use the adjoint method to perform the integration $\hat{x}_{t_i} = \hat{x}_{t_0} + \int_{t_0}^{t_i} f_\theta(x(t), t)dt; \quad x(0) = \hat{x}_{t_0}$.

6.5 Experiments

All experiments were performed with the TrajectoryNet framework with a network of consisting of three layers with LeakyReLU activations. Optimization was performed on 10,000 iterations of batches of size 1,000 using the dopri5 solver [57] with both absolute and relative tolerances set to 1×10^{-5} and the ADAM optimizer [103] with learning rate 0.001, and weight decay 5×10^{-5} as in [81]. We evaluate using three TrajectoryNet models with different regularization terms. The Base model refers to a standard normalizing flow. +E adds L_{energy}, +D adds $L_{density}$,+V adds $L_{velocity}$, and +G adds L_{growth}.

Comparison to Existing Methods. Since there are no ground truth methods to calculate the trajectory of a single cell we evaluate our model using interpolation of held-out timepoints. We leave out an intermediary timepoint and measure the Kantorovich-distance also known as the earth mover's distance (EMD) between the predicted and held-out distributions. For EMD lower is more accurate. We compare the distribution interpolated by TrajectoryNet with four other distributions. The previous timepoint, the next timepoint, a random timepoint and the McCann interpolant in the discrete OT solution as used in [181].

6.5.1 Artificial Data

For artificial data where we have known paths, we can measure the mean squared error (MSE) predicted by the model based on the first timepoint. Here we leave out the middle

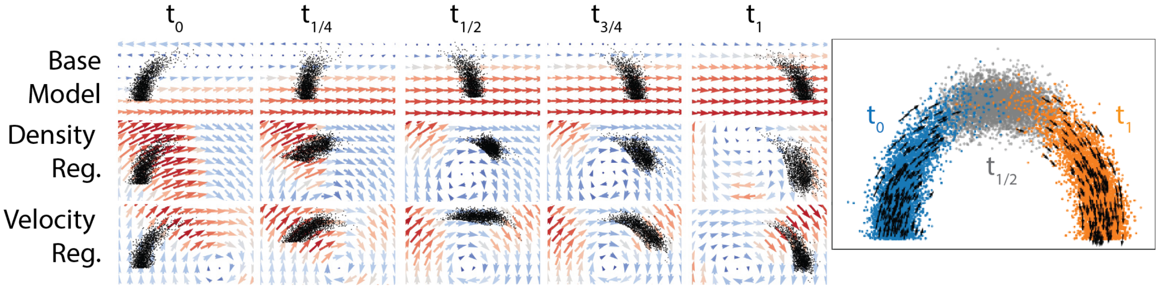


Figure 6.3: Density regularization or velocity regularization can be used to follow a 1D manifold in 2D.

	EMD			MSE		
	Arch	Cycle	Tree	Arch	Cycle	Tree
Base	0.691	0.037	0.490	0.300	0.190	0.218
Base + D	0.607	0.049	0.373	0.236	0.191	0.145
Base + V	**0.243**	0.033	**0.143**	**0.107**	**0.068**	**0.098**
Base + D + V	0.415	0.034	0.252	0.156	0.081	0.132
OT	0.644	**0.032**	0.492	0.252	0.192	0.196
prev	1.086	0.035	1.092	0.652	0.192	0.666
next	1.090	0.035	1.068	0.659	0.192	0.689
rand	0.622	0.406	0.420	0.243	0.346	0.161

Table 6.1: Shows the Wasserstein distance EMD and MSE for artificial datasets between the left out timepoint and the predicted points for our two generated datasets. Mean over 3 seeds.

timepoint $t_{1/2}$ for training then calculate the MSE between the predicted point at time $t_{1/2}$ and the true point at $t_{1/2}$ for 5000 sampled trajectories. This gives a measure of how accurately we can model simple dynamical systems.

We first test TrajectoryNet on two datasets where points lie on a 1D manifold in 2D with Gaussian noise (See Figure 6.4). First two half Gaussians are sampled with means zero and one in one dimension. These progressions are then lifted onto curved manifolds in two dimensions either an arch or a tree mimicking a differentiating system where we have two sampled timepoints that have some overlap. Table 6.1 shows the Wasserstein distance (EMD) and the mean squared error for different interpolation methods between the interpolated distribution at $t_{1/2}$ and the true interpolated distribution at $t_{1/2}$. Because optimal transport considers the shortest Euclidean distance, the base model and OT methods follow the lowest energy path, which is straight across. With density regularization or velocity regularization TrajectoryNet learns paths that follow the density manifold. Figure 6.3 and

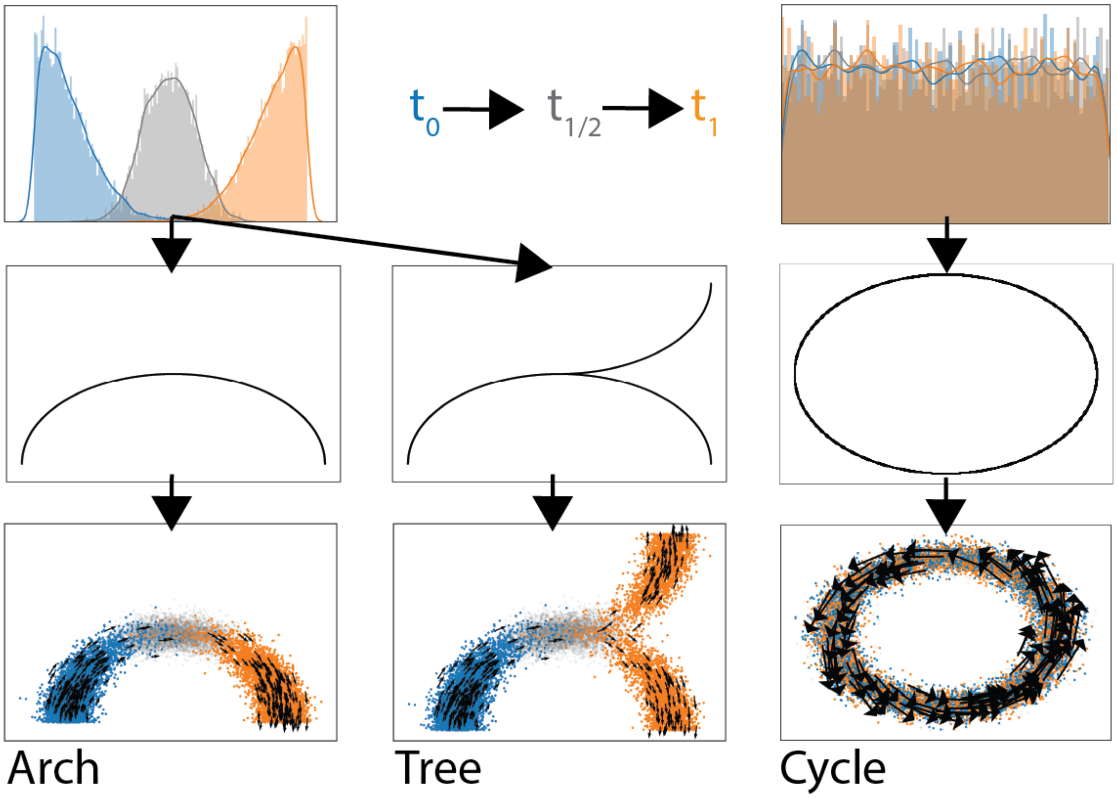


Figure 6.4: A 1D distribution of data over time embedded in two dimensions along a smooth manifold. On a single branch (left), with a tree structure (center), and circle (right).

Figure 6.10 demonstrate how TrajectoryNet with density or velocity regularization learns to follow the manifold.

A third artificial dataset shows the necessity of using velocity estimates for some data. Here we have an unchanging distribution of points distributed uniformly over the unit circle, but are traveling counterclockwise at $\pi/5$ radians per unit time. This is similar to the cell-cycle process in adult systems. Without velocity estimates it is impossible to pick up this type of dynamical system. This is illustrated by the MSE of the cycle dataset using velocity regularization in Table 6.1.

6.5.2 Single-Cell Data

We run our model on 5D PCA due to computational constraints, but note that computation time scales roughly linearly with dimension for our test cases (See Appendix 6.C), which

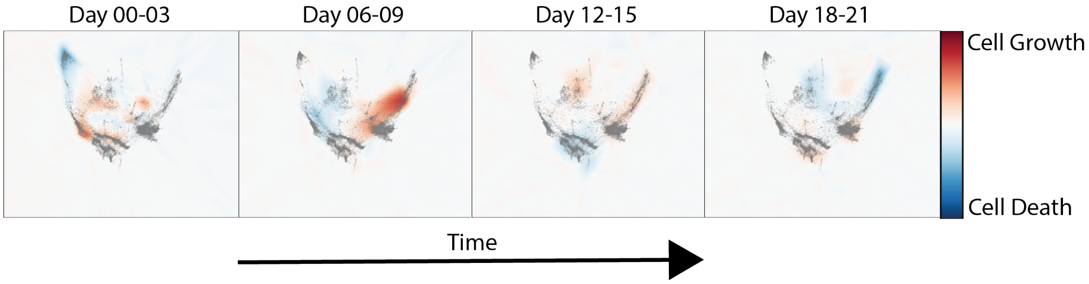


Figure 6.5: Cell growth model learned on Embryoid Body Data [148]

is consistent to what was found in Grathwohl et al. [81]. Since there are no ground truth trajectories in real data, we can only evaluate using distributional distances. We do leave-one-out validation, training the model on all but one of the intermediate timepoints then evaluating the EMD between the validation data and the model's predicted distribution. We evaluate and compare our method on two single-cell RNA sequencing datasets.

	rep1	rep2	mean
Base	0.888 ± 0.07	0.905 ± 0.06	0.897 ± 0.06
Base + D	0.882 ± 0.03	0.895 ± 0.03	0.888 ± 0.03
Base + V	0.900 ± 0.09	0.898 ± 0.10	0.899 ± 0.10
Base + D + V	**0.851** ± 0.08	**0.866** ± 0.07	**0.859** ± 0.07
OT	1.098	1.095	1.096
prev	1.628	1.573	1.600
next	1.324	1.391	1.357
rand	1.333	1.288	1.311

Table 6.2: Shows the Wasserstein distance between the left out timepoint and the predicted distribution for various methods on a 4 timepoint mouse embryo cortex dataset. Mean and standard deviation over 3 seeds.

Mouse Cortex Data. [2] The first dataset has structure similar to the Arch toy dataset. It consists of cells collected from mouse embryos at days E12.5, E14.5, E16, and E17.5. In Figure 6.6(d) we can see at this time in development of the mouse cortex the distribution of cells moves from a mostly neural stem cell population at E12.5 to a fairly developed and differentiated neuronal population at E17.5 [48, 101]. The major axis of variation is neuron development. Over the 4 timepoints we have 2 biological replicates that we can

[2] For videos of the dynamics learned by TrajectoryNet see `http://github.com/krishnaswamylab/TrajectoryNet`

use to evaluate variation between animals. In Table 6.2, we can see that TrajectoryNet outperforms baseline models, especially when adding density and velocity information. The curved manifold structure of this data, and gene expression data in general means that methods that interpolate with straight paths cannot fully capture the structure of the data. Since TrajectoryNet models full paths between timepoints, adding density and velocity information can bend the cell paths to follow the manifold utilizing all available data rather than two timepoints as in standard optimal transport.

Embryoid body Data. Next, we evaluate on a differentiating Embryoid body scRNA-seq time course. Figure 6.7 shows this data projected into two dimensions using a non-linear dimensionality reduction method called PHATE [148]. This data consists of 5 timepoints of single cell data collected in a developing human embryo system (Day 0-Day 24). See Figure 6.5 for a depiction of the growth rate. Initially, cells start as a single stem cell population, but differentiate into roughly 4 cell precursor types. This gives a branching structure similar to our artificial tree dataset. In Table 6.3 we show results when each of the three intermediate timepoints are left out. In this case velocity regularization does not seem to help, we hypothesis this has to do with the low unspliced RNA counts present in the data (See Figure 6.11). We find that energy regularization and growth rate regularization help only on the first timepoint, and that density regularization helps the most overall.

We can also project trajectories back to gene space. This gives insights into when populations might be distinguishable. In Figure 6.8, we demonstrate how TrajectoryNet can be projected back to the gene space. We sample cells from the end of the four main branches, then integrate TrajectoryNet backwards to get their paths through gene space. This recapitulates known biology in Moon et al. [148]. See appendix 6.D for a more in-depth treatment.

6.6 Conclusion

TrajectoryNet computes dynamic optimal transport between distributions of samples at discrete times to model realistic paths of samples continuously in time. In the single-cell

	t=1	t=2	t=3	mean
Base	0.764	0.811	0.863	0.813
Base + D	0.759	**0.783**	0.811	**0.784**
Base + V	0.816	0.839	0.865	0.840
Base + D + V	0.930	0.806	**0.810**	0.848
Base + E	. 0.737	0.896	0.842	0.825
Base + G	**0.700**	0.913	0.829	0.814
OT	0.791	0.831	0.841	0.821
prev	1.715	1.400	0.814	1.309
next	1.400	0.814	1.694	1.302
rand	0.872	1.036	0.998	0.969

Table 6.3: Shows the Wasserstein distance (EMD) between the left out timepoint and the predicted distribution for various methods on the 5 timepoint Embryoid body dataset.

case, TrajectoryNet "reanimates," cells which are destroyed by measurement to recreate a continuous-time trajectory. This is also relevant when modeling any underlying system that is high-dimensional, dynamic, and non-linear. In this case, existing static OT methods are under-powered and do not interpolate well to intermediate timepoints between measured ones. Existing dynamic OT methods (non-neural network based) are computationally infeasible for this task.

In this work we integrate multiple priors and assumptions into one model to bias TrajectoryNet towards more realistic dynamic optimal transport solutions. We demonstrated how this gives more power to discover hidden and time specific relationships between features. In future work, we would like to consider stochastic dynamics [125] and learning the growth term together with the dynamics.

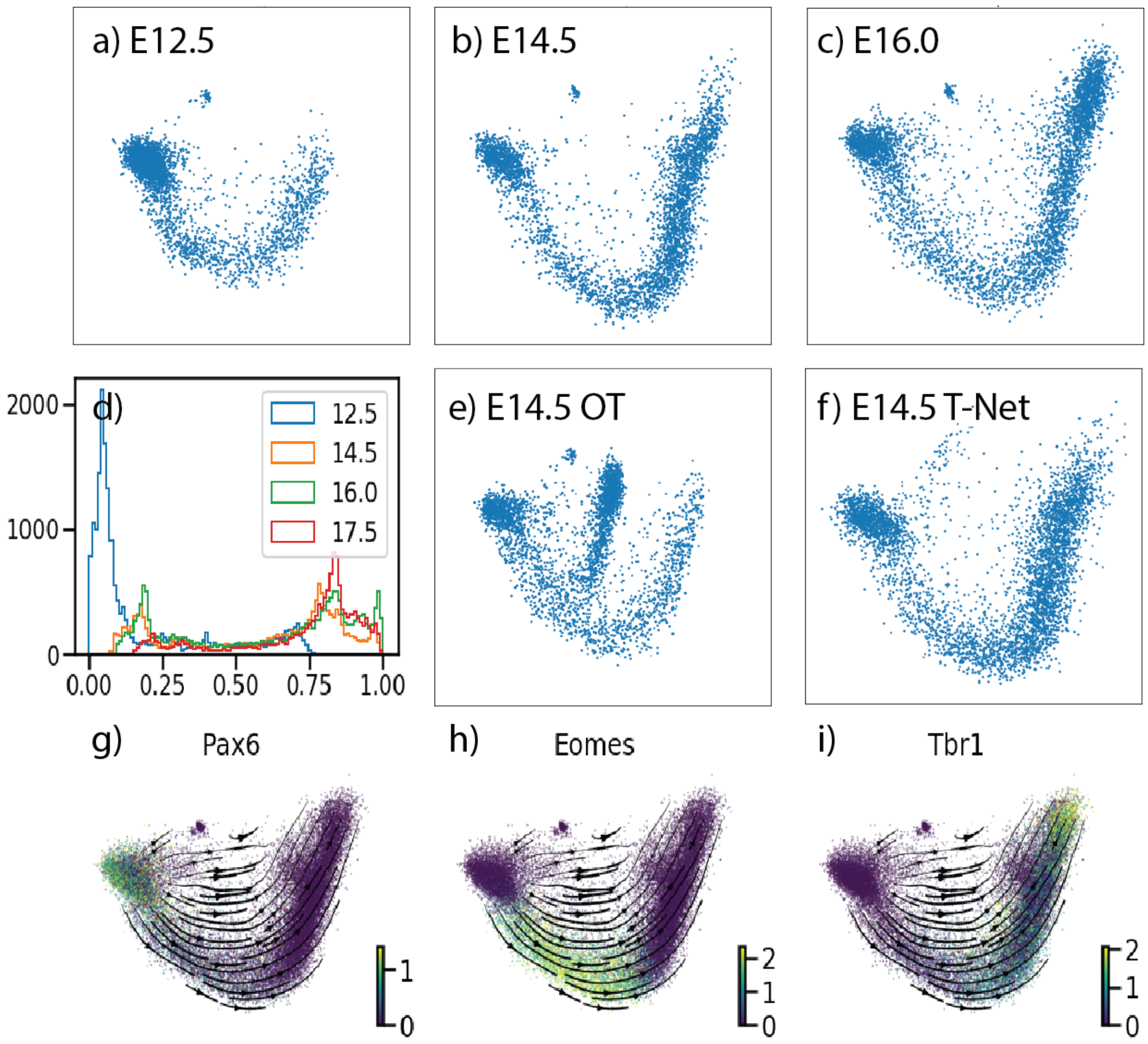


Figure 6.6: Shows the first 2 PCs of the mouse cortex dataset. (a-c) show the distributions for the first three timepoints. (d) shows the distribution of cells over PC1. the interpolated points for E14.5 using (e) static OT, and (f) TrajectoryNet with density regularization. (g-i) shows expression of three markers of early (Pax6) mid (Eomes) and late (Tbr1) stage neurons.

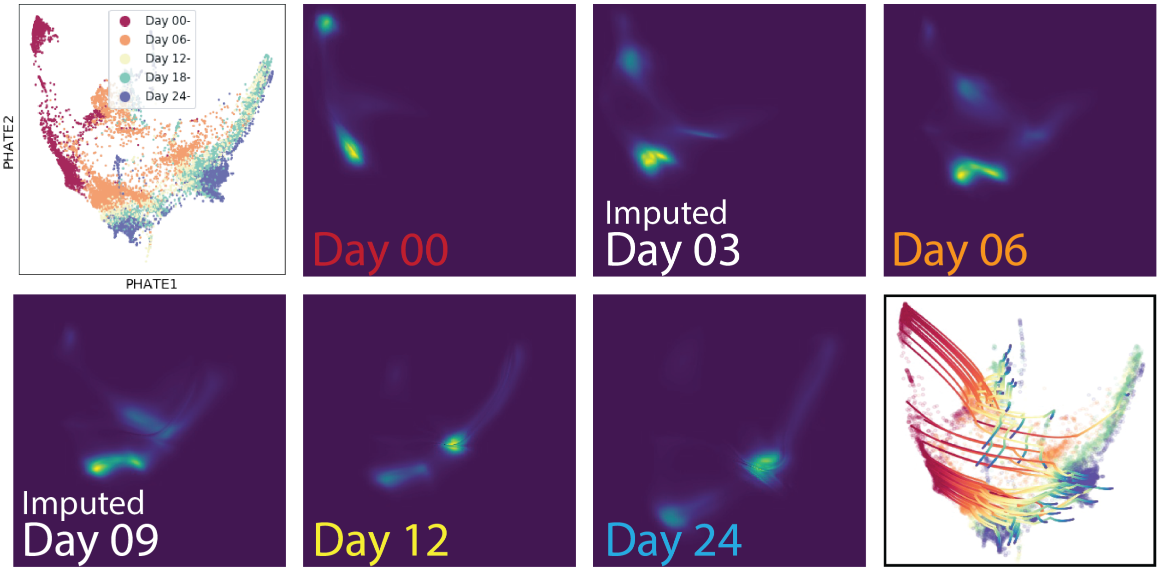


Figure 6.7: Shows the Embryoid body dataset projected into 2D with PHATE [148] with paths and densities imputed using TrajectoryNet.

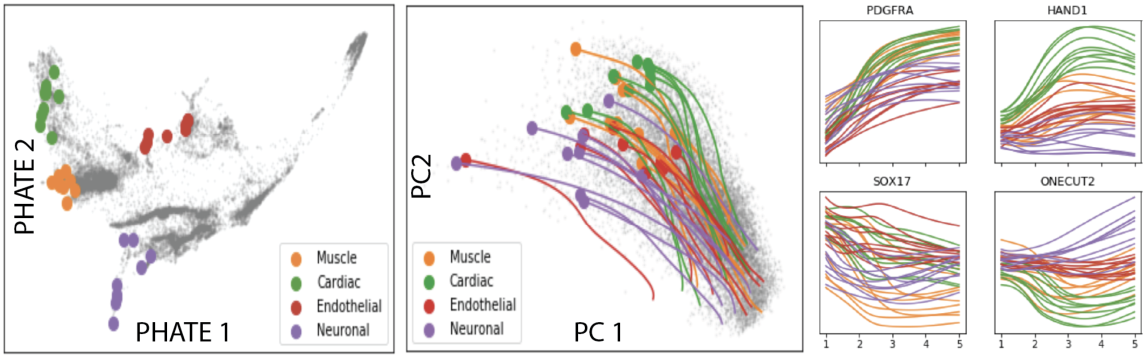


Figure 6.8: For curated endpoints, shows location on PHATE dimensions, TrajectoryNet paths projected into PCA space, and trajectories for 4 genes.

Appendix

6.A Technical Details

6.A.1 Proof of Theorem 6.4.1

First, we apply the Lagrange multiplier method by introducing the variable λ to the minimization problem of equation 6.5 subject to constraints equation 6.4. As we always begin with the base distribution and at any time t, x is defined by $f(x,t)$ and the initial value $x(t_0) = x_0$, which has $\mathrm{KL}(P(t_0,\cdot) \parallel \mu) = 0$.

$$\begin{aligned} \inf_{(P,f)} \sup_{\lambda} (t_1 - t_0) \mathop{\mathbb{E}}_{x_0 \sim \mu} \int_{t_0}^{t_1} P(x,t) \left|f(x,t)\right|^2 dt \\ + \lambda \mathrm{KL}(P(t_1,\cdot) \parallel \nu) \end{aligned} \tag{6.14}$$

The expectation part is equivalent to $\int_{\mathbb{R}^d} \int_{t_0}^{t_1} P(x,t) \left|f(x,t)\right|^2 dtdx$, and we use interchangeably in the the remaining of the proof. Since the KL divergences are non-negative, $\lambda \geq 0$. The optimal solution of the min-max problem is the optimal solution to the original problem. Consider the true minimal loss given by the optimal solution to be c, we know that $L(\lambda) \leq c$, where

$$\begin{aligned} L(\lambda) = \sup_{\lambda \geq 0} \inf_{(P,f)} (t_1 - t_0) \mathop{\mathbb{E}}_{x_0 \sim \mu} \int_{t_0}^{t_1} P(x,t) \left|f(x,t)\right|^2 dt \\ + \lambda \mathrm{KL}(P(t_1,\cdot) \parallel \nu) \end{aligned} \tag{6.15}$$

In order to show that the solution of the max-min problem converges to c, we first show that it is monotonic in λ. For easier reading, set $E = \int_{\mathbb{R}^d} \int_{t_0}^{t_1} P(x,t) \left|f(x,t)\right|^2 dtdx$, $M = \mathrm{KL}(P(t_1, \cdot) \parallel \nu)$. Both E and M are functions of f. For any pair of E, M values, if $\lambda_1 > \lambda_2$, $E + \lambda_1 M > E + \lambda_2 M$. Thus the maximum and minimum of the function $L = E + \lambda M$ is also monotonic in λ, and it will converge to the supremum.

Next, we show that the divergence term $M(f)$ decreases monotonically as λ increases, and it converges to 0 as λ goes to infinity. For a given λ, let $f^*_\lambda = \arg\inf E(f) + \lambda M(f)$. By definition, $E(f^*_{\lambda_1}) + \lambda_1 M(f^*_{\lambda_1}) \leq E(f^*_{\lambda_2}) + \lambda_1 M(f^*_{\lambda_2})$, and $E(f^*_{\lambda_1}) + \lambda_2 M(f^*_{\lambda_1}) \geq E(f^*_{\lambda_2}) + \lambda_2 M(f^*_{\lambda_2})$. Thus $(\lambda_1 - \lambda_2)(M(f^*_{\lambda_1}) - M(f^*_{\lambda_2})) \leq 0$. If $\lambda_1 > \lambda_2$,then $M(f^*_{\lambda_1}) - M(f^*_{\lambda_2}) \leq 0$. The sequence $M(f)$ decreases monotonically as λ increases. Because $L(\lambda)$ is upper bounded by c and $M(f) \geq 0$, $M(f)$ converges to zero as λ goes to infinity.

Now we have shown that $L(\lambda)$ is a monotone sequence and is upper bounded by c, and that the divergence term M converges to zero, we next show that L converges to c, and that the optimal solution of the max-min problem in equation 6.15 is the optimal solution of the original problem. Since the divergence term is non-negative, we have a lower bound for $L(\lambda)$ as

$$\begin{aligned} \forall \lambda, \quad L(\lambda) \geq \inf E(f) & \\ \text{s.t.} \quad M(f) \leq M(f^*_\lambda) & \end{aligned} \tag{6.16}$$

Because $L(\lambda)$ is monotonically increasing, and $M(f)$ is monotonically decreasing, $H(\lambda) = \inf E(f)$ increases as λ increases.

Lemma 6.A.1.

$$\begin{aligned} \forall \epsilon > 0, \quad \forall f, \quad s.t. \quad M(f) \leq \epsilon, & \\ \exists \hat{f}, \quad s.t. \quad M(\hat{f}) = 0, \quad and & \\ E(\hat{f}) - E(f) \leq & \frac{D^2}{\sqrt{2T}} \sqrt[4]{\epsilon}(1 + \sqrt[4]{\epsilon}) \end{aligned}$$

where D is the diameter of the probability space and T is the transformation completion time.

Proof. For a certain λ, starting from the base distribution μ, at time T the distribution is transformed, by following f, to ν', and $\mathrm{KL}(\nu' \parallel \nu) = \epsilon$. Now consider a different transformation $\hat{f}$, which is composed of two part: the first part is an accelerated f, so that ν' is achieved by time $T/(1+\xi)$, and the second part is transforming ν' to ν in the remaining time of $\frac{\xi T}{(1+\xi)}$. Thus at time T, by following $\hat{f}$, we achieve zero divergence, $M(\hat{f}) = 0$. The new transformation $\hat{f}$ has an increased E, from the acceleration and the additional transformation.

$$\begin{aligned} E(\hat{f}) = & \frac{T}{1+\xi} \int_{\mathbb{R}^d} \int_0^{\frac{T}{1+\xi}} P\left(x, \left(1+\xi\right)t\right) \\ & \qquad |(1+\xi)f(x,(1+\xi)t)|^2 dtdx \\ & + \int_Z \int_Y \int_{\frac{T}{1+\xi}}^{T} \mathcal{M}(y,z) \frac{|z-y|^2}{(\frac{\xi}{1+\xi}T)^2} dtdydz \end{aligned} \tag{6.17}$$

where Z has distribution ν, Y has distribution ν', and $\mathcal{M}(y,z)$ is a mapping from Y to Z. The first part of $E(\hat{f})$ is just $E(f)$. The second part is upper bounded by $\frac{TV(Y,Z)D^2}{\frac{\xi}{1+\xi}T}$, where $TV(Y,Z)$ is the total variation between Y and Z, which is in turn upper bounded by $\sqrt{\epsilon/2}$. We choose $\xi = \sqrt[4]{\epsilon}$. □

By definition, $E(\hat{f}) \geq c$, as c is the infimum at zero divergence. Assuming $L(\lambda)$ converges to $c - \alpha$, and $\alpha > 0$, $c - \alpha \geq E(f)$. Then by Lemma 6.A.1, $\forall \epsilon, \quad \alpha < \frac{D^2}{\sqrt{2}T} \sqrt[4]{\epsilon}(1 + \sqrt[4]{\epsilon})$, and we have a contradiction. Now we complete the proof that $L(\lambda)$ converges to c and that for a large enough λ, the solution of the max-min problem equation 6.15 is the solution of equation 6.5 when subject to conditions equation 6.4.

6.B Growth Rate Model Training

Our growth network $G(x,t)$ is trained to match the discrete unbalanced optimal transport problem with entropic regularization:

$$\begin{aligned}\gamma = \text{argmin}_\gamma &< \gamma, M >_F + \lambda\big(\sum_{i,j} \gamma_{i,j}\log(\gamma_{i,j})\big) \\ &+ \alpha KL(\gamma 1, \mu) + \beta KL(\gamma^T 1, \nu) \\ &\text{s.t. } \gamma \geq 0\end{aligned} \tag{6.18}$$

Where $< \cdot, \cdot >_F$ is the Frobenius norm of elementwise matrix multiplication of the transportation matrix γ and the cost matrix M, and where λ, α, β are regularization constants > 0 on the source and target unbalanced distributions μ, ν. Then the growth rate of each cell i in μ to ν is then

$$g_i = \gamma_{(i,\cdot)}1 \tag{6.19}$$

In our experiments we set $\lambda = 0.1, \beta = 10000$ and tune α for reasonable growth rates. This gives a growth rate at every observed cell; however, our model needs a growth rate defined continuously at every measured timepoint. For this, we learn a neural network that is trained to match the growth rate at measured cells and equal to one at negative sampled points. We use a simple form of negative sampling, for each batch of real points we sample an equal sized batch of points from a uniform distribution over the [-1,1] hypercube, where these negative points are given a growth rate value of 1. The network is trained with mean squared error loss to match these growth rates at all measured times.

6.C Scaling with Dimension

Runtime Considerations. Existing numerical methods for solving dynamic OT rely on proximal splitting methods over a discretized staggered grid [17, 156, 165]. This results in a non-smooth but convex optimization problem over these grid points. However, the number of grid points scales exponentially with the dimension, so these methods are only applicable in low dimensions. TrajectoryNet scales polynomially with the dimension. See Figure 6.9 for empirical measurements.

To test the computation time with dimension we run TrajectoryNet for 100 batches of 1000 points on the mouse cortex dataset over different dimensionalities. For hardware we use a single machine with An AMD Ryzen Threadripper 2990WX 32-core Processor, 128GB of memory, and three Nvidia TITAN RTX GPUs. Our model is coded in the Pytorch framework [158]. We count the total number of function evaluations (both forward and backward) divide the total time by this. In Figure 6.9, you can see the seconds per evaluation is roughly linear with the dimensionality of the data. This does not imply convergence of the model is linear in dimension, only that computation per iteration is linear. As suggested in Grathwohl et al. [81], number of iterations until convergence is a function of how complicated the distributions are, and less dependent on the ambient dimension itself. By learning flows along a manifold with $L_{density}$, our method may scale closer to the intrinsic dimensionality of the data rather than the ambient.

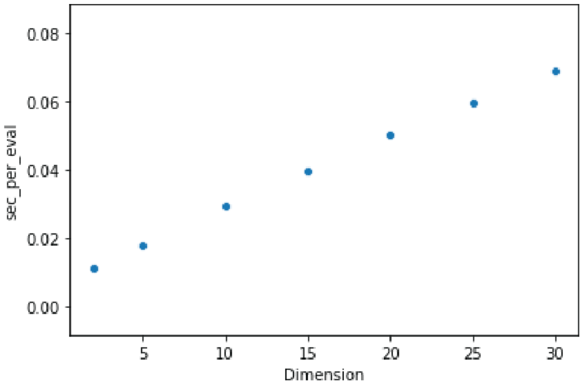


Figure 6.9: The computation per evaluation is roughly linear in terms of dimension.

6.D Biological Considerations

Quality control and normalization is important when estimating RNA-velocity from existing single cell measurements. We suspect that the RNA-velocity measurements from the Embryoid body data may be suspect given the low number of unspliced RNA counts present. In Figure 6.11 we can see that each timepoint consists of around 10%-20% of unspliced RNA.

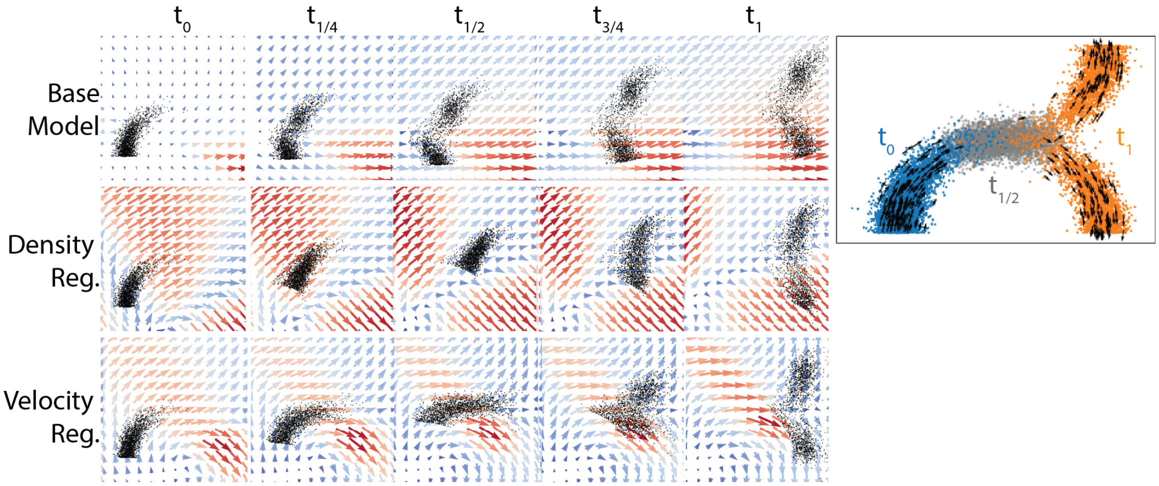


Figure 6.10: Density regularization or velocity regularization can be used to follow a 1D manifold in 2D.

This is relatively low relative to numbers in other recent works [113, 22, 99]. Low unspliced RNA counts leads to more noise in the estimates of RNA velocity and lower quality.

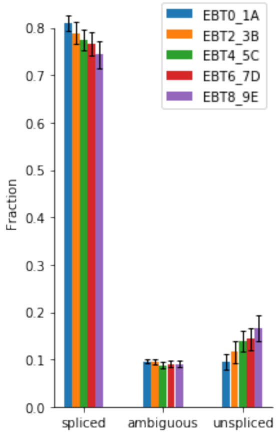


Figure 6.11: Shows the ratio of spliced, ambiguous, and unspliced RNA counts over the 5 timepoints in the Embryoid body dataset. Mean unspliced here is around 10%-20% of total counts, in other systems this is near 30% [113].

In Figure 6.8 we showed how TrajectoryNet can be projected back to the gene space. These projections can be used to infer the differences much earlier in time than they can be identified in the gene space. Here we have four populations that are easily identified by marker genes or clustering at the final timepoint. Since all four populations emerge from a single relatively uniform stem cell population, the question becomes how early can we identify the features of progenitor cells, the cells leading to these differentiated populations. Since TrajectoryNet models cells as probabilities continuously over time, we can find the path for each differentiated cell in earlier timepoints. This allows inferences such as the fact that HAND1, a gene that is generally high in cardiac cells, is high at earlier timepoints, and may even start to distinguish the population as early as day 6. A gene like ONECUT2 is only starts to distinguish neuronal populations at later timepoints. For further information on this particular system see [148] Figure 6.

6.E Reproducibility

To foster reproducibility, we provide as many details as possible on the experiments in the main paper. Code is available at `http://github.com/krishnaswamylab/TrajectoryNet`.

6.E.1 2D Examples

In Figure 6.2 we transport a Gaussian to an s-curve. The Gaussian consists of 10000 points sampled from a standard normal distribution. The s-curve is generated using the sklearn function `sklearn.datasets.make_s_curve` with noise of 0.05, and 10000 samples. We then take the first and third dimension, and multiply by 1.5 for the proper scaling. To generate the OT subplot we used the Mccann interpolant from 200 points sampled from the Gaussian. To generate panel (d), we used the procedure detailed in the beginning of Section 6.5 to train TrajectoryNet, then sampled 200 points from a Gaussian distribution and used the adjoint with these points as the initial state at time t_0 to generate points at time t_1. For panel (e) we added an energy regularization with $\lambda_e = 0.1$ and $\lambda_j = 1$. These were found by experimentation, although parameters in the range of $\lambda_e = [0.01, 1]$ and $\lambda_j = [0.1, 1]$ were largely visually similar.

To generate the arch and tree datasets we started with two half Gaussians $\mathcal{N}(\cdot, \frac{1}{2\pi})$ at mean zero and one (as pictured in Figure 6.4) with 5000 points each, then found the Mccann interpolant at $t_{1/2}$ as the test distribution. We then lift these into 2d by embedding on the half circle of radius 1 and adding noise $\mathcal{N}(0, 0.1)$ to the radius. To generate velocity, we add a velocity tangent to the circle for each point. For the tree dataset we additionally flip (randomly) half of the points with $x > 1$ over the line $y = 1$.

For the Cycle dataset, we start with 5000 uniformly sampled points around the circle, with radius as $\mathcal{N}(1, 0.1)$ We then add an arrow tangent to the circle with magnitude $\pi/5$, Thus in one time unit the points should move 1/10 of the way around the circle.

6.E.2 Single Cell Datasets

Both single cell datasets were sequenced using 10X sequencing. The Embryoid body data can be found here[3] and consists of roughly 30,000 cells unfiltered, and 16,000 cells after filtering. The mouse cortex dataset is not currently publicly available, but consists of roughly 20,000 cells after filtering. For both datasets no batch correction was used. Raw sequences were processed with CellRanger. We then used velocyto [113] to produce the unspliced and spliced count matrices. We then used the default parameters in ScVelo [22] to generate velocity arrows on a PCA embedding. These include count normalization across rows, selection of the 3000 most variable genes, filtering of low quality genes, and smoothing counts between cells.

For parameters we did a grid search over $\lambda_{density} \in \{0, 0.1, 0.01\}$, $\lambda_{velocity} \in \{0, 0.001, 0.0001\}$. For Base+E we did a search of $\lambda_{energy} \in \{1.0, 0.1, 0.01\}$, A more extensive search could lead to better results. We intended to show how these regularizations can be used and demonstrate the viability of this approach rather than fully explore parameter space.

6.E.3 Software Versioning

The following software versions were used. `scvelo==0.1.24`, `torch==1.3.1`, `torchdiffeq==0.0.1`, `velocyto==0.17.17`, `scprep==1.0.3`, `scipy==1.4.1`, `scikit-learn==0.22`, `scanpy==1.4.5`

[3] https://doi.org/10.17632/v6n743h5ng.1

Part III

Optimal Transport in Graphs

Chapter 7

DiffusionEMD

7.1 Introduction

With the profusion of modern high dimensional, high throughput data, the next challenge is the integration and analysis of collections of related datasets. Examples of this are particularly prevalent in single cell measurement modalities where data (such as mass cytometry, or single cell RNA sequencing data) can be collected in a multitude of patients, or in thousands of perturbation conditions [186]. These situations motivate the organization and embedding of datasets, similar to how we now organize data points into low dimensional embeddings, e.g., with PHATE [148], tSNE [204], or diffusion maps [46]). The advantage of such organization is that we can use the datasets as rich high dimensional features to characterize and group the patients or perturbations themselves. In order to extend embedding techniques to entire datasets, we have to define a distance between datasets, which for our purposes are essentially high dimensional point clouds. For this we propose a new form of Earth Mover's Distance (EMD), which we call *Diffusion EMD*[1], where we model the datasets as distributions supported on a common data affinity graph. We provide two extremely fast methods for computing Diffusion EMD based on an approximate multiscale kernel density estimation on a graph.

Optimal transport is uniquely suited to the formulation of distances between entire

[1] Python implementation is available at `https://github.com/KrishnaswamyLab/DiffusionEMD`.

datasets (each of which is a collection of data points) as it generalizes the notion of the shortest path between two points to the shortest set of paths between distributions. Recent works have applied optimal transport in the single-cell domain to interpolate lineages [181, 219, 198], interpolate patient states [197], integrate multiple domains [52], or similar to this work build a manifold of perturbations [39]. All of these approaches use the standard *primal* formulation of the Wasserstein distance. Using either entropic regularization approximation and the Sinkhorn algorithm [50] to solve the discrete distribution case or a neural network based approach in the continuous formulation [8]. We will instead use the *dual* formulation through the well-known Kantorovich-Rubinstein dual to efficiently compute optimal transport between many distributions lying on a common low-dimensional manifold in a high-dimensional measurement space. This presents both theoretical and computational challenges, which are the focus of this work.

Specifically, we will first describe a new Diffusion EMD that is an L^1 distance between density estimates computed using multiple scales of diffusion kernels over a graph. Using theory on the Hölder-Lipschitz dual norm on continuous manifolds [121], we show that as the number of samples increases, Diffusion EMD is equivalent to the Wasserstein distance on the manifold. This formulation reduces the computational complexity of computing K-nearest Wasserstein-neighbors between m distributions over n points from $O(m^2n^3)$ for the exact computation to $\tilde{O}(mn)$ with reasonable assumptions on the data. Finally, we will show how this can be applied to embed large sets of distributions that arise from a common graph, for instance single cell datasets collected on large patient cohorts.

Our contributions include: 1. A new method for computing EMD for distributions over graphs called Diffusion EMD. 2. Theoretical analysis of the relationship between Diffusion EMD and the snowflake of a standard EMD. 3. Fast algorithms for approximating Diffusion EMD. 4. Demonstration of the differentiability of this framework. 5. Application of Diffusion EMD to embedding massive multi-sample biomedical datasets.

7.2 Preliminaries

We now briefly review optimal transport definitions and classic results from diffusion geometry.

Notation. We say that two elements A and B are equivalent if there exist $c, C > 0$ such that $cA \leq B \leq CA$, and we denote $A \simeq B$. The definition of A and B will be clear depending on the context.

Optimal Transport. Let μ, ν be two probability distributions on a measurable space Ω with metric $d(\cdot, \cdot)$, $\Pi(\mu, \nu)$ be the set of joint probability distributions π on the space $\Omega \times \Omega$, where for any subset $\omega \subset \Omega$, $\pi(\omega \times \Omega) = \mu(\omega)$ and $\pi(\Omega \times \omega) = \nu(\omega)$. The 1-Wasserstein distance W_d also known as the earth mover's distance (EMD) is defined as:

$$W_d(\mu, \nu) := \inf_{\pi \in \Pi(\mu,\nu)} \int_{\Omega \times \Omega} d(x, y)\pi(dx, dy). \tag{7.1}$$

When μ, ν are discrete distributions with n points, then equation 7.1 is computable in $O(n^3)$ with a network-flow based algorithm [165].

Let $\| \cdot \|_{L_d}$ denote the Lipschitz norm w.r.t. d, then the dual of equation 7.1 is:

$$W_d(\mu, \nu) = \sup_{\|f\|_{L_d} \leq 1} \int_{\Omega} f(x)\mu(dx) - \int_{\Omega} f(y)\nu(dy). \tag{7.2}$$

This formulation is known as the Kantorovich dual with f as the witness function. Since it is in general difficult to optimize over the entire space of 1-Lipschitz functions, many works optimize the cost over a modified family of functions such as functions parameterized by clipped neural networks [8], functions defined over trees [115], or functions defined over Haar wavelet bases [75].

Data Diffusion Geometry Let $(\mathcal{M}, d_{\mathcal{M}})$ be a connected Riemannian manifold, we can assign to $\mathcal{M}$ sigma-algebras and look at $\mathcal{M}$ as a "metric measure space". We denote by Δ the Laplace-Beltrami operator on $\mathcal{M}$. For all $x, y \in \mathcal{M}$ let $h_t(x, y)$ be the heat kernel, which is the minimal solution of the heat equation:

$$\left(\frac{\partial}{\partial t} - \Delta_x\right) h_t = 0, \tag{7.3}$$

with initial condition $\lim_{t\to 0} h_t(x, y) = \delta_y(x)$, where $x \mapsto \delta_y(x)$ is the Dirac function centered at y, and Δ_x is taken with respect to the x argument of h_t. Note that as shown in Grigor'yan et al. [83], the heat kernel captures the local intrinsic geometry of $\mathcal{M}$ in the sense that as $t \to 0$, $\log h_t(x, y) \simeq -d^2_{\mathcal{M}}(x, y)/4t$.

Here, in Sec. 7.4.2 (Theorem 7.4.2) we discuss another topological equivalent of the geodesic distance with a diffusion distance derived from the heat operator $\boldsymbol{H}_t := e^{-t\Delta}$ that characterizes the solutions of the heat equation (equation 7.3), and is related to the heat kernel via $\boldsymbol{H}_t f = \int h_t(\cdot, y) f(y) dy$ [see 114, 46, 83, for further details].

It is often useful (particularly in high dimensional data) to consider data as sampled from a lower dimensional manifold embedding in the ambient dimension. This manifold can be characterized by its local structure and in particular, how heat propagates along it. Coifman and Lafon [46] showed how to build such a propagation structure over discrete data by first building a graph with affinities

$$(\boldsymbol{K}_\epsilon)_{ij} := e^{-\|x_i - x_j\|_2^2/\epsilon} \tag{7.4}$$

then considering the density normalized operator $\boldsymbol{M}_\epsilon := \boldsymbol{Q}^{-1}\boldsymbol{K}_\epsilon\boldsymbol{Q}^{-1}$, where $\boldsymbol{Q}_{ii} := \sum_j (\boldsymbol{K}_\epsilon)_{ij}$. Lastly, a Markov diffusion operator is defined by

$$\boldsymbol{P}_\epsilon := \boldsymbol{D}^{-1}\boldsymbol{M}_\epsilon, \text{ where } \boldsymbol{D}_{ii} := \sum_j (\boldsymbol{M}_\epsilon)_{ij}. \tag{7.5}$$

Both $\boldsymbol{D}$ and $\boldsymbol{Q}$ are diagonal matrices. By the law of large numbers, the operator $\boldsymbol{P}_\epsilon$ admits a natural continuous equivalent $\tilde{\boldsymbol{P}}_\epsilon$, i.e., for n i.i.d. points, the sums modulo n converge to the integrals. Moreover, in Coifman and Lafon [46, Prop. 3] it is shown that $\lim_{\epsilon\to 0} \tilde{\boldsymbol{P}}_\epsilon^{t/\epsilon} = e^{-t\Delta} = \boldsymbol{H}_t$. In conclusion, the operator $\boldsymbol{P}_\epsilon$ converges to $\tilde{\boldsymbol{P}}_\epsilon$ as the sample size increases and $\tilde{\boldsymbol{P}}_\epsilon$ provide an approximation of the Heat kernel on the manifold. Henceforth, we drop the subscript of $\boldsymbol{P}_\epsilon$ to lighten the notation, further we will use the notation $\boldsymbol{P}_\epsilon$ for the operator as well as for the matrix (it will be clear in the context).

7.3 EMD Through the L^1 Metric Between Multiscale Density Estimates

The method of efficiently approximating EMD that we will consider here is the approximation of EMD through density estimates at multiple scales. Previous work has considered densities using multiscale histograms over images [93], wavelets over images and trees [187, 75] and densities over hierarchical clusters [115]. Diffusion EMD also uses a hierarchical set of bins at multiple scales but with smooth bins determined by the heat kernel, which allows us to show equivalence to EMD with a ground distance of the manifold geodesic. These methods are part of a family that we call *multiscale earth mover's distances* that first compute a set of multiscale density estimates or histograms where L^1 differences between density estimates realizes an effective *witness function* and have (varying) equivalence to the Wasserstein distance between the distributions. This class of multiscale EMDs are particularly useful in computing embeddings of distributions where the Wasserstein distance is the ground distance, as they are amenable to fast nearest neighbor queries. We explore this application further in Sec. 7.4.6 and these related methods in Sec. 7.B.1 of the Appendix.

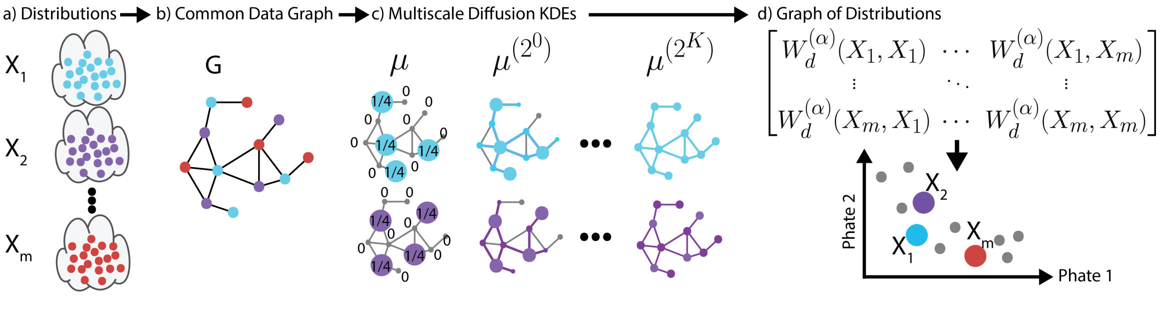

a) Distributions
b) Common Data Graph
c) Multiscale Diffusion KDEs
d) Graph of Distributions
X_1
X_2
X_m
G
μ
$\mu^{(2^0)}$
$\mu^{(2^K)}$
1/4
0
$\begin{bmatrix} W_d^{(\alpha)}(X_1, X_1) & \cdots & W_d^{(\alpha)}(X_1, X_m) \\ \vdots & \ddots & \vdots \\ W_d^{(\alpha)}(X_m, X_1) & \cdots & W_d^{(\alpha)}(X_m, X_m) \end{bmatrix}$
Phate 2
Phate 1
X_2
X_1
X_m

Figure 7.1: Diffusion EMD first embeds datasets into a common data graph G, then takes multiscale diffusion KDEs for each of the datasets. These multiscale KDEs are then used to compute the Diffusion Earth Mover's Distance between the datasets that can be used in turn to create graphs and embeddings (PHATE [147] shown here) of the datasets.

7.4 Diffusion Earth Mover's Distance

We now present the Diffusion EMD, a new Earth Mover's distance based on multiscale diffusion kernels as depicted in Fig. 7.1. We first show how to model multiple datasets as distributions on a common data graph and perform multiscale density estimates on this graph.

7.4.1 Data Graphs and Density Estimates on Graphs

Let $\mathcal{X} = \{X_1, X_2, \ldots, X_m\}$, $\cup_{j=1}^m X_j \subseteq \mathcal{M} \subset \mathbb{R}^d$, be a collection of datasets with $n_i = |X_i|$ and $n = \sum_i n_i$. Assume that the X_i's are independently sampled from a common underlying manifold $(\mathcal{M}, d_{\mathcal{M}})$ which is a Riemannian closed manifold (compact and without boundary) immersed in a (high dimensional) ambient space $\mathbb{R}^d$, with geodesic distance $d_{\mathcal{M}}$. Further, assume that while the underlying manifold is common, each dataset is sampled from a different distribution over it, as discussed below. Such collections of datasets arise from several related samples of data, for instance single cell data collected on a cohort of patients with a similar condition.

Here, we consider the datasets in $\mathcal{X}$ as representing distributions over the common data manifold, which we represent in the finite setting as a common data graph $G_{\mathcal{X}} = (V, E, w)$ with $V = \cup_{j=1}^m X_j$ and edge weights determined by the Gaussian kernel (see equation 7.4), where we identify edge existence with nonzero weights. Then, we associate each X_i with a density measure $\boldsymbol{\mu}_i^{(t)} : V \to [0, 1]$, over the entire data graph. To compute such measures, we first create indicator vectors for the individual datasets on it, let $\mathbf{1}_{X_i} \in \{0, 1\}^n$ be a vector where for each $v \in V, \mathbf{1}_{X_i}(v) = 1$ if and only if $v \in X_i$. We then derive a kernel density estimate by applying the diffusion operator constructed via equation 7.5 over the graph G to these indicator functions to get scale-dependent estimators

$$\boldsymbol{\mu}_i^{(t)} := \frac{1}{n_i} \boldsymbol{P}^t \mathbf{1}_{X_i}, \tag{7.6}$$

where the scale t is the diffusion time, which can be considered as a meta-parameter [e.g., as used in 31] but can also be leveraged in multiscale estimation of distances between distributions as discussed here. Indeed, as shown in Burkhardt et al. [31], at an appropriately tuned single scale, this density construction yields a discrete version of kernel density estimation.

7.4.2 Diffusion Earth Mover's Distance Formulation

We define the Diffusion Earth Mover's Distance between two datasets $X_i, X_j \in \mathcal{X}$ as

$$W_{\alpha,K}(X_i, X_j) := \sum_{k=0}^{K} \|T_{\alpha,k}(X_i) - T_{\alpha,k}(X_j)\|_1 \tag{7.7}$$

where $0 < \alpha < 1/2$ is a meta-parameter used to balance long- and short-range distances, which in practice is set close to $1/2$, K is the maximum scale considered here, and

$$T_{\alpha,k}(X_i) := \begin{cases} 2^{-(K-k-1)\alpha}(\boldsymbol{\mu}_i^{(2^{k+1})} - \boldsymbol{\mu}_i^{(2^k)}) & k < K \\ \boldsymbol{\mu}_i^{(2^K)} & k = K \end{cases} \tag{7.8}$$

Further, to set K, we note that if the Markov process governed by $\boldsymbol{P}$ converges (i.e., to its stationary steady state) in polynomial time w.r.t. $|V|$, then one can ensure that beyond $K = O(\log |V|)$, all density estimates would be essentially indistinguishable as shown by the following lemma, whose proof appears in the Appendix:

Lemma 7.4.1. *There exists a $K = O(\log |V|)$ such that $\boldsymbol{\mu}_i^{(2^K)} \simeq \boldsymbol{P}^{2^K} \mathbf{1}_{X_i} \simeq \phi_0$ for every $i = 1, \dots, n$, where ϕ_0 is the trivial eigenvector of $\boldsymbol{P}$ associated with the eigenvalue $\lambda_0 = 1$.*

To compute the Diffusion EMD $W_{\alpha,K}$ in equation 7.7 involves first calculating diffusions of the datasets $\boldsymbol{\mu}$, second calculating differences and weighting them in relation to their scale, this results in a vector per distribution of length $O(nK)$, and finally computing the L^1 norm between them. The most computationally intensive step, computing the diffusions, is discussed in further detail in Sec. 7.4.4.

7.4.3 Theoretical Relation to EMD on Manifolds

We now provide a theoretical justification of the Diffusion EMD defined via equation 7.7 by following the relation established in Leeb and Coifman [121] between heat kernels and the EMD on manifolds. Leeb and Coifman [121] define the following ground distance for EMD over $\mathcal{M}$ by leveraging the geometric information gathered from the L^1 distances between kernels at different scales.

Definition 7.4.1. The diffusion ground distance between $x, y \in \mathcal{M}$ is defined as

$$D_\alpha(x, y) := \sum_{k \geq 0} 2^{-k\alpha} \| h_{2^{-k}}(x, \cdot) - h_{2^{-k}}(y, \cdot) \|_1,$$

for $\alpha \in (0, 1/2)$, the scale parameter $K \geq 0$ and $h_t(\cdot, \cdot)$ the heat kernel on $\mathcal{M}$.

Note that $D_\alpha(\cdot, \cdot)$ is similar to the diffusion distance defined in Coifman and Lafon [46], which was based on L^2 notions rather than L^1 here. Further, the following result from Leeb and Coifman [121, see Sec. 3.3] shows that the distance $D_\alpha(\cdot, \cdot)$ is closely related to the intrinsic geodesic one $d_{\mathcal{M}}(\cdot, \cdot)$.

Theorem 7.4.2. *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a closed manifold with geodesic distance $d_{\mathcal{M}}$, and let $\alpha \in (0, 1/2)$. The metric $D_\alpha(\cdot, \cdot)$ defined via Def. 7.4.1 is equivalent to $d_{\mathcal{M}}(\cdot, \cdot)^{2\alpha}$.*

The previous theorem justifies why in practice we let α close to 1/2, because we want the snowflake distance $d_{\mathcal{M}}^{2\alpha}(\cdot, \cdot)$ to approximate the geodesic distance of $\mathcal{M}$. The notion of equivalence established by this result is such that $D_\alpha(x, \cdot) \simeq d(x, \cdot)^{2\alpha}$. It is easy to verify that two equivalent metrics induce the same topology. We note that while here we only consider the Heat kernel, a similar result holds (see Theorem 7.A.1 in the Appendix) for a more general family of kernels, as long as they satisfy certain regularity conditions.

For a family of operators $(\boldsymbol{A}_t)_{t \in \mathbb{R}^+}$ we define the following metric on distributions; let μ and ν be two distributions:

$$\begin{aligned} \widehat{W}_{\boldsymbol{A}_t}(\mu, \nu) = {} & \| \boldsymbol{A}_1 (\mu - \nu) \|_1 \\ & + \sum_{k \geq 0} 2^{-k\alpha} \| (\boldsymbol{A}_{2^{-(k+1)}} - \boldsymbol{A}_{2^{-k}})(\mu - \nu) \|_1. \end{aligned} \tag{7.9}$$

The following result shows that applying this metric for the family of operators $\boldsymbol{H}_t$ to get $\widehat{W}_{\boldsymbol{H}_t}$ yields an equivalent of the EMD with respect to the diffusion ground distance $D_\alpha(\cdot, \cdot)$.

Theorem 7.4.3. *The EMD between two distributions μ, ν on a closed Riemannian manifold*

$(\mathcal{M}, d_{\mathcal{M}})$ w.r.t. the diffusion ground distance $D_\alpha(\cdot,\cdot)$, defined via Def. 7.4.1, given by

$$W_{D_\alpha}(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathcal{M}\times\mathcal{M}} D_\alpha(x,y)\pi(dx,dy), \tag{7.10}$$

is equivalent to $\widehat{W}_{\boldsymbol{H}_t}$. That is $W_{D_\alpha} \simeq \widehat{W}_{\boldsymbol{H}_t}$, where $\boldsymbol{H}_t$ is the Heat operator on $\mathcal{M}$.

Proof. In Proposition 15 of Leeb and Coifman [121], it is shown that $\mathcal{M}$ is separable w.r.t. $D_\alpha(\cdot,\cdot)$, hence we can use the Kantorovich-Rubinstein theorem. We let Λ_α, the space of functions that are Lipschitz w.r.t. $D_\alpha(\cdot,\cdot)$ and $\|\cdot\|_{\Lambda_\alpha^*}$, the norm of its dual space Λ_α^*. The norm is defined by

$$\|T\|_{\Lambda_\alpha^*} := \sup_{\|f\|_{\Lambda_\alpha}\leq 1} \int_{\mathcal{M}} f dT.$$

In Theorem 4 of Leeb and Coifman [121], it is shown that both W_{D_α} and $\widehat{W}_{\boldsymbol{H}_t}$ are equivalent to the norm $\|\cdot\|_{\Lambda_\alpha^*}$. □

We consider the family of operators $(\boldsymbol{P}_\epsilon^{t/\epsilon})_{t\in\mathbb{R}^+}$, which is related to the continuous equivalent of the stochastic matrix defined in equation 7.5. In practice, we use this family of operators to approximate the heat operator $\boldsymbol{H}_t$. Indeed, when we take a small value of ϵ, as discussed in Sec. 7.2, we have from Coifman and Lafon [46] that this is a valid approximation.

Corollary 7.4.4. *Let $\boldsymbol{P}_\epsilon$ be the continuous equivalent of the stochastic matrix in equation 7.5. For ϵ small enough, we have:*

$$\widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}} \simeq W_{D_\alpha}. \tag{7.11}$$

Equation 7.11 motivates our use of equation 7.7 to compute the Diffusion EMD. The idea is to take only the first K terms in the infinite sum $\widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}$ and then choosing $\epsilon := 2^{-K}$ would give us exactly equation 7.7. We remark that the summation order of equation 7.7 is inverted compared to equation 7.11, but in both cases the largest scale has the largest weight. Finally, we state one last theorem that brings our distance closer to the Wasserstein w.r.t. $d_{\mathcal{M}}(\cdot,\cdot)$; we refer the reader to the Appendix for its proof.

Theorem 7.4.5. *Let $\alpha \in (0, 1/2)$ and $(\mathcal{M}, d_{\mathcal{M}})$ be a closed manifold with geodesic $d_{\mathcal{M}}$. The Wasserstein distance w.r.t. the diffusion ground distance $D_\alpha(\cdot,\cdot)$ is equivalent to the Wasserstein distance w.r.t the snowflake distance $d_{\mathcal{M}}(\cdot,\cdot)^{2\alpha}$ on $\mathcal{M}$, that is $W_{D_\alpha} \simeq W_{d^{2\alpha}_{\mathcal{M}}}$.*

Corollary 7.4.6. *For each $1 \leq i, j \leq m$, let $X_i, X_j \in \mathcal{X}$ be two datasets with size n_i and n_j respectively, and let μ_i and μ_j be the continuous distributions corresponding to the ones of X_i and X_j, let K be the largest scale and put $N = \min(K, n_i, n_j)$. Then, for sufficiently big $N \to \infty$ (implying sufficiently small $\epsilon = 2^{-K} \to 0$):*

$$W_{\alpha,K}(X_i, X_j) \simeq W_{d^{2\alpha}_{\mathcal{M}}}(\mu_i, \mu_j), \tag{7.12}$$

for all $\alpha \in (0, 1/2)$.

In fact we can summarize our chain of thought as follows

$$\begin{aligned} W_{\alpha,K}(X_i, X_j) &\overset{(a)}{\simeq} \widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}(\mu_i, \mu_j) &&\overset{(b)}{\simeq} \widehat{W}_{\boldsymbol{H}_t}(\mu_i, \mu_j) \\ &\overset{(c)}{\simeq} W_{D_\alpha}(\mu_i, \mu_j) &&\overset{(d)}{\simeq} W_{d^{2\alpha}_{\mathcal{M}}}(\mu_i, \mu_j), \end{aligned}$$

where the approximation *(a)* is due to the fact that the discrete distributions on X_i and X_j converge respectively to μ_i and μ_j when $\min(n_i, n_j) \to \infty$. Further, $W_{\alpha,K}(X_i, X_j)$ approximate the infinite series $\widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}(\mu_i, \mu_j)$ as in equation 7.9 when $K \to \infty$, note also that we take $\epsilon = 2^{-K}$ so that the largest scale in equation 7.7 is exactly 2^K. The approximation in *(b)* comes from the approximation of the heat operator as in Coifman and Lafon [46], *(c)* comes from Theorem 7.4.3 and *(d)* comes from Theorem 7.4.2.

7.4.4 Efficient Computation of Dyadic Scales of the Diffusion Operator

The most computationally intensive step of Diffusion EMD requires computing dyadic scales of the diffusion operator $\boldsymbol{P}$ times $\boldsymbol{\mu}$ to estimate the density of $\boldsymbol{\mu}^{(t)}$ at multiple scales which we will call $\boldsymbol{b}$. After this embedding, Diffusion EMD between two embeddings $\boldsymbol{b}_i, \boldsymbol{b}_j$ is computed as $|\boldsymbol{b}_i - \boldsymbol{b}_j|$, i.e. the L^1 norm of the difference. Computing the embedding $\boldsymbol{b}$ naively by first powering $\boldsymbol{P}$ then right multiplying $\boldsymbol{\mu}$, may take up to 2^K matrix multiplications which is infeasible for even moderately sized graphs. We assume two properties of $\boldsymbol{P}$ that makes

Algorithm 2 Chebyshev embedding

```
Input: $n \times n$ graph kernel $\boldsymbol{K}$, $n \times m$ distributions $\boldsymbol{\mu}$, maximum scale $K$, and snowflake constant $\alpha$.
Output: $m \times (K+1)n$ distribution embeddings $\boldsymbol{b}$
$\boldsymbol{Q} \leftarrow Diag(\sum_i \boldsymbol{K}_{ij})$
$\boldsymbol{K}^{norm} \leftarrow \boldsymbol{Q}^{-1}\boldsymbol{K}\boldsymbol{Q}^{-1}$
$\boldsymbol{D} \leftarrow Diag(\sum_i \boldsymbol{K}^{norm}_{ij})$
$\boldsymbol{M} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{K}^{norm}\boldsymbol{D}^{-1/2}$
$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T = \boldsymbol{M}$;    $\boldsymbol{U}$ orthogonal, $\boldsymbol{\Sigma}$ Diagonal
$\boldsymbol{\mu}^{(2^0)} \leftarrow \boldsymbol{P}\boldsymbol{\mu} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{M}\boldsymbol{D}^{1/2}\boldsymbol{\mu}$
for $k = 1$ to $K$ do
    $\boldsymbol{\mu}^{(2^k)} \leftarrow \boldsymbol{P}^{2^k}\boldsymbol{\mu} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{U}(\boldsymbol{\Sigma})^{2^k}\boldsymbol{U}^T\boldsymbol{D}^{1/2}\boldsymbol{\mu}$
    $\boldsymbol{b}_{k-1} \leftarrow 2^{(K-k-1)\alpha}(\boldsymbol{\mu}^{(2^k)} - \boldsymbol{\mu}^{(2^{k-1})})$
end for
$\boldsymbol{b}_K \leftarrow \boldsymbol{\mu}^{(2^K)}$
$\boldsymbol{b} \leftarrow [\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_K]$
```

this computation efficient in practice. First, that $\boldsymbol{P}$ is sparse with order $\tilde{O}(n)$ non-zero entries. This applies when thresholding $\boldsymbol{K}_\epsilon$ or when using a K-nearest neighbors graph to approximate the manifold [204, 148]. Second, that $\boldsymbol{P}^t$ is low rank for large powers of t.

While there are many ways to approximate dyadic scales of $\boldsymbol{P}$, we choose from two methods depending on the number of distributions m compared to the number of points in the graph n. When $m \ll n$, we use a method based on Chebyshev approximation of polynomials of the eigenspectrum of $\boldsymbol{P}$ as shown in Alg. 2. This method is efficient for sparse $\boldsymbol{P}$ and a small number of distributions [188]. For more detail on the error incurred by using Chebyshev polynomials we refer the reader to Trefethen [203, Chap. 3]. In practice, this requires for the approximating polynomial of J terms, computation of mJ (sparse) matrix vector multiplications for a worst case time complexity of $O(Jmn^3)$, but in practice is $\tilde{O}(Jmn)$ where J is a small constant (see Fig. 7.5(e)). However, while asymptotically efficient, when $m \gg n$ in practice this can be inefficient as it requires many multiplications of the form $\boldsymbol{P}\boldsymbol{\mu}$.

In the case where $m \gg n$, approximating powers of $\boldsymbol{P}$ and applying these to the $n \times m$ collection of distributions $\boldsymbol{\mu}$ once is faster. This method is also useful when the full set of distributions is not known and can be applied to new distributions one at a time in a data streaming model. A naive approach for computing $\boldsymbol{P}^{2^K}$ would require K dense $n \times n$

matrix multiplications. However, as noted in Coifman and Maggioni [47], for higher powers of $\boldsymbol{P}$, we can use a much smaller basis. We use an algorithm based on interpolative decomposition [128, 23] to reduce the size of the basis, and subsequently the computation time, at higher scales. In this algorithm we first determine the approximate rank of $\boldsymbol{P}^{2^k}$ using an estimate of the density of the eigenspectrum of $\boldsymbol{P}$ as in Dong et al. [56]. We then alternate steps of downsampling the basis to the specified rank of $\boldsymbol{P}^{2^k}$ with (randomized) interpolative decomposition with steps of powering $\boldsymbol{P}$ on these bases. Informally, the interpolative decomposition selects a representative set of points that approximate the basis well. In the worst case this algorithm can take $\tilde{O}(mn^3)$ time to compute the diffusion density estimates, nevertheless with sparse $\boldsymbol{P}$ with a rapidly decaying spectrum, this algorithm is $\tilde{O}(mn)$ in practice. For more details see Alg. 5 and Sec. 7.C of the Appendix.

7.4.5 Subsampling Density Estimates

The density estimates created for each distribution are both large and redundant with each distribution represented by a vector of $(K + 1) \times n$ densities. However, as noted in the previous section, $\boldsymbol{P}^{2^k}$ can be represented on a smaller basis, especially for larger scales. Intuitively, the long time diffusions of nodes that are close to each other are extremely similar. Interpolative decomposition [128, 23] allows us to pick a set of points to center our diffusions kernels such that they approximately cover the graph up to some threshold on the rank. In contrast, in other multiscale EMD methods the bin centers or clusters are determined randomly, making it difficult to select the number of centers necessary. Furthermore, the relative number of centers at every scale is fixed, for example, Quadtree or Haar wavelet based methods [93, 75] use 2^{d^k} centers at every scale, and a clustering based method [115] selects C^k clusters at every scale for some constant C. Conversely, in Diffusion EMD, by analyzing the behavior of $\boldsymbol{P}^{2^k}$, we intelligently select the number of centers needed at each scale based on the approximate rank of $\boldsymbol{P}^{2^k}$ up to some tolerance at each scale. This does away with the necessity of a fixed ratio of bins at every scale, allowing adaptation depending on the structure of the manifold and can drastically reduce the size representations (see Fig. 7.5(c)). For the Chebyshev polynomials method, this subsampling is done post computation of diffusion scales, and for the method based on approximating $\boldsymbol{P}^{2^k}$ directly the subsampling happens during computation. To this point, we have described a method to embed distributions on a graph into a set of density estimates whose size depends on the data, and the spectrum decay of $\boldsymbol{P}$. We will now explore how to use these estimates for exploring the Diffusion EMD metric between distributions.

7.4.6 Diffusion EMD Based Embeddings of Samples

Our main motivation for a fast EMD computed on related datasets is to examine the space of the samples or datasets themselves, i.e., the higher level manifold of distributions. In terms of the clinical data, on which we show this method, this would be the relationship between patients themselves, as determined by the EMD between their respective single-cell peripheral blood datasets. Essentially, we create a kernel matrix $\boldsymbol{K}_{\mathcal{X}}$ and diffusion operator $\boldsymbol{P}_{\mathcal{X}}$ between datasets where the samples are nodes on the associated graph. This diffusion operator $\boldsymbol{P}_{\mathcal{X}}$ can be embedded using diffusion maps [46] or visualized with a method like PHATE [147] that collects the information into two dimensions as shown in Sec. 7.5. We note this higher level graph can be a sparse KNN graph, particularly given that a diffusion operator on the graph can allow for global connections to be reformed via t-step path probabilities.

Multiscale formulations of EMD as in equation 7.8 are especially effective when searching for nearest neighbor distributions under the Wasserstein metric [93, 10] as this distance forms a normed space, i.e., a space where the metric is induced by the L_1 norm of the distribution vectors and their differences. Data structures such as kd-trees, ball trees, locality sensitive hashing, are able to take advantage of such normed spaces for sub-linear neighbor queries. This is in contrast to network-flow or Sinkhorn type approximations that require a scan through all datapoints for each nearest neighbor query as this metric is not derived from a norm.

7.4.7 Gradients of the Earth Mover's Distance

One of the hindrances in the use of optimal transport-based distances has been the fact that it cannot be easily incorporated into deep learning frameworks. Gradients with respect to the EMD are usually found using a trained Lipschitz discriminator network as in

Wasserstein-GANs [8], which requires unstable adversarial training, or by taking derivatives through a small number of iterations of the Sinkhorn algorithm [69, 25, 77, 133], which scales with $O(n^2)$ in the number of points. Tree based methods that are linear in the number of points do not admit useful gradients due to their hard binning over space, giving a gradient of zero norm almost everywhere.

We note that, given the data diffusion operator, the computation of Diffusion EMD is differentiable and, unlike Tree-based EMD, has smooth bins and therefore a non-zero gradient norm near the data (as visible in Fig. 7.2). Further, computation of the gradient only requires powers of the diffusion operator multiplied by the indicator vector describing the distribution on the graph. In fact, as mentioned in the supplementary material (Sec. 7.C.1) for each $v \in V$, the gradient of the Diffusion EMD $\partial W_{\alpha,K}(X_i, X_j)/\partial v$ depends mainly on the gradients $\partial \boldsymbol{P}_\epsilon^{2^k}/\partial v$ for $0 \leq K$ which can be expressed in terms of the gradient of the Gaussian kernel $\partial \boldsymbol{K}_\epsilon/\partial v$. This last quantity is easy to compute. In Sec. 7.C.1 of the Appendix, we give an exact process on computing the gradient of the Diffusion EMD.

7.5 Results

In this section, we first evaluate the Diffusion EMD on two manifolds where the ground truth EMD with a geodesic ground distance is known, a swiss roll dataset and spherical MNIST [45]. On these datasets where we have access to the ground truth geodesic distance we show that Diffusion EMD is both faster and closer to the ground truth than comparable methods. Then, we show an application to a large single cell dataset of COVID-19 patients where the underlying metric between cells is thought to be a manifold [147, 111]. We show that the manifold of patients based on Diffusion EMD by capturing the graph structure, better captures the disease state of the patients.

Experimental Setup. We consider four baseline methods for approximating EMD: QuadTree(D) [10] which partitions the dataspace in half in each dimension up to some specified depth D, ClusterTree(C, D) [115] which recursively clusters the data with C clusters up to depth D using the distance between clusters to weight the tree edges, and the convolutional Sinkhorn distance [192] with the same graph as used in Diffusion EMD. QuadTree and ClusterTree are fast to compute. However, because they operate in the ambient space, they do not represent the geodesic distances on the manifold in an accurate way. The convolutional Sinkhorn method represents the manifold well but is significantly slower even when using a single iteration. For more details on related work and the experimental setup see Sections 7.B and 7.D of the Appendix respectively.

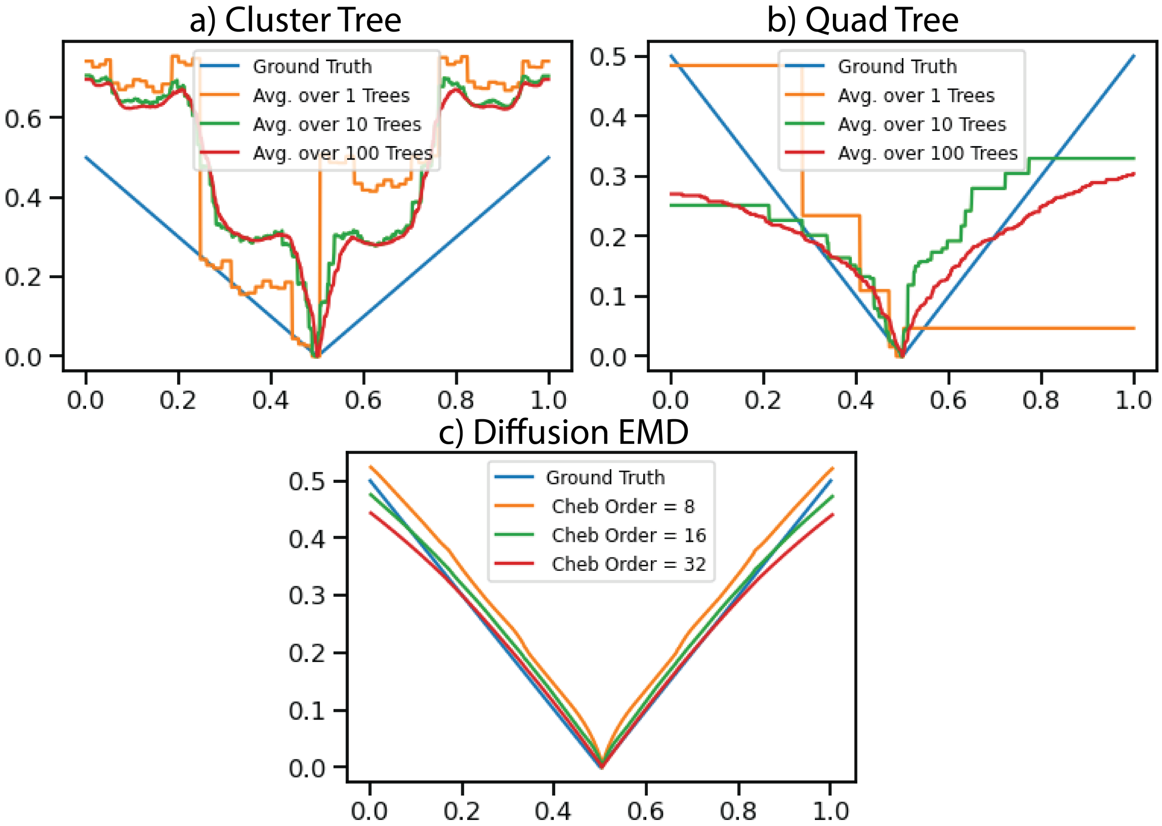


Figure 7.2: Wasserstein distance of indicator distributions $\mathbf{1}_x, x \in [0,1]$ from $\mathbf{1}_{0.5}$ computed using linear EMD methods L^2 distance: (a) ClusterTree (b) QuadTree and (c) Diffusion EMD.

1D data. We first illustrate the advantages of using Diffusion EMD over tree-based methods (ClusterTree and QuadTree) on a line graph with 500 points spaced in the interval $[0,1]$. In Fig. 7.2 we depict the Wasserrstein distance between an indicator function at each of these 500 points ($\mathbf{1}_x$) and an indicator function at $x = 0.5$. In the case of indicator functions the Wasserstein distance is exactly the ground distance. We show three approximations of each method, varying the number of trees and the Chebyshev polynomial order respectively. It is clear that Diffusion EMD achieves a much better approximation of the ground truth primarily due to its use of smooth bins.

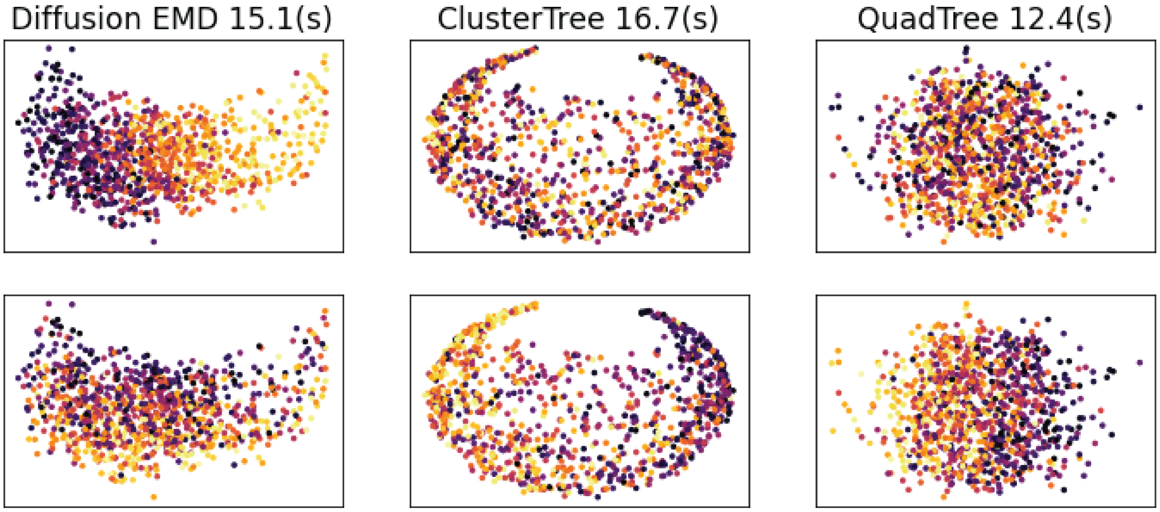


Figure 7.3: Swiss roll dataset embeddings of $m = 1000$ distributions with $n = 10,000$ total points rotated into 10D colored by ground truth 2D sheet axes. Diffusion EMD recreates the manifold better in similar time.

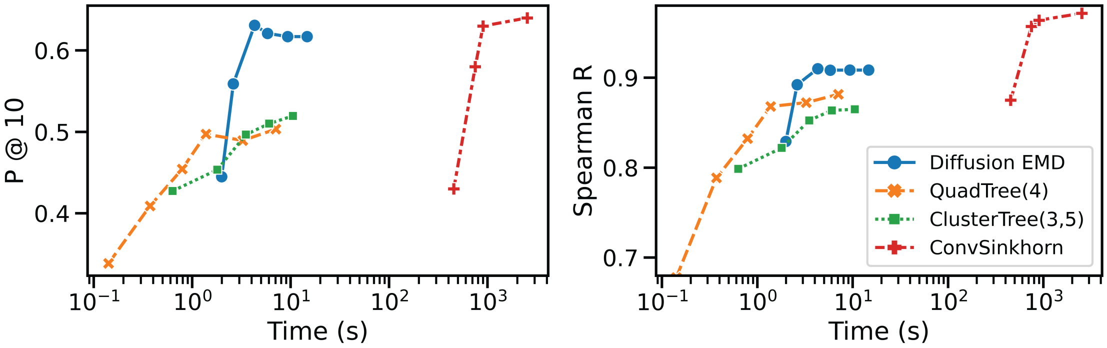


Figure 7.4: Accuracy of methods measured via P@10 (left) and Spearman coefficient (right), against their (log scaled) computation time in seconds on the swiss roll dataset. Variations of methods are over Chebyshev approximation order for Diffusion EMD, # of trees for tree methods, and number of iterations for conv. Sinkhorn. Diffusion EMD is more accurate than tree methods and orders of magnitude faster than conv. Sinkhorn even with a single iteration.

Swiss roll data. The next application we explore is to a dataset where we have distributions on a manifold for which the geodesic distance is easily computable. In this way

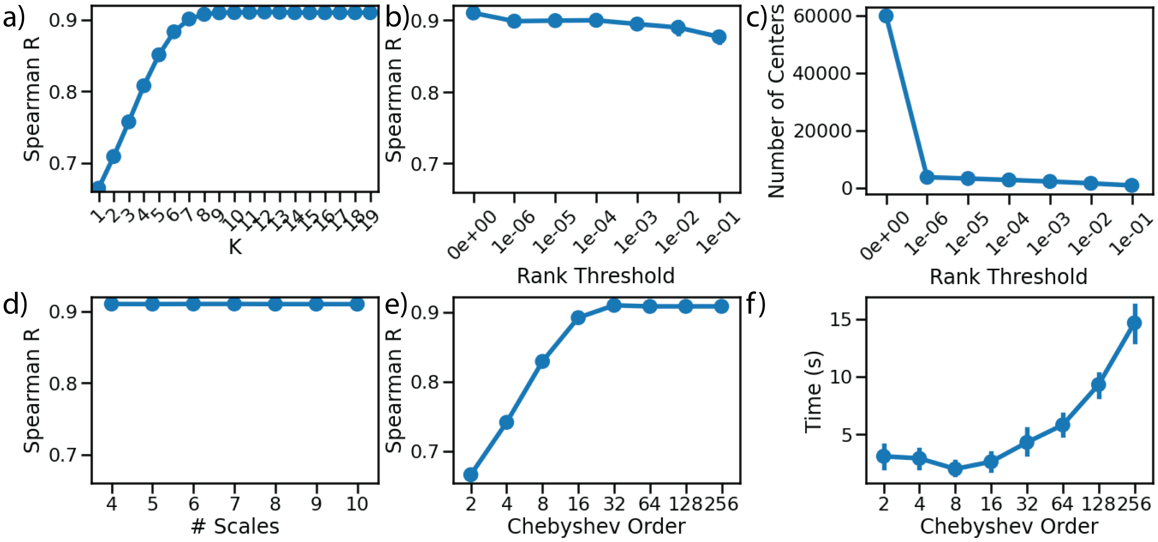


Figure 7.5: Ablation study of major parameters for Chebyshev polynomial approximation on the swiss roll dataset. Mean and std. over 10 runs over (a) values of the maximum scale K, (b) the rank threshold in interpolative decomposition, (c) the total number of centers in the L^1 representation which drops with decomposition, (d-e) performance against the # of scales, and the order of the polynomial, both are very stable after a certain point, and (f) time vs. the Chebyshev order.

we can compare to the ground truth EMD between distributions. We generate $m = 100$ Gaussians on the swiss roll with 100 points each for a total of $n = 10,000$ points on the graph. We compare each method on two metrics, the 10-nearest neighbor accuracy measured (P@10) where a P@10 of 1 means that the 10-nearest neighbors are the same as the ground truth. We compare the rankings of nearest neighbors over the entire dataset using the Spearman-ρ correlation coefficient, which measures the similarity of nearest neighbor rankings. This coefficient ranges between -1 for inversely ranked lists and 1 for the same ranks. This measures rankings over the entire dataset equally rather than only considering the nearest neighbors. Visually, we show embeddings of the swiss roll in Fig. 7.3, where the 2D manifold between distributions is best captured by Diffusion EMD given a similar amount of time.

In Fig. 7.4, we investigate the time vs. accuracy tradeoff of a number of fast EMD methods on the swiss roll. We compare against the ground truth EMD which is calculated with the exact EMD on the "unrolled" swiss roll in 2D. We find that Diffusion EMD is more accurate than tree methods for a given amount of time and is much faster than the convolutional Sinkhorn method and only slightly less accurate. To generate multiple models for each dataset we vary the number of trees for tree methods, the Chebyshev order for Diffusion EMD, and the number of iterations for convolutional Sinkhorn. We search over and fix other parameters using a grid search as detailed in Sec. 7.D of the Appendix.

In Fig. 7.5, we vary parameters of the Chebyshev polynomial algorithm of Diffusion EMD. Regarding performance, we find Diffusion EMD is stable to the number of scales chosen after a certain minimum maximum scale K, Chebyshev polynomial order, and the number of scales used. By performing interpolative decomposition with a specified rank threshold on $\boldsymbol{P}^{2^k}$ we can substantially reduce the embedding size at a small cost to performance Fig. 7.5(b,c).

Table 7.1: Classification accuracy, P@10, Spearman ρ and runtime (in minutes) on 70,000 distributions from Spherical MNIST.

	Accuracy	P@10	Spearman ρ	Time
Diff. EMD	**95.94**	**0.611**	**0.673**	34m
Cluster	91.91	0.393	0.484	30m
Quad	79.56	0.294	0.335	**16m**

Spherical MNIST. Next, we use the Spherical MNIST dataset to demonstrate the efficacy of the interpolative decomposition based approximation to Diffusion EMD, as here $m \gg n$. Each image is treated as a distribution (of pixel intensities) over the sphere. To evaluate the fidelity of each embedding, we evaluate the 1-NN classification on the embedding vectors in addition to P@10 and Spearman coefficient in Tab. 7.1. Diffusion EMD creates embeddings that better approximate true EMD over the sphere than tree methods in a similar amount of time, which in this case also gives better classification accuracy.

Single cell COVID-19 patient data. The COVID-19 pandemic has driven biologists to generate vast amounts of cellular data on hospitalized patients suffering from severe disease. A major question in clinicians minds is determining a priori which patients may be at risk for worse outcomes, including requiring increased ventilatory support and increased risk of

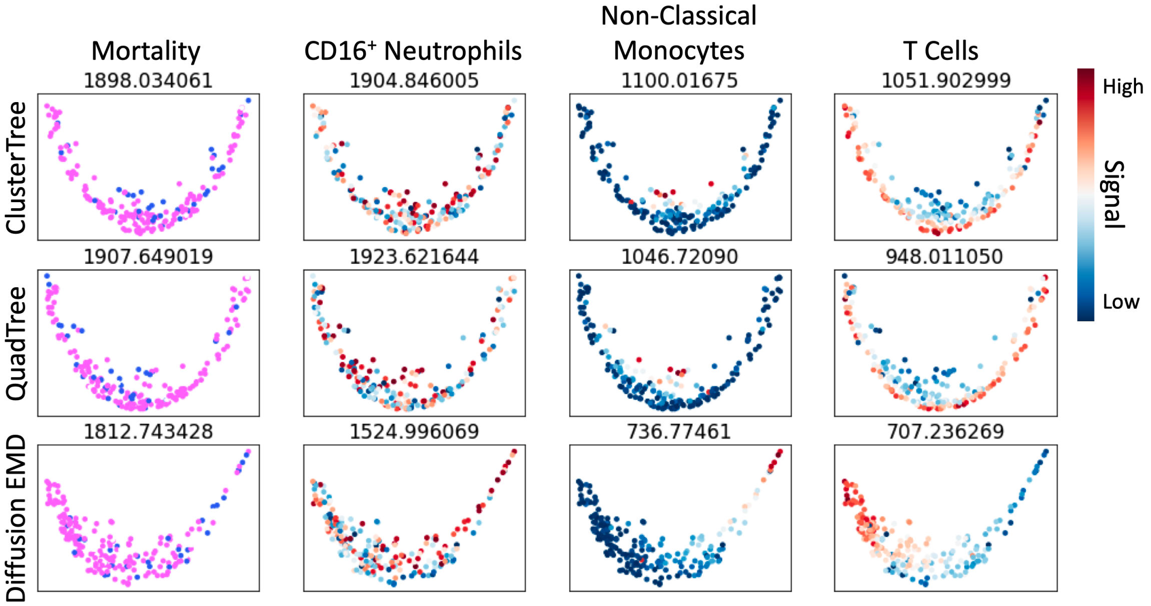


Figure 7.6: Embedding of 210 patients through different manifold constructions. Visualizing patient eventual mortality and cell types predictive of disease outcome on each manifold. Laplacian smoothness reported on each signal for each manifold.

mortality. Certain cell types found in the blood, such as CD16$^+$ Neutrophils, T cells and non-classical monocytes, have been associated with and predictive of mortality outcome. Ideally, a manifold of patients would find these cellular populations to occupy a region of high mortality for CD16$^+$ Neutrophils and non-classical monocytes and low mortality for T cells. In order to construct a manifold of patients suffering from COVID-19, we analyzed 210 blood samples from 168 patients infected with SARS-CoV-2 measured on a myeloid-specific flow cytometry panel, an expanded iteration of a previously published dataset [136]. We embedded 22 million cells from these patients into a common combined cell-cell graph with 27,000 nodes as defined in Sec. 7.4.1. We then computed Diffusion EMD and other methods on these datasets. Diffusion EMD is computed by using indicator vectors for each patient converted to density estimates as in equation 7.6.

On an informative embedding of patients, similar patients, with similar features (such as mortality) would localize on the manifold and thus the important features should be smooth over the manifold. Furthermore, cell types which are correlated with outcome either positively or negatively should also be smooth and either correlated or anticorrelated with outcome. To quantify this, we compute the smoothness with respect to the patient manifold by using a *Laplacian quadratic form* with respect to the 10-NN graph between patients. Convolutional Sinkhorn does not scale to this data, so we compare a patient manifold created with Diffusion EMD to ones created with QuadTree and ClusterTree. Diffusion EMD is able to use the manifold of cells where QuadTree and ClusterTree are built in the ambient space. In Fig. 7.6 we visualize relevant signals over the patients using PHATE [148] overlayed with the quadratic smoothness of the signal over the graph. While the mortality signal appeared enriched in the right branches of both the Diffusion EMD and QuadTree manifolds, it did not localize as well on the ClusterTree manifold. Both CD16$^+$ Neutrophils and non-classical monocytes appeared smoother over the Diffusion EMD manifold than the comparison manifolds. Since both cell types are associated with mortality, it was interesting to see them both enriched in high mortality region of the Diffusion EMD manifold but not the others. Finally, T cells, which are negatively correlated with mortality appeared smoothly enriched in the Diffusion EMD manifold in a region with no mortality. In QuadTree and ClusterTree constructions, T cells appeared enriched throughout the manifold, no localizing smoothly to a region with low mortality. These experiments show that the patient manifold constructed with Diffusion EMD is more informative, smoothly localizing key signals, such as patient outcome and predictive cell types. For a more details see Sec. 7.D of the Appendix.

7.6 Conclusion

In this work we have introduced Diffusion EMD, a multiscale distance that uses heat kernel diffusions to approximate the earth mover's distance over a data manifold. We showed how Diffusion EMD can efficiently embed many samples on a graph into a manifold of samples in $\tilde{O}(mn)$ time more accurately than similarly efficient methods. This is useful in the biomedical domain as we show how to embed COVID-19 patient samples into a higher level patient manifold that more accurately represents the disease structure between patients. Finally, we also show how to compute gradients with respect to Diffusion EMD, which opens the possibility of gradients of the earth mover's distance that scale linearly with the dataset size.

Appendix

We first analyze the theoretical framework of Diffusion EMD in Appendix 7.A. Next we discuss related worki n Appendix 7.B, with a particular focus on multiscale methods for EMD. In Appendix 7.C we provide further detail on the two algorithms for computing Diffusion EMD, the first based on Chebyshev approximation, and the second based on directly powering the diffusion operator while reducing the basis. We discuss gradients of Diffusion EMD in section 7.C.1. Finally we provide further experimental details in Appendix 7.D.

7.A General framework and proofs

7.A.1 General framework

We now recall some useful results from Leeb and Coifman [121]. We will only present an overview, for a rigorous exposition, we suggest Leeb [119], Leeb and Coifman [121] and Grigor'yan and Liu [82].

We let $\mathscr{Z}$ be a sigma-finite measure space of dimension n, $d(\cdot,\cdot)$ its intrinsic distance and ξ its associated sigma-finite measure. In order to evaluate the EMD between two measures on $\mathscr{Z}$, one would need to know $d(\cdot,\cdot)$. Either directly, by using equation 7.1, or implicitly, to define the space of Lipschitz functions with respect to $d(\cdot,\cdot)$ in equation 7.2. One of the contributions of this paper is to define a kernel based metric $D_\alpha(\cdot,\cdot)$, where the kernel depends on $\alpha \in (0,1)$, and show that the metrics $D_\alpha(\cdot,\cdot)$ and $d(\cdot,\cdot)$ are closely related. Next, the objective is to use $D_\alpha(\cdot,\cdot)$ as the ground distance to compute the EMD in its dual form. That is a norm between two distributions acting on the space of Lipschitz functions with respect to $D_\alpha(\cdot,\cdot)$. To do so, the authors define Λ_α the space of functions that are Lipschitz with respect to $D_\alpha(\cdot,\cdot)$, and its dual Λ^*_α; the space of measures acting on $f \in \Lambda_\alpha$. Further, in order to use the Kantorovich–Rubinstein theorem, they show that the space $\mathscr{Z}$ is separable with respect to the metric $D_\alpha(\cdot,\cdot)$. Thus, the norm of the dual space $\|\cdot\|_{\Lambda^*_\alpha}$ can be used to compute the EMD with $D_\alpha(\cdot,\cdot)$ as the ground distance. Lastly, they define two norms $\|\cdot\|^{(1)}_{\Lambda^*_\alpha}$ and $\|\cdot\|^{(2)}_{\Lambda^*_\alpha}$ that are equivalent to $\|\cdot\|_{\Lambda^*_\alpha}$ on Λ^*. In practice, these norms are much faster to compute.

The metric $D_\alpha(\cdot,\cdot)$ is defined using a family of kernels $\{a_t(\cdot,\cdot)\}_{t\in\mathbb{R}^+}$ on $\mathscr{Z}$. For each kernel, we define an operator $\boldsymbol{A}_t$ as $(\boldsymbol{A}_t f)(x) = \int_{\mathscr{Z}} a_t(x,y)f(y)d\xi(y)$. These kernels must respect some properties:

- The semigroup property: for all $s,t>0, \boldsymbol{A}_t\boldsymbol{A}_s = \boldsymbol{A}_{t+s}$;
- The conservation property: $\int_{\mathscr{Z}} a_t(x,y)d\xi(y) = 1$;
- The integrability property: there exists $C>0$ such that $\int_{\mathscr{Z}}|a_t(x,y)|d\xi(y) < C$, for all $t>0$ and $x\in\mathscr{Z}$.

Considering only the dyadic times, that is $t = 2^{-k}$, we define the kernel $p_k(\cdot,\cdot) := a_{2^{-k}}(\cdot,\cdot)$ and the operator $\boldsymbol{P}_k = \boldsymbol{A}_{2^{-k}}$, for all $k\in\mathbb{N}$. By leveraging the local geometric information gathered from the L^1 distance between two measures

$$D_k(x,y) := \|p_k(x,\cdot) - p_k(y,\cdot)\|_1,$$

the authors define the following multiscale metric

$$D_\alpha(x,y) := \sum_{k\geq 0} 2^{-k\alpha} D_k(x,y).$$

To interpret $D_k(\cdot,\cdot)$ in an intuitive way, consider the case where $a_t(\cdot,\cdot)$ defines a random walk on $\mathscr{Z}$. As a consequence, $a_t(x,B(y,r))$ is the probability to move from x to a point in $B(y,r)$ in t steps. Moreover, for any $x\in\mathscr{Z}$, $a_t(x,\cdot)$ defines a distribution, therefore $D_k(x,y)$ is the L^1 distance between two distributions induced by the points x and y. Since these distributions depend on the number of steps t, it is clever to consider a distance that includes many scales, just like $D_\alpha(\cdot,\cdot)$. Another property needs to be verified by the kernel $a_t(\cdot,\cdot)$, this property depends on the distance $D_\alpha(\cdot,\cdot)$. Namely, the geometric property: there exist $C > 0$ and $\alpha \in (0,1)$ such that for all $k \in \mathbb{N}$ and $x \in \mathcal{Z}$

$$\int_{\mathcal{Z}} |p_k(x,y)| D_\alpha(x,y) d\xi(y) \leq C 2^{-k\alpha}.$$

We need to add three stronger regularity conditions on the kernel $a_t(\cdot,\cdot)$, for $D_\alpha(\cdot,\cdot)$ to be closely related to the intrinsic distance $d(\cdot,\cdot)$:

1. An upper bound on the kernel: there exist a non-negative, monotonic decreasing function $\phi : \mathbb{R}^+ \to \mathbb{R}$ and $\beta > 0$ such that for any $\gamma < \beta$, the function verifies

   $$\int_0^\infty \tau^{n+\gamma-1} \phi(\tau) d\tau$$

   and

   $$|a_t(x,y)| \leq t^{-\frac{n}{\beta}} \phi\left(\frac{d(x,y)}{t^{1/\beta}}\right).$$

2. Hölder continuity estimate: there exist $\Theta > 0$ sufficiently small, such that, for all $t \in (0,1]$ and all $x, y \in \mathcal{Z}$, the distance verifies $d(x,y) \leq t^{1/\beta}$ and for all $u \in \mathcal{Z}$ the difference between kernels is bounded

   $$|a_t(x,u) - a_t(y,u)| \leq t^{-\frac{n}{\beta}} \left(\frac{d(x,y)}{t^{1/\beta}}\right)^\Theta \phi\left(\frac{d(x,y)}{t^{1/\beta}}\right).$$

3. A local lower bound: there exist a monotonic decreasing function $\psi : \mathbb{R}^+ \to \mathbb{R}$ and $R > 0$, such that, for all $t \in (0,1]$ and all x, y where $d(x,y) < R$, we have

   $$|a_t(x,y)| \geq t^{-\frac{n}{\beta}} \psi\left(\frac{d(x,y)}{t^{1/\beta}}\right).$$

It is shown that the heat kernel on a closed Riemannian manifold respects all these conditions [121, see Sec. 3.3].

Definition 7.A.1. A distance $d(\cdot,\cdot)^b$ is a snowflake of a distance $d(\cdot,\cdot)$ if $b \in (0,1)$. Moreover, the Hölder space is the space of functions that are Lipschitz continuous w.r.t. a snowflake distance, hence the terminology used in Leeb and Coifman [121].

We now state an important theorem from Leeb and Coifman [121, Thm. 2].

Theorem 7.A.1. *Consider a sigma-finite measure space $\mathcal{Z}$ of dimension n with metric $d(\cdot,\cdot)$ and a measure ξ such that $\xi(B(x,r)) \simeq r^n$. If the family of kernels $\{a_t(\cdot,\cdot)\}_{t\in\mathbb{R}^+}$ respect the condition 1,2 and 3, then, for $0 < \alpha < \min(1, \Theta/\beta)$, the distance $D_\alpha(\cdot,\cdot)$ is equivalent to the thresholded snowflake distance* $\min[1, d(\cdot,\cdot)^{\alpha\beta}]$.

Remark. In our case, because we used the heat kernel on a closed Riemannian manifold $\mathcal{M}$, we had $D_\alpha(\cdot,\cdot) \simeq d_\mathcal{M}(\cdot,\cdot)^{2\alpha}$ (Thm. 7.4.2). This can be justified from Leeb and Coifman [121, Cor. 2]. Using the same notation as in the corollary, we define $C := \max_{x,y} d_\mathcal{M}(x,y)^{2\alpha}$ (which is finite due to the assumptions on $\mathcal{M}$), thus we can bound the constant B

$$B = \frac{B}{d_\mathcal{M}(x,y)^{2\alpha}} d_\mathcal{M}(x,y)^{2\alpha} \geq \frac{B}{C} d_\mathcal{M}(x,y)^{2\alpha}.$$

The previous theorem closely links the two considered distances. It also motivated the goal to compute the EMD w.r.t. $D_\alpha(\cdot,\cdot)$. In Leeb and Coifman [121][Prop. 15], it is shown that $\mathcal{Z}$ is a separable space with respect to the metric $D_\alpha(\cdot,\cdot)$. Hence, we can use the Kantorovich-Rubinstein theorem to express the EMD in its dual form.

First, we need to define the space of functions that are Lipschitz with respect to $D_\alpha(\cdot,\cdot)$. For a fix $\alpha \in (0,1)$, we note this space by Λ_α. It corresponds to the set of functions f on $\mathcal{Z}$ such that the norm

$$\|f\|_{\Lambda_\alpha} := \sup_x |f(x)| + \sup_{x\neq y} \frac{|f(x) - f(y)|}{D_\alpha(x,y)}$$

is finite. Next, we need to define the space dual to Λ_α, which is noted by Λ_α^*. For a function $f \in \Lambda_\alpha$ and a L^1 measure T, we define $\langle f, T\rangle := \int_\mathcal{Z} f dT$. The space Λ_α^* is the space of L^1 measure with the norm

$$\|T\|_{\Lambda_\alpha^*} := \sup_{\|f\|_{\Lambda_\alpha}\leq 1} \langle f, T\rangle.$$

In practice, this norm would still be computationally expensive. However, the authors show that the norms

$$\|T\|^{(1)}_{\Lambda^*_\alpha} := \|P_0^* T\|_1 + \sum_{k\geq 0} 2^{-k\alpha} \|(P^*_{k+1} - P^*_k)T\|_1$$
$$\|T\|^{(2)}_{\Lambda^*_\alpha} := \|P_0^* T\|_1 + \sum_{k\geq 0} 2^{-k\alpha} \|(P^*_{k} - P^*_0)T\|_1$$

are equivalent to the norm $\|\cdot\|_{\Lambda^*_\alpha}$ on Λ^*_α [121, Thm. 4], where P^*_k is the adjoint of P_k. Finally, using the Kantorovich-Rubinstein theorem, we get

$$\inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathcal{Z}\times\mathcal{Z}} D_\alpha(x,y)\pi(dx,dy) \|\mu-\nu\|_{\Lambda^*_\alpha} \simeq \|\mu-\nu\|^{(1)}_{\Lambda^*_\alpha} \simeq \|\mu-\nu\|^{(2)}_{\Lambda^*_\alpha},$$

where $D_\alpha(\cdot,\cdot) \simeq \min[1, d(\cdot,\cdot)^{\beta\alpha}]$ and $d(\cdot,\cdot)$ is the ground distance of $\mathcal{Z}$.

In conclusion, using $\|\cdot\|^{(1)}_{\Lambda^*_\alpha}$ or $\|\cdot\|^{(2)}_{\Lambda^*_\alpha}$ yields a norm equivalent to $\|\cdot\|_{\Lambda^*_\alpha}$. The norm $\|\mu - \nu\|_{\Lambda^*_\alpha}$ is equal to the Wasserstein distance between the distributions μ and ν with respect to the ground distance $D_\alpha(\cdot,\cdot)$.

7.A.2 Proofs of section 4.2

Lemma 1. Assuming that $\boldsymbol{P}$ converges (i.e. to its stationary distribution) in polynomial time w.r.t. $|V|$, then there exists a $K = O(\log|V|)$ such that $\boldsymbol{\mu}_i^{(2^K)} \simeq \boldsymbol{P}^{2^K}\mathbf{1}_{X_i} \approx \phi_0$ for every $i = 1,\ldots,n$, where ϕ_0 is the trivial eigenvector of $\boldsymbol{P}$ associated with the eigenvalue $\lambda_0 = 1$.

Proof. First we notice that $\boldsymbol{P}$ is reversible with respect to $\boldsymbol{\pi}_i = \boldsymbol{D}_{ii}/\sum_i\sum_j \boldsymbol{D}_{ij}$. Since $\boldsymbol{P}$ is ergodic it converges to its unique stationary distribution $\boldsymbol{\pi}$. Moreover, we assumed this convergence to be in polynomial time w.r.t. $|V|$, i.e. we define

$$\Delta_i(k) := \frac{1}{2}\sum_j |\boldsymbol{P}^{2^k}_{ij} - \boldsymbol{\pi}_j|$$

and the mixing time for a $\epsilon > 0$

$$\tau_i(\epsilon) := \min\{k : \Delta_i(k') < \epsilon, \forall\, k' \geq k\}.$$

Then, by our assumption, there exist $2^K = \tau_i(\epsilon) = O(|V|)$, thus $K = O(\log |V|)$. Intuitively, for all $k \geq K$, each row of the matrix $\boldsymbol{P}^{2^k}$ is approximately equal to $\boldsymbol{\pi}$ (w.r.t. ϵ), as a consequence $\boldsymbol{P}^{2^k}\mathbf{1}_{X_i} \approx \phi_0$.

□

Theorem 3. Let $\alpha \in (0, 1/2)$ and $(\mathcal{M}, d_{\mathcal{M}})$ be a closed manifold with geodesic $d_{\mathcal{M}}$. The Wasserstein distance w.r.t. the diffusion ground distance $D_\alpha(\cdot,\cdot)$ is equivalent to the Wasserstein distance w.r.t the snowflake distance $d_{\mathcal{M}}(\cdot,\cdot)^{2\alpha}$ on $\mathcal{M}$, that is $W_{D_\alpha} \simeq W_{d_{\mathcal{M}}^{2\alpha}}$.

Proof. We will prove this for a more general framework, we let two metrics $d_1(\cdot,\cdot)$ and $d_2(\cdot,\cdot)$ on a sigma-finite measure space $\mathcal{Z}$, and let μ and ν be two measures. We assume $d_1 \simeq d_2$, that is there exist two constants $c, C > 0$ such that for all $x, y \in \mathcal{Z}$ we have $c\, d_1(x,y) \leq d_2(x,y) \leq C\, d_1(x,y)$. Using the same notation as in equation 7.1, for all $\pi \in \Pi(\mu,\nu)$ we have

$$c \int_{\mathcal{Z}\times\mathcal{Z}} d_1(x,y)\pi(dx,dy) \leq \int_{\mathcal{Z}\times\mathcal{Z}} d_2(x,y)\pi(dx,dy) \leq C \int_{\mathcal{Z}\times\mathcal{Z}} d_1(x,y)\pi(dx,dy)$$

then

$$\begin{aligned} c \inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathcal{Z}\times\mathcal{Z}} d_1(x,y)\pi(dx,dy) &\leq \inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathcal{Z}\times\mathcal{Z}} d_2(x,y)\pi(dx,dy) \\ &\leq C \inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathcal{Z}\times\mathcal{Z}} d_1(x,y)\pi(dx,dy) \end{aligned}$$

which is the same as $cW_{d_1}(\mu,\nu) \leq W_{d_2}(\mu,\nu) \leq CW_{d_1}(\mu,\nu)$, this proves that whenever d_1 and d_2 are equivalent then W_{d_1} and W_{d_2} are equivalent as well. □

Corollary 2.1. Let $\boldsymbol{P}_\epsilon$ be the continuous equivalent of the stochastic matrix in equation 7.5. For ϵ small enough, we have:

$$\widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}} \simeq W_{D_\alpha}. \tag{7.13}$$

Proof. Note that by Theorem 7.4.3 it is equivalent to show that $\widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}} \simeq \widehat{W}_{\boldsymbol{H}_t}$. First note that according to [46] we have $\|\boldsymbol{P}_\epsilon^{t/\epsilon} - \boldsymbol{H}_t\|_{L^2(\mathcal{M})} \to 0$ as $\epsilon \to 0$, note that since $\mathcal{V}\mathrm{ol}(\mathcal{M}) < \infty$ (closed manifold) and according to Cauchy-Schwarz inequality we get $\|\boldsymbol{P}_\epsilon^{t/\epsilon} - \boldsymbol{H}_t\|_{L^1(\mathcal{M})} \to 0$

as $\epsilon \to 0$. Generally speaking, convergence in L^2 implies convergence in L^1 when the manifold is closed. Now put $\boldsymbol{D}_{\epsilon,t} = \boldsymbol{P}_\epsilon^{t/\epsilon} - \boldsymbol{H}_t$, then we have $\|\boldsymbol{D}_{\epsilon,t}\|_{L^1(\mathcal{M})} \to 0$ as $\epsilon \to 0$. Let μ and ν be two measures and let $\delta > 0$, choose $\epsilon > 0$ small enough so that $\|\boldsymbol{D}_{t,\epsilon}\gamma\|_1 < \delta\|\gamma\|_1$ for all $t > 0$, where $\gamma = \mu - \nu$,

$$\begin{aligned}\widehat{W}_{\boldsymbol{D}_{\epsilon,t}}(\mu,\nu) &= \|\boldsymbol{D}_{\epsilon,1}\gamma\|_1 + \sum_{k\geq 0} 2^{-k\alpha} \left\| \left(\boldsymbol{D}_{\epsilon,2^{-(k+1)}} - \boldsymbol{D}_{\epsilon,2^{-k}} \right) \gamma \right\|_1 \\ &\leq \|\boldsymbol{D}_{\epsilon,1}\gamma\|_1 + \sum_{k\geq 0} 2^{-k\alpha} \left\| \boldsymbol{D}_{\epsilon,2^{-(k+1)}}\gamma \right\|_1 + 2^{-k\alpha} \left\| \boldsymbol{D}_{\epsilon,2^{-k}}\gamma \right\|_1 \\ &\leq \delta\|\gamma\|_1 + 2\,\delta\,\|\gamma\|_1 \sum_{k\geq 0} 2^{-k\alpha} \\ &= \delta\|\gamma\|_1 \left(1 + \frac{2}{1-2^{-\alpha}}\right).\end{aligned}$$

This proves that for all $t > 0$, for all $\delta > 0$ there exists an $\epsilon > 0$ sufficiently small such that

$$\widehat{W}_{\boldsymbol{D}_{\epsilon,t}}(\mu,\nu) \leq \delta\|\mu - \nu\|$$

Note also that using the reverse triangle inequality we can easily show that

$$\left| \widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}(\mu,\nu) - \widehat{W}_{\boldsymbol{H}_t}(\mu,\nu) \right| \leq \widehat{W}_{\boldsymbol{D}_{\epsilon,t}}(\mu,\nu)$$

, let $\delta > 0$ then for $\epsilon > 0$ small enough we get

$$\widehat{W}_{\boldsymbol{H}_t}(\mu,\nu) - \delta\|\mu-\nu\|_1 \leq \widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}(\mu,\nu) \leq \widehat{W}_{\boldsymbol{H}_t}(\mu,\nu) + \delta\|\mu-\nu\|_1,$$

according to [212] we can get lower bounds of the heat kernel (which implies lower bounds for the heat operator), we have $\widehat{W}_{\boldsymbol{H}_t}(\mu,\nu) \geq \|\boldsymbol{H}_1(\mu-\nu)\|_1 \geq C\|\mu-\nu\|_1$ for some $C > 0$. If $\delta < C/2$ then for sufficiently small $\epsilon > 0$,

$$\frac{1}{2}\widehat{W}_{\boldsymbol{H}_t}(\mu,\nu) \leq \widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}(\mu,\nu) \leq \frac{3}{2}\widehat{W}_{\boldsymbol{H}_t}(\mu,\nu)$$

which completes the proof. □

Corollary 3.1. For each $1 \leq i, j \leq m$ let $X_i, X_j \in \mathcal{X}$ be two datasets with size n_i and n_j respectively, and let μ_i and μ_j be the continuous distributions corresponding to the ones of X_i and X_j, let K be the largest scale and put $N = \min(K, n_i, n_j)$. Then, for sufficiently big $N \to \infty$ (implying sufficiently small $\epsilon = 2^{-K} \to 0$):

$$W_{\alpha,K}(X_i, X_j) \simeq W_{d_{\mathcal{M}}^{2\alpha}}(\mu_i, \mu_j), \tag{7.14}$$

for all $\alpha \in (0, 1/2)$.

Proof. First define

$$\begin{aligned} W_{\alpha,K,\epsilon}(X_i, X_j) :=& \|\boldsymbol{\mu}_i^{(1/\epsilon)} - \boldsymbol{\mu}_j^{(1/\epsilon)}\|_1 \\ &+ \sum_{k=0}^{K-1} 2^{-k\alpha} \|(\boldsymbol{\mu}_i^{(2^{-(k+1)}/\epsilon)} - \boldsymbol{\mu}_i^{(2^{-k}/\epsilon)}) - (\boldsymbol{\mu}_j^{(2^{-(k+1)}/\epsilon)} - \boldsymbol{\mu}_j^{(2^{-k}/\epsilon)})\|_1, \end{aligned} \tag{7.15}$$

Note that for $\epsilon = 2^{-K}$ we have $W_{\alpha,K,\epsilon}(X_i, X_j) = W_{\alpha,K}(X_i, X_j)$. Since $n_i \to \infty$ and $n_j \to \infty$, then using Monte-Carlo integration we get

$$\lim_{n_i, n_j \to \infty} W_{\alpha,K,\epsilon}(X_i, X_j) = W_{\alpha,K,\epsilon}(\mu_i, \mu_j)$$

where $W_{\alpha,K,\epsilon}(\mu_i, \mu_j)$ has the same expression as in equation 7.15 (replacing the discrete measure $\boldsymbol{\mu}_i$'s with the continuous measures μ_i's),

$$\begin{aligned} W_{\alpha,K,\epsilon}(\mu_i, \mu_j) =& \|\boldsymbol{P}_\epsilon^{1/\epsilon}\mu_i - \boldsymbol{P}_\epsilon^{1/\epsilon}\mu_j\|_1 \\ &+ \sum_{k=0}^{K-1} 2^{-k\alpha} \|(\boldsymbol{P}_\epsilon^{2^{-(k+1)}/\epsilon}\mu_i - \boldsymbol{P}_\epsilon^{2^{-k}/\epsilon}\mu_i) - (\boldsymbol{P}_\epsilon^{2^{-(k+1)}/\epsilon}\mu_j - \boldsymbol{P}_\epsilon^{2^{-k}/\epsilon}\mu_j)\|_1 \end{aligned}$$

Note that by definition of $\widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}$ we have $\lim_{K\to\infty} W_{\alpha,K,\epsilon}(\mu_i, \mu_j) = \widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}(\mu_i, \mu_j)$ and thus

$$\lim_{N\to\infty} W_{\alpha,K,\epsilon}(X_i, X_j) = \widehat{W}_{\boldsymbol{P}_\epsilon^{t/\epsilon}}(\mu_i, \mu_j)$$

Combining this result with Corollary 7.4.4 yields $W_{\alpha,K,\epsilon}(X_i, X_j) \simeq W_{D_\alpha}(\mu_i, \mu_j)$ for N large enough and ϵ small enough. Now if we take $\epsilon = 2^{-K}$ and apply Theorem 7.4.5 we get

$W_{\alpha,K}(X_i, X_j) \simeq W_{d_{\mathcal{M}}^{2\alpha}}(\mu_i, \mu_j)$ as required.

□

7.B Related Work

Wavelet-based linear time approximations of the earth mover's distance have been investigated by Indyk and Thaper [93] who used randomly shifted multi-scale grids to compute EMD between images. Shirdhonkar and Jacobs [187] expanded this to wavelets over $\mathbb{R}^n$ showing the benefit of using smooth bins over the data, and the relationship to the dual problem. Leeb and Coifman [121] investigated using wavelets over more general metric spaces. We build off of this theory and efficiently approximate this distance on discrete manifolds represented by sparse kernels.

Another line of work in approximations to the Wasserstein distance instead works with tree metrics over the data [120, 115, 10, 179]. These tree methods also linear time however are built in the ambient dimension and do not represent graph distances as well as graph based methods as shown in Sec. 8.5. Averaging over certain tree families can linked to approximating the EMD with a Euclidean ground distance in $\mathbb{R}^n$ [93]. Many trees may be necessary to smooth out the effects of binning over a continuous space as is evident in Figure 7.2. Diffusion EMD uses multiple scales of smooth bins to reduce this effect, thus giving a smoother distance which is approximately equivalent to a snowflake of the L^2 distance on the manifold.

A third line of work considers entropy regularized Wasserstein distances [50, 18, 192]. These methods show that the transportation plan of the regularized 2-Wasserstein distance is a rescaling of the heat kernel accomplished by the iterative matrix rescaling algorithm known as the Sinkhorn algorithm. In particular, Solomon et al. [192] links the time parameter t in the heat kernel H_t to the entropy regularization parameter and performs Sinkhorn scaling with H_t. While applying the heat kernel is efficient, to embed m distributions with the entropy regularized Wasserstein distance is $O(m^2)$ as all pairwise distances must be considered. Next we explore the linear time algorithms based on multiscale smoothing for compute the earth mover's distance.

7.B.1 Multiscale Methods for Earth Mover's Distance

Let a (possibly randomized) transformation T map distributions on Ω to a set of multiscale bins $\boldsymbol{b}$, $T : \mu(\Omega) \to \boldsymbol{b}$, these methods define a T such that

$$W_d(\mu, \nu) \approx \mathbb{E}\|T(\mu) - T(\nu)\|_1. \tag{7.16}$$

Where the approximation, the randomness, and the transform depend on the exact implementation. All of these methods have two things in common, they smooth over multiple scales

Indyk and Thaper [93] presented one of the early methods of this type. They showed that by computing averages over a set of randomly placed grids at dyadic scales. These grids work well for images where the pixels form a discrete set of coordinates.

Shirdhonkar and Jacobs [187] showed how to generalize this work over images to more general bin types, more specifically they showed how wavelets placed centered on a grid could replace the averages in Indyk and Thaper [93]. This work linked the earth mover's distance in the continuous domain to that of the discrete through wavelets in $\mathbb{R}^d$, showing that for some wavelets the EMD approximation was better than the previous grid method. In Diffusion EMD we generalize these wavelets to the graph domain using diffusion wavelets on the graph allowing Wasserstein distances with a geodesic ground distance.

Kolouri et al. [109] in Sliced Wasserstein distances showed how the Wasserstein distance could be quickly approximated by taking the Wasserstein distance along many one dimensional slices of the data. This can be thought of binning along one dimension where the cumulative distribution function is represented along n bins (one for each point) with each bin encompassing one more point than the last.

Le et al. [115] generalized the Sliced Wasserstein distance back to trees, giving a new method based on multi-level clustering where the data is partitioned into smaller and smaller clusters where the bins are the clusters. They demonstrated this method on high dimensional spaces where previous methods that had d^{2^k} bins at scale k were inefficiently using bins. By clustering based on the data, their ClusterTree performs well in high dimensions. However, by clustering they lose the convergence to the Wasserstein distance with Euclidean ground distance of previous methods for efficiency in high dimensions.

Diffusion EMD uses smooth diffusion wavelets over the graph which can give geodesic distances unlike previous multiscale methods. Furthermore, Diffusion EMD selects a number of bins that depends on the rank of the data at each dyadic scale, which can lead to smaller representations depending on the data. This is summarized in Table 7.2, which details these differences.

Table 7.2: Comparison of Multiscale Methods for Earth Mover's Distance

Method	Scales	Bins per scale	Data Dependent Centers	Smooth Bins	Geodesic Distances
Indyk and Thaper [93]	Dyadic	d^{2^k}	No	No	No
Shirdhonkar and Jacobs [187]	Dyadic	d^{2^k}	No	Yes	No
Kolouri et al. [109]	n	1	Yes	No	No
Le et al. [115]	Specified	C^k	Yes	No	No
Diffusion EMD	Dyadic	Rank Dependent	Yes	Yes	Yes

7.C Algorithm Details

In this section we present two algorithms for computing the Diffusion EMD, the Chebyshev approximation method and the interpolative decomposition method. The Chebyshev method is more effective when the number of distributions is relatively small. The interpolative decomposition method is more effective when the number of distributions is large relative to the size of the manifold. We also detail how to subsample the normed space based on the spectrum of $\boldsymbol{P}$ which can be approximated quickly.

First we define the approximate rank up to precision δ of a matrix $A \in \mathbb{R}^{n \times n}$ as:

$$R_\delta(A) := \# \{ i : \sigma_i(A) / \sigma_0(A) \geq \delta \} \tag{7.17}$$

Where $\sigma_i(A)$ is the ith largest singular value of the matrix A. The approximate ranks of dyadic powers of $\boldsymbol{P}^{2^k}$ are useful for determining the amount of subsampling to do at each scale either after an application of the Chebyshev method or during the computation of (approximate) dyadic powers of $\boldsymbol{P}$. We note that based on the density of singular values of $\boldsymbol{P}$, which is quickly computable using the algorithm presented in Dong et al. [56], the approximate rank of all dyadic powers of $\boldsymbol{P}$ can be calculated without computing powers of $\boldsymbol{P}$.

Chebyshev Approximation of Diffusion EMD We first note that $\boldsymbol{P}^{2^k}$ can be computed spectrally using a filter on its eigenvalues. Let $\boldsymbol{M} = \boldsymbol{D}^{1/2}\boldsymbol{P}\boldsymbol{D}^{-1/2}$ be the symmetric conjugate of $\boldsymbol{P}$. Then $\boldsymbol{M}$ is symmetric and has eigenvalues lying in the range $-1 \leq \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_n \leq 1$. $\boldsymbol{M}$ can be decomposed into $\boldsymbol{U\Sigma U}^T$ for orthonormal $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ a diagonal matrix of $[\lambda_0, \lambda_1, \ldots, \lambda_n]$. We then express $\boldsymbol{P}^{2^k}$ as a filter on the eigenvalues of the $\boldsymbol{M}$:

$$\boldsymbol{P}^{2^k} = \boldsymbol{D}^{-1/2}\boldsymbol{U}(\boldsymbol{\Sigma})^{2^k}\boldsymbol{U}^T\boldsymbol{D}^{1/2} \tag{7.18}$$

We compute the first J Chebyshev polynomials of $\boldsymbol{P}^j\boldsymbol{\mu}$ then use the polynomial filter on the eigenvalues $h(\sigma) = \sigma^{2^k}$ to compute diffusions of the distributions, and reweight these diffusion bins as specified in Algorithm 2. This algorithm requires J sparse matrix multiplications of $\boldsymbol{P}$ and $\boldsymbol{\mu}$. For a total time complexity of $\tilde{O}(Jmn)$ when $\boldsymbol{P}$ is sparse.

Interpolative Decomposition Approximation of Diffusion EMD Our second method proceeds by directly approximating multiplication by the matrix $\boldsymbol{P}^{2^k}$. The naive solution of computing dyadic powers of $\boldsymbol{P}$ on the original basis quickly leads to a dense $n \times n$ matrix. Coifman and Maggioni [47] observed that $\boldsymbol{P}^{2^k}$ is of low rank, and therefore multiplication by $\boldsymbol{P}^{2^k}$ can be approximated on a smaller basis. In that work they introduced a method of iteratively reducing the size of the basis using rank-revealing pivoted sparse QR decomposition, then computing $\boldsymbol{P}^{2^{k+1}} = \boldsymbol{P}^{2^k}\boldsymbol{P}^{2^k}$. By reducing the size of the basis and orthogonalizing the basis is kept small and $\boldsymbol{P}^{2^k}$ is kept sparse on its corresponding basis. For sparse $\boldsymbol{P}$ this gives an algorithm that is $O(n \log^2 n)$ for computing dyadic powers of a sparse matrix.

Our algorithm follows the same lines as theirs however we use interpolative decomposition to reduce the size of the basis, and do not do so at early stages where the size of the basis may still be a significant fraction of n. Interpolative decomposition allows for an easy interpretation: we are selecting a well spaced set of points that acts as a basis for higher levels of $\boldsymbol{P}^{2^k}$. For more details on interpolative decomposition we refer the reader to Liberty et al. [128], Bermanis et al. [23]. The algorithm in Coifman and Maggioni [47] also computes powers of $\boldsymbol{P}^{2^k}$ in the original basis by projecting the multiplication in the reduced basis back to the original. We do not bother computing $\boldsymbol{P}^{2^k}\boldsymbol{\mu}$ in the original basis, but instead use the reduced basis directly to compare distributions against. Denote the basis at level k as ϕ_k, which is really a subset of ϕ_{k-1}, then ϕ_k can be thought of as the centers of the bins at level k of our multiscale embedding. We only compare distributions at these representative centers rather than at all datapoints. This is summarized in Algorithm 5, creating an embedding for each distribution whose length is dependent on the rank of $\boldsymbol{P}$.

Sampling the Diffusions at Larger Scales Interpolative decomposition is an integral part of Algorithm 5. However, it can also be useful in reducing the size of the distribution representations of Algorithm 2 which uses the Chebyshev approximation. Without sampling the Chebyshev algorithm centers a bin at every datapoint at every scale. However, for larger scales this is unnecessary, and we can take advantage of the low rank of $\boldsymbol{P}^{2^k}$ with interpolative decomposition. In practice, we use the top 6 largest scales, in fact the distance shows very little sensitivity to small scales (see Fig. 7.5(d)). In fact, this representation can be compressed without a significant loss in performance as shown in Fig. 7.5(b,c). We apply the rank threshold to the spectrum of $\boldsymbol{P}^{2^k}$, which can be computed from the density of states algorithm described above, to determine the number and location of centers to keep. With $\delta = 10^{-6}$ this reduces the number of centers from $60,000$ to $< 4,000$ with a small loss in performance. Rather than selecting a fixed number of centers per scale as in previous methods, this allows us to vary the number of centers per scale as necessary to preserve the rank at each scale.

Time Complexity Here, we assume that $\boldsymbol{P}$ has at most $\tilde{O}(n) = O(n \log^c n)$ nonzero entries. In this case, interpolative decomposition in Algorithm 3 has complexity $O(k \cdot m \cdot n \log n)$, and the randomized version in Algorithm 4 has complexity $O(km + k^2 n)$ [128]. Algorithm 2 has complexity $O(Jmn \log^c n)$ which is dominated by calculation of the J powers of $\boldsymbol{P}^j\boldsymbol{\mu}$. The computation time of Algorithm 5 is dominated by the first randomized interpolative decomposition step. When we set $\gamma = O(\log^c n)$, which controls when the application of RandID occurs, this gives a total time complexity for Algorithm 5 of $\tilde{O}(mn) = O(n \log^{2c} n + mn \log^b n)$, where b and c are constants controlled by γ. As γ in-

Algorithm 3 Interpolative Decomposition

```
Input: $m \times n$ matrix $A$ and number of subsamples $k$ s.t. $k < \min\{m, n\}$.
Output: $m \times k$ matrix $B$ and $k \times n$ matrix $C$ s.t. $\|AP - BC\|_2 \leq \sqrt{4k(n-k)+1}\sigma_{k+1}(A)$, where $B$ consists of $k$ columns of $A$.
Perform pivoted QR decomposition on $A$ s.t. $AP = QR$, where $P$ is a permutation matrix of $A$.
Where Q and R are decomposed as the following with $Q_1$ is a $m \times k$ matrix, $Q_2$ is $m \times (n-k)$, etc.
    $Q = [Q_1 \mid Q_2]; \qquad R = \begin{bmatrix} R_{11} & R_{12} \\ \mathbf{0}_{k \times n-k} & R_{22} \end{bmatrix}$
$B \leftarrow Q_1 R_{11}$
$C \leftarrow [I_k | R_{11}^{-1} R_{12}]$
```

Algorithm 4 Randomized Interpolative Decomposition: RandID($\boldsymbol{A}, j, k, l$)

```
Input: $m \times n$ matrix $A$, random sample sizes $j, l$, and number of subsamples $k$ s.t. $\min\{m, n\} > j > k > l$.
Output: $m \times k$ matrix $B$ and $k \times n$ matrix $C$ s.t. $\|A - BC\|_2 \leq \sqrt{4k(n-k)+1}\sigma_{k+1}(A)$ w.h.p., where $B$ consists of $k$ columns of $A$.
$G_1 \leftarrow \mathcal{N}(0,1)^{j \times m}$
$W \leftarrow G_1 A$
Perform the (deterministic) interpolative decomposition on $W$ in Algorithm 3 to get $\tilde{B}, C$ s.t. $\|W - \tilde{B}D\|_2$ small.
$G_2 \leftarrow \mathcal{N}(0,1)^{l \times j}$
$W_2 \leftarrow G_2 W$
$\tilde{B}_2 \leftarrow G_2 \tilde{B}$
Find the $k$ indices of $\tilde{B}_2$ that are approximately equal to columns in $W_2$, $[i_1, i_2, \ldots, i_k]$.
$\boldsymbol{b} \leftarrow [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_K]$
```

creases, b increases and c decreases. The first term is dominated by the first interpolative decomposition setup, and the second is the calculation of $\boldsymbol{\mu}^{(2^k)}$. It follows that when $m \gg n$, it is helpful to set γ larger for a better tradeoff of b and c. In practice we set $\gamma = n/10$ which seems to offer a reasonable tradeoff between the two steps. For small m in practice Algorithm 2 is significantly faster than Algorithm 5.

7.C.1 Gradients of the Diffusion EMD

In this section we will develop the computation of the gradient of the Diffusion EMD $W_{\alpha,K}(X_i, X_j)$, we will show that its gradient depend only on the gradient of the Gaussian kernel matrix $\boldsymbol{K}_\epsilon$ which is easy to compute. We start by recalling some basic facts about the gradient of matrices,

Algorithm 5 Interpolative Decomposition diffusion densities $T_{id}(\boldsymbol{K}, \boldsymbol{\mu}, K, \alpha, \delta, \gamma)$

```
Input: $n \times n$ graph kernel $\boldsymbol{K}$, $n \times m$ distributions $\boldsymbol{\mu}$, maximum scale $K$, snowflake constant $\alpha$, rank threshold $\delta$, and basis recomputation threshold $\gamma$.
Output: Distribution embeddings $\boldsymbol{b}$
$\boldsymbol{Q} \leftarrow Diag(\sum_i \boldsymbol{K}_{ij})$
$\boldsymbol{K}^{norm} \leftarrow \boldsymbol{Q}^{-1}\boldsymbol{K}\boldsymbol{Q}^{-1}$
$\boldsymbol{D} \leftarrow Diag(\sum_i \boldsymbol{K}^{norm}_{ij})$
$\boldsymbol{M} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{K}^{norm}\boldsymbol{D}^{-1/2}$
$\boldsymbol{\mu}^{(2^0)} \leftarrow \boldsymbol{P}\boldsymbol{\mu} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{M}\boldsymbol{D}^{1/2}\boldsymbol{\mu}$
$\boldsymbol{\mu}_0 \leftarrow \boldsymbol{\mu}$
$n_0 \leftarrow n$
for $k = 1$ to $K$ do
  if $R_\delta(\boldsymbol{M}^{2^k}) < \gamma$ then
    $\boldsymbol{B}_k, \boldsymbol{C}_k \leftarrow \mathrm{RandID}(\boldsymbol{M}_{k-1}, R_\delta(\boldsymbol{M}^{2^k}) + 8, R_\delta(\boldsymbol{M}^{2^k}), 5)$
    $n_k \leftarrow R_\delta(\boldsymbol{M}^{2^k})$
  else
    $\boldsymbol{B}_k, \boldsymbol{C}_k, n_k \leftarrow \boldsymbol{B}_{k-1}, \boldsymbol{I}_{n_{k-1}}, n_{k-1}$
  end if
  $\boldsymbol{M}_k \leftarrow \boldsymbol{C}_k\boldsymbol{M}_{k-1}\boldsymbol{M}_{k-1}^T\boldsymbol{C}_k^T$
  $\boldsymbol{\mu}_k \leftarrow \boldsymbol{C}_k\boldsymbol{\mu}_{k-1}$
  $\boldsymbol{\mu}^{(2^k)} \leftarrow \boldsymbol{D}^{-1/2}\boldsymbol{M}_k\boldsymbol{D}^{1/2}\boldsymbol{\mu}_k$
  $\boldsymbol{b}_{k-1} \leftarrow 2^{(K-k-1)\alpha}(\boldsymbol{\mu}^{(2^k)} - \boldsymbol{C}_k\boldsymbol{\mu}^{(2^{k-1})})$
end for
$\boldsymbol{b}_K \leftarrow \boldsymbol{\mu}^{(2^K)}$
$\boldsymbol{b} \leftarrow [\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_K]$
```

1. For a matrix P and a vector μ we have $\nabla\|P\mu\|_1 = \mathrm{sign}(P\mu)\mu^T\nabla P$.

2. For an invertible matrix P, the gradient of the inverse of P is $\nabla(P^{-1}) = -P^{-1}\nabla(P)P^{-1}$.

3. Let $n \in \mathbb{N}^*$ the gradient of P^n is

$$\nabla P^n = \sum_{\ell=0}^{n-1} P^\ell \nabla(P) P^{n-\ell-1}.$$

Using the definition of the Diffusion EMD, we have

$$\begin{aligned}\nabla W_{\alpha,K}(X_i,X_j) =& \nabla\|\boldsymbol{P}_\epsilon^{2^K}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_i)\|_1+\sum_{k=0}^{K-1}2^{-(K-k-1)\alpha}\nabla\|\left(\boldsymbol{P}_\epsilon^{2^{k+1}}-\boldsymbol{P}_\epsilon^{2^k}\right)(\boldsymbol{\mu_j}-\boldsymbol{\mu}_i)\|_1\\ =&\mathrm{sign}\left(\boldsymbol{P}_\epsilon^{2^K}(\boldsymbol{\mu}_j-\boldsymbol{\mu}_i)\right)(\boldsymbol{\mu}_j-\boldsymbol{\mu}_i)^T\nabla(\boldsymbol{P}_\epsilon^{2^K})+\sum_{k=0}^{K-1}2^{-(K-k-1)\alpha}\\ &\mathrm{sign}\left(\boldsymbol{P}_\epsilon^{2^{k+1}}-\boldsymbol{P}_\epsilon^{2^k}\right)(\boldsymbol{\mu_j}-\boldsymbol{\mu}_i))^T\nabla(\boldsymbol{P}_\epsilon^{2^{k+1}}-\boldsymbol{P}_\epsilon^{2^k}).\end{aligned}$$

The last formula tells us that the gradient of $W_{\alpha,K}(X_i,X_j)$ depends only on the gradient of the powers of $\boldsymbol{P}_\epsilon$ which in turn can be expressed in terms of the gradients $\boldsymbol{P}_\epsilon$. We have:

$$\begin{aligned}\nabla P_\epsilon &= \nabla(D_\epsilon^{-1}M_\epsilon D_\epsilon^{-1})\\ &= \nabla(D_\epsilon^{-1})M_\epsilon + D_\epsilon^{-1}\nabla(M_\epsilon)\\ &= -D_\epsilon^{-1}\nabla(D_\epsilon)D_\epsilon^{-1}M_\epsilon + D_\epsilon^{-1}\nabla(M_\epsilon)\\ &= -D_\epsilon^{-1}\nabla(\mathrm{diag}(M_\epsilon\mathbf{1}))D_\epsilon^{-1}M_\epsilon + D_\epsilon^{-1}\nabla(M_\epsilon)\\ &= -D_\epsilon^{-1}\mathrm{diag}(\nabla(M_\epsilon)\mathbf{1})D_\epsilon^{-1}M_\epsilon + D_\epsilon^{-1}\nabla(M_\epsilon),\end{aligned}$$

therefore in order to compute ∇P_ϵ we need to compute ∇M_ϵ,

$$\begin{aligned}\nabla M_\epsilon &= \nabla(Q_\epsilon^{-1}K_\epsilon Q_\epsilon^{-1})\\ &= \nabla(Q_\epsilon^{-1})K_\epsilon Q_\epsilon^{-1} + Q_\epsilon^{-1}\nabla(K_\epsilon)Q_\epsilon^{-1} + Q_\epsilon^{-1}K_\epsilon\nabla(Q_\epsilon^{-1})\\ &= -Q_\epsilon^{-1}\nabla(Q_\epsilon)Q_\epsilon^{-1}K_\epsilon Q_\epsilon^{-1} + Q_\epsilon^{-1}\nabla(K_\epsilon)Q_\epsilon^{-1} - Q_\epsilon^{-1}K_\epsilon Q_\epsilon^{-1}\nabla(Q_\epsilon)Q_\epsilon^{-1}\\ &= -Q_\epsilon^{-1}\nabla(\mathrm{diag}(K_\epsilon\mathbf{1}))Q_\epsilon^{-1}K_\epsilon Q_\epsilon^{-1} + Q_\epsilon^{-1}\nabla(K_\epsilon)Q_\epsilon^{-1} - Q_\epsilon^{-1}K_\epsilon Q_\epsilon^{-1}\nabla(\mathrm{diag}(K_\epsilon\mathbf{1}))Q_\epsilon^{-1}\\ &= -Q_\epsilon^{-1}\mathrm{diag}(\nabla(K_\epsilon)\mathbf{1})Q_\epsilon^{-1}K_\epsilon Q_\epsilon^{-1} + Q_\epsilon^{-1}\nabla(K_\epsilon)Q_\epsilon^{-1} - Q_\epsilon^{-1}K_\epsilon Q_\epsilon^{-1}\mathrm{diag}(\nabla(K_\epsilon)\mathbf{1})Q_\epsilon^{-1}\end{aligned}$$

hence the gradient of M_ϵ can be written only in terms of the gradient of K_ϵ. This process shows that the gradient of the EMD diffusion can be expressed in terms of the gradient of the Gaussian kernel only, which may make it useful in future applications where fast gradients of the earth mover's distance are necessary. For example, in distribution matching where previous methods use gradients with respect to the Sinkhorn algorithm [69], which scales with $O(n^2)$. In this application, computation of gradients of Diffusion EMD would be $O(n)$. What makes this possible is a smooth kernel and smooth density bins over the graph. In contrast to Diffusion EMD, multiscale methods that use non-smooth bins have non-smooth gradients with respect to K_ϵ, and so are not useful for gradient descent type algorithms. We leave further investigation into the gradients of Diffusion EMD to future work.

7.D Experiment Details

We ran all experiments on a 36 core Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz with 512GB of RAM. We note that our method is extremely parallelizable, consisting of only matrix operations, thus a GPU implementation could potentially speed up computation. For fair comparison we stick to CPU implementations and leave GPU acceleration to future work. We provide custom implementations of all methods in the `DiffusionEMD` package, which is available on Github. We base our Sinkhorn implementation on that in the python optimal transport package (POT) [68], and our tree methods off of the sklearn nearest neighbors trees [159].

7.D.1 Metrics used

P@10. P@10 is a metric that measures the overlap in the 10-nearest neighbors between the nearest neighbors of the EMD method and the ground truth nearest neighbors, which is calculated using exact OT on the geodesic ground distance. P@10 is useful for distinguishing the quality of distance calculations locally, whereas the next metric measures the quality more globally. This may be particularly important in applications where only the quality of nearest neighbors matters as in Sect 7.4.6, or in applications of Wasserstein nearest neighbor classification [93, 10].

Spearman-ρ. The Spearman's rank correlation coefficient (also known as Spearman-ρ) measures the similarity in the order of two rankings. This is useful in our application because while the 1-Wasserstein and 2-Wasserstein may have large distortion as metrics, they will have a high Spearman-ρ in all cases. Spearman-ρ is defined for two rankings of the data where d_i is the difference in the rankings of each for each datapoint i as

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

This is useful for quantifying the global ranking of distributions in the graph, which shows the more global behavior of EMD methods. Since locally manifolds with low curvature look very similar to $\mathbb{R}^d$, where d is the intrinsic dimension of the manifold, for distributions very close together on the manifold using a Euclidean ground distance may be acceptable, however for distributions far away from each other according to a geodesic ground distance, approximation with a Euclidean ground distance may perform poorly.

7.D.2 Line Example

In Fig 7.2 we used a cluster tree with 4 levels of 4 clusters for the cluster tree, and a quad tree of depth 4. These are mostly for illustrative purposes so we there was no parameter search for ClusterTree or QuadTree. As the number of trees gets larger ClusterTree and QuadTree start to converge to some smooth solution. ClusterTree has a number of bumps caused by the number of clusters parameter that do not disappear even with many trees. Quadtree converges to a smooth solution as the number of trees gets large in this low dimensional example.

7.D.3 Swiss Roll

On the swiss roll example we compared Diffusion EMD to ClusterTree, QuadTree, and the convolutional Sinkhorn method. Here we describe the parameter setup in each of these methods for Figure 7.3, 7.4, and 7.5.

For Fig. 7.3 we chose parameters that approximately matched the amount of time between methods on this dataset to the Diffusion EMD with default settings using the Chebyshev algorithm. This was 150 cluster trees of depth 8 with 3 cluster, and 20 quad trees of depth 4. We noticed that the speed of quadtree diminishes exponentially with dimension.

For Fig. 7.4 we did a grid search over parameters for QuadTree and Cluster Tree to find the best performing settings of depth and number of clusters. We searched over depth $d \in [2 \ldots 10]$ and number of clusters $C \in [2 \ldots 5]$ for ClusterTree. We found a depth of 4 for QuadTree and 3 Clusters of depth 5 for ClusterTree gave the best tradeoff of performance vs. time on this task. These parameters were tried while fixing the number of trees to 10. We fixed this depth and number of clusters for subsequent experiments varying the number of trees in the range $[1 \ldots 100]$ for QuadTree and in the range $[1 \ldots 15]$ for ClusterTree. The maximum of this range exceeded the time allotment of Diffusion EMD in both cases. For the convolutional Sinkhorn method we fixed parameter $t = 50$ for H_t, and fixed the maximum number of iterations in the range $[1 \ldots 100]$. This took orders of magnitude longer than the multiscale methods in all cases, with comparable performance to Diffusion EMD on the P@10 metric and better performance using the Spearman-ρ metric. We note that a similar multi-step refinement scheme as implemented in Backurs et al. [10] could be used here.

7.D.4 MNIST

To construct the Spherical MNIST dataset [45], we projected the 70,000 images in classical MNIST onto a spherical graph containing 25,088 nodes. Every pixel in each MNIST image was projected onto four nodes on the northern hemisphere. These projections were treated as signals over a spherical graph.

We ran Diffusion EMD, ClusterTree, and QuadTree on this dataset to generate embeddings in L1 space. Diffusion EMD was run with the diffusion wavelet approximations described in the paper, using an epsilon value of 0.001. To best compare accuracy, we chose parameters for QuadTree and ClusterTree that approximately matched the runtime of Diffusion EMD. For QuadTree, we used a depth of 20 and 10 clusters. For ClusterTree, we used a depth of 5. However, we note that both ClusterTree and QuadTree experienced a variance of 10-30 minutes between subsequent runtimes with the same parameters, probably due to differences of random initialization. We did not run Convolutional Sinkhorn on this dataset, as the projected runtime exceeded 24 hours.

We calculated a ground truth EMD between 100 of the projected MNIST digits using the exact EMD implementation of Python Optimal Transport. More accurate comparisons might be obtained using more points, but computation of this 100 by 100 distance matrix took over 24 hours, necessitating severe subsampling, and once again highlighting the need for fast and accurate approximations of EMD.

For each algorithm, we obtained an accuracy score by running a kNN classifier on the embeddings using an even train/test split considering only the nearest neighbors. We then computed the Spearman ρ and P@10 score between each method and the exact EMD ground truth.

7.D.5 Single-cell COVID-19 Dataset

One hundred sixty eight patients with moderate to severe COVID-19 [140] were admitted to YNHH and recruited to the Yale Implementing Medical and Public Health Action Against Coronavirus CT (IMPACT) study. From each patient, blood samples were collected to characterize patient cellular responses across timepoints to capture the spectrum of disease. In total, the composition of peripheral blood mononuclear cell (PBMC) was measured by a myeloid-centric flow cytometry on 210 samples. Finally, clinical data was extracted from the electronic health record corresponding to each biosample timepoint to allow for clinical correlation of findings as shown previously. In the main analysis, poor or adverse outcomes were defined by patient death, while good outcomes were defined by patient survived. In order to analyze the 22 million cells generated in this study, we performed k-means clustering, setting k to 27,000, and took the resultant cluster centroids as a summarization of the cellular state space as done previously in [111].

To test the quality of the organization of patients using EMD methods, we test the Laplacian smoothness on the 10-nearest neighbors graph created for each method. The Laplacian smoothness of a function on the nodes is denoted $f^T L f$ where $L = D - A$ is the (combinatorial) Laplacian of the adjacency matrix A. This is equivalent to computing the following sum:

$$\sum_{(i,j) \in E} (f_i - f_j)^2$$

where E is the edge set of the kNN graph. This measures the sum of squared differences of f along the edges of the kNN graph, which measures the smoothness of f over the graph.

Various cell types have been shown to be associated with mortality from COVID-19 infection, including CD16$^+$ Neutrophils, T cells and non-classical monocytes. Previously, T cells-to-CD16$^+$ Neutrophils ratios have been reported to be predict of outcome, with low ratios predicting mortality and high ratios predicting survival [124]. Even outside of ratios, the absolute counts to T cells [43] and CD16$^+$ Neutrophils [141] have been shown to be associated with mortality. Finally, non-classical monocytes have also been shown to be enriched in patients with more severe outcomes [161], further supporting our findings.

Chapter 8

Unbalanced Transport on Graphs and Geometric Domains

8.1 Introduction

The task of comparing probability distributions is applicable to a wide variety of machine learning problems, giving rise to popular ϕ-divergences such as the Kullback-Leibler (KL), Hellinger, or total variation divergences, which all compare the ratio between probability density functions of the two distributions, while ignoring the underlying geometry of their support (or on which they are defined). The Earth Mover's Distance (EMD), also known as the Monge-Kantorovich or Wasserstein Distance, explicitly takes into account this underlying geometry via a domain-specific ground distance, which yields many advantages when processing empirical probability distributions [165, 10]. Here, we show that earth mover's distances are useful in a new domain: that of graph signals. In modern relational machine learning, we encounter large graphs that arise via interactions between entities in many domains [24, 184]. Features of such entities can be considered as signals on the graph. In many cases, entities can also be modeled directly as signals on existing graphs between connected concepts or observations. For such signals, which often tend to be noisy, we propose a new unbalanced graph earth mover's distance, and use it to organize the signals and determine relationships between them.

Since modern knowledge graphs can contain tens (Cora) [24] to hundreds (SNOMED-CT) [184] of thousands of nodes, there is a great need for this measure to be highly computationally efficient. However, while the Wasserstein Distance is intuitively attractive, it presents computational challenges and has only become feasible for large problems with approximation, which is the focus of this work. First, based on the recent diffusion EMD method [200], we show that an efficient unbalanced EMD between signals can be computed as the difference between graph convolutions of the signal with multiscale graph kernels — which can be computed in linear time with convergence guarantees without solving an optimization problem. We call our distance *unbalanced diffusion earth mover's distance (UDEMD).*

While previous work on Wasserstein distance embedding mostly focused on its relation to the balanced optimal transport problem [93, 187, 75, 115, 200], we propose an unbalanced Wasserstein embedding approach between large number of distributions defined as signals on graphs. Since signals and particularly graph signals tend to be noisy, an *unbalanced* transport, which can choose not to transport parts of the data space when it is inefficient to do so, leads to more robust distances between graph distributions that are less sensitive to outliers in the signal.

We apply our unbalanced diffusion EMD to medical knowledge graphs using Systemized Nomenclature of Medicine - Clinical Terms (SNOMED-CT) [184]. Here, we take patient records obtained from the Medical Information Mart for Intensive Care version III (MIMIC-III), a database containing information for patients who were admitted to the hospital at a tertiary academic medical center in Boston, MA, USA and required intensive care unit level of care [98], and extract concepts from discharge summaries (notes written by physicians to summarize a patient's hospital stay) via MetaMap, a tool for discovering concepts in biomedical text [9]. We then model the patient (collection of concepts extracted from the discharge summary) as a signal on the SNOMED-CT knowledge graph. We show that unbalanced diffusion EMD can be used to find meaningful distances between patients such that a patient-kernel can be created which successfully clusters patients into different categories, and allows us to find relationships between patient features.

We also show application of this approach to single cell RNA-sequencing data where we can model both cells as signals on gene interaction graphs or genes as signals on cell-similarity graphs. In cases where the gene regulatory network is well known, researchers [92, 15] have shown that affinity between cells can be computed as an earth mover's distance. We show that UDEMD runs orders of magnitude faster than the Sinkhorn method, and network simplex methods used in those works, while maintaining accuracy. On the other hand, in cases where the gene regulatory network is not well known, we can derive groupings of genes that function similarly by modeling genes as expression values over single cells. This type of signal is notoriously noisy as it is afflicted by dropout [167] (i.e., the measurement technology effectively captures 10 percent of the transcriptome). Here we show that the UDEMD provides robust distances that recapitulate ground truth gene groupings in single cell data from peripheral blood mononuclear cells (PBMCs).

The remainder of the paper is structured as follows. In section 8.2 we review graph signal processing, the Wasserstein metric and the unbalanced Wasserstein metric. In section 8.3, we formulate the problem of embedding a robust earth mover's distance over a graph into L^1. In section 8.4 we formulate the embedding of the unbalanced diffusion earth mover's distance both theoretically and algorithmically, and in section 8.5 we empirically establish the efficiency, accuracy, and robustness of UDEMD.

8.2 Preliminaries

In this section we review graph signal processing, the Wasserstein metric and embedding based methods for approximating it, and unbalanced optimal transport.

Notation We say that two items A and B are equivalent if there exist $c, C > 0$ such that $cA \leq B \leq CA$, and we denote $A \simeq B$. When two metrics are equivalent, this is also known as topological equivalence.

Graph Signal Processing We consider a graph $G = (V, E)$ equipped with a family of graph signals $\mathcal{F} = \{f_i : V \to \mathbb{R}; i \in 1, \dots m\}$. In this work we will show how to build a kernel over this family of functions that measures the similarity between signals on the graph. One way to do this would be to measure the sum of squared edge weights of the two signals known as Tikhonov regularization i.e.

$$\sum_{e_{ab} \in E} ((f_i - f_j)(a) - (f_i - f_j)(b))^2 \tag{8.1}$$

or with an L^1 norm, known as graph total variation [220]. However, these two functions only penalize local differences in the signals, ignoring differences outside of the immediate neighborhood of each node. Here, we perform unbalanced transport that takes into account larger scales and has an intuitive interpretation as $n \to \infty$, namely that of the Wasserstein metric over the geodesic metric.

The Wasserstein Metric The Wasserstein metric is a notion of distance between two measures μ, ν on a measurable space Ω endowed with a metric $d(\cdot, \cdot)$, which is referred as the ground distance. The primal formulation of the p-Wasserstein distance W_d, also known as the earth mover's distance (EMD), is defined as:

$$W_d^p(\mu, \nu) := \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\Omega \times \Omega} d(x, y)^p \pi(dx, dy) \right)^{1/p}, \tag{8.2}$$

where $\Pi(\mu, \nu)$ is the set of joint probability distributions π on the space $\Omega \times \Omega$, such that for any subset $\omega \subset \Omega$, $\pi(\omega \times \Omega) = \mu(\omega)$ and $\pi(\Omega \times \omega) = \nu(\omega)$. Also of interest is the entropy regularized Wasserstein distance [50], which reduces the computation to $O(n^2)$. This algorithm is extremely parallelizable, and works quite well even for a small number of iterations [10], and there are many works investigating how to scale this to larger problems. However, when comparing a large number of signals, we must solve the optimization for each pair of signals, i.e. $O(m^2)$ optimizations. For this reason we turn to methods that approximate the dual of the Wasserstein metric.

For $p \leq 1$, for any metric d, d^p is still a metric and for separable Ω, the Kantorovich-Rubinstein dual holds:

$$W_d^p(\mu, \nu) = \sup_{f : |f(x) - f(y)| \leq d(x,y)^p} \int_\Omega f(x) \mu(dx) - \int_\Omega f(y) \nu(dy). \tag{8.3}$$

Here, f is commonly known as the witness function, particularly in the context of integral probability measures, as it "witnesses" the differences between μ and ν over Ω. Since it is in general difficult to optimize over the entire space of p-Hölder functions, many works optimize the cost over a modified family of functions such as functions parameterized by neural networks [8, 84, 199], functions defined over trees [93, 115], and wavelet bases [187, 75].

Unbalanced Optimal Transport There are numerous formulations of unbalanced optimal transport both to accommodate problems with unequal masses and to provide robustness to outlier points [165, 11]. In general these can be formulated as a mixture between a pure optimal transport problem and a ϕ-divergence:

$$\mathrm{UW}^\alpha(\mu, \nu) = \inf_{\mu', \nu'} \left\{ \mathrm{W}^\alpha(\mu', \nu') + \lambda_\mu D_\phi(\mu || \mu') + \lambda_\nu D_\phi(\nu || \nu') \right\}. \tag{8.4}$$

Where λ_μ, λ_ν control the relative cost of mass creation / destruction compared to transportation, and D_ϕ is a ϕ-divergence, also known as an f-divergence or a Csiszár divergence, which is any divergence that can be formulated as $D_\phi(P||Q) = \int_\Omega \phi\left(\frac{dP}{dq}\right) dQ$, for some convex ϕ such that $\phi(1) = 0$. Common choices of ϕ are $\phi(x) := x \log x$ for the KL divergence or $\phi(x) := \frac{1}{2}|x - 1|$ for the total variation distance. In the optimal transport literature, most often considered is the KL-divergence formulation which can be solved efficiently in the case of entropic regularized problem as shown in [44, 129], but cannot be optimized stochastically as is possible in the balanced case, limiting scalability [77, 66]. In the exact case, the TV-unbalanced problem can be solved by adding a "dummy point" that is connected to every point with equal cost [32, 160]. However, adding a dummy point removes the metric structure necessary for dual-based Wasserstein distances. Recently [150] showed that the TV-unbalanced problem can be solved through cost truncation, which is the variant of unbalanced transport considered in this work. We note that for this case, the total variation unbalanced Wasserstein distance can be simplified with $\lambda = \min\{\lambda_\mu, \lambda_\nu\}$ as

$$\text{TV-UW}^\alpha(\mu, \nu) = \inf_s \left\{ \mathrm{W}^\alpha(\mu + s, \nu) + \lambda \mathrm{TV}(\mu + s, \mu) \right\}. \tag{8.5}$$

This simplification is possible as the total variation is symmetric. Intuitively, we can think of equation 8.5 as minimizing over the "teleporting" mass s, that is too costly to transport.

It is not immediately obvious that this is efficiently computable. We will show, however, that there is an embedding of distributions to vectors where the the L^1 distance between vectors is equivalent to the TV-UW between the distributions.

8.3 Problem Setup

We consider a graph $G = (V, E)$ equipped with a family of graph signals $\mathcal{F} = \{f_i : V \to \mathbb{R}; i \in 1, \dots m\}$. Denote the (sparse) graph adjacency matrix $\boldsymbol{A}$. Then the associated (lazy) Markov kernel $\boldsymbol{P}$ is defined as $\boldsymbol{P} := 1/2(\boldsymbol{D}^{-1}\boldsymbol{A} + \boldsymbol{I})$ where $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{D}$ is the degree matrix of $\boldsymbol{A}$. We will assume $|E| > |V|$ which is true for all connected graphs and refer to $n = |V|$ and $m = |\mathcal{F}|$. In this work we will show how to build a kernel over this family of functions that measures the similarity between signals over the graph at various scales, that can then be used to derive a robust earth mover's distance between these signals.

We will start with a few desirable properties of this earth mover's distance function: 1) satisfies the properties of a distance, 2) robustness to noise in terms of small perturbations of values over graph nodes, 3) efficiency and scalability for large graphs that arise in modern machine learning contexts.

A naive method for comparing signals on graphs is to simply take an L^p pointwise difference between graph signals. Another method known as graph total variation [220] takes into account essentially one step of smoothing via the graph adjacency matrix and then takes an L_1 distance. However, this method would not be sensitive to perturbations of graph signals, either in terms of 1) additive noise, 2) outlier components of the signal. As shown in Figure 8.1a, additive noise can interfere with pointwise distance metrics. Thus we turn to a diffusion-based formulation of EMD from [200] which effectively smooths signals before comparison.

Smoothing the signal f on the graph via a Markov graph kernel $\boldsymbol{P}$, i.e., $f' = \boldsymbol{P}^t f$ would yield a less noisy signal due to local vertex averaging or equivalently, spectral low pass filtering. However, how much to smooth this signal is unclear. To resolve this we turn to the recently proposed Diffusion EMD [200], which uses multi-scale heat kernels and is defined as follows:

Definition 8.3.1. Diffusion EMD The Diffusion Earth Mover's Distance between two signals μ, ν as

$$\mathrm{DEMD}_\alpha(\mu, \nu) := \sum_{v \in V} \sum_{k=0}^{O(\log|V|)} \|g_{\alpha,k}(\mu(v)) - g_{\alpha,k}(\nu(v))\|_1 \tag{8.6}$$

where $0 < \alpha < 1/2$ is a meta-parameter used to balance long- and short-range distances, which in practice is set close to 1/2, and

$$g_{\alpha,k}(\mu(v)) := 2^{-(K-k-1)\alpha}\big(\boldsymbol{\mu}^{(k+1)} - \boldsymbol{\mu}^{(k)}\big) \tag{8.7}$$

Where $\boldsymbol{\mu}^{(t)}$ is short for $\boldsymbol{\mu}^{(t)}(v) = (\boldsymbol{P}^{2^t}\mu)(v)$

Intuitively, we can think about this as looking at the L^1 differences of the signals at multiple scales, where a scale represents an amount of blur, or low-pass filtering of the signal. Distances of this type have been explored in the setting of continuous manifolds in [121] drawing on work on generalized heat kernels on manifolds [82].

A second type of noise that may occur in real signals could be outliers as shown in Fig. 8.1, while smoothing can diminish the magnitude of outliers, they can add a lot of spurious cost to EMD measures as they require transport (see blue arrow in Fig. 8.1). We turn to the unbalanced transport explained in the next section to create robustness to such outliers.

8.4 Unbalanced Diffusion Earth Mover's Distance

While diffusion EMD presented in [200] can provide an earth mover's distance between graph signals, its formulation is not motivated by considering noisy signals on graphs, but rather geared to avoid high dimensional density estimation by considering intrinsic geometry defined via sparse affinity matrices constructed over input data. Here, we focus on utilizing EMD to organize graph signals. Therefore, we are interested in distances that are immune to outlier spikes in the signals. While the multiscale smoothing proposed in [200] is effective in handling noisy perturbation of the signals, it is less effective at dealing with outlier vertex components of the signal. However, as we show here, the construction can be adapted to consider unbalanced transport, which is essentially based on the idea that a more faithful earth mover's distance is given by an optimal transport in which we ignore some of the mass – particularly, mass that requires large transport costs. To incorporate this idea in the dual formulation used in Def. 8.3.1, we decouple the scale selection (i.e., $k = 1, \dots, \log n$) from the size of the graph, an instead set it to ignore larger scales, which are also more difficult to compute. This yields the Unbalanced Diffusion EMD (UDEMD) defined below, which is topologically equivalent to the total variation unbalanced Wasserstein distance.

Definition 8.4.1. Unbalanced Diffusion EMD The Unbalanced Diffusion Earth Mover's Distance (UDEMD) between two signals μ, ν as

$$\mathrm{UDEMD}_{\alpha,K}(\mu,\nu) := \sum_{v \in V} \sum_{k=0}^{K} \| g_{\alpha,k}(\mu(v)) - g_{\alpha,k}(\nu(v)) \|_1 \tag{8.8}$$

where α and $g_{\alpha,k}$ are defined as in Def. 8.3.1, K is the maximum scale considered, which relates to the unbalancing threshold (see Fig. 8.2 and discussion below in Sec. 8.4.1).

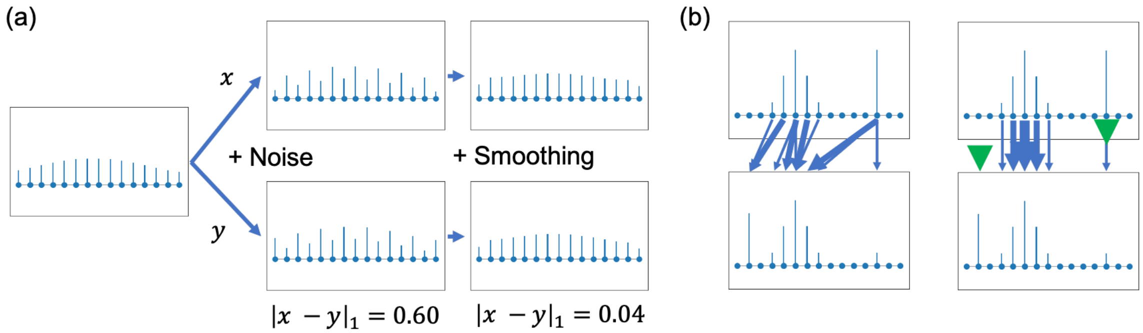


Figure 8.1: (a) Illustrates effect of smoothing component of UDEMD on noise. (b) Illustrates balanced and unbalanced transport maps.

8.4.1 Equivalence to (Unbalanced) Wasserstein Distance

In [150], it was shown that truncated-cost optimal transport distances were equivalent to unbalanced Wasserstein distances, and that they are useful in outlier detection. However, in [150] the proposed implementation was done via truncated the Sinkhorn-OT. Here we show a similar result for the Unbalanced Diffusion EMD from Def. 8.4.1, i.e., showing that with scale truncation it is equivalent an unbalanced Wasserstein distance. We first adapt Theorem 3.1 from [150] in the following Lemma 8.4.1, which will in turn be combined with Lemma 8.4.2 to yield this result.

Lemma 8.4.1. *The Wasserstein distance with a truncated ground distance* $d_\lambda(x, y) = \min\{\lambda, d(x, y)\}$ *for some constant* λ *and distance* d *is equivalent to a total variation unbalanced Wasserstein distance for some constant* λ*, i.e.,*

$$W_{d_\lambda}(\mu, \nu) = \inf_s \{W_d(\mu + s, \nu) + \lambda TV(\mu + s, \mu)\}. \tag{8.9}$$

Intuitively, any mass transported here (captured by s) at the truncated cost level λ, can be equivalently seen as being destroyed at the input point for cost $\lambda/2$ and created at the target point $\lambda/2$ for a total cost of λ.

The theory developed in [200] assumed that the support of the considered distributions was a closed Riemannian manifold. In such a case, $\mathrm{DEMD}_\alpha(\mu, \nu)$ from Def. 8.3.1 will converge to a distance that is equivalent to the Wasserstein distance defined with the geodesic distance on the manifold. This theory referenced work from [121] that showed the convergence of a similar multiscale distance on continuous Riemannian manifolds. The following lemma extends this theory (together with results from [121]) to show that, more generally, UDEMD from Def. 8.4.1 will converge to a Wasserstein distance where the ground distance is a thresholded geodesic.

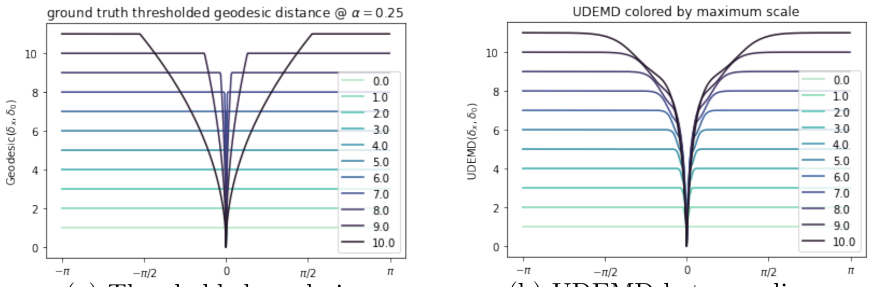
Lemma 8.4.2. $\mathrm{UDEMD}_{\alpha,K}(\mu, \nu)$ *is equivalent to a Wasserstein distance* $W_{d_\lambda}(\mu, \nu)$*, defined as in Lemma 8.4.1, with the ground distance being a truncated geodesic distance on the manifold, i.e.,* $d_\lambda(x, y) = \min\{\lambda, \rho(x, y)\}$ *for* $\lambda > 0$*.*

Proof. We present a proof sketch here and leave the full proof to the supplement, the main part of the proof follows the same lines as in Corollary 3.1 of [200]. In equation 8.6, an anisotropic kernel $\boldsymbol{P}$ is used, which can be shown to converge to the Heat kernel on a Riemannian manifold (see [47], Prop.3). In [121], it is shown that the construction of equation 8.6 using the Heat Kernel will converge to a metric that is equivalent to the Wasserstein with ground distance $\min\{1, \rho(x, y)^{2\alpha}\}$, where ρ is the geodesic on the manifold. Because the metrics $\min\{1, \rho(x,y)^{2\alpha}\}$ and $\min\{\lambda, \rho(x,y)^{2\alpha}\}$ are equivalent (under the equivalence notion in [200]) for $\lambda > 0$, the Wasserstein distances induced by these metrics are also equivalent. □

By combining together Lemmas 8.4.1 and 8.4.2, we have that the UDEMD metric from Def. 8.4.1 is equivalent to an unbalanced optimal transport metric. Formally, using the equivalence notation from [200], we have

$$\mathrm{UDEMD}_{\alpha,K}(\mu,\nu) \simeq \inf_{s} \{W_\rho(\mu + s, \nu) + \lambda TV(\mu + s, \mu)\} \tag{8.10}$$

We note that while our result here establishes a relation between these two metrics, it does not directly quantify the relation between the λ and K. We leave careful theoretical and rigorous study of this relation to future work[1], but mention here that we observe empirically, as shown in Fig. 8.2, the choice of K indeed acts in a similar way to the threshold λ on the ground distance. For more discussion of this relation, we refer to the supplement.



(a) Thresholded geodesic

(b) UDEMD between diracs

Figure 8.2: On a ring graph $n = 500$ compares the UDEMD to the thresholded ground distance, this suggests that UDEMD closely approximates the thresholded ground distance with $\lambda \approx 2^K$.

8.4.2 An Efficient UDEMD Algorithm

To compute the UDEMD defined in Def. 8.4.1, we present Algorithm 6 with time complexity $O(2^K m|E|)$, which is similar to algorithms used in graph neural networks.

Our algorithm scales well with the size of the graph, the number of distributions m and number of points n, but poorly with the maximum scale K. However, we note that

[1] We deem the required nuanced treatment of scaling constants in equivalence bounds out of scope here

Algorithm 6 Unbalanced diffusion EMD embedding UDEMD$(\boldsymbol{A}, \boldsymbol{\mu}, K, \alpha) \rightarrow \boldsymbol{b}$

```
**Input:** $n \times n$ graph adjacency $\boldsymbol{A}$, $n \times m$ distributions $\boldsymbol{\mu}$, maximum scale $K$, and snowflake constant $\alpha$.
**Output:** $m \times (K+1)n$ distribution embeddings $\boldsymbol{b}$
$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$
$\boldsymbol{\mu}^{(2^0)} \leftarrow \boldsymbol{\mu}$
for $k = 1$ to $K$ do
  $\boldsymbol{\mu}^{(2^k)} \leftarrow \boldsymbol{P}^{2^k} \boldsymbol{\mu}^{(2^{k-1})}$
  $\boldsymbol{b}_{k-1} \leftarrow 2^{(K-k-1)\alpha}(\boldsymbol{\mu}^{(2^k)} - \boldsymbol{\mu}^{(2^{k-1})})$
end for
$\boldsymbol{b}_K \leftarrow \boldsymbol{\mu}^{(2^K)}$
$\boldsymbol{b} \leftarrow [\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_K]$
$\tilde{\boldsymbol{b}} \leftarrow \text{Subselect}(\boldsymbol{b})$
```

already in Def. 8.3.1, originating from [200], the maximum scale considered for DEMD was of order $O(\log |V|)$, derived from the convergence of the heat kernel to its steady state (i.e., given by $\psi_0 \simeq \boldsymbol{P}^{2^K}$ for $K = O(\log |V|)$). Such scale would still require $O(n)$ matrix-vector products, or $O(\log n)$ dense matrix multiplications. Here, on the other hand, we decouple the tuning of K and find much smaller maximum scales suffice, and in fact (as discussed in Sec. 8.4.1) correspond to a well characterized unbalanced earth mover's distance on the underlying geometry of the graph. This leads to our algorithm here (unlike the one in [200]) emphasizing preferable scaling properties for small K, while enabling implementation on GPU.

In practice, the bottleneck for computation of the UDEMD between many distributions is not computing the embeddings but comparing embeddings. Previous work has made use of locality sensitive hashing [93], subsampling of significant elements [187], random sampling [115], and interpolative decomposition [200] to speed up nearest neighbor search in Wasserstein space [10]. While all of these methods are compatible with UDEMD, we stick to random sampling over the nodes for simplicity.

8.5 Results

In this section, we show that UDEMD is an efficient and robust method for measuring distances between graph signals and then using the distances to find embeddings and organization of the signals (often entities such as patients).

Experimental setup We compare UDEMD to a GPU implementation of numerically stabilized Sinkhorn optimization that includes minibatching of sets of distributions. However, despite this, this method runs out of memory when there are beyond 10,000 nodes in the graph. We note that all of these methods of this type require solving m^2 optimizations, even when looking for nearest neighbors. For discussion of other methods of this type and implementation details we refer to the supplement. Unless otherwise noted, we set $K = 4$ and $\alpha = 0.5$.

Spherical data test case To test the speed and robustness of UDEMD we present a dataset where we have knowledge of the intrinsic ground distances and where we can vary the number of points and distributions. For this dataset we sample m Gaussian-like functions uniformly distributed on the unit sphere with 10 points each for a total of $n = 10m$ points (See supplement for more details). We add a random noise spike at a random location on the sphere to check robustness to this type of noise. The goal is to predict the neighboring distributions on the sphere. We find that UDEMD is significantly more scalable and find that there is a sweet spot in terms of K at $K = 4$ for this dataset. The UDEMD with $K = 4$ performs significantly better than the balanced DEMD case with this type of noise. This supports the claim that setting $K \ll O(\log n)$ is beneficial in real world datasets. UDEMD also outperforms the graph-TV distance as it is both faster and more accurate at $K = 1$, and more accurate overall.

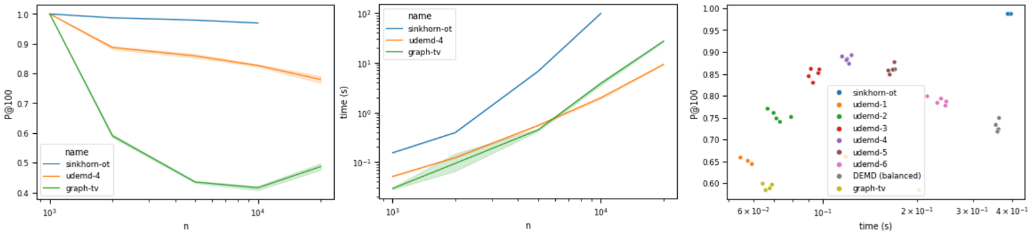
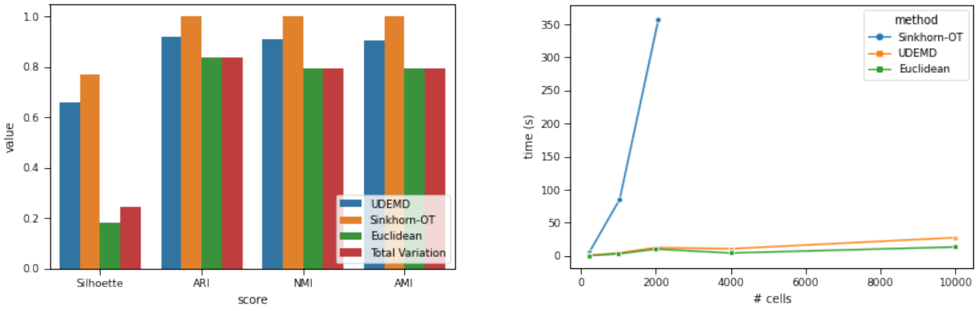


Figure 8.3: UDEMD is more scalable than Sinkhorn-OT and performs better than graph total variation. (left) Shows performance as measured by P@100, the fraction of the 100 nearest neighbors predicted correctly, against problem size. (middle) Shows time against problem size, and (right) shows performance vs. time on a problem size of $n = 2000$ for different choices of K. We find a good tradeoff at $K = 4$.

Single-cell data with cells as signals over gene graphs We consider 206 cells from the K562 human lymphoblast cell line as signals over a known 10000-node gene graph in single-cell RNA seq data [132]. We measure the distance between cells based on their transport on this gene graph. This was recently independently proposed by [92] and [15], who showed that OT over the gene graph can provide better distances between cells than Euclidean measures. We measure the performance of these methods based on how well the resulting distance matrix between cells matches the clusters according to four scores: Silhouette score, the adjusted Rand index, the normalized mutual information, and the adjusted mutual information. In Fig. 8.4, we see that UDEMD performs almost as well as Sinkhorn-OT, and much better than the Euclidean and total variation distances that do not take into account the gene graph as well as much faster than Sinkhorn-OT, scaling almost as well as Euclidean distances due to the embedding. Note however, that using balanced transport (see 8.4b right) degrades the accuracy. The balanced transport compared here is the original diffusion EMD from [200] which is not a thresholded distance, and thus noise in the data are able to perturb the accuracy of the distances. For more examples of the interplay between scale and sensitivity to outliers see supplement.



(a) Performance metrics

(b) Computation time comparison

Figure 8.4: UDEMD achieves better clustering than Euclidean and total variation (TV) distances, and performs similarly well to Sinkhorn-OT but is much more scalable with similar scalability to Euclidean and TV distances. (a) performance in terms of Silhouette score, adjusted rand index (ARI), normalized mutual information (NMI), and adjusted mutual information (AMI).

Single-cell data with genes as signals over cell graphs Next we applied our approach to 4,360 peripheral blood mononuclear cells (PBMCs) measured via single cell RNAseq publicly available on the 10X platform. We consider 3 curated gene sets that are explanatory for this dataset. We compare the distances between genes using UDEMD, Euclidean, total variation and Sinkhorn-OT distances. We can see that the genes canonical for monocytes (orange), T cells (green) and B cells (blue) all appear to be closely positioned to one another and separate between the groups in our embedding in contrast to a Euclidean distance embedding of the genes where the clusters are less clear (Figure 8.5a). Visualizing the UDEMD distance between our 46 genes in a heatmap, we can identify the three clusters as dark blocks of low distance (Figure 8.5b). This is quantified in (Figure 8.5c), where UDEMD performs the best on 3/4 metrics. Last, we tried to see how diffusion time scale impacted silhouette score, identifying that score maximized at a timescale of $K = 10$ and did not improve with higher scales.

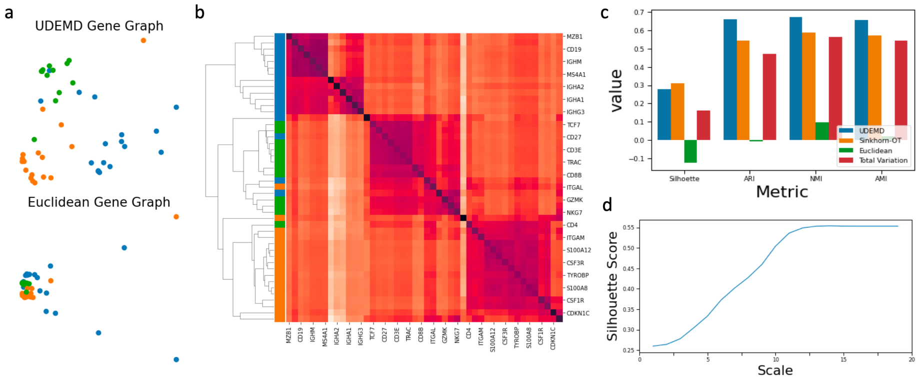


Figure 8.5: (a) Visualization of gene graphs of 46 genes canonical for different cell types using UDEMD and Euclidean ground distances (blue for B cells, orange for monocytes and green for T cells), (b) heat map of gene distances (c) clustering performance (d) silhouette score vs. maximum diffusion scale K.

A patient concept knowledge graph We consider a knowledge graph constructed from medical concepts captured in clinical documentation and reporting. SNOMED-CT is a widely used collection of terms and concepts with defined relationships considered to be an international standard for medical concepts captured from the electronic health record. We used 52,150 discharge summaries from MIMIC-III, which contain all information about a patient's hospital course. The SNOMED-CT concepts were extracted using a tool called MetaMap [9], which is a program developed and administered by the National Library of Medicine designed to extract relevant medical concepts out of biomedical text. Each patient encounter was linked to a single discharge summary, and after processing was

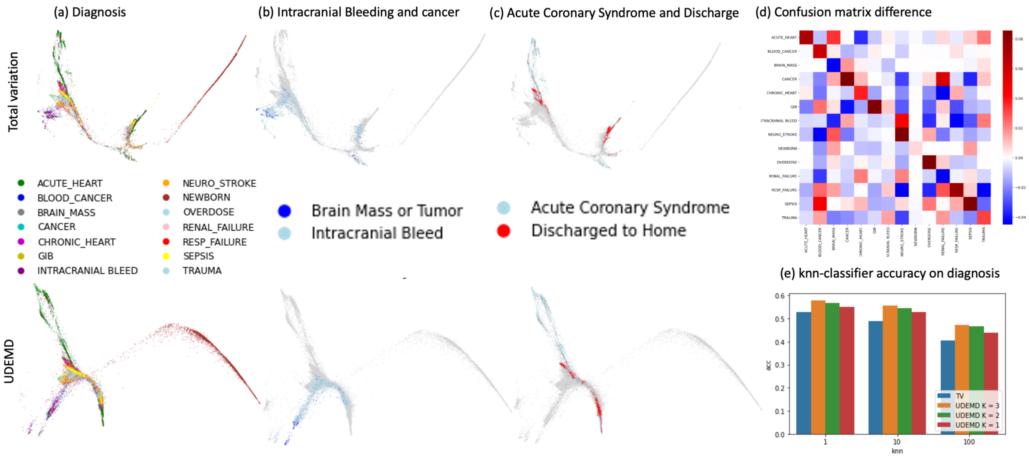


Figure 8.6: Embeddings of patients modeled as signals over the Snomed-CT graph using TV distance (a top) and using UDEMD distance (a bottom), colored by patient diagnosis. UDEMD better organizes the space as noted by selected terms in (b-c), difference of confusion matrices in (d) and k-nearest neighbors classification accuracy on the diagnosis in (e). In (b) note that the TV embedding (top) creates a spurious separation (due to noise in the signal) between subsets of patients who display intracranial bleeding that is not distinguished by diagnosis. On the other hand the UDEMD embedding (bottom) shows a continuum of patients with this diagnosis. The same holds for patients with brain mass or tumor shown in green. (c) UDEMD embedding organizes patients with acute coronary syndrome as a continuous trajectory with patients who are discharged (with milder cases) towards the bottom and more severe cases towards the top. The TV embedding again splits this trajectory.

represented by a vector of medical concepts. These medical concepts were then used as signals on the SNOMED-CT knowledge graph, which link all relevant concepts together. The metadata used to label patients included primary diagnosis, which was stored in the MIMIC-III database, and the discharge status after the conclusion of the hospital stay.

One of the advantages of the UDEMD-based embedding is the identification of clinically meaningful overlaps that may not be apparent from the single primary diagnosis recorded in the database. Patients with a primary diagnosis of intracranial bleeding (bleeding in the brain) can also have primary brain masses and tumors. Compared to the spurious fragmentation of patients with the same diagnosis of intracranial bleeding into several clusters in the TV embedding, UDEMD consolidated patients with the same diagnosis of intracranial bleeding and specifically grouped those that may have had bleeding due to a primary brain mass or tumor (See Figure 8.6b). Interestingly, UDEMD also identified patients who were predicted to have intracranial bleeding to have the diagnosis of stroke with higher accuracy, reflecting consistency with the fact that a subtype of stroke (hemorrhagic) is due to intracranial bleeding (Figure 8.6d).

UDEMD is also useful for identifying risk trajectories of patients with the same primary diagnosis as reflected by the utilization of healthcare services after discharge (see Figure 8.6c). UDEMD preserves the continuum of patients with the primary diagnosis of acute coronary syndrome and organizes them by high risk (discharged to extended stays at other facilities for further recuperation) at the top and low risk (discharged home) at the bottom of the trajectory.

8.6 Conclusion

In this work we explored the use of earth mover's distance to organize signals on large graphs. We presented an unbalanced extension of Diffusion EMD, which we showed is equivalent to the total variation unbalanced Wasserstein distance between signals on a graph. We showed how to compute nearest neighbors in this space in time log-linear in the number of nodes in the graph and the number of signals. Finally, we demonstrated how this can be applied to entities which can be modeled as signals on graphs between genes, cells, and biomedical concepts.

Appendix

In Section 8.A we further explore the theory of UDEMD as the size of the graph increases. In section 8.B we explore how the maximum scale K relates to the total variation weight empirically. In section 8.C we provide further details on the experiments and implementations.

8.A Theoretical Results

In this section we further describe some of the theoretical results, namely how UDEMD behaves as the number of points gets large and are sampled from some underlying manifold.

Lemma 2. $\mathrm{UDEMD}_{\alpha,K}(\mu,\nu)$ is equivalent to a Wasserstein distance $W_{d_\lambda}(\mu,\nu)$, defined as in Lemma 8.4.1, with the ground distance being a truncated geodesic distance on the manifold, i.e., $d_\lambda(x,y) = \min\{\lambda, d(x,y)\}$ for $\lambda > 0$.

Proof. We consider a close Riemannian manifold $\mathcal{M}$ with geodesic $d(x,y)$. We note h_t the heat kernel on $\mathcal{M}$ and $\boldsymbol{H}_t$ its corresponding operator. It is known (see [121], [82]) that h_t respects the semigroup property, the conservation property, the integrability property, Hölder continuity estimate and can be locally upper and lower bounded. We let

$$D_k(x,y) := \|h_{2^{-k}}(x,\cdot) - h_{2^{-k}}(y,\cdot)\|_1,$$

the authors in [121] define the following multiscale metric

$$D_\alpha(x,y) := \sum_{k\geq 0} 2^{-k\alpha} D_k(x,y).$$

Moreover, they show it is equivalent to the metric $d(x,y)^{2\alpha}$, for $\alpha \in (0,1/2)$. Now, the main idea is that instead of computing the Wasserstein distance with respect to the geodesic, one could compute it using $D_\alpha(x,y)$ which we note W_{D_α}. For two distributions μ and ν on $\mathcal{M}$, it is shown in [121] that

$$\widehat{W}_{\boldsymbol{H}}(\mu,\nu) := \|\boldsymbol{H}_1(\mu-\nu)\|_1 + \sum_{k\geq 0} 2^{-k\alpha}\|(\boldsymbol{H}_{2^{-(k+1)}} - \boldsymbol{H}_{2^{-k}})(\mu-\nu)\|_1 \simeq W_{D_\alpha}.$$

Using the heat kernel on $\mathcal{M}$, we are able to define a metric $\widehat{W}_{\boldsymbol{H}}(\mu,\nu)$ which is equivalent to W_{D_α}, the Wasserstein with respect to $d(x,y)^{2\alpha}$. However, we cannot directly use the heat kernel, therefore we refer to two other results from [200] (see proof of corollary 2.1 and corollary 3.1) building upon [47].

Lemma 2.1. Let $\boldsymbol{A}_\epsilon$ be such that $\lim_{\epsilon\to 0} \boldsymbol{A}_\epsilon^{t/\epsilon} = \boldsymbol{H}_t$, for ϵ small enough, we have:

$$\widehat{W}_{\boldsymbol{A}_\epsilon^{t/\epsilon}} \simeq W_{D_\alpha}.$$

For example, it is shown that the anisotropic and isotropic operator $\boldsymbol{A}_t$ defined in [47] will indeed converge to the heat kernel when $\epsilon \to 0$. The next lemma can be seen as a bridge between discrete and continuous, it corresponds to Corollary 3.1 in [200]. Because we have a finite set of points, we can only approximate the operator $\boldsymbol{A}_t$. For the isotropic operator, one could use the approximation as in [47]. By defining $k_t(x,y) = e^{-d(x,y)^2/t}$,

$$d_t(x) = \sum_y k_t(x,y) \text{ and } p_t(x,y) = k_t(x,y)/d_t(x),$$

the operator is $(\boldsymbol{P}_t f)(x) = \sum_y p_t(x,y)f(y)$. As discussed in [47], this sum converges to an integral (up to a normalization constant), thus to the operator $\boldsymbol{A}_t$ when the number of observations tends to infinity.

Lemma 2.2. Let $X_i, X_j \in \mathcal{X}$ be two datasets with size n_i and n_j respectively, and let μ_i and μ_j be the continuous distributions corresponding to the ones of X_i and X_j, let K be the largest scale and put $N = \min(K, n_i, n_j)$. Then, for sufficiently big $N \to \infty$ (implying sufficiently small $\epsilon = 2^{-K} \to 0$):

$$DEMD_{\alpha,K}(X_i, X_j) \simeq W_{d^{2\alpha}_{\mathcal{M}}}(\mu_i, \mu_j),$$

for all $\alpha \in (0, 1/2)$.

In our case, X_i and X_j are two signals on a graph.

Finally, we note that for all $\lambda > 0$, the metrics $\{1, d(x,y)\} \simeq \min\{\lambda, d(x,y)\}$, indeed for $\lambda > 1$ we have

$$(1/\lambda)\min\{\lambda, d(x,y)\} \leq \min\{1, d(x,y)\} \leq \min\{\lambda, d(x,y)\}$$

and

$$\min\{(1/\lambda), d(x,y)\} \leq \min\{1, d(x,y)\} \leq \lambda \min\{(1/\lambda), d(x,y)\}.$$

On a close Riemannian manifold we can define the constant in the minimum to be the largest geodesic distance, thus $d(x,y)^{2\alpha} \simeq \min\{\lambda, d(x,y)^{2\alpha}\}$ and the Wasserstein distances induced by these metrics are also equivalent. Therefore we know that DEMD is equivalent to a class of Wasserstein with thresholded ground distance, i.e. $\{W_{d_\lambda}\}_{\lambda>0}$. Since UDEMD is a special case of DEMD where the largest scale does not depend on the mixing time, it is also equivalent to the class of Wasserstein with thresholded ground distance.

□

8.A.1 Robustness to Corruption

Many related works exist on unbalanced transport we suggest [165, Sec. 10.2] for a broad overview. The thresholded Wasserstein distance considered here has benefits both computationally and in terms of robustness. Intuitively, for the standard optimal transport problem under an adversarial noise model where the adversary is allowed to add a small fraction ϵ of points at an arbitrary location, the adversary is able to affect the Wasserstein distance by roughly $\epsilon\lambda$, where $\lambda := \max C$ is defined as the maximum cost over the domain. In many domains the maximum cost, $\max C$ can be arbitrarily large e.g. $\mathbb{R}^d$, with a thresholded cost we immediately limit the power of this type of adversarial noise.

8.B Unbalanced Transport

In this section we investigate the relationship between the maximum scale K of UDEMD and the threshold λ, which can be equivalently expressed as the weight of the total variation coefficient in equation 8.5. The theory in sections 8.4.1 and Appendix 8.A suggest a relationship between K and λ. However, the exact relationship is impossible to determine due to the constants in the notion of equivalence used in Lemma 8.4.2. Instead we investigate this relationship empirically in Figure 8.7. We use the same toy sphere dataset with 10000 points uniformly distributed over the unit sphere equipped with a 10-nearest-neighbor graph.

Here we see that the UDEMD of specified K is closely related to the thresholded arc length distance (or geodesic) on the sphere manifold.

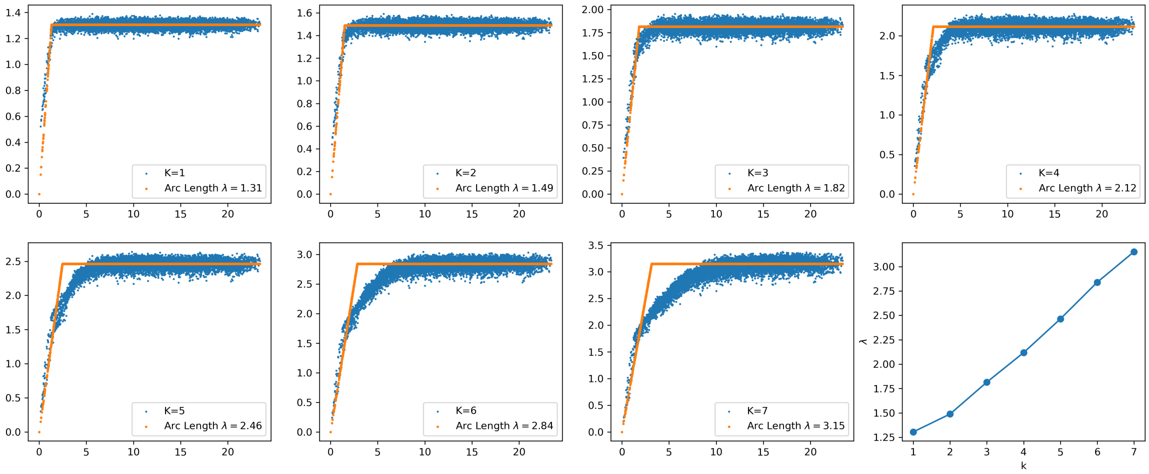


Figure 8.7: Compares the thresholded ground distance with threshold level λ (orange) with the UDEMD for different maximum scales $K \in \{1 \ldots 7\}$ (blue) on the sphere with $n = 10000$. Larger K corresponds to larger λ as shown in the bottom right figure in a roughly linear relationship.

8.C Experiment Details

In this section we present details of our experiments and implementations including information on our evaluation metrics, dataset preprocessing, and algorithmic implementations.

8.C.1 Metrics

Clustering Metrics For the single cell datasets we use four methods to measure how well distance matrices match ground truth clusterings of distributions. These four metrics the Silhouette score, the adjusted rand index, the normalized mutual information, and the adjusted mutual information measure how well two clusterings of the data align. The Silhoette score measures how similar a point is to points in its own cluster as compared to those in other clusters with some distance metric. To compare distance matrices Silhoette score relative to some ground truth clustering, we measure the Silhoette score using the distance matrix of interest.

This is in contrast to the ARI, NMI, and AMI scores which all compare two clusterings of the data. For these methods, we first cluster the distance matrix using spectral clustering, then compare this clustering with the ground truth. The adjusted rand index measures the similarity of two cluster by counting the pairs of samples that assigned to the predicted vs. true clusters. This is then normalized so that a score of zero should correspond to random clusters and a score of 1 corresponds to perfectly matching clusters up to a label permutation. Scores here range from -1 to 1. The NMI scores the mutual information of the between clusterings and ranges from 0 to 1. Finally, the AMI is similar to the NMI however it is adjusted (downwards) to adjust for random chance labelings.

All of these metrics are implemented using `sklearn.metrics` [159].

Embedding Metrics with Labels To evaluate the quality of patient embeddings where we have patient labels, we measure the k-nearest-neighbor classification accuracy on the embeddings for different values of k. We choose $k \in \{1, 10, 100\}$ to evaluate the performance of embeddings at different neighborhood scales in Figure 8.6e.

8.C.2 Dataset Details

Cell line dataset For this dataset we preprocess the data as in [92] to get a matrix of cosine distances between genes. We use this cosine distance matrix for the Sinkhorn-OT cost matrix, which we note requires n^2 memory to store. To construct our graph for UDEMD we connect genes with cosine distance < 0.6. We choose this parameter for an appropriate level of sparsity. We then compute the embeddings according to Algorithm 2, and subselect to 10% of the elements of the embedding to get the results in Figure 8.4. This reduces the time to compute nearest neighbors over the embeddings using a ball tree algorithm from `sklearn` [159]. This strategy should enable time complexities that scale linearly with the number of cells instead of quadratic scaling in [92] and cubic in [15].

PBMC Dataset Peripheral Blood Mononuclear Cells (PBMCs) are immune cells that travel in the blood to identify and traffic to sites of inflammation. To build a graph between cells we first normalize each cell to have 10,000 gene counts, then square root transform elementwise to weight genes appropriately. We then construct a graph with a anisotropic alpha-decay kernel as in [148], which in practice behaves similarly to a k-nearest-neighbors graph.

PBMCs include T cells, B cells and monocytes, each with unique functions defined by distinct transcriptomic states. Recently [54], described a set of genes which define each of these cell types: for T cell: CD3D, CD3E, CD3G, TRAC, CD4, TCF7, IL7R, CD8A, CD8B, GZMK, CCL5, NKG7, for B cells: CD19, MS4A1, CD79A, CD79B, MZB1, IGHD, IGHM, CD38,XBP1, CD27, SLAMF7, IGHA1, IGHA2, IGHG1, IGHG2, IGHG3, IGHG4 and for Monocytes: VCAN, FCN1, S100A8, S100A9, CD14, ITGAL, ITGAM, CSF3R, CSF1R, CX3CR1,TYROBP, LYZ, S100A12, FCN1, FCGR3A, FCGR3B, ITGAL, ITGAM, CSF3R, CSF1R, CX3CR1, CDKN1C, MS4A7. We used these curated lists as ground truth clusters to score gene distances.

MIMIC-III / SNOMED-CT dataset MIMIC-III discharge summary text from 52,150 unique patient admissions were extracted from the NOTEEVENTS table subset by CATEGORY as "Discharge summary" written by physician providers. Diagnoses metadata were extracted from the admissions table under DIAGNOSIS, which is the free text diagnosis entered by clinical providers on admission to the hospital. Out of 15,693 distinct diagnoses, 14 categories were grouped based on clinical expertise, with the reference for specific groupings in the code provided.

MetaMap processing of the raw text from the discharge summaries used the setting of composite phrases (Q) with 3 elements, and reducing the number of candidate concepts using the prune option to 30 (recommended setting to avoid out-of-memory errors during processing). For text that still was unable to be processed, we used the setting of disabling composite phrases (Q=0) and setting the prune option to 10. We used the 2018 version of MetaMap with the 2018AB relaxed version terminology, USABase containing SNOMED-CT concepts.

Relationships between concepts in SNOMED-CT were subset to the Clinical Findings concept model, with a total of 19 types of edges. All relationship types were used to create edges between concepts into a large sparse adjacency matrix with 437,855 x 437,855 nodes with 3,672,277 edges in total. The final extracted concepts from the discharge summaries had a total of 24,826 nodes due to redundant and inactivated concepts. This may have been due to the fact that the MetaMap extractor was the 2018 version, whereas the SNOMED-CT knowledge graph is the 2021 version (the 2018 version was unavailable).

8.C.3 Method Details

Our method UDEMD is implemented using Pytorch [158] and Pytorch Geometric [67] for fast GPU computation with sparse graphs. We compare this to a Sinkhorn-OT algorithm as implemented in [92] which is also implemented in Pytorch. We note that there are other similar implementations of the Sinkhorn algorithm such as those available in the python optimal transport package [68] or through alternative libraries such as KeOps [37]. While these implementations may vary in their efficiency, we note that any implementation of the Sinkhorn algorithm with a full cost matrix will scale asymptotically at least with n^2.

In contrast, UDEMD for sparse adjacency matrices is able to capitalize on the structure of total variation unbalanced transport's limited radius or distance threshold, which helps to localize computation to local neighborhoods, making it unnecessary to operate on the full $n \times n$ distance matrix. For specific implementations we refer to the code. We note that UDEMD's implementation is similar to that of graph neural networks (GNNs) [216], and hence can take advantage of frameworks that allow for efficient training of these GNNs. While the implementations are similar we note that the goal of UDEMD as an unsupervised approximation of the unbalanced Wasserstein distance is quite different than that of GNNs.

8.C.4 Compute Details

All experiments were run on a single machine equipped with an AMD Ryzen Threadripper 2990WX 32-Core Processor, 128GB of ram, and two NVIDIA TITAN RTX GPUs.

Chapter 9

Discussion and Future Work

This thesis combines ideas from graph signal processing, deep learning, and optimal transport to improve the interpretability, accuracy, and efficiency of data processing algorithms. Melding these ideas together leverages different aspects of the geometry of data. No tool for is best at processing all types of data. By taking the best parts of each this thesis takes a small step in understanding the data with weak supervision from multiple perspectives.

9.1 Future Work

Looking forward, despite the progress made in this thesis and by other related work, there remain many open problems as the data collection and computational resources continue to evolve. What may be the best algorithm for the task today will not be tomorrow. Building on the work in this thesis, there are a number of open directions and problems. We split this future work into a few different directions, first we consider further applications of the algorithms developed in this thesis. These algorithms are still in development and may need further adaptation to specific data and applications. Second, we consider further theoretical results. Many of these algorithms are developed based on what works, and further generalization could help to gain a better understanding of the underlying benefits and biases of these works.

9.1.1 Further Applications of Algorithms and Methods

Application of TrajectoryNet to Mammospheres In Chapter 6 we presented an algorithm for learning the dynamics of signal cells in a continuous way. In ongoing work we are applying this method to a triple-negative breast cancer dataset. The well studied epithelial-to-mesenchymal transition (EMT) process delineates how cells, and in particular cancer cells, transform from sticky structural epithelial cells to free floating mesenchymal cells. This is important in understanding how cancer can metastasis and spread to new locations. Equally important but less well understood is the mesenchymal-to-epithelial transition (MET) process. When cancer cells spread to a new location, to take root they transition at least partially back into an epithelial state. Studying how this transition occurs may lead to new drug targets preventing metastasis.

We study the MET transition with a single-cell dataset of five timepoints of 3D cultured mammosphere organoids. Using a combination of TrajectoryNet and Granger causality analysis we hope to uncover the MET program — the sequence of transcription factors and genes that lead a mesenchymal breast cancer cell to transition to a more epithelial state.

A secondary goal of this project is to understand the culture better. Only 10% of the cells in the cell line become mammospheres, and understanding what makes these 10% of cells different is key to studying these cells and understanding the MET biology. Our preliminary findings suggest that cell cycle as well as the E-cadherin (CDH1) and CAV1 genes are related to this process.

Application of Diffusion EMD to Patient Derived Organoid (PDO) Drug screen Diffusion EMD in Chapter 7 and unbalanced Diffusion EMD in Chapter 8 suggest ways to compute optimal transport distances between a large number of distributions very quickly. In this application with data from Chris Tape's Lab at Univeristy College London and in collaboration with Maria Ramos we have a set of over 5,000 single-cell datasets in different patient, drug, and co-culture conditions. This dataset gives us a chance to understand the effects of drug and co-culture at a large scale. However, given the large size of this dataset it is difficult to understand the effects of these different conditions. We aim to apply Diffusion EMD and other multiscale approximations of the the earth mover's distance to understand the local effects of these conditions.

9.1.2 Theoretical Generalization

Diffusion EMD and unbalanced Diffusion EMD explore multiscale approximations to the earth mover's distance. Both of these methods rely on interesting multiscale approximations to markov random walks, diffusion, or powers of sparse matrices. These works suggest possible improvements to other work which use diffusion on graphs built from data. In particular, by noticing that the rank of $\boldsymbol{P}^t$ for some diffusion matrix $\boldsymbol{P}$ shrinks rapidly with t, we can use low rank approximations for large t. While this is a spectral approximation, there are also interesting avenues in spatial approximation, which we explore briefly in integrated diffusion with local denoising [112]. By using a multiscale diffusion operator we were able to denoise the manifold locally. In the future we can use ideas from interpolative decomposition and wavelets to design smoothly integrating spatial sections of the manifold.

Another direction of inquiry in the direction of the unbalanced variant of Diffusion EMD is into the relationship between the maximum scale K, and the λ balancing the Wasserstein distance with the total variation distance. In Chapter 8 we showed an empirical relationship, however a precise theoretical relationship is more difficult to show, requiring a more fine grained analysis of both diffusion distances and the scale parameter's effect on the distance threshold.

Trajectory inference in single cell genomics with TrajectoryNet has many advantages. By directly learning the full and continuous gene trajectories we are able to regularize trajectories based on the geometry, which is not possible when optimizing over a discrete matching. However, this raises a number of questions on identifiability. Specifically, in what cases can the origin or ancestor cells be determined for a given population? What are the tradeoffs for using such a continuous method? and finally, are there cases which are identifiable either in a discrete time context that are not identifiable in a continuous time context or vice-versa?

9.1.3 Casual Learning

The methods presented in this thesis are statistical not causal. While they may be useful for predicting in a similar setting, they are not necessarily useful when the underlying conditions change. This is the problem tackled by causal inference. With the further assumption of a causal model, we can attempt to make inferences about counter-factuals and situations which are not present in the data. While some of this work is applied to perturbation data, namely Diffusion EMD (Chapter 7), we do not assume a causal model. Doing so might allow us to make further inferences into counter-factuals and other out of domain questions as in [201]. Single-cell genomics has the power to sample the distribution of possible cellular states in a given condition, using a combination of optimal transport and causal inference will allow us to measure how a shift in the latent causes affects different parts of the distribution differently.

Bibliography

[1] Martın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. *OSDI*, page 21, 2016.

[2] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent Space Autoregression for Novelty Detection. In *CVPR*, 2019.

[3] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[4] Alessandro Achille and Stefano Soatto. Emergence of Invariance and Disentanglement in Deep Representations. *ArXiv170601350 Cs Stat*, June 2017.

[5] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. *ArXiv180506725 Cs*, November 2018.

[6] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[7] Jerone T A Andrews, Edward J Morton, and Lewis D Griffin. Detecting Anomalous Data Using Auto-Encoders. *Int. J. Mach. Learn. Comput.*, 6(1):6, 2016.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. In *ICML*, 2017.

[9] Alan R. Aronson and François-Michel Lang. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236, 2010. ISSN 1067-5027. doi: 10.1136/jamia.2009.002733. URL `https://doi.org/10.1136/jamia.2009.002733`.

[10] Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Scalable Nearest Neighbor Search for Optimal Transport. *ICML*, 2020.

[11] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Robust Optimal Transport with Applications in Generative Modeling and Domain Adaptation. *ArXiv201005862 Cs*, October 2020.

[12] Pablo Barceló, Egor V Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2020.

[13] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[14] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory*, pages 624–638. Springer, 2004.

[15] Riccardo Bellazzi, Andrea Codegoni, Stefano Gualandi, Giovanna Nicora, and Eleonora Vercesi. The Gene Mover's Distance: Single-cell similarity via Optimal Transport. *ArXiv210201218 Cs Math Q-Bio*, March 2021.

[16] Jean-David Benamou. Numerical resolution of an "unbalanced" mass transport problem. *ESAIM: M2AN*, 37(5):851–868, 2003.

[17] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3): 375–393, 2000.

[18] Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative Bregman Projections for Regularized Transportation Problems. *SIAM J. Sci. Comput.*, 2014.

[19] Jean-David Benamou, Thomas O. Gallouët, and François-Xavier Vialard. Second-Order Models for Optimal Transport and Cubic Splines on the Wasserstein Space. *Found Comput Math*, 19(5):1113–1143, 2019.

[20] Sean C. Bendall, Kara L. Davis, El-ad David Amir, Michelle D. Tadmor, Erin F. Simonds, Tiffany J. Chen, Daniel K. Shenfeld, Garry P. Nolan, and Dana Pe'er. Single-Cell Trajectory Detection Uncovers Progression and Regulatory Coordination in Human B Cell Development. *Cell*, 157(3):714–725, 2014.

[21] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 00010782. doi: 10.1145/361002.361007.

[22] Volker Bergen, Marius Lange, Stefan Peidli, F. Alexander Wolf, and Fabian J. Theis. Generalizing RNA velocity to transient cell states through dynamical modeling. *BioRxiv 820936*, 2019.

[23] Amit Bermanis, Amir Averbuch, and Ronald R. Coifman. Multiscale data sampling and function extension. *Applied and Computational Harmonic Analysis*, 34(1):15–29, January 2013. ISSN 10635203. doi: 10.1016/j.acha.2012.03.002.

[24] Aleksandar Bojchevski and Stephan Gunnemann. Deep Gaussian Embedding of Graphs:unsupervised Inductive Learning via Ranking. *ICLR*, 2021.

[25] Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. Wasserstein barycentric coordinates: Histogram regression using optimal transport. *ACM Transactions on Graphics*, 35(4), 2016.

[26] K. M. Borgwardt, C. S. Ong, S. Schonauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl 1):i47–i56, June 2005. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/bti1007.

[27] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: Identifying Density-Based Local Outliers. In *ACM SIGMOD*, page 12, Dalles, TX, 2000.

[28] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.*, 34(4):18–42, 2017. ISSN 1053-5888. doi: 10.1109/MSP.2017.2693418.

[29] D. S. Broomhead and D Lowe. Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks. *R. Signals Raar Establ.*, Memorandum 4148, 1988.

[30] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*, 2014.

[31] Daniel B. Burkhardt, Jay S. Stanley, Alexander Tong, Ana Luisa Perdigoto, Scott A. Gigante, Kevan C. Herold, Guy Wolf, Antonio J. Giraldez, David van Dijk, and Smita Krishnaswamy. Quantifying the effect of experimental perturbations in single-cell RNA-sequencing data using graph signal processing. Technical report, BiorXiv, 2020.

[32] Luis Caffarelli and Robert McCann. Free boundaries in optimal transport and Monge-Ampère obstacle problems. *Ann. Math.*, 171(2):673–730, March 2010. ISSN 0003-486X. doi: 10.4007/annals.2010.171.673.

[33] Guilherme O. Campos, Arthur Zimek, Jörg Sander, Ricardo J. G. B. Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E. Houle. On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study.

Data Min Knowl Disc, 30(4):891–927, July 2016. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-015-0444-8.

[34] Raghavendra Chalapathy and Sanjay Chawla. Deep Learning for Anomaly Detection: A Survey. *ArXiv190103407 Cs Stat*, January 2019.

[35] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Robust, Deep and Inductive Anomaly Detection. In *ECML*. Springer International Publishing, 2017. ISBN 978-3-319-71248-2 978-3-319-71249-9. doi: 10.1007/978-3-319-71249-9_3.

[36] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly Detection using One-Class Neural Networks. *ArXiv180206360 Cs Stat*, February 2018.

[37] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021. URL http://jmlr.org/papers/v22/20-275.html.

[38] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31*, 2018.

[39] William S. Chen, Nevena Zivanovic, David van Dijk, Guy Wolf, Bernd Bodenmiller, and Smita Krishnaswamy. Uncovering axes of variation among single-cell cancer specimens. *Nat Methods*, 17(3):302–310, March 2020. ISSN 1548-7091, 1548-7105. doi: 10.1038/s41592-019-0689-z.

[40] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *ArXiv160603657 Cs Stat*, June 2016.

[41] Yongxin Chen, Giovanni Conforti, and Tryphon T. Georgiou. Measure-Valued Spline Curves: An Optimal Transport Viewpoint. *SIAM J. Math. Anal.*, 50(6):5947–5968, 2018.

[42] Yunqiang Chen, Xiang Zhou, and Thomas Huang. One-class SVM for learning in image retrieval. In *ICIP*, volume 1, pages 34–37, Thessaloniki, Greece, 2001. IEEE. ISBN 978-0-7803-6725-8. doi: 10.1109/ICIP.2001.958946.

[43] Zeyu Chen and E. John Wherry. T cell responses in patients with COVID-19. *Nature Reviews Immunology*, 20(9):529–536, 2020.

[44] Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Unbalanced optimal transport: Dynamic and Kantorovich formulations. *Journal of Functional Analysis*, 274(11):3090–3123, 2018.

[45] Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Convolutional Networks for Spherical Signals. In *ICML*, 2017.

[46] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006. ISSN 10635203. doi: 10.1016/j.acha.2006.04.006.

[47] Ronald R. Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, July 2006. ISSN 10635203. doi: 10.1016/j.acha.2006.04.004.

[48] Justin Cotney, Rebecca A. Muhle, Stephan J. Sanders, Li Liu, A. Jeremy Willsey, Wei Niu, Wenzhong Liu, Lambertus Klei, Jing Lei, Jun Yin, Steven K. Reilly, Andrew T. Tebbenkamp, Candace Bichsel, Mihovil Pletikos, Nenad Sestan, Kathryn Roeder, Matthew W. State, Bernie Devlin, and James P. Noonan. The autism-associated chromatin modifier CHD8 regulates other autism risk genes during human neurodevelopment. *Nat Commun*, 6(1):6404, 2015.

[49] Sergio Martinez Cuesta, Syed Asad Rahman, Nicholas Furnham, and Janet M. Thornton. The Classification and Evolution of Enzyme Function. *Biophysical Journal*, 109 (6):1082–1086, 2015. ISSN 00063495. doi: 10.1016/j.bpj.2015.04.020.

[50] Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems 26*, pages 2292–2300, 2013.

[51] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *Adv. Neural Inf. Process. Syst. 30*, 2016.

[52] Pinar Demetci, Rebecca Santorella, Björn Sandstede, William Stafford Noble, and Ritambhara Singh. Gromov-Wasserstein optimal transport to align single-cell multi-omics data. Preprint, Bioinformatics, April 2020.

[53] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A Survey on GANs for Anomaly Detection. *ArXiv190611632 Cs Stat*, June 2019.

[54] Jiarui Ding, Xian Adiconis, Sean K. Simmons, Monika S. Kowalczyk, Cynthia C. Hession, Nemanja D. Marjanovic, Travis K. Hughes, Marc H. Wadsworth, Tyler Burks, Lan T. Nguyen, John Y. H. Kwon, Boaz Barak, William Ge, Amanda J. Kedaigle, Shaina Carroll, Shuqiang Li, Nir Hacohen, Orit Rozenblatt-Rosen, Alex K. Shalek, Alexandra-Chloé Villani, Aviv Regev, and Joshua Z. Levin. Systematic comparative analysis of single cell rna-sequencing methods. *bioRxiv*, 2019. doi: 10.1101/632216. URL `https://www.biorxiv.org/content/early/2019/05/23/632216`.

[55] Paul D. Dobson and Andrew J. Doig. Distinguishing Enzyme Structures from Non-enzymes Without Alignments. *Journal of Molecular Biology*, 330(4):771–783, July 2003. ISSN 0022-2836. doi: 10.1016/S0022-2836(03)00628-4.

[56] Kun Dong, Austin R. Benson, and David Bindel. Network Density of States. *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 1152–1161, July 2019. doi: 10.1145/3292500.3330891.

[57] J.R. Dormand and P.J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

[58] R. M. Dudley. The Speed of Mean Glivenko-Cantelli Convergence. *Ann. Math. Stat.*, 40(1):40–50, 1969.

[59] Emilien Dupont. Learning Disentangled Joint Continuous and Discrete Representations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and

R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 710–720. Curran Associates, Inc., 2018.

[60] Tapio Elomaa, Heikki Mannila, and Hannu Toivonen. Fast Outlier Detection in High Dimensional Spaces. In *PKDD*, Lecture Notes in Computer Science ; Lecture Notes in Artificial Intelligence, Helsinki, Finland, 2002. Springer. ISBN 978-3-540-44037-6.

[61] Matthias Erbar, Martin Rumpf, Bernhard Schmitzer, and Stefan Simon. Computation of optimal transport on discrete metric measure spaces. *Numer. Math.*, 144(1):157–200, January 2020. ISSN 0029-599X, 0945-3245. doi: 10.1007/s00211-019-01077-z.

[62] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognit.*, 58:121–134, October 2016. ISSN 00313203. doi: 10.1016/j.patcog.2016.03.028.

[63] Florian Erhard, Marisa A. P. Baptista, Tobias Krammer, Thomas Hennig, Marius Lange, Panagiota Arampatzi, Christopher S. Jürges, Fabian J. Theis, Antoine-Emmanuel Saliba, and Lars Dölken. scSLAM-seq reveals core features of transcription dynamics in single cells. *Nature*, 571(7765):419–423, 2019.

[64] Federico Errica, Davide Bacciu, Marco Podda, and Alessio Micheli. A Fair Comparison of Graph Neural Networks for Graph Classification. In *ICLR*, page 14, 2020.

[65] Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, N Siddharth, Brooks Paige, and Dana H Brooks. Structured Disentangled Representations. *AISTATS*, page 10, 2019.

[66] Kilian Fatras, Younes Zine, Rémi Flamary, Rémi Gribonval, and Nicolas Courty. Learning with minibatch Wasserstein: Asymptotic and gradient properties. *AISTATS*, 2020.

[67] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. *ArXiv190302428 Cs Stat*, April 2019.

[68] Remi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurelie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Leo Gautheron, Nathalie T H Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. POT: Python Optimal Transport. *JMLR*, 22, 2021.

[69] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo, and Tomaso Poggio. Learning with a Wasserstein Loss. *Adv. Neural Inf. Process. Syst. 28*, 2015.

[70] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Diffusion Scattering Transforms on Graphs. *Int. Conf. Mach. Learn.*, 2019.

[71] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability of Graph Scattering Transforms. *Adv. Neural Inf. Process. Syst. 33*, 2019.

[72] Feng Gao, Guy Wolf, and Matthew Hirn. Geometric scattering for graph data analysis. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2122–2131, 2019.

[73] Feng Gao, Guy Wolf, and Matthew Hirn. Geometric Scattering for Graph Data Analysis. In *International Conference on Machine Learning*, pages 2122–2131, May 2019.

[74] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[75] Matan Gavish, Boaz Nadler, and Ronald R Coifman. Multiscale Wavelets on Trees, Graphs and High Dimensional Data: Theory and Applications to Semi Supervised Learning. In *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

[76] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning Generative Models with Sinkhorn Divergences. *ArXiv170600292 Stat*, October 2017.

[77] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning Generative Models with Sinkhorn Divergences. In *AISTATS*, 2018.

[78] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[79] Vladimir Gligorijevic, P. Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C. Taylor, Ian M. Fisk, Hera Vlamakis, Ramnik J. Xavier, Rob Knight, Kyunghyun Cho, and Richard Bonneau. Structure-Based Protein Function Prediction using Graph Convolutional Networks. Preprint, Bioinformatics, October 2019.

[80] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734, Montreal, Que., Canada, 2005. IEEE. ISBN 978-0-7803-9048-5. doi: 10.1109/IJCNN.2005.1555942.

[81] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. In *7th International Conference on Learning Representations*, 2019.

[82] Alexander Grigor'yan and Liguang Liu. Heat kernel and Lipschitz–Besov spaces. *Forum Math.*, 27(6), January 2015. ISSN 0933-7741, 1435-5337. doi: 10.1515/forum-2014-0034.

[83] Alexander Grigor'yan, Jiaxin Hu, and Ka-Sing Lau. Heat Kernels on Metric Measure Spaces. In *Geometry and Analysis of Fractals*, volume 88, pages 147–207, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-43919-7 978-3-662-43920-3. doi: 10.1007/978-3-662-43920-3_6.

[84] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *ArXiv170400028 Cs Stat*, March 2017.

[85] Laleh Haghverdi, Maren Büttner, F Alexander Wolf, Florian Buettner, and Fabian J Theis. Diffusion pseudotime robustly reconstructs lineage branching. *Nat Methods*, 13(10):845–848, 2016.

[86] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *Adv. Neural Inf. Process. Syst. 31*, 2017.

[87] Tatsunori B Hashimoto, David K Gifford, and Tommi S Jaakkola. Learning Population-Level Diffusions with Generative Recurrent Networks. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2417–2426, 2016.

[88] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier Detection Using Replicator Neural Networks. In *Data Warehousing and Knowledge Discovery*, volume 2454, pages 170–180, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-44123-6 978-3-540-46145-6. doi: 10.1007/3-540-46145-0_17.

[89] Gert-Jan Hendriks, Lisa A. Jung, Anton J. M. Larsson, Michael Lidschreiber, Oscar Andersson Forsman, Katja Lidschreiber, Patrick Cramer, and Rickard Sandberg. NASC-seq monitors RNA synthesis in single cells. *Nat Commun*, 10(1):3138, 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-11028-9.

[90] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK. In *ICLR*, page 22, 2017.

[91] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a Definition of Disentangled Representations. *ArXiv181202230 Cs Stat*, December 2018.

[92] Geert-Jan Huizing, Gabriel Peyré, and Laura Cantini. Optimal Transport improves cell-cell similarity inference in single-cell omics data. Preprint, Systems Biology, March 2021.

[93] Piotr Indyk and Nitin Thaper. Fast image retrieval via embeddings. In *3rd International Workshop on Statistical and Computational Theories of Vision*, 2003.

[94] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative Models for Graph-Based Protein Design. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 33*, pages 15820–15831, 2019.

[95] Lesley A. Inker and Ronald D. Perrone. Assessment of kidney function. https://www.uptodate.com/, June 2018.

[96] Jörn-Henrik Jacobsen, Edouard Oyallon, Stéphane Mallat, and Arnold W. M. Smeulders. Multiscale Hierarchical Convolutional Networks. *ArXiv170304140 Cs Stat*, March 2017.

[97] Arun Jambulapati, Aaron Sidford, and Kevin Tian. A Direct $\tilde{O}(1/\epsilon)$ Iteration Parallel Algorithm for Optimal Transport. In *Advances in Neural Information Processing Systems 32*, pages 11359–11370, 2019.

[98] A Johnson, L Bulgarelli, T Pollard, S Horng, LA Celi, and R Mark. Mimic-iv (version 1.0), 2020.

[99] Sabina Kanton, Michael James Boyle, Zhisong He, Malgorzata Santel, Anne Weigert, Fátima Sanchís-Calleja, Patricia Guijarro, Leila Sidow, Jonas Simon Fleck, Dingding Han, Zhengzong Qian, Michael Heide, Wieland B. Huttner, Philipp Khaitovich, Svante Pääbo, Barbara Treutlein, and J. Gray Camp. Organoid single-cell genomic atlas uncovers human-specific features of brain development. *Nature*, 574(7778):418–422, 2019.

[100] L V Kantorovich. On the Translocation of Masses. *Dokl. Akad. Nauk*, 1942.

[101] Yuta Katayama, Masaaki Nishiyama, Hirotaka Shoji, Yasuyuki Ohkawa, Atsuki Kawamura, Tetsuya Sato, Mikita Suyama, Toru Takumi, Tsuyoshi Miyakawa, and Keiichi I. Nakayama. CHD8 haploinsufficiency results in autistic-like phenotypes in mice. *Nature*, 537(7622):675–679, 2016.

[102] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[103] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*, 2015.

[104] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *arXiv:1312.6114 [Cs, Stat]*, December 2013.

[105] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems 29*, 2016.

[106] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *4th Int. Conf. Mach. Learn.*, September 2016.

[107] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. In *Bayesian Deep Learning Workshop NeurIPS 2016*, 2016.

[108] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal The International Journal on Very Large Data Bases*, 8(3-4):237–253, February 2000. ISSN 1066-8888, 0949-877X. doi: 10.1007/s007780050006.

[109] Soheil Kolouri, Yang Zou, and Gustavo K. Rohde. Sliced Wasserstein Kernels for Probability Distributions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5258–5267, Las Vegas, NV, USA, June 2016. IEEE. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.568.

[110] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized Sliced Wasserstein Distances. In *Advances in Neural Information Processing Systems 32*, pages 261–272, 2019.

[111] Manik Kuchroo, Jessie Huang, Patrick Wong, Jean-Christophe Grenier, Dennis Shung, Alexander Tong, Carolina Lucas, Jon Klein, Daniel Burkhardt, Scott Gigante,

Abhinav Godavarthi, Benjamin Israelow, Ji Eun Oh, Julio Silva, Takehiro Takahashi, Camila D Odio, John Fournier, Dela Cruz, Albert I Ko, F Perry Wilson, Julie Hussin, Guy Wolf, and Smita Krishnaswamy. Multiscale PHATE Exploration of SARS-CoV-2 Data Reveals Multimodal Signatures of Disease. *BioRxiv*, 2020.

[112] Manik Kuchroo, Abhinav Godavarthi, Alexander Tong, Guy Wolf, and Smita Krishnaswamy. Multimodal data visualization, denoising and clustering with integrated diffusion. In *IEEE MLSP*, 2021.

[113] Gioele La Manno, Ruslan Soldatov, Amit Zeisel, Emelie Braun, Hannah Hochgerner, Viktor Petukhov, Katja Lidschreiber, Maria E. Kastriti, Peter Lönnerberg, Alessandro Furlan, Jean Fan, Lars E. Borm, Zehua Liu, David van Bruggen, Jimin Guo, Xiaoling He, Roger Barker, Erik Sundström, Gonçalo Castelo-Branco, Patrick Cramer, Igor Adameyko, Sten Linnarsson, and Peter V. Kharchenko. RNA velocity of single cells. *Nature*, 560(7719):494–498, 2018.

[114] John Lafferty and Guy Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6(5):129–163, 2005.

[115] Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. Tree-Sliced Variants of Wasserstein Distances. In *Advances in Neural Information Processing Systems 33*, 2019.

[116] Andrew Leaver-Fay, Michael Tyka, Steven M. Lewis, Oliver F. Lange, James Thompson, Ron Jacak, Kristian Kaufman, P. Douglas Renfrew, Colin A. Smith, Will Sheffler, Ian W. Davis, Seth Cooper, Adrien Treuille, Daniel J. Mandell, Florian Richter, Yih-En Andrew Ban, Sarel J. Fleishman, Jacob E. Corn, David E. Kim, Sergey Lyskov, Monica Berrondo, Stuart Mentzer, Zoran Popović, James J. Havranek, John Karanicolas, Rhiju Das, Jens Meiler, Tanja Kortemme, Jeffrey J. Gray, Brian Kuhlman, David Baker, and Philip Bradley. Rosetta3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. *Methods Enzymol*, 487:545–574, 2011. ISSN 0076-6879. doi: 10.1016/B978-0-12-381270-4.00019-6.

[117] Yann LeCun, B. Boser, J. S. Denker, D Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropogation Applied to Handwritten Zip Code Recognition. In *Neural Computation*, 1989.

[118] Alex R Lederer and Gioele La Manno. The emergence and promise of single-cell temporal-omics approaches. *Current Opinion in Biotechnology*, 63:70–78, 2020.

[119] William Leeb. *Topics in Metric Approximation.* PhD thesis, Yale University, 2015.

[120] William Leeb. The mixed Lipschitz space and its dual for tree metrics. *Applied and Computational Harmonic Analysis*, 44(3):584–610, May 2018. ISSN 10635203. doi: 10.1016/j.acha.2016.06.008.

[121] William Leeb and Ronald Coifman. Hölder–Lipschitz Norms and Their Duals on Spaces with Semigroups, with Applications to Earth Mover's Distance. *J Fourier Anal Appl*, 22(4):910–953, August 2016. ISSN 1069-5869, 1531-5851. doi: 10.1007/s00041-015-9439-5.

[122] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 3538–3545. Association for the Advancement of Artificial Intelligence, 2018.

[123] Wenfei Li, Jun Wang, Jian Zhang, and Wei Wang. Molecular simulations of metal-coupled protein folding. *Current Opinion in Structural Biology*, 30:25–31, February 2015. ISSN 0959-440X. doi: 10.1016/j.sbi.2014.11.006.

[124] Xiaoming Li, Chao Liu, Zhi Mao, Minglu Xiao, Li Wang, Shuang Qi, and Feihu Zhou. Predictive values of neutrophil-to-lymphocyte ratio on disease severity and mortality in COVID-19 patients: a systematic review and meta-analysis. *Critical Care*, 24(1), 2020.

[125] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable Gradients for Stochastic Differential Equations. *Artif. Intell. Stat.*, 2020.

[126] Renjie Liao, Alexander Schwing, Richard S Zemel, and Raquel Urtasun. Learning Deep Parsimonious Representations. In *NeurIPS*, 2016.

[127] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S Zemel. LanczosNet: Multi-Scale Deep Graph Convolutional Networks. In *ICLR*, 2019.

[128] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, December 2007. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0709640104.

[129] Matthias Liero, Alexander Mielke, and Giuseppe Savaré. Optimal Entropy-Transport problems and a new Hellinger–Kantorovich distance between positive measures. *Invent. math.*, 211(3):969–1117, 2018.

[130] Ofir Lindenbaum, Jay Stanley, Guy Wolf, and Smita Krishnaswamy. Geometry Based Data Generation. In *Advances in Neural Infromation Processing Systems 31*, pages 1400–1411, 2018.

[131] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-Based Anomaly Detection. *ACM Trans. Knowl. Discov. Data*, 6(1):1–39, March 2012. ISSN 15564681. doi: 10.1145/2133360.2133363.

[132] Longqi Liu, Chuanyu Liu, Andrés Quintero, Liang Wu, Yue Yuan, Mingyue Wang, Mengnan Cheng, Lizhi Leng, Liqin Xu, Guoyi Dong, Rui Li, Yang Liu, Xiaoyu Wei, Jiangshan Xu, Xiaowei Chen, Haorong Lu, Dongsheng Chen, Quanlei Wang, Qing Zhou, Xinxin Lin, Guibo Li, Shiping Liu, Qi Wang, Hongru Wang, J. Lynn Fink, Zhengliang Gao, Xin Liu, Yong Hou, Shida Zhu, Huanming Yang, Yunming Ye, Ge Lin, Fang Chen, Carl Herrmann, Roland Eils, Zhouchun Shang, and Xun Xu. Deconvolution of single-cell multi-omics layers reveals regulatory heterogeneity. *Nat Commun*, 10(1):470, December 2019. ISSN 2041-1723. doi: 10.1038/s41467-018-08205-7.

[133] Ruishan Liu, James Zou, and Akshay Balsubramani. Learning Transport Cost From

Subset Correspondence. In *International Conference on Learning Representations*, page 15, 2020.

[134] Nikos K. Logothetis, Jon Pauls, Mark Augath, Torsten Trinath, and Axel Oeltermann. Neurophysiological investigation of the basis of the fMRI signal. *Nature*, 412(6843): 150–157, July 2001. ISSN 0028-0836, 1476-4687. doi: 10.1038/35084005.

[135] Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. Break the Ceiling: Stronger Multi-scale Deep Graph Convolutional Networks. In *Advances in Neural Information Processing Systems 33*, 2019.

[136] Carolina Lucas, , Patrick Wong, Jon Klein, Tiago B. R. Castro, Julio Silva, Maria Sundaram, Mallory K. Ellingson, Tianyang Mao, Ji Eun Oh, Benjamin Israelow, Takehiro Takahashi, Maria Tokuyama, Peiwen Lu, Arvind Venkataraman, Annsea Park, Subhasis Mohanty, Haowei Wang, Anne L. Wyllie, Chantal B. F. Vogels, Rebecca Earnest, Sarah Lapidus, Isabel M. Ott, Adam J. Moore, M. Catherine Muenker, John B. Fournier, Melissa Campbell, Camila D. Odio, Arnau Casanovas-Massana, Roy Herbst, Albert C. Shaw, Ruslan Medzhitov, Wade L. Schulz, Nathan D. Grubaugh, Charles Dela Cruz, Shelli Farhadian, Albert I. Ko, Saad B. Omer, and Akiko Iwasaki. Longitudinal analyses reveal immunological misfiring in severe COVID-19. *Nature*, 584:463–469, 2020.

[137] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[138] Evan Z. Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R. Bialas, Nolan Kamitaki, Emily M. Martersteck, John J. Trombetta, David A. Weitz, Joshua R. Sanes, Alex K. Shalek, Aviv Regev, and Steven A. McCarroll. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell*, 161(5):1202–1214, 2015.

[139] Stéphane Mallat. Group Invariant Scattering. *Commun. Pure Appl. Math.*, LXV: 1331–1398, 2012.

[140] John C Marshall, Srinivas Murthy, Janet Diaz, N K Adhikari, Derek C Angus, Yaseen M Arabi, Kenneth Baillie, Michael Bauer, Scott Berry, Bronagh Blackwood, Marc Bonten, Fernando Bozza, Frank Brunkhorst, Allen Cheng, Mike Clarke, Vu Quoc Dat, Menno de Jong, Justin Denholm, Lennie Derde, Jake Dunning, Xiaobin Feng, Tom Fletcher, Nadine Foster, Rob Fowler, Nina Gobat, Charles Gomersall, Anthony Gordon, Thomas Glueck, Michael Harhay, Carol Hodgson, Peter Horby, YaeJean Kim, Richard Kojan, Bharath Kumar, John Laffey, Denis Malvey, Ignacio Martin-Loeches, Colin McArthur, Danny McAuley, Stephen McBride, Shay McGuinness, Laura Merson, Susan Morpeth, Dale Needham, Mihai Netea, Myoung-Don Oh, Sabai Phyu, Simone Piva, Ruijin Qiu, Halima Salisu-Kabara, Lei Shi, Naoki Shimizu, Jorge Sinclair, Steven Tong, Alexis Turgeon, Tim Uyeki, Frank van de Veerdonk, Steve Webb, Paula Williamson, Timo Wolf, and Junhua Zhang. A minimal common outcome measure set for COVID-19 clinical research. *The Lancet Infectious Diseases*, 20(8):e192–e197, 2020.

[141] Matthew L. Meizlish, Alexander B. Pine, Jason D. Bishai, George Goshua, Emily R. Nadelmann, Michael Simonov, C-Hong Chang, Hanming Zhang, Marcus Shallow, Parveen Bahel, Kent Owusu, Yu Yamamoto, Tanima Arora, Deepak S. Atri, Amisha Patel, Rana Gbyli, Jennifer Kwan, Christine H. Won, Charles Dela Cruz, Christina Price, Jonathan Koff, Brett A. King, Henry M. Rinder, F. Perry Wilson, John Hwa, Stephanie Halene, William Damsky, David van Dijk, Alfred I. Lee, and Hyung J. Chun. A neutrophil activation signature predicts critical illness and mortality in COVID-19. medRxiv, DOI: 10.1101/2020.09.01.20183897, 2020.

[142] Wenwen Min, Juan Liu, and Shihua Zhang. Network-Regularized Sparse Logistic Regression Models for Clinical Risk Prediction and Biomarker Discovery. *IEEE CompBio*, 15(3):944–953, May 2018. ISSN 1545-5963. doi: 10.1109/TCBB.2016.2640303.

[143] Yimeng Min, Frederik Wenkel, and Guy Wolf. Scattering gcn: Overcoming oversmoothness in graph convolutional networks. *arXiv preprint arXiv:2003.08414*, 2020.

[144] Vivek Modi, Qifang Xu, Sam Adhikari, and Roland L. Dunbrack. Assessment of

Template-Based Modeling of Protein Structure in CASP11. *Proteins*, 84(Suppl 1): 200–220, September 2016. ISSN 0887-3585. doi: 10.1002/prot.25049.

[145] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Hist. Académie R. Sci.*, 1781.

[146] Kevin R. Moon, David van Dijk, Zheng Wang, Daniel Burkhardt, William Chen, Antonia van den Elzen, Matthew J Hirn, Ronald R Coifman, Natalia B Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing transitions and structure for high dimensional data exploration. *bioRxiv*, 2017. doi: 10.1101/120378. URL `https://www.biorxiv.org/content/early/2017/12/01/120378`.

[147] Kevin R. Moon, Jay S. Stanley, Daniel Burkhardt, David van Dijk, Guy Wolf, and Smita Krishnaswamy. Manifold learning-based methods for analyzing single-cell RNA-sequencing data. *Current Opinion in Systems Biology*, 7:36–46, February 2018. ISSN 24523100. doi: 10.1016/j.coisb.2017.12.008.

[148] Kevin R. Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B. Burkhardt, William S. Chen, Kristina Yim, Antonia van den Elzen, Matthew J. Hirn, Ronald R. Coifman, Natalia B. Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing structure and transitions in high-dimensional biological data. *Nat Biotechnol*, 37(12): 1482–1492, 2019.

[149] John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, Torsten Schwede, and Anna Tramontano. Critical assessment of methods of protein structure prediction (CASP)—Round XII. *Proteins Struct. Funct. Bioinforma.*, 86(S1):7–15, 2018. ISSN 1097-0134. doi: 10.1002/prot.25415.

[150] Debarghya Mukherjee, Aritra Guha, Justin Solomon, Yuekai Sun, and Mikhail Yurochkin. Outlier-Robust Optimal Transport. *ICML*, 2020.

[151] Boris Muzellec and Marco Cuturi. Subspace Detours: Building Transport Plans that are Optimal on Subspace Projections. In *Advances in Neural Information Processing Systems 32*, pages 6917–6928, 2019.

[152] Jim Oeppen and James W. Vaupel. Broken Limits to Life Expectancy. *Science*, 296 (5570):1029–1031, 2002.

[153] Seiji Ogawa and Tso-Ming Lee. Magnetic resonance imaging of blood vessels at high fields: In vivo and in vitro measurements and image simulation. *Magn. Reson. Med.*, 16(1):9–18, 1990. ISSN 1522-2594. doi: 10.1002/mrm.1910160103.

[154] Sergey Ovchinnikov, Hahnbeom Park, David E. Kim, Frank DiMaio, and David Baker. Protein structure prediction using Rosetta in CASP12. *Proteins Struct. Funct. Bioinforma.*, 86(S1):113–121, 2018. ISSN 1097-0134. doi: 10.1002/prot.25390.

[155] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep Learning for Anomaly Detection: A Review. *ArXiv200702500 Cs Stat*, July 2020.

[156] Nicolas Papadakis, Gabriel Peyré, and Edouard Oudet. Optimal Transport with Proximal Splitting. *SIAM J. Imaging Sci.*, 7(1):212–238, 2014.

[157] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems 30*, pages 2338–2347, 2017.

[158] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 33*, pages 8026–8037, 2019.

[159] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. Scikit-learn: Machine Learning in Python. *Mach. Learn. PYTHON*, 2011.

[160] Ofir Pele and Michael Werman. Fast and robust Earth Mover's Distances. In *2009*

IEEE 12th International Conference on Computer Vision, pages 460–467, Kyoto, September 2009. IEEE. ISBN 978-1-4244-4420-5. doi: 10.1109/ICCV.2009.5459199.

[161] Brandt D. Pence. Severe COVID-19 and aging: are monocytes the key? *GeroScience*, 42(4):1051–1061, 2020.

[162] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. OCGAN: One-class Novelty Detection Using GANs with Constrained Latent Representations. *ArXiv190308550 Cs*, March 2019.

[163] Michael Perlmutter, Guy Wolf, and Matthew Hirn. Geometric scattering on manifolds. In *NeurIPS 2018 Workshop on Integration of Deep Learning Theories*, 2018.

[164] Michael Perlmutter, Feng Gao, Guy Wolf, and Matthew Hirn. Understanding graph neural networks with asymmetric geometric scattering transforms. *arXiv preprint arXiv:1911.06253*, 2019.

[165] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport*. arXiv:1803.00567, 2019.

[166] Stanislav Pidhorskyi, Ranya Almohsen, Donald A. Adjeroh, and Gianfranco Doretto. Generative Probabilistic Novelty Detection with Adversarial Autoencoders. In *arXiv:1807.02588 [Cs]*, July 2018.

[167] Peng Qiu. Embracing the dropouts in single-cell RNA-seq analysis. *Nat Commun*, 11 (1):1169, December 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-14976-9.

[168] Adityanarayanan Radhakrishnan, Karren Yang, Mikhail Belkin, and Caroline Uhler. Memorization in Overparameterized Autoencoders. *ArXiv181010333 Cs Stat*, September 2019.

[169] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. In *MOD*, page 12, Dalles, TX, 2000.

[170] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1530–1538, 2015.

[171] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *Proceedings of the 29th International Conference on Machine Learning*, pages 833–840, 2011.

[172] Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. *ArXiv170303717 Cs Stat*, March 2017.

[173] Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Lucas Deecke, Shoaib A Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep One-Class Classification. In *ICML*, page 10, Stockholm, Sweden, 2018.

[174] Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Müller. A Unifying Review of Deep and Shallow Anomaly Detection. *ArXiv200911732 Cs Stat*, September 2020.

[175] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially Learned One-Class Classifier for Novelty Detection. In *CVPR*, June 2018. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00356.

[176] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic Routing Between Capsules. In *31st Conference on Neural Information Processing Systems*, October 2017.

[177] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. A comparison of single-cell trajectory inference methods. *Nat Biotechnol*, 37(5):547–554, 2019.

[178] Mayu Sakurada and Takehisa Yairi. Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. In *MLSDA*, Australia, 2014. ISBN 978-1-4503-3159-3. doi: 10.1145/2689746.2689747.

[179] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Fast Unbalanced Optimal Transport on a Tree. *Adv. Neural Inf. Process. Syst. 34*, 2020.

[180] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini. The Graph Neural Network Model. *IEEE Trans. Neural Netw.*, 20(1):61–80, January 2009. ISSN 1045-9227, 1941-0093. doi: 10.1109/TNN.2008.2005605.

[181] Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, Lia Lee, Jenny Chen, Justin Brumbaugh, Philippe Rigollet, Konrad Hochedlinger, Rudolf Jaenisch, Aviv Regev, and Eric S. Lander. Optimal-Transport Analysis of Single-Cell Gene Expression Identifies Developmental Trajectories in Reprogramming. *Cell*, 176(4): 928–943.e22, 2019.

[182] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *IPML*, 2017.

[183] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.*, 13(7):1443–1471, July 2001. ISSN 0899-7667, 1530-888X. doi: 10.1162/089976601750264965.

[184] Stefan Schulz and Gunnar O. Klein. Snomed ct – advances in concept mapping, retrieval, and ontological foundations. selected contributions to the semantic mining conference on snomed ct (smcs 2006). *BMC Medical Informatics and Decision Making*, 8(1):S1, 2008. ISSN 1472-6947. doi: 10.1186/1472-6947-8-S1-S1. URL https://doi.org/10.1186/1472-6947-8-S1-S1.

[185] Manu Setty, Michelle D Tadmor, Shlomit Reich-Zeliger, Omer Angel, Tomer Meir Salame, Pooja Kathail, Kristy Choi, Sean Bendall, Nir Friedman, and Dana Pe'er. Wishbone identifies bifurcating developmental trajectories from single-cell data. *Nature Biotechnology*, 34(6):637, 2016.

[186] Eric Shifrut, Julia Carnevale, Victoria Tobin, Theodore L. Roth, Jonathan M. Woo, Christina T. Bui, P. Jonathan Li, Morgan E. Diolaiti, Alan Ashworth, and Alexander Marson. Genome-wide CRISPR screens in primary human t cells reveal key regulators of immune function. *Cell*, 175(7):1958–1971.e15, 2018.

[187] Sameer Shirdhonkar and David W. Jacobs. Approximate earth mover's distance in linear time. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, 2008. IEEE. ISBN 978-1-4244-2242-5. doi: 10.1109/CVPR.2008.4587662.

[188] David I Shuman, Pierre Vandergheynst, and Pascal Frossard. Chebyshev polynomial approximation for distributed signal processing. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–8, Barcelona, Spain, June 2011. IEEE. ISBN 978-1-4577-0512-0. doi: 10.1109/DCOSS.2011.5982158.

[189] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[190] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices, 1964.

[191] Justin Solomon. Optimal Transport on Discrete Domains. *ArXiv180107745 Cs Math*, May 2018.

[192] Justin Solomon, Fernando de Goes, Gabriel Peyre, Univ Paris-Dauphine, and Marco Cuturi. Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains. In *ACM Transactions on Graphics*, 2015.

[193] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R. G. Lanckriet. On integral probability metrics, \phi-divergences and binary classification. *ArXiv09012698 Cs Math*, January 2009.

[194] Austin Stone, Huayan Wang, Michael Stark, Yi Liu, D. Scott Phoenix, and Dileep George. Teaching Compositionality to CNNs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 732–741, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.85.

[195] David M.J. Tax and Robert P.W. Duin. Support Vector Data Description. *Mach. Learn.*, 54(1):45–66, January 2004. ISSN 0885-6125. doi: 10.1023/B:MACH.0000008084.60811.49.

[196] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma. Statistical evaluation of the Predictive Toxicology Challenge 2000-2001. *Bioinformatics*, 19(10):1183–1193, July 2003. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btg130.

[197] Alexander Tong and Smita Krishnaswamy. Interpolating optimal transport barycenters of patient manifolds, July 2020.

[198] Alexander Tong, Jessie Huang, Guy Wolf, David van Dijk, and Smita Krishnaswamy. TrajectoryNet: A Dynamic Optimal Transport Network for Modeling Cellular Dynamics. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[199] Alexander Tong, Guy Wolf, and Smita Krishnaswamy. Fixing Bias in Reconstruction-based Anomaly Detection with Lipschitz Discriminators. In *IEEE MLSP*, Espoo, Finland, 2020.

[200] Alexander Tong, Guillaume Huguet, Amine Natik, Kincaid MacDonald, Manik Kuchroo, Ronald Coifman, Guy Wolf, and Smita Krishnaswamy. Diffusion Earth Mover's Distance and Distribution Embeddings. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 10336–10346. PMLR, 2021.

[201] William Torous, Florian Gunsilius, and Philippe Rigollet. An Optimal Transport Approach to Causal Inference. *ArXiv210805858 Econ Math Stat*, August 2021.

[202] Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J Lennon, Kenneth J Livak, Tarjei S Mikkelsen, and John L Rinn. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nat Biotechnol*, 32(4):381–386, 2014.

[203] Lloyd N. Trefethen. *Approximation Theory and Approximation Practice*. Society for Industrial and Applied Mathematics, 2013.

[204] Laurens van der Maaten and Geoffrey E Hinton. Visualizing Data using t-SNE. *J. Mach. Learn. Res.*, 2008.

[205] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *6th Int. Conf. Learn. Represent.*, 2018.

[206] Cedric Villani. *Optimal Transport, Old and New*. Springer, June 2008. ISBN 978-3-540-71050-9.

[207] Cedric Villani. *Optimal Transport: Old and New*. Springer, Berlin, 2009.

[208] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *JMLR*, pages 3371–3408, 2010.

[209] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *JMLR*, pages 3371–3408, 2010.

[210] C. H. Waddington. The epigenotype. *Endeavour*, 1:18–20, 1942.

[211] Nikil Wale, Ian A Watson, and George Karypis. Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification. *Knowl. Inf. Syst.*, 2008.

[212] Feng-Yu Wang et al. Sharp explicit lower bounds of heat kernels. *Annals of Probability*, 25(4):1995–2006, 1997.

[213] Jonathan Weed and Francis Bach. Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance. *Bernoulli*, 25(4A):2620–2648, 2019.

[214] Caleb Weinreb, Samuel Wolock, Betsabeh K. Tusi, Merav Socolovsky, and Allon M. Klein. Fundamental limits on dynamic inference from single-cell snapshots. *Proc Natl Acad Sci USA*, 115(10):E2467–E2476, 2018.

[215] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chem. Sci.*, 9(2):513–530, 2018. ISSN 2041-6520, 2041-6539. doi: 10.1039/C7SC02664A.

[216] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *ArXiv190100596 Cs Stat*, January 2019.

[217] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful Are Graph Neural Networks? *ICLR*, 2019.

[218] Pinar Yanardag and S.V.N. Vishwanathan. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, pages 1365–1374, Sydney, NSW, Australia, 2015. ACM Press. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2783417.

[219] Karren D Yang and Caroline Uhler. Scalable Unbalanced Optimal Transport Using Generative Adversarial Networks. In *7th International Conference on Learning Representations*, page 20, 2019.

[220] Zhenzhang Ye, Thomas Möllenhoff, Tao Wu, and Daniel Cremers. Optimization of Graph Total Variation via Active-Set-based Combinatorial Reconditioning. *AISTATS*, 2020.

[221] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *ArXiv13112901 Cs*, November 2013.

[222] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient GAN-Based Anomaly Detection. *ArXiv180206222 Cs Stat*, 2018.

[223] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable Convolutional Neural Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, Salt Lake City, UT, June 2018. IEEE. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00920.

[224] Manqi Zhao and Venkatesh Saligrama. Anomaly Detection with Score functions based on Nearest Neighbor Graphs. In *NeurIPS*, 2009.

[225] Dengyong Zhou and Bernhard Schölkopf. A regularization framework for learning from graph data. In *ICML workshop on statistical relational learning and Its connections to other fields*, volume 15, pages 67–8, 2004.

[226] Naihui Zhou, Yuxiang Jiang, Timothy R. Bergquist, Alexandra J. Lee, Balint Z. Kacsoh, Alex W. Crocker, Kimberley A. Lewis, George Georghiou, Huy N. Nguyen, Md Nafiz Hamid, Larry Davis, Tunca Dogan, Volkan Atalay, Ahmet S. Rifaioglu, Alperen Dalkıran, Rengul Cetin Atalay, Chengxin Zhang, Rebecca L. Hurto, Peter L. Freddolino, Yang Zhang, Prajwal Bhat, Fran Supek, José M. Fernández, Branislava Gemovic, Vladimir R. Perovic, Radoslav S. Davidović, Neven Sumonja, Nevena Veljkovic, Ehsaneddin Asgari, Mohammad R.K. Mofrad, Giuseppe Profiti, Castrense Savojardo, Pier Luigi Martelli, Rita Casadio, Florian Boecker, Heiko Schoof, Indika Kahanda, Natalie Thurlby, Alice C. McHardy, Alexandre Renaux, Rabie Saidi, Julian Gough, Alex A. Freitas, Magdalena Antczak, Fabio Fabris, Mark N. Wass, Jie Hou, Jianlin Cheng, Zheng Wang, Alfonso E. Romero, Alberto Paccanaro, Haixuan Yang, Tatyana Goldberg, Chenguang Zhao, Liisa Holm, Petri Törönen, Alan J. Medlar, Elaine Zosa, Itamar Borukhov, Ilya Novikov, Angela Wilkins, Olivier Lichtarge, Po-Han Chi, Wei-Cheng Tseng, Michal Linial, Peter W. Rose, Christophe Dessimoz, Vedrana Vidulin, Saso Dzeroski, Ian Sillitoe, Sayoni Das, Jonathan Gill Lees, David T. Jones, Cen Wan, Domenico Cozzetto, Rui Fa, Mateo Torres, Alex War-

wick Vesztrocy, Jose Manuel Rodriguez, Michael L. Tress, Marco Frasca, Marco Notaro, Giuliano Grossi, Alessandro Petrini, Matteo Re, Giorgio Valentini, Marco Mesiti, Daniel B. Roche, Jonas Reeb, David W. Ritchie, Sabeur Aridhi, Seyed Ziaeddin Alborzi, Marie-Dominique Devignes, Da Chen Emily Koo, Richard Bonneau, Vladimir Gligorijević, Meet Barot, Hai Fang, Stefano Toppo, Enrico Lavezzo, Marco Falda, Michele Berselli, Silvio C.E. Tosatto, Marco Carraro, Damiano Piovesan, Hafeez Ur Rehman, Qizhong Mao, Shanshan Zhang, Slobodan Vucetic, Gage S. Black, Dane Jo, Erica Suh, Jonathan B. Dayton, Dallas J. Larsen, Ashton R. Omdahl, Liam J. McGuffin, Danielle A. Brackenridge, Patricia C. Babbitt, Jeffrey M. Yunes, Paolo Fontana, Feng Zhang, Shanfeng Zhu, Ronghui You, Zihan Zhang, Suyang Dai, Shuwei Yao, Weidong Tian, Renzhi Cao, Caleb Chandler, Miguel Amezola, Devon Johnson, Jia-Ming Chang, Wen-Hung Liao, Yi-Wei Liu, Stefano Pascarelli, Yotam Frank, Robert Hoehndorf, Maxat Kulmanov, Imane Boudellioua, Gianfranco Politano, Stefano Di Carlo, Alfredo Benso, Kai Hakala, Filip Ginter, Farrokh Mehryary, Suwisa Kaewphan, Jari Björne, Hans Moen, Martti E.E. Tolvanen, Tapio Salakoski, Daisuke Kihara, Aashish Jain, Tomislav Šmuc, Adrian Altenhoff, Asa Ben-Hur, Burkhard Rost, Steven E. Brenner, Christine A. Orengo, Constance J. Jeffery, Giovanni Bosco, Deborah A. Hogan, Maria J. Martin, Claire O'Donovan, Sean D. Mooney, Casey S. Greene, Predrag Radivojac, and Iddo Friedberg. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biology*, 20(1):244, 2019. ISSN 1474-760X. doi: 10.1186/s13059-019-1835-8.

[227] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Stat. Anal. Data Min. ASA Data Sci. J.*, 5(5):363–387, 2012. ISSN 1932-1872. doi: 10.1002/sam.11161.

[228] Dongmian Zou and Gilad Lerman. Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis*, 2019. ISSN 10635203. doi: 10.1016/j.acha.2019.06.003.