

Machine Learning with Topological Data Analysis

A Dissertation Presented for the
Doctor of Philosophy
Degree

The University of Tennessee, Knoxville

Ephraim Robert Love

May 2021

© by Ephraim Robert Love, 2021
All Rights Reserved.

Dedication

“There are two ways to live. You can live as if nothing is a miracle. You can live as if everything is a miracle.”
- Dr. Albert Einstein

Dedicated to my parents who always encouraged my sense of wonder. You taught me to see the many miracles.

Acknowledgments

I would first like to express my deepest thanks to my advisor, Dr. Vasileios Maroulas, who fought for me to be able to pursue my passions. Dr. Maroulas welcomed me into the world of mathematics and believed in me.

I would like to thank my committee: Dr. Gunnar Carlsson, Dr. Blair Christian, Dr. Theodore Papamarkou, and Dr. Ioannis Sgouralis. You have each provided me with indispensable advice throughout this endeavor.

I would like to acknowledge the National Science Foundation which partially supported my research via grant MCB-1715794.

Mentorship has played a critical role in my journey through the education system. I would like to extend a special thanks, in order of appearance in my life, to several of my mentors: Shoshana Love, Dr. Karl Malcolm, Cheryl Wells, Dr. Brian Min, Dr. Nicholas Nagle, Dr. Blair Christian, Guruprasad Kora, and Dr. Jason Reminick; each of whom selflessly provided me critical support when I needed it most.

Finally, I am eternally indebted to my wife, Sarah Jayne Love, for the love and support she has provided, encouraging me to pursue my doctorate, and throughout my studies. I wouldn't have started or finished without Sarah by my side. Despite the hard work, and occasional exam-induced terror, these years have been some of the best of my life thanks mostly to Sarah's presense in them.

Abstract

Topological Data Analysis (TDA) is a relatively new focus in the fields of statistics and machine learning. Methods of exploiting the geometry of data, such as clustering, have proven theoretically and empirically invaluable. TDA provides a general framework within which to study topological invariants (shapes) of data, which are more robust to noise and can recover information on higher dimensional features than immediately apparent in the data. A common tool for conducting TDA is persistence homology, which measures the significance of these invariants. Persistence homology has prominent realizations in methods of data visualization, statistics and machine learning. Extending ML with TDA and persistence homology offers greater interpretability and generalizability than can be achieved with current state of the art tools. My work explores several applications of TDA and persistence homology. TDA has previously been used to track the movements of intracellular bodies and it follows that it could also be used as a tool to examine the cytoskeleton of cells as a network captured in confocal microscopic images. I show recent and developing work in an application of persistence homology for the classification of intra-cellular networks. In addition to persistence homology, methods from TDA can be used to partition data into hierarchical clusters. I explore empirical and theoretical findings from topological clustering of images and graphs in order to propose a novel generalization on the convolutional neural network (CNN) and contrast its empirical results with the traditional CNN.

Table of Contents

Introduction	1
1 Literature Review	4
1.1 Foundations of Topological Data Analysis	4
1.1.1 Topological Invariants and Combinatorics	4
1.1.2 Simplicial Complexes	5
1.1.3 Homology and Filtrations	6
1.1.4 Persistence Homology	8
1.1.5 Statistical Developments	11
1.1.6 Applications in Signals	14
1.1.7 Conclusion	16
1.2 Cytoskeletal Actin Network Classification	17
1.3 Topological Study of Neural Networks	19
1.3.1 Neural Networks	19
1.3.2 Persistence Features as Input to NNs	20
1.3.3 TDA in NNs	22
1.3.4 TDA in CNNs	22
1.3.5 Topologically Inspired CNNs	23
1.4 Conclusions	23
2 Persistence Homology Classification of Cytoskeletal Actin Filament Networks	24
2.1 Background	24
2.2 Persistence Homology and Filament Networks	26
2.2.1 Data	26
2.2.2 Persistence Homology	27
2.3 Filament network classifier	29
2.3.1 Distance-based network classifier	29
2.3.2 Non-distance-based network classifier	32
2.4 Classification result	33
2.4.1 d_p^c -based classifier	33
2.4.2 non-distance based	35

2.4.3	Comparison to other classifiers	36
2.4.4	Application to classifying myosin mutants	36
2.5	Clustering of filament networks	37
2.6	Conclusion	38
3	Topological CNNs	44
3.1	Introduction	45
3.2	Background: CNNs	48
3.3	Topological Convolutional Layers	52
3.3.1	2D Images	53
3.3.2	Video	57
3.3.3	Gabor filters versus Klein bottle filters	59
3.4	2D Images	61
3.4.1	Experiments and Results	61
3.4.2	Details of methods	66
3.5	Video	72
3.5.1	Experiments and Results	72
3.5.2	Details of methods	76
	Conclusion	80
	Bibliography	81
	Vita	90

List of Tables

2.1	Accuracy rate of classifiers	39
3.1	Size, dimensions and sources of all image datasets used in this work.	63
3.2	Test and training splits for numeric image datasets.	67
3.3	Metaparameters correspondent to figures in this work.	67
3.4	Video dataset metadata.	74
3.5	Train/test split for UCF-101.	77
3.6	Metaparameters for video experiment figures.	77
3.7	Residual block for our 12-convolutional layer ResNet where x is an input tensor, data passes from top to bottom, so $x + y$ is the returned tensor. . . .	78
3.8	Structure of our 12-convolutional layer ResNet with order of operations indicated by descending through the table. The first column of the table indicates the kernel size of a convolutional layer or layers in a residual block.	78

List of Figures

1.1	Examples of betti numbers used to characterize topological holes in 0 to 3-dimensional space.	7
1.2	Simplices in increasing dimensions from 0-dimensional to 3-dimensional connected components. From left to right: a 0-dimensional point to a 3-dimensional tetrahedron.	7
1.3	Data in the center and their potential simplicial complex-representations on either side. The left complex consisting of 1 0-dimensional connected component, 2 1-dimensional connected components, 1 2-dimensional connected component and 1 3-dimensional connected component. In the right complex, the 2-dimensional face is not yet fully connected, and so represents a 2-dimensional loop.	7
1.4	The union-of-spheres with radius (ϵ) increasing from left to right. The leftmost figure has 100 0-dimensional connected components with very small spheres about the data, the second figure has grown the spheres sufficiently to merge the original components into 1 1-dimensional connected component and 1 2-dimensionanl hole. The third shows a very large radius at which all of the spheres have merged and we have 1 connected component again.	9
1.5	Example of a union-of-spheres at which the Čech and VR filtrations would differ. In this example, if t is sufficiently small, than the VR complex would consider these to be a 3-way connected component, where the Čech filtration would leave them as pairwise connected with a hole between them.	9
1.6	Approach depiction of making persistence homology. Beginning from $\epsilon = 0$, there are 5 components. We show the key ϵ values where components are born or die. When $\epsilon = 0.5$, 5 components stay alive. When $\epsilon = 0.71$, 4 components connect into one connected component, forming a 1 hole, thus 3 components die and 1 hole is born. When $\epsilon = 0.85$, 2 components further connect into 1, therefore, 1 more component dies. Finally, when $\epsilon = 1$, the only 1 hole dies and only the 1 connected component survives. No additional homological change while ϵ is grown to 1.25. The persistence homology of this complex is summarized in the persistence barcode and diagrams in Figure 1.7.	10

1.7	A persistence barcode and two forms of the persistence diagram. These data belong to the simplices in the union-of-spheres example in Figure 1.5. Notice that the right is simply the rotation of the line $birth = death$ on the left to the x -axis on the right.	15
1.8	An example of a realization of a persistence landscape from a persistence diagram. Note that λ_0 appears dominated by the VR computation's chosen limit of $\epsilon \leq 3$ and do to the simplicity of the example.	15
1.9	An example of a level-set embedding, from a 1-dimensional pseudocontinuous function, to a persistence barcode and to a persistence diagram. Note that in the persistence diagram, the diagonal ($birth = death$), indicates the addition of infinite points of infinite density to the diagonal for purposes of measuring the distance of sets with uneven cardinalities. This figure was taken from [1] pg. 18.	18
1.10	Example of a confocal microscopic image of a potato root hair cell. This cell is an example of a cell which has had the production of a particular myosin protein knocked-out through mutation. The image is very long and narrow, since the cell is a hair. The images are in very high resolution, but include a great deal of noise, as well as real features which pass in and out of focus since this is a single slice of a 3-dimensional object. The image was graciously provided by Jinmo Gu.	18
1.11	Demonstration of 1 and 2 dimensional binning methods for the vectorization of persistence barcodes, landscapes, diagrams, and images. There are many methods of vectorization, retaining various aspects of the prior persistence information, so these are merely meant to demonstrate several commonly used methods. Notably, the method I chose for the PD, does not produce a standard error for the holes. This might naturally motivate a researcher to choose another method when they have a very small number of important, higher dimensional features in much of their data. Also note that the vectorizations we chose for the barcode and landscape are discretized 1-dimensional signals and the vectorized persistence image is a 2-dimensional signal. This means that all of these could be treated as “flat” vectors, or operated on as signals, such as done in a CNN.	21
2.1	Filament networks. Panel (a) shows three filament networks generated by 825, 1650 and 3300 cross-linkers, respectively, in a $20\mu\text{m} \times 20\mu\text{m}$ area. Each network contains 100 filaments which are represented as red lines. The blue dots are the barbed ends of these filaments. Panel (b) shows the locations of the actin beads that make up the filaments exhibited in Panel (a).	28
2.2	Demonstration filament network. This network contains 3 filaments. Points are sampled along the filaments, in order to produce a point cloud from which persistence homology can be studied.	30

2.3	Investigation of the persistence homology of the demonstration filament network in Figure 2.2. The first row of figures, shows the growing ϵ -spheres about the sampled points of the fillaments. The second row shows the corresponding persistence barcode. The third row shows the corresponding persistence diagram. The columns progress with the algorithm, right-to-left.	30
2.4	Example of one filament construction (B) from the supplied point cloud (A), rasterization (C), re-sampling (D), construction of persistence diagram (E) and vectorization of persistence diagram (F). The point cloud in (A) was supplied with an index that allowed for the construction of the filaments in (B); note the coloring. This was projected onto a 200x200 raster (C). The raster was sampled into 3 sets (colored) of 1000 points (D) without replacement and the x,y coordinates are used to generate a VR complex. The persistence diagrams and vectorizations in (E) and (F) are colored the same as (D), to show the variability that arises in the vectorized persistence diagram features when this approach is taken.	34
2.5	Example of a raw WT cell image.	39
2.6	Plot of cell's persistence information principal components. Points are colored by treatment.	40
2.7	WGSS with several numbers of clusters (values of k).	41
2.8	Plot of cell's persistence information principle components. Points are colored by treatment and class assignments from k-means clustering are indicated by shape. The asterisks indicate the cluster centers.	42
2.9	Diagram showing misclassified cells based on k-means clustering with $k = 2$. The groups are separated by column and the Xs indicate cells that do not agree with the majority in the cluster. The left column is mostly mutant, so the X's indicate wildtypes. The right side is mostly wildtype, so the Xs indicate mutants.	43
3.1	Visual guide to CNN (green rectangle) and TCNN (red rectangles) architectures. A typical CNN layer for image classification takes in a 2-dimensional array (a) as an input image and convolves this image through a system of spatially localized filters (b) to produce multiple slices of processed data (c) called feature maps. The feature maps in (c) are flattened to a column vector in (d) and passed through a fully-connected layer to the output nodes (e) which are of cardinality equal to the number of classes in the prediction space. The TCNN modifies the typical CNN framework by specifying the weights in (b) to lie on a topological manifold such as the circle or Klein bottle. We indicate this choice by the dashed triangle in (b) and red rectangles indicating the selection of circle or Klein weights. This shows the behavior of CF and KF type TCNN layers. The COL and KOL type TCNN layers also modify the weights of a traditional CNN layer but in a different way: They localize weights with respect to the circle and Klein bottle topologies yet, as usual, instantiate them randomly and allow them to vary during training.	50

3.2	A visual representation of the neurons and weights in a KOL with threshold $s = 2$. Each intersection of grid lines on the two Klein bottles is a neuron. The red arrows depict the nonzero weights corresponding to the given input neuron.	56
3.3	Table of weights (top row) and activations (bottom row) for 3 convolutional layers (a),(b),(c) evaluated on a handwritten 5 from MNIST (initial data shown in Figure 3.1). The first column (a) is a normal CNN layer (NOL) trained on one epoch of MNIST data followed by 2 fully connected linear layers. This network has a testing accuracy of approximately 99%. Second (b) is a Circle Filters (CF) layer. Third (c) is a Klein Filters (KF) layer. . .	63
3.4	Two synthetic experiments on noisy MNIST data, designed to test model assumptions. The first column displays the results of the experiment where Gaussian noise is added to the training data but not the testing data. The second column displays the results of the experiment where the training data is the original MNIST data and the testing data are corrupted by Gaussian noise. The first row is testing accuracy and the second row is training loss. .	67
3.5	Sweep of distributions for synthetic MNIST data varying the mean and variance from which class-noise distributions are drawn. The first column shows the tested values of τ and the second column shows the tested values of ω^2 . The first row shows accuracies when training on data with Gaussian noise and testing on unaltered test set. The second row shows accuracies when training on unaltered training data and testing on noise-added test data. We show results after 1 epoch and 5 epochs of training.	68
3.6	Left: Comparisons of testing accuracy after training on 1,000 images. Right: Full comparison of testing accuracy (y-axes) over a single epoch. MNIST was trained on 60,000, SVHN on 50,032, and USPS on 7291 images.	69
3.7	Comparisons of testing accuracy and validation loss when generalizing from SVHN to MNIST and vice versa.	69
3.8	Comparisons of testing accuracy and validation loss when generalizing from CIFAR to Kaggle and vice versa.	70
3.9	Example of a small set of gabor filters and activations on a handwritten 5 from MNIST.	70
3.10	Top panel: testing accuracies on 3 digit datasets and 2 cats and dogs datasets after 100 training images. Bottom panel: testing accuracies after 1 full epoch of training. The KF model consistently performs similarly or better than the Gabor filters.	71
3.11	Examples of one frame from Weizmann (top row) and KTH (bottom row) videos from the same classes (columns).	74
3.12	Testing accuracy of a 12 layer ResNet trained on UCF-101.	75
3.13	Top 1, 5, and 10 testing overall classification accuracies as a function of epochs for both the TCNN ResNet (left) and the CNN ResNet (right).	75

3.14 Testing accuracy of a TCNN and a CNN trained on the KTH dataset and tested on the Weizmann dataset.	77
--	----

Introduction

A key thrust in statistical research is the pursuit of methods to extract novel information from limited data. Many researchers are currently focused on developing, analyzing and improving deep learning methods which, in a relatively opaque box, perform this high dimensional feature extraction. These many-layered networks capitalize on high-dimensional, latent features in data. To this point in time, reliable, automated methods are still unavailable for understanding the complete pipeline of feature generation and classification for most methods of deep learning. Topological data analysis (TDA) offers an alternative perspective on how to identify higher-dimensional features with convenient mathematical tools to transform, analyze and utilize these features [2, 3, 4, 5]. TDA could be framed as a generalization or an extension of many existing network topology methods, but TDA has several advantages stemming from recent theoretical mathematical developments, which will be discussed in my first chapter. From these novel formulations, researchers are developing measures, tests, classifiers and more. TDA is a promising avenue for data scientists interested in pushing the known boundaries of statistics and machine learning (ML).

The work presented herein is focused on the development of ML for signals, taking a topological perspective on the signal data. ML of spatial signals (digital images or videos) for the purpose of studying the context of these signals, or identifying objects within signals, has a large corpus with several distinct generations of work. Generally, this task requires a summary of the high dimensional digitized signal into some lower dimensional feature space which can be used to group or classify signals or elements of signals. Naturally, the strategies taken in tackling this task in one generation reflect both the state of mathematical and statistical ideas and the feasibility of their computation, the latter of which grows at an exponential pace. Early ML methods for image classification used crafted filters to dramatically reduce the dimensionality of images, such that a statistical tool like logistic regression could be used on a vector of summary information. In recent years, computation and large, labeled datasets have grown so cheap that we no longer craft filters at all and instead learn the relevant filters through gradient decent and backpropagation in the framework of a deep neural network [6].

The development of deep learning comes with incredible power, but also comes at notable cost. Deep neural networks are powerful ML tools, but how they arrive at their final predictions can be extremely difficult to interpret. This has lead at times to erroneous

inference based on network predictions, but also to researchers mistaking the fundamental function of general structures of the network [7]. In this work, I take a hybrid approach to signal processing wherein I use new tools from TDA both to summarize images into features (as in the classical image processing methods), and to construct deep learning models which are more easily interpreted and studied. The former is a sensible approach in small n cases, while the latter generalizes to deep learning and large n cases. Both of these studies seek to improve -situationally applicable- state of the art tools for learning spatial signals.

In Chapter 1, I provide a review of key background information for my work as well as a review of current state-of-the-art methods. TDA encompasses “a large class of data analysis methods that measure shape and connectivity.” [8] I draw a rough arc from topological features to their persistent representations in data. I demonstrate how to create topological filtrations through the construction of simplicial complexes. I then cover the more specific background to the development of the methods presented in the following two chapters. Finally, I review widely used contemporary applications of TDA as well as applications of TDA for neural networks, which is a new and rapidly developing corpus. This chapter is written primarily for an audience of data scientists.

In Chapter 2, I present an application of TDA, to study the connectivity of intracellular structures as they are captured in confocal microscopic images. The actin cytoskeleton is a complex network of proteins that is present in all eukaryotic cells and is critical to cellular functions [9]. The state-of-the-art methods for studying the topology of actin networks is time consuming, relies on significant preprocessing and human interjections, and results only in data only loosely correlated with dependent variables [10]. I demonstrate the utility of TDA to discern between several classes of simulated cells, differing in numbers of binding proteins which contribute to the connectivity of the actin network. This work is motivated in part by prior work showing the utility of TDA to track intracellular movements [11]. I demonstrate a novel method of data augmentation for TDA to improve these results. I demonstrate how to bridge the methods developed on synthetic data to the microscopy domain and analyze experimental data comparing cells with and without a genetic knockout. I further demonstrate the utility of TDA by constructing an unsupervised clustering, which allows for the examination of the clustering outcomes and errors in the original space of microscopic images. This work demonstrates and extends the utility of TDA in a biological imaging application.

In Chapter 3, I propose a theoretically and empirically motivated, novel generalization on the convolutional neural network (CNN). This framework draws on findings about the long-run topologies of well trained CNNs as well as theoretical knowledge about the space of natural images. I contrast the empirical results of a variety of such formulations with each other and a typical CNN. This work serves as a template for the development of limitless models with augmented topologies based on empirical knowledge and/or theoretical connections to a logical manifold. I demonstrate the efficacy of this generalization in several image and video classification experiments. These models are more easily interpreted and studied. The models train faster and can be trained on less data. The models can generalize better,

especially under a common scenario wherein one would like to train on a relatively simple dataset and generalize to much more complex dataset. The formalism, framework and empirical results of TCNNs are all elucidated.

As a whole, this work explores and advances applications of TDA, a new tool with broad use in machine learning. In this work I begin by reviewing the contemporary literature on TDA. I then demonstrate a method of summarizing images of networks into measures of the robustness of topological invariants which are used to classify and cluster the images. This can be seen as an augmented approach to older signal/image processing methods, which are improved upon by taking advantage of the topology of data. This method is particularly valuable in a complex setting with a small set of experimental data. I then demonstrate the utility of TDA and topology in order to study and generalize a method of deep learning which I call the TCNN. This is a novel generalization of the newer methods of deep learning which are particularly appropriate in situations with large, labeled datasets. In sum, the work presented herein will demonstrate the utility of a topological treatment of machine learning.

Chapter 1

Literature Review

1.1 Foundations of Topological Data Analysis

Topological data analysis (TDA) refers broadly to a set of analytic, computational and statistical methods stemming from the study of the shape and connectivity of data [12, 8]. All data can be considered as vectors in a space and therefore have corresponding shapes. The most familiar methods exploiting the shape of data are clustering methods. In density clustering for instance, one attempts to partition a space into relatively dense and sparse regions by considering distances on bijections of a nearest neighbors graph of the data, developing a set of regions [13]. TDA borrows from the fields of algebraic topology, computational geometry and from many traditional statistical methods, to study higher dimensional topological properties of data [8]. In this section I will summarize the most common path taken in statistical applications of TDA to transform data to measures of topology, and subsequently to statistics and machine learning.

1.1.1 Topological Invariants and Combinatorics

The key mathematical features of analysis in statistical TDA are homologic invariants (differentiable geometric features). Homologic invariants have a convenient hierarchical characterization that is expressed through a homology number. These homology numbers are just one type of topological invariant. Whereas geometric invariants typically require many detailed calculations, homology is a summary of geometric features which can be formulated in a less computationally intensive manner. The crossover from homology to statistics is neutral in some sense, since both disciplines' motivation lies in the summarizing of information. Geometers have long been interested in the behaviors of manifolds, a type of topological space like a smooth surface about an open set. The field of algebraic topology was in some ways born from Euler's description of his eponymous, summarizing characteristic. For a 3-dimensional pseudomanifold (a manifold that can actually have singularities from that smooth surface, like a pinched torus), the Euler characteristic [14] is provided in

Equation 1.1.1. V is the number of vertices, E is the number of edges, F is the number of faces, and C is the number of cells. $X(M) = 0$ for manifold M 's.

$$X(M) = V - E + F - C$$

The Euler characteristic relates the number of vertices, edges, faces and cells in a 3-dimensional pseudomanifold or polyhedron. It may be immediately apparent with the Euler characteristic alone how one can construct combinatoric relations of topological features. However, convention is to use the homology numbers, or betti numbers instead. Betti numbers were introduced as a dimensional generalization of Euler's work by the works of Riemann, Betti and finally Poincarè (the betti numbers will be our reference to homology features for this paper). The index n of the betti number β_n refers to the dimension of the space described (see Figure 1.1). Betti numbers can be independent across indices in n -dimensional submanifolds, but often the betti numbers are related. For instance, $\beta_0 = \beta_1$ for general 3-dimensional pseudomanifolds.

A body of early work in algebraic topology analyzed and characterized interesting features and behaviors of manifolds. This involved showing the relationships of dimensionality, spaces and connectedness through the betti numbers[14]. Table 2 shows how n -dimensional (in this case 0 to 3-dimensional) shapes would be mapped to betti numbers, β_n . β_0 counts the number of 0-dimensional components, β_1 the number of 1-dimensional holes, β_2 the number of 2-dimensional loops and β_3 the number of 3-dimensional open cells. Betti numbers can describe manifolds, polyhedra and, as we will now examine, simplicial complexes in $\mathbb{R}^n$.

1.1.2 Simplicial Complexes

A simplicial complex is a collection of simple pieces ("simplices") that can be represented as connected components (connected subgraphs) on a graph. Simplices are the simplest representations of n -dimensional triangles, i.e. the 0-simplex is a point, the 1-simplex is a line, the 2-simplex is a triangle and the 3-simplex is a tetrahedron. Clearly then, the simplicial index k results in a simplex of $k + 1$ vertices (or the connection of vertices results in a simplex with index k). A simplex induces a face (a closed set in $\mathbb{R}^2$) in 2 dimensions and a cell (a closed set in $\mathbb{R}^3$) in 3 dimensions. Also notice then that we have $k - 1$ faces and $k - 2$ cells in completely connected simplices. Homological features can be reliably filtered from properly constructed simplices, because the Nerve Theorem shows that an open covering of a simplicial complex has a 1 to 1 combinatorial relationship with a topological space[1]. Figure 1.2 shows examples of several simplices.

Simplicial complexes can be constructed of data and are a powerful transformation in that they not only tell us about the connectedness of the data, but about spaces in the data and the connectedness of those spaces. Figure 1.3 shows an example of how data might be transformed into a simplicial complex. A simplicial complex is a set of simplices with a few governing rules. Points, lines, faces and cells belong to their higher order connected simplices

in a hierarchical structure. Note that the vertices of the simplices give rise to a face and a cell, but they also give rise to an open space and a structure of connections. The pairwise connection of $k+1$ or more vertices does not induce a face, only complete connectedness of at least $k+1$ vertices (similar to maximal cliques in computation) induces a face. Therefore, one must be careful in one's definition of simplicial complex construction and choose one of several algorithms.

In the interest of a clear explanation of simplicial complex-generation, or the homology filtration process, I will describe a common algorithm used to generate simplicial complexes. The full mathematical explanation is beyond the scope of this brief review. Hopefully, description of the following procedure can help the reader gain an intuition for what is meant by analyzing the homologies of data with simplicial complexes.

The key tool to extend TDA to the generation of statistical summaries is a tool which transforms data to a summary of homological features. Consider a finite set of data $\{x_1, x_2, \dots, x_i\}$ as nodes (points) on a graph with no edges (connections between the points). As discussed, a simplex is the abstraction of a triangle into n -dimensional space. So, to begin with, we have i unconnected 0-dimensional simplices (also commonly called “connected components”). This collection of 0-dimensional simplices is our first simplicial complex.

Now, consider a parameter ϵ which describes a rate of growth. For conciseness, assume that ϵ is isotropic in all dimensions of our data. That is, as we grow ϵ , we grow closed spheres about each data point of radius ϵ in the space of our data. When spheres intersect at a given radius, we record the deaths of all but one of the components as they become a new connected simplex. I will refer to this process as the union-of-spheres method for constructing the simplicial complex.

As the radius grows, we may encounter a loop in the data (shown in the second tile of Figure 1.4). We record this as a new homology feature which is 2-dimensions and gets the hierarchical betti number β_1 . This loop dies when all its member components connect; i.e. it returns to a feature of hierarchy β_0 . In the next section we will look more closely at the definition of intersections and connections as it is critical to the computation of the complex and collection of homological information. This procedure stays the same for higher dimensional homology features. We choose to terminate the algorithm when we have the union of all of the spheres, and we are left with one completely connected component. At this point, the continuation of the algorithm will not produce additional homological information.

1.1.3 Homology and Filtrations

The union-of-spheres algorithm described in the previous section is one method for generating a Čech or Vietoris-Rips (VR) *filtration* of the data. Both filtrations are methods of transforming data to a topological space with some nice properties; particularly stability in the face of lightly perturbations. There are many interesting mapping procedures that have uses in topology, but we will only consider the Čech and VR which are simple to explain and computationally feasible. These filtrations are widely utilized in the TDA

$\beta_0=1$	$\beta_0=1, \beta_1=1$	$\beta_0=1, \beta_1=1$	$\beta_0=1, \beta_1=2$	$\beta_0=1, \beta_1=1, \beta_2=1$	$\beta_0=1, \beta_1=0, \beta_2=1$
					

Figure 1.1: Examples of betti numbers used to characterize topological holes in 0 to 3-dimensional space.



Figure 1.2: Simplices in increasing dimensions from 0-dimensional to 3-dimensional connected components. From left to right: a 0-dimensional point to a 3-dimensional tetrahedron.

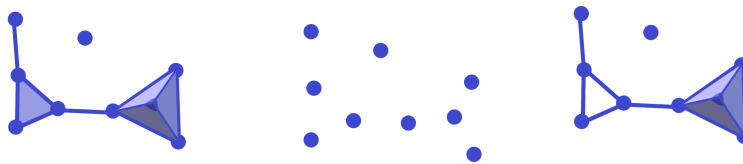


Figure 1.3: Data in the center and their potential simplicial complex-representations on either side. The left complex consisting of 1 0-dimensional connected component, 2 1-dimensional connected components, 1 2-dimensional connected component and 1 3-dimensional connected component. In the right complex, the 2-dimensional face is not yet fully connected, and so represents a 2-dimensional loop.

statistical literature. The Čech and VR filtrations essentially differ on whether they consider the union of $k + 1$ components to be the true geometric union, or based on a pairwise simplification. Compared to the Čech filtration, the VR uses a cutoff parameter t ($d < t$ where d is the maximum of a distance between the spheres) when deciding whether to consider 3 or more spheres fully merged to reduce computational complexity (see Figure 4 for a graphic representation). In the Čech complex, merging of simplices does not occur until spheres fully overlap (not just kissing at a single point). This subtle difference in procedure is essential to whether the outcome are homotopic (Čech) or homologic (VR) invariants and results in a significant difference in computational complexity where the VR is much less complex to compute than the Čech. It should be noted that both of these filtrations are “hard” computational problems and this is crucial to understanding how and why data homology may or may not be studied. This will be discussed further in Section 1.3. If the reader is interested to know more about the details of homology filtrations, a good place to get started is [1].

To illustrate the union of spheres algorithm, I provide a depiction of a simple network in Fig. 1.6. We show the data points in 2-dimensional space, investigating ϵ from 0 to 1.25 and recording 0-dim and 1-dim homological features. These features are interchangeably referred to as connected components and holes respectively. More exactly, 0-dim homological features measure the connectedness of the data and 1-dim homological features capture the number of one dimensional holes in the data. When ϵ is relatively small, five individual components are present. When ϵ increases to 0.71, four ϵ -balls in the upper half area have intersected and four components merge into one connected component. This simultaneously gives rise to a hole. When ϵ is enlarged to 0.85, all components merge into one connected component, though the hole survives. When ϵ is continuously increases to 1, the only 1-dim hole vanishes. No extra information is gained while ϵ is grown to 1.25. The persistence barcode and diagrams correspondent to these data are given in Figure 1.7.

1.1.4 Persistence Homology

Persistence homology is the title given to the study of homological information in data. A definition of persistence that well encapsulates its meaning to persistence homology is “the continuance of an effect after its cause is removed” [15]. Consider a loop represented by a finite sample of data in a 2-dimensional space. When inferring about the underlying shape of the data, one could argue the data causes a loop. Perturb the data and make it very noisy and the homology will remain remarkably intact, i.e. the homology “persists” in the face of perturbations. How persistent or robust a higher dimensional feature is can translate to our confidence in the features existence as measured in homology. From this understanding of the persistence of homology, it is immediately evident why these robust summaries of data have interested statisticians and machine learning experts. Persistence homology is of particular utility for common machine learning (ML) tasks such as classification and clustering.

In TDA, we define persistence in a more rigorous sense to be the *death* – *birth* times of a homology feature. In the union-of-spheres algorithm described previously, these times

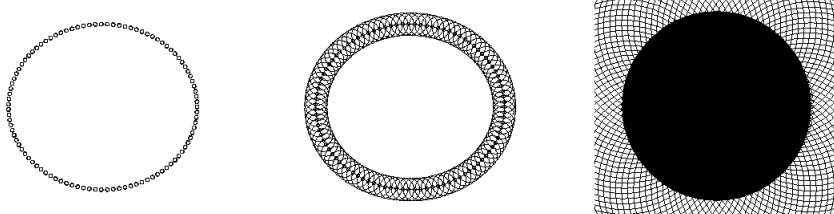


Figure 1.4: The union-of-spheres with radius (ϵ) increasing from left to right. The leftmost figure has 100 0-dimensional connected components with very small spheres about the data, the second figure has grown the spheres sufficiently to merge the original components into 1 1-dimensional connected component and 1 2-dimensionanl hole. The third shows a very large radius at which all of the spheres have merged and we have 1 connected component again.

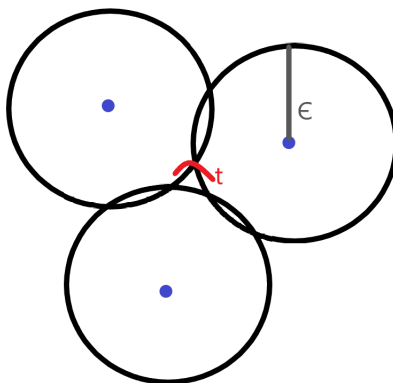


Figure 1.5: Example of a union-of-spheres at which the Čech and VR filtrations would differ. In this example, if t is sufficiently small, than the VR complex would consider these to be a 3-way connected component, where the Čech filtration would leave them as pairwise connected with a hole between them.

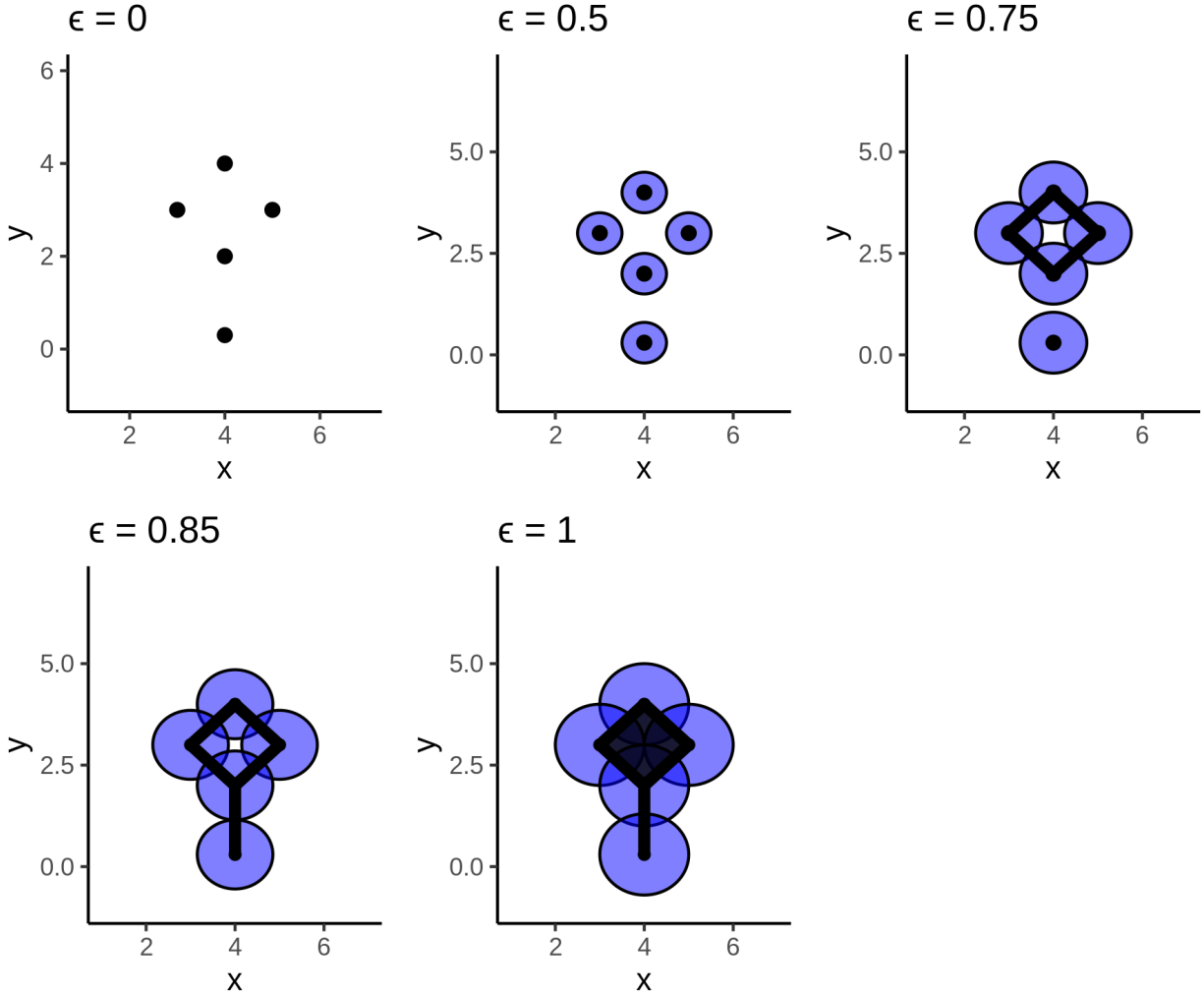


Figure 1.6: Approach depiction of making persistence homology. Beginning from $\epsilon = 0$, there are 5 components. We show the key ϵ values where components are born or die. When $\epsilon = 0.5$, 5 components stay alive. When $\epsilon = 0.71$, 4 components connect into one connected component, forming a 1 hole, thus 3 components die and 1 hole is born. When $\epsilon = 0.85$, 2 components further connect into 1, therefore, 1 more component dies. Finally, when $\epsilon = 1$, the only 1 hole dies and only the 1 connected component survives. No additional homological change while ϵ is grown to 1.25. The persistence homology of this complex is summarized in the persistence barcode and diagrams in Figure 1.7.

would be radii at which simplices are born (connected components are born at $\epsilon = 0$; higher dimensional features are born later) or die (are absorbed into larger connected components). Persistence information can be recorded/represented in many ways including the two plots in Figure 1.7. In these persistence diagrams (PDs), the black dots represent the connected components of 1-dimension and the red triangles represent loops of 2-dimension. The y-axis of the graph on the right of Figure 1.7 has simply had the transformation $persistence = T(birth, death) = death - birth$, so that it's coordinates are $(birth, persistence)$, where as the graph on the left has coordinates $(birth, death)$. It is convention to use this rotation for some later applications. These graphs seem to show a very persistent loop in a noisy set. Despite the loop's persistence, in empirical application these loops are would be extremely difficult to observe in the raw data. This result will be discussed more in the applications section.

I also provide a persistence barcode example in Figure 1.7, since it is has recently become a commonly used filtration in ML applications of TDA. A persistence barcode is simply another parameterization of the betti numbers of a persistence diagram, wherein data are discrete line segments (1-dimensional signals) over the radii at which a found and stratified by the hierarchy of betti numbers (see [16] for an excellent overview of persistence barcodes). Each simplex in the union-of-spheres algorithm has a horizontal line in the barcode, which begins at *birth* and terminates at *death*. The persistence barcode was not frequently used in application, because of its hierarchical, 1-dimensional nature. However, because it can be thought of as a signal, it has become a common feature added to ML signal analyses, as I will discuss in Subsection 1.3.2.

The great advantage of studying homology groups is that they are relatively insensitive to perturbation. Stability of persistent homology has been shown in several ways. A detailed proofs is shown in [17]. They create a framework for “soft” and “hard” stability of persistent homology depending on how one pulls features from the homology filtrations. They explain that persistent homology can be thought of as just the homological filtrations, or their second order linear projections. The authors show that both of these approaches are stable, but with caveats. Both barcode and discretized versions of the analysis are shown to be 1-Lipschitz continuous and continuously differentiable and thus suitable for interesting methods like filtrations based on a sliding window or kernel. This brings us to a discussion of the common statistical applications of TDA.

1.1.5 Statistical Developments

In real-world analyses of data with noise, purely geometric modes of analysis, which seem obvious at first glance, are often fragile or difficult to compute. Persistence homology is a remarkable tool for data analysis, particularly due to its robustness in these real-data settings and for its ability to deliver reliable summaries of geometric insights. So far, I have explained the underpinnings of TDA from the algebraic topology and persistence homology literature. I've shown an example of a common algorithm used to generate filtrations and explained how the generated features are can be recorded in a variety of persistence modules.

I have tried to provide helpful resources along the way for the more in-depth mathematical ideas which could be helpful to review. With this background covered, we are lead to ask how we might implement measures, tests, learning, etc. on persistence homology. In this section I present a few key statistical methods for comparing persistence homologies.

As an example, let's consider the case of comparing persistence diagrams. When comparing two persistence diagrams, the most common traditional approaches is the calculation of a Bottleneck or Wasserstein distance [8]. While the Bottleneck is often used as an example in the mathematical literature, it does not appear to perform well in practice, so it is usually the Wasserstein distance used in papers [18, 19, 20]. Recently, a novel formulation called the d_c^p distance has been shown to be a situational improvement over the Wasserstein and Bottleneck. Bottleneck, Wasserstein and d_c^p distances are shown below. In addition to the persistence diagram distances, we should also examine the landscape distance, since these have also become very commonly used metric.

Persistence diagram distances are meant to summarize the pairwise dissimilarity in a set of persistence diagrams, which are themselves random sets. Since these distances are pairwise, the outcome is a matrix of dimension $n \times n$, with 0s in the diagonal and symmetric upper and lower triangles. In both the bottleneck and Wasserstein distances, the diagonal of the diagram, (*birth* = *death*), of each diagram is given infinite points of infinite density for the purpose of measuring the distance between sets of uneven cardinality. Points close to the *birth-death* line are matched to these artificial points. The bottleneck distance takes the bijection of all the points between the two sets, minimizing the overall distance, and simply takes the maximal distance between two points as the summary distance. The Wasserstein distance by contrast allows the user to choose a parameter $p \geq 1$, which is frequently set to 1 or 2 and corresponds to the p^{th} moment of the Wasserstein distance [18]. The effect is to proportion the distance between sets on the distance between all matchings in the minimal distance bijection.

Definition 1.1 (Wasserstein Distance).

$$W_p(D_x, D_y) = \left(\inf_{\eta: D_x \rightarrow D_y} \sum_{x \in D_x} \|x - \eta(x)\|_\infty^p \right)^{1/p}$$

Where $\|\cdot\|_\infty$ is the maximum norm and where $\eta : D_1 \rightarrow D_2$ is a bijection on the two persistence diagrams.

Definition 1.2 (Bottleneck Distance).

$$W_\infty(D_x, D_y) = \inf_{\eta: D_x \rightarrow D_y} \sup_{x \in D_x} \|x - \eta(x)\|_\infty$$

where the notation is the same as in Definition 1.1. The bottleneck distance is relatively insensitive to small changes in η , since it uses only the largest-distanced matching in η to summarize the distance between sets.

The d_c^p distance adds a novel method for controlling sensitivity to data near the diagonal, often called “topological noise”, and ignored by the Wasserstein and bottleneck distances. The d_c^p distance adds the constant c which is a regularization penalty applied to unmatched (off-cardinality and close to the diagonal) points. Rather than matching points on the set of higher cardinality to the diagonal as in the Wasserstein and Bottleneck distances, these points are assigned the penalty c . This c can then be optimized with a method such as minimizing the training error over a k -folds validation experiment. This formulation has been shown to have desirable stability attributes[21]. I am providing the formulation that the authors use, which replaces the function η with Π which is instead a finite set of permutations of bijections. So, rather the infimum over η , they find the minimum in Pi .

Definition 1.3 (d_c^p Distance). *Let D_x and D_y be two persistence diagrams with cardinalities n and m respectively such that $n \leq m$ and denote $D_x = \{x_1, \dots, x_n\}$, $D_y = \{y_1, \dots, y_m\}$. Let $c > 0$ and $1 \leq p < \infty$ be fixed parameters. The d_c^p distance between two persistence diagrams D_x and D_y is*

$$d_c^p(D_x, D_y) = \left(\frac{1}{m} \left(\min_{\pi \in \Pi_m} \sum_{l=1}^n \min(c, \|x_l - y_{\pi(l)}\|_\infty)^p + c^p |m - n| \right) \right)^{\frac{1}{p}}$$

where Π_m is the set of permutations of $(1, \dots, m)$. If $m < n$, define $d_c^p(D_x, D_y) := d_c^p(D_y, D_x)$.

There are several methods of comparing persistence homology outside of the persistence diagram comparisons described so far in this section. One of the common formulations, which has some convenient statistical results is the landscape distance [22]. In this method, one converts $(birth, death)$ coordinates to a system of orthogonal lambda functions (with coordinates (m, h) mapped as in Definition 1.4), which is called the persistence landscape. It is shown that this new space is a Hilbert space and that estimates of the mean in this space abide the central limit theorem. It is proven to be stable and one can reference the pdf, how to calculate Z-scores, confidence intervals, etc. for this parameterization. It is also shown that the distance between landscape diagrams (Definition 1.4) is a stable lower bound on the Wasserstein bottleneck distance. All of these details make the landscape distance a simple and clean formulation for statistical applications.

Definition 1.4 (Landscape Distance).

$$m = \frac{death + birth}{2}, \quad h = \frac{death - birth}{2}$$

After persistence points have been mapped to (m, h) there is a rescaled rank function $\lambda : \mathbb{R}^2 \rightarrow \mathbb{R}$

$$\lambda(m, h) = \begin{cases} \beta^{t-m, t+m} & \text{if } h \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The persistence landscape is then given by $\lambda : \mathbb{N} \times \mathbb{R} \rightarrow \overline{\mathbb{R}}$

$$\lambda(k, t) = \sup(m \geq 0 | \beta^{t-m, m+t} \geq k)$$

The functions of λ can then be grouped into sequences $\lambda_k : \mathbb{R} \rightarrow \overline{\mathbb{R}}$. The set of all λ_k is then the persistence landscape.

An example of the transformation from point cloud, to persistence diagram, to persistence landscape is given in Figure 1.8.

Yet another important method to analyze persistence information is a transformation from the persistence diagram to a persistence image [23]. To transform a diagram to an image, one first takes a linear transformation T of a persistence diagram B where the diagram is tilted along the *birth-death* line by $T(x, y) = (x, y - x)$ (as shown in Figure 1.7). Second, one picks a weighting function which assigns a mean and variance to a Gaussian, which is then fit about each point in the $T(B)$. The summation of these Gaussians is discretized into a pixelated image, which is the PI. See Adams Et. al [24] for a result on stability and multiple weighting functions. I provide a visual example of a PI and a binned PI in Figure 1.11. The PI comes from a continuous, additive Gaussian function and the choice of grid size in the discretization step depends on the data and mode of analysis. I provide an example with a PI in high resolution, and then a binning to more reasonable size vector for construction of an NN discriminator.

Up to this time, many of TDA applications either stop at simple visual analyses (for instance [25, 26]) or use ML methods on top of the persistent homology or persistent landscape data. [27] show a method for image classification using the Wasserstein distance on a kernel filtration, and plug this into an SVM. [28] show a method for converting barcode data into an input to a CNN and show how this can be thought of as a generalized form of the residual learning system. Visual and ML methods are intriguing, but some researchers are building more robust statistical approaches as well.

While some researchers have pursued transformations like the landscape distance to form parameter estimates, others have pursued a Bayesian framework working on the persistence diagram itself [18, 20, 29]. These studies have shown promising results on synthetic data even with bad/uninformative priors. They have also shown strong empirical results. In these papers the researchers can impose a minimally constrained model that assumes only basic continuity of signals (not smooth or time evolved). There will be further discussion of some of the applications of the Bayesian framework to TDA in the applications section.

1.1.6 Applications in Signals

Statistical approaches to signal processing are some of the most heavily studied sub-fields of applied statistics. Signals are simply functions describing processes. In time-series these are often one-dimensional signals, dependent on time, i.e. $f(t) = z$. Signal processing methods are generally aimed at analyzing the evolving structure of such a signal over the course of the

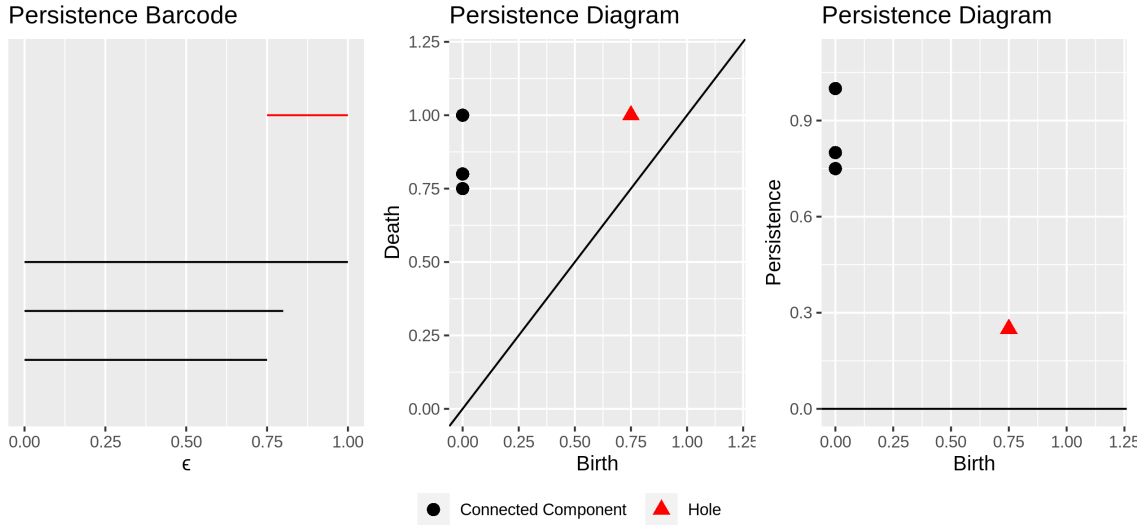


Figure 1.7: A persistence barcode and two forms of the persistence diagram. These data belong to the simplices in the union-of-spheres example in Figure 1.5. Notice that the right is simply the rotation of the line $birth = death$ on the left to the x -axis on the right.

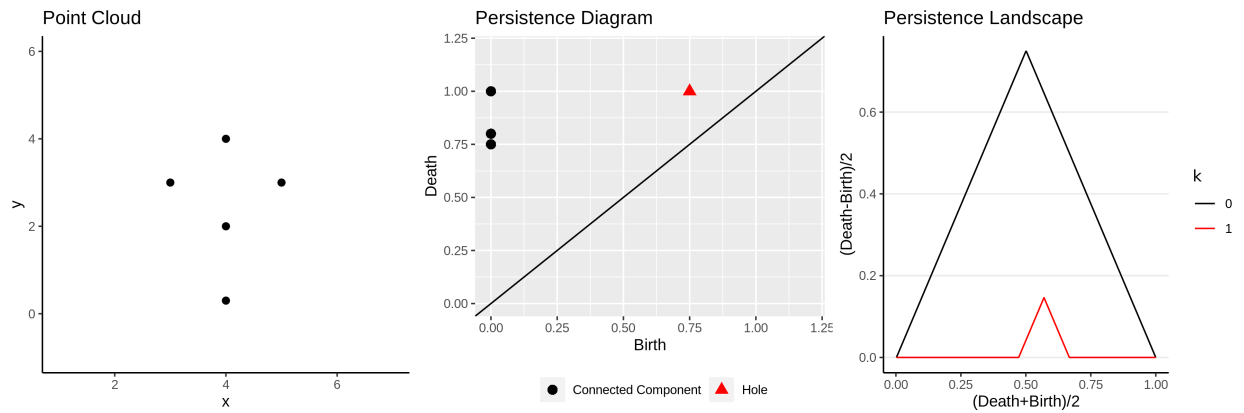


Figure 1.8: An example of a realization of a persistence landscape from a persistence diagram. Note that λ_0 appears dominated by the VR computation's chosen limit of $\epsilon \leq 3$ and do to the simplicity of the example.

dependent variable. Signals may represent a continuous process, but in analysis signal data are always represented by a finite, discrete sampling. Images are also often treated as signals in statistical analysis. Images can be thought of similarly to their 1-dimensional counterparts, as a discrete sampling of a 2-dimensional signal, $f(x, y) = z$. This perspective reflects that images also represent smoothly evolving, continuous processes. Major applications of statistical signal processing include image, video, financial and biological/physiological signals.

To analyze 1-dimensional signals with TDA, we must map signals to a topological space. One example of a signal embedding to a topological space is illustrated in [1]. In their mapping, they grow a parameter a , which records births when it intersects local minima and records deaths when those minima are closed. Minima and maxima are nested in this embedding, such that the longest persistence feature is the largest shift in amplitude, and all other filtrations are nested within. This is because a is raised across the entire function, and so the entire function is considered simultaneously. See Figure 1.9 for an illustration of this embedding, from the 1-dimensional function, to a persistence barcode, to a persistence diagram. This is a simple example of how to extract topological features from a 1-dimensional signal. The authors also show that this mapping works for 2 and 3-dimensional loops and tori.

Another popular method for extracting topological features from signal data is the sliding window or kernel filter. As discussed above, homology filtrations have been shown to have a stability property which makes them suitable not just for the classical level-set filtration, but can also be generated from a moving window[11, 30, 31, 27]. In these papers, 1-dimensional time series are converted into point clouds through a sliding window operation, where the functions value is taken at regular intervals of time t and $t + \tau$. The union of closed spheres-approach is then used to indirectly generate the homology features of the signal from this new point cloud.

Statistical TDA is a relatively new field and signal processing is one of its newest and most exciting applications. As discussed, TDA is a mathematical toolbox used to observe and summarize the shape-like structures of data. TDA for signals is much like the methods described in the previous process, usually capitalizing on the continuously evolving structure of signals, but incorporates the insights of computational and algebraic topology. Topological invariants have been shown in many studies to be robust to high levels of noise in signals. They have also been shown to be powerful descriptors in mixed chaotic and unchaotic signals, and classifiers between sampled signals. TDA signal methods have been developed with non-parametric and now Bayesian implementations, mostly around the framework of persistent homology.

1.1.7 Conclusion

This review of the foundational mathematics and concepts for TDA was provided in order to demonstrate to the reader the breadth of field. Hopefully, this summary of the available

tools of TDA shows how one might approach applications to datasets with inherent spatial information or signals, or even n -dimensional, unstructured data. The following sections of the literature review will discuss several problems for which these tools can provide path forward.

1.2 Cytoskeletal Actin Network Classification

In the prior section I covered the fundamentals of TDA including a demonstration of 2 groups of methods for producing the topological filtrations: the VR complex for point clouds and the level-set embedding for signals. In this section I will demonstrate the development of a ML application for the study of a physical, intra-cellular networks called the actin cytoskeleton. I provide an example image, from confocal microscopy, of a cell with focus on the actin network in Figure 1.10. This image is one data-point and can be thought of as a 2-dimensional, discretized signal. Alternatively, we can threshold the image to consider pixels with values above a threshold to be the points of a point cloud. I will attempt to motivate a method using the perspective of the data as a point cloud and computing the VR complex.

The actin cytoskeleton is a complex network of proteins that is present in all eukaryotic cells. In addition to its function as cellular scaffolding, the actin cytoskeleton enables several basic cellular functions including the control of cellular shape and direction of movement [32]. These basic functions are critical to many higher order physiological processes such as cell division, expansion, mobility and motility[33].

In the actin cytoskeleton, actin filament organization is thought to be governed partially by the interaction of filaments and partially by myosin motor proteins. Actin filaments are polar structures, polymerized by globular actin proteins. Many actin-binding proteins have potential to bind to actin filaments at various sites along the filament. These binding proteins allow actin filaments to assemble and disassemble spatiotemporally. The binding proteins give rise to a dense cross-linking where filaments develop into networks at with many filaments and binding sites. To understand certain behaviors of cells, it is of tremendous importance to understand the processes that govern actin filament network organization. A key driver of these processes may be the relationship between actin-binding proteins, individual filaments and emergent networks.

One method to study the actin filament network generating process is to simulate the dynamic structure of filament networks, combining theory from theoretical physical with experimental stochastic simulation. With this methodology one can control the known factors, which will affect the structure of networks. Varying these initial conditions enables researchers to compare the conditional difference in outcomes of the simulated networks. This experimental strategy can provide an opportunity to independently examine the role each factor plays in the process. These factors could include the cross-linker density(number of cross-linkers per certain area), cross-linker stiffness, maximum angle that can exist between

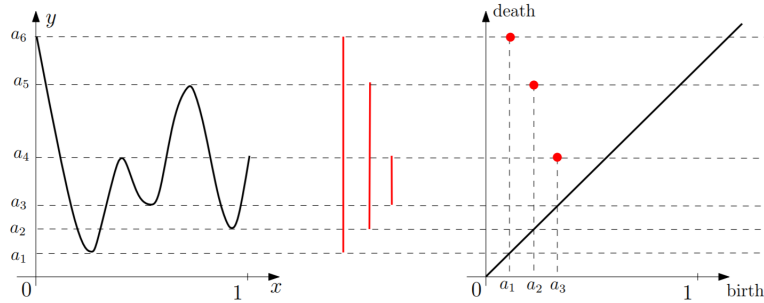


Figure 1.9: An example of a level-set embedding, from a 1-dimensional pseudocontinuous function, to a persistence barcode and to a persistence diagram. Note that in the persistence diagram, the diagonal ($birth = death$), indicates the addition of infinite points of infinite density to the diagonal for purposes of measuring the distance of sets with uneven cardinalities. This figure was taken from [1] pg. 18.



Figure 1.10: Example of a confocal microscopic image of a potato root hair cell. This cell is an example of a cell which has had the production of a particular myosin protein knocked-out through mutation. The image is very long and narrow, since the cell is a hair. The images are in very high resolution, but include a great deal of noise, as well as real features which pass in and out of focus since this is a single slice of a 3-dimensional object. The image was graciously provided by Jinmo Gu.

two filament segments to be crosslinked, and so on [33, 34]. In this work, I propose a novel machine learning approach to classify filament networks generated with varied cross-linker density.

1.3 Topological Study of Neural Networks

In the first section I covered key background to TDA and in the second section I motivated a fairly typical, though augmented TDA analysis. In this section I will discuss a novel way of understanding the inner-workings of neural networks (NNs) with TDA. Applying TDA to study NN is a new method of answering questions that began to be asked in the late 1980's. These questions have the general form: Can we derive theoretical and/or empirical shortcuts to the design or optimization of NNs by studying the topology of generalized networks? One could argue that this fits into the literature on network meta-analysis or common meta-parameter optimization. Viewing network topology through the lens of algebraic topology, it does seem possible to assimilate all of these ideas under a common umbrella, since algebraic topology allows us to optimize with topological invariants, similar to how we might optimize with model meta-parameters. This section will discuss some key background, concepts and methods to use TDA to empirically study NN topology.

In the following subsections, I will show how topology has been studied in the context of multi-layer NNs. I will show a new body of literature emerging on the empirical study of deep networks with TDA. I will also present several intriguing, though difficult to categorize, state of the art methods that fuse TDA and NNs in a meaningful manner (I will also try to qualify my idea of what is meaningful).

1.3.1 Neural Networks

It has been shown that NNs can approximate arbitrary, continuous functions on $\mathbb{R}^n$ [35]. Since at least the early aughts, NN's have been in favor for many pattern recognition tasks [36]. However, 20 years ago, computational power put tremendous pressure on researchers to find efficient NN formulations, even for small (by today's standard) networks.

Already in the early 90's, a procedure was demonstrated for constructing a topologically optimal three-layer NN. This work applies several foundational works from the mathematical literature to show that one can estimate the size of the latent dimensions in the data space to and limit an NN to the smallest number of nodes and edges necessary to model approximate this space [37]. This is important since one would like to find a network with minimal size and complexity to reduce symmetries in the solution space and find more efficient approximations. While this procedure makes sense for small networks, it is computationally infeasible for large networks and big data. So, while NNs are very powerful tools, finding computationally efficient typologies for efficient approximations is a difficult problem, particularly in modern applications. Recently, a few studies have made use of persistent homology to study topological optimization of networks [38, 39].

For simplicity, I will categorize advances in the use of TDA for NN research in two broad categories. The first category (with significantly more work at present) are papers which use persistence homology features as inputs to an NN. The second, are papers which develop generalization, optimization or extensions of NNs through empirically motivated work from TDA.

1.3.2 Persistence Features as Input to NNs

By far, the most commonly published method of incorporating persistence homology is the incorporation of some vectorized form of a topological filtration, usually in the form of a barcode, landscape or a persistence image (closely related to the persistence diagram). I show several examples of common vectorizations of persistence schemes in Figure 1.11.

If the desire is to feed a persistence barcode, landscape or diagram to an NN, some discretization of the signal must be performed (recall that persistence filtrations produce a continuous, hierarchical signal which can also be viewed as a point cloud). In order to do this, one must choose a fixed number of bins for the analysis. In the case of barcodes and landscapes, this would be a single choice of width. In the case of persistence landscapes, it would be a two dimensional binning and could therefore have independent choices for width and height of the bins.

In Figure 1.11 I demonstrate several methods for vectorizing persistence representations. In the cases of the persistence barcode and landscape, I normalize the x-axis to the interval $[0, 1]$. In the case of the persistence image, I normalize both axes. I then divide each domain into 8 equal partitions of width $\xi = .125$. In the persistence image, I apply this in both dimensions, giving a square lattice. Since persistence barcode data may have uneven cardinalities, I chose to sum data by dimension to get two signals for connected components (black) and holes (red), as demonstrated in the bar graph. The persistence landscape is a continuous signal, and so one could choose any function $\mathbb{R}^n \rightarrow \mathbb{R}$ such as taking the minimum, maximum, etc. For this demonstration, I chose to take the mean in each partition. The persistence diagram is vectorized by simply taking the mean and standard deviation of the persistence of the data. Instead, one could use alternative functions, and/or birth and death rather than persistence. These are all choices that should be made carefully by the researcher based on their data and objective.

Once one has a vectorized representation of the persistence information, it is trivial to use this as input to any number of algorithms. Support vector machines (SVMs), NNs, clustering algorithms (such as k-means), and others have all been shown effective in application. In the case of NNs, the vectorized persistence information is input to the 0^{th} -layer. If the data are already a single column, such as in the vectorized landscape, this can be input directly. If the vectorized data are greater than 1-dimensional, such as the persistence image, then they must first be unrolled into a single, “flat”, column. Alternatively, one can use the signal attribute of some vectorization methods to input the data first to convolutional layers and treat the persistence information as an n -dimensional signal. The rationale for learning a convolutional

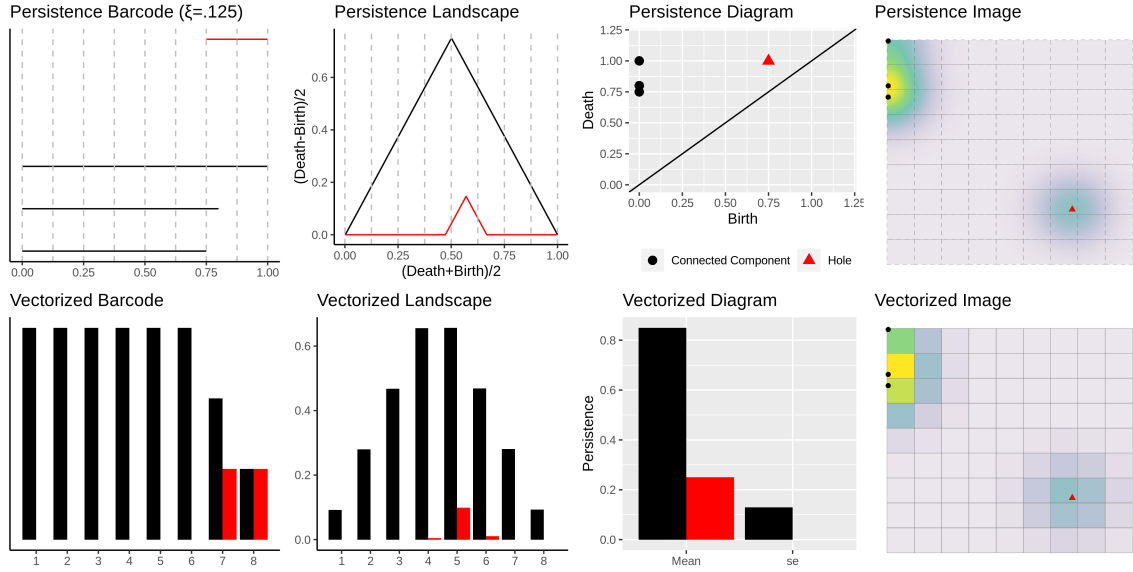


Figure 1.11: Demonstration of 1 and 2 dimensional binning methods for the vectorization of persistence barcodes, landscapes, diagrams, and images. There are many methods of vectorization, retaining various aspects of the prior persistence information, so these are merely meant to demonstrate several commonly used methods. Notably, the method I chose for the PD, does not produce a standard error for the holes. This might naturally motivate a researcher to choose another method when they have a very small number of important, higher dimensional features in much of their data. Also note that the vectorizations we chose for the barcode and landscape are discretized 1-dimensional signals and the vectorized persistence image is a 2-dimensional signal. This means that all of these could be treated as “flat” vectors, or operated on as signals, such as done in a CNN.

structure, separate from the fully connected network, for persistence information is not clear, but it has been applied with success none-the-less.

1.3.3 TDA in NNs

A common strategy is to incorporate persistence barcode data as direct inputs to an NN, as in [40, 28]. These papers vary widely in application, but in all cases a CNN is used primarily to optimize over persistence homologies. Persistence barcodes allowed TopologyNet “outperforms other methods in the predictions of protein-ligand binding affinities and mutation induced protein stability changes” [40]. A drawback to these studies, from the perspective of a methodologist, is that they make a number of choices in the pre and post processing, bootstrapping, etc. on top the fact that NN learning is an inherently difficult classifier to understand with a variety of choices in parameterization. This combinatorial array of choices and the complexity if the classifier can make it difficult to generalize results. These methods (with the exception of the vectorized landscape) also rely on a vectorization of persistence information that is known to be unstable, despite its empirically demonstrated success. The vectorization of the barcode and persistence diagram could be problematic [21], while the vectorized landscape and persistence image have been shown to be stable [41, 17].

1.3.4 TDA in CNNs

Theoretically, a discretized signal of arbitrary dimension can be used as input to a CNN. CNNs balance the learning of global and local features, by enforcing a graph structure on the data which is based in the signal’s space. Since persistence barcodes and landscapes can be viewed as 1-dimensional signals, they may be go through these sparsified convolutions prior to being passed to a dense network. Several papers have used this strategy to great effect [42, 43]. However, they do not make a comparison of inputting the persistence features to the dense NN layers, and skipping the convolution, so it is difficult to surmise the relative advantage that the convolutional layer provides. It is also unclear precisely what statistical, mathematical justifications would drive this choice.

The paradox of deeper networks and persistent homology is that large NNs often necessitate large training datasets. Persistence homology is of course a difficult computation and so the compute time when including the computation of the simplicial complexes and filtrations is very large relative to the time needed to train even very large NNs. Therefore, one must have an objective that is greatly advanced through topological information for it to be worthwhile to include persistence homology into a deep learning problem.

One approach that has been developed to incorporate persistent homology to image classification through CNNs, uses the homology filtrations of “exemplar” image patches as an imputation. In this method, one first trains a typical CNN. Second, one selects the most important patches in the final convolutional layer across the classes. Third, one computes the persistence homologies of these patches. Fourth, one uses a second CNN, but replaces the dense network with a random forest model with the raw image features as well as the

persistence homologies for the patches based on KL-divergence to the “exemplar” patches. This approach was found to be highly efficacious in clock time and accuracy [44]. Creative relaxations such as this will be necessary to include topological information into machine learning for big data while keeping computational overhead more reasonable.

1.3.5 Topologically Inspired CNNs

The study of meta-structures on NNs has become an important sub-field in ML research. Since NNs are graphs with non-trivial topology and so one can use persistence homology to measure their shapes. In this framework, one can construct a finite number of candidate models and compare their performance with their homology signatures, via persistence diagram distances such as the Wasserstein distance or a metric on the persistence landscape. Alternatively, one could optimize for a known, optimal homology. It has been empirically shown that there is a close relationship between homology and network capacity [45]. This work seems to suggest that future work should be done to implement useful, novel topological structures in NNs. My own work, in Chapter 3 extends these ideas to a novel generalization of the CNN based on these empirical findings.

1.4 Conclusions

In this literature review I provided an abbreviated background on the foundations of TDA and provided a review of state-of-the-art applications. First, a guide to the foundations of TDA should be included in my dissertation, because it is a newer field and a short guide should aid the reader’s imagination in the following review and research chapters. Second, consolidation of the literature on novel applications of TDA is critical at this juncture as the popularity of these applications has exploded in recent years and it is now being used in a broad array of fields. Finally, I give the relevant background literature for the next two chapters. I have shown that TDA is a young, but proliferating body of work with many promising theoretical and applied contributions. The following two chapters will build on this background and review by developing several novel applications.

Chapter 2

Persistence Homology Classification of Cytoskeletal Actin Filament Networks

This second chapter focuses on an application of TDA to microbiological data. The work herein is novel both in its use of TDA to classify cytoskeletal networks and in its use of resampling of data and vectorization of persistence information to form a classifier. I tackle a preliminary classification problem on simulated cytoskeletons, testing many conventional and TDA-based methods. I use the knowledge that actin filaments represents continuous paths which can be viewed as a graph and recover the topology of these networks from data via persistence homology. I introduce a subsampling approach, which I show to improve classification accuracy. This approach generates several point clouds per network data-point, which are each transformed to a persistence diagram. I compare several classifiers and find that a machine learned approach on a vectorized form of these resampled diagrams gives very high accuracy on synthetic data. I validate this method on real, confocal microscopy data. Finally, I demonstrate an alternative clustering approach using k-means, which allows us to do an in depth examination of the cluster assignments and misclassifications in the original data space.

2.1 Background

The actin cytoskeleton is a complex network of proteins that is present in all eukaryotic cells. In addition to its function as cellular scaffolding, the actin cytoskeleton enables several basic cellular functions including the control of cellular shape and direction of movement [46]. These basic functions are critical to many higher order physiological processes such as cell division, expansion, mobility and motility[9].

In the actin cytoskeleton, actin filament organization is thought to be governed partially by the interaction of filaments and partially by myosin motor proteins. Actin filaments are polar structures, polymerized by globular actin proteins. Many actin-binding proteins

have potential to bind to actin filaments at various sites along the filament. These binding proteins allow actin filaments to assemble and disassemble spatiotemporally. The binding proteins give rise to a dense cross-linking where filaments develop into networks at with many filaments and binding sites. To understand certain behaviors of cells, it is of tremendous importance to understand the processes that govern actin filament network organization. A key driver of these processes may be the relationship between actin-binding proteins, individual filaments and emergent networks.

The goal of this work is to develop a framework for the classification of actin networks. The images of actin networks that we are interested in are captured via confocal microscopy. These images are very high resolution (typically around 2500x500 pixels with $0.04 \mu\text{m}^2$ pixels), but still suffer from several types of noise such as: filaments moving through the focal plain, rounding of the cell at the edges, neighboring cells polluting the image, changes in microscopic conditions/settings, and many more. Therefor, confocal microsocopy has the advantage of providing a lot of high quality data at the expense of also including many noisy data. This means that in order to automatically study these images, without interjections by researchers (eliminating potential for an introduction of unforeseen bias), a tool is called for which is highly robust to these types of noise.

In the fast developing field of machine learning, topological data analysis (TDA) has become increasingly popular as a tool for noisy network and signal classification. To date, researchers have used TDA to solve many real-world problems including signal identification [47], materials classification [48, 20], shape recognition [49, 50], histologic image analysis [51, 52, 53], ecology of human mobility [54, 55], and cosmology [56, 57]. A review of TDA and its applications is provided in [58]. A sub-method of TDA, persistence homology, is a popular method used to measure differences in topological features, due to its robustness in the face of perturbation of data. Persistence homology records when homological features (connections and voids) appear and vanish within data. These patterns vary between data. All of the appearances and disappearances of homological features are summarized in persistence barcodes and/or diagrams. In this work, geometric features of filament networks are encoded into persistence diagrams and a method of classification on the vectorization of the persistence space is demonstrated (a persistence space is not itself a vector space, so it has no mean for instance). I compare this approach to a traditional, distance-based classifications which attempts to summarize the similarities of the actin network topologies in the persistence space.

Our investigation of classification methods is aided by a high quality dataset of simulated actin networks, which is used to benchmark each method. The outputs of simulations, which combine theoretical physical properties with experimental stochastic simulation in order to emualte actin network dyanmics, were provided in order to compare methods. With this methodology one can control the known factors, which will affect the structure of networks. Varying these initial conditions enables researchers to compare the conditional difference in outcomes of the simulated networks. This experimental strategy can provide an opportunity to independently examine the role each factor plays in the process. These factors

could include the cross-linker density (number of cross-linkers per certain area), cross-linker stiffness, maximum angle that can exist between two filament segments to be crosslinked, and so on [9, 59]. This control mechanism also allows for the testing of many methods, on a highly controlled and clean dataset, in order to test sensitivity and to compare between methods.

After testing several methods of actin network classification on the simulated data, the top performing method is chosen and adapted to the microscopy images. A classification is performed attempting to separate myosin-mutant and wildtype *Arabidopsis* root cells.

The structure of this work is as follows: Section 2 describes the data and introduces the relevant background of persistence homology. Section 3 demonstrates two algorithms for classifying filament networks. Section 4 exhibits the numerical results. Section 5 gives conclusions and a discussion of future directions.

2.2 Persistence Homology and Filament Networks

To quantify the differences in filament networks, a transformation is required of our data which reveals its hidden geometric features. This transform is performed by the construction of simplicial complexes in a manner typical of persistence homology. The 2-dimensional coordinates of sampled points along the filaments are taken as the initial nodes. As discussed in Chapter 1, simplicial complexes provide a bridge between the data space and a topological space in which computation of distances between sets of data points can be realized. A simplicial complex is a finite collection of simplices of different dimensions such that faces of simplices are also simplices, and intersections of the simplices are either empty or a face of both [60]. In particular, higher dimensional simplices are constructed from lower dimensional simplices. Vertices are 0-dim simplices. A 1-dim simplex is called an edge and is created by its two vertices as faces (note that a higher dimensional edge is constructed from lower dimensional points). A 2-dim simplex or a triangle has three edges as faces. Further more, a 3-dim simplex or a tetrahedron has four triangles as faces, another nesting of several lower dimensional features to build one of higher dimension.

2.2.1 Data

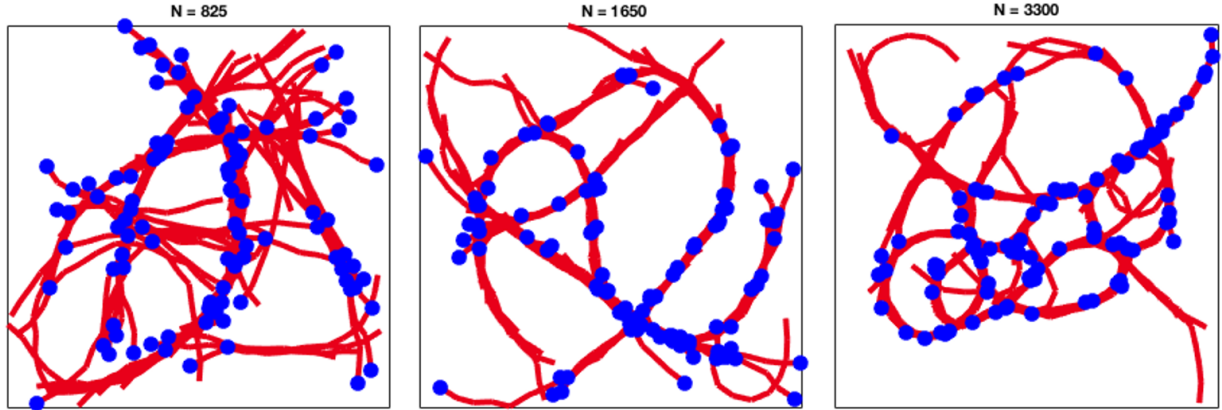
Microscopy data: Microscopy data were provided in the form of one grayscale image per cell. The actin filaments fluoresce and so the intensity of each pixel of an image can be thought of as indicating the likely presence of a filament in that region of the image. In order to study the homology of an actin network, we must perform a topological transformation on the data. Since the images contain hundreds of thousands of pixels, we chose to sample from the images a set of points, where the probability of choosing a pixel is proportionate to the pixel’s intensity. We can make a choice of a number of points that we think is likely to sufficiently summarize a network. We can now perform our topological transformation on these new point clouds.

Simulated data: Our synthetic data come from simulations with varied numbers of crosslinking proteins. As discussed, actin filaments are thought to be organized by cross-linking on actin-binding proteins. Filaments and inter-filament structure can then be simulated by a physical model [9, 59]. The change of initial conditions in a eukaryotic cell will cause variation in later measurement of filament networks. Our network data is simulated by three different cross-linker densities. Higher cross-linker density means more opportunities for filaments to be cross-linked, i.e. the binding and unbinding processes can be more active. As shown in Fig. 2.1(a), three kinds of filaments networks were simulated with different numbers of cross-linkers: 825, 1650 and 3300. All simulated cells were bound by a $20\text{ }\mu\text{m} \times 20\text{ }\mu\text{m}$ square. Therefore, the cross-linker density of each network is 2.06, 4.13 and 8.25 per μm^2 , respectively. In each network, there are a total of 100 filaments with average length $10\text{ }\mu\text{m}$, where filaments are model as polar worm-like chain in red and blue dots represent barbed ends of these filaments. We also record the locations of the actin beads that make up the filaments, which are shown as small black circles in Fig. 2.1(b). Each actin bead is of radius $0.5\text{ }\mu\text{m}$. Our interest is in developing an automated method to accurately classify cross-linker density of filament networks from simulated network data. The actin beads of the simulated networks will act as our point clouds in the topological transformations of these synthetic data.

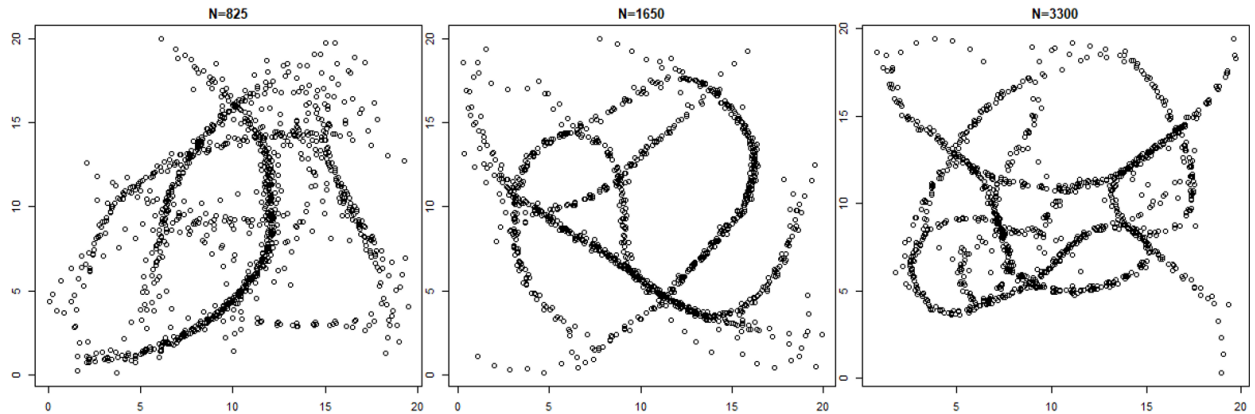
2.2.2 Persistence Homology

In order to build simplicial complexes, we adopt the procedure of forming Vietoris-Rips complexes on each dataset (actin network) by introducing a sequence of ϵ -balls with increasing radius ϵ centered at each data point (a sampled pixel for image data or an actin bead in the synthetic data). Simplicial complexes are constructed based on intersections of these ϵ -balls and each value of ϵ corresponds to an unordered group of homological features, which is called a homology group. Considering values of ϵ as a timeline, we only record when a homological feature appears and disappears. These indexes are called the birth times and death times of a particular homological feature. Moreover, the lifespan (death minus birth) of a homological feature is referred to as the feature’s persistence. A set of homological features gives rise to a set of persistence measures. At the end of this procedure, when radius ϵ is sufficient larger so that the homology group remains unchanged even by further increase the radius, information of a filament network’s persistence homology (the set of persistent homology measurements) is summarized in a persistence diagram.

Figure 2.3 depicts the process of discovering and summarizing the persistence homology of a simplified filament network. When $\epsilon = 0$, the sampled points of the filament network are each their own connected component. As the ϵ -spheres grow, connected components begin to merge. In the first column of the figure, when $\epsilon = 0.3$, several of the sampled points have already connected. We denote the points which have died (having merged into a larger connected component), by ending their bar in the persistence barcode below, and plotting a point at 0 on the x-axis and at their precise time of death on the y-axis in the persistence diagram below that. When $\epsilon \approx 1$, two holes form. The holes are evident in the top row,



(a)



(b)

Figure 2.1: Filament networks. Panel (a) shows three filament networks generated by 825, 1650 and 3300 cross-linkers, respectively, in a $20\text{ }\mu\text{m} \times 20\text{ }\mu\text{m}$ area. Each network contains 100 filaments which are represented as red lines. The blue dots are the barbed ends of these filaments. Panel (b) shows the locations of the actin beads that make up the filaments exhibited in Panel (a).

second column, of the figure, where one can see one larger and one smaller hole. We begin plotting bars for the holes in the corresponding persistence barcode below. Note that there are not yet records of the holes in the persistence diagram, because we require the death time of the holes in order to plot them in the 2-dimensional diagram.

In the third column of Figure 2.3, when $\epsilon = 1.1$, the smaller hole has closed, and become part of a larger connected component. Now a single point is plotted in the persistence diagram below and the bar is terminated.

As the algorithm progresses, the larger hole eventually dies and its corresponding bar in the persistence barcode is terminated. A point is added at the corresponding birth and death in the persistence diagram (a second red triangle appears in the final column). The algorithm could continue to $\epsilon = \inf$, but it is evident in this example that no more homologic information will be discovered as the spheres continue to grow. We choose to arbitrarily terminate the algorithm at $\epsilon = 2$, terminate the final bar in the final barcode, and plot at $(0,2)$ in the persistence diagram to denote final death time.

Overall, persistence homology indirectly summarizes the hidden shape of the data and transcribes this shape to the persistence diagram. With the persistent homology of each point cloud, a classifier can be generated either from the distance [18] between persistence diagrams or by alternative vectorizations of the diagrams [41, 61, 21].

2.3 Filament network classifier

Once we have persistence diagrams corresponding to the point clouds of actin networks, we are ready to classify these networks. In this work, we propose two methodologies as candidates for a filament network classifier, one is distance-based method while the other one is based on a vectorization.

2.3.1 Distance-based network classifier

Given any two persistence diagrams, we need a way to quantify the difference between them in the space of persistence diagrams. In TDA, two methods are commonly used to calculate the distance between persistence diagrams: the Bottleneck and Wasserstein distances [41, 61, 60, 62, 58]. These distances calculate the optimal (minimal) cost in matching the points between two persistence diagrams. They assume infinitely many points of infinite multiplicity on the diagonal (where birth equals death), so that off-diagonal points are matched to a point in this artificial set. In addition to the Wasserstein and Bottleneck distances, in this work we adopt a new distance, called d_p^c distance, which is proposed in [18] and has been proved to be stable in [21]. The cardinality of a persistence diagram may carry important information in applications, especially for those homological features which die very quickly and may be considered as insignificant in Wasserstein distance. However, the d_p^c distance accounts uneven cardinalities between persistence diagrams by assigning a regularization term with the parameter c rather than connecting extra points to the diagonal as is done

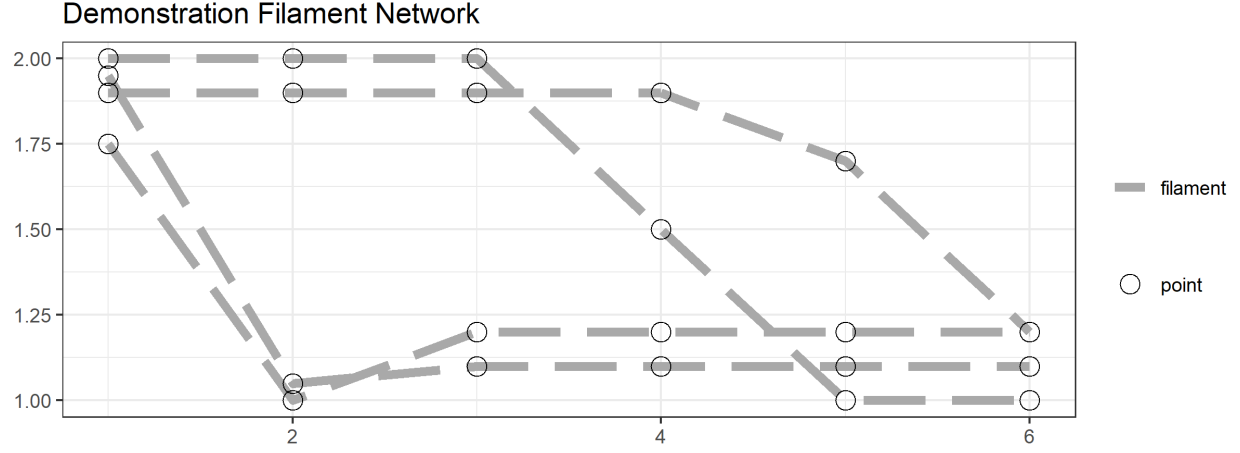


Figure 2.2: Demonstration fillament network. This network contains 3 filaments. Points are sampled along the filaments, in order to produce a point cloud from which persistence homology can be studied.

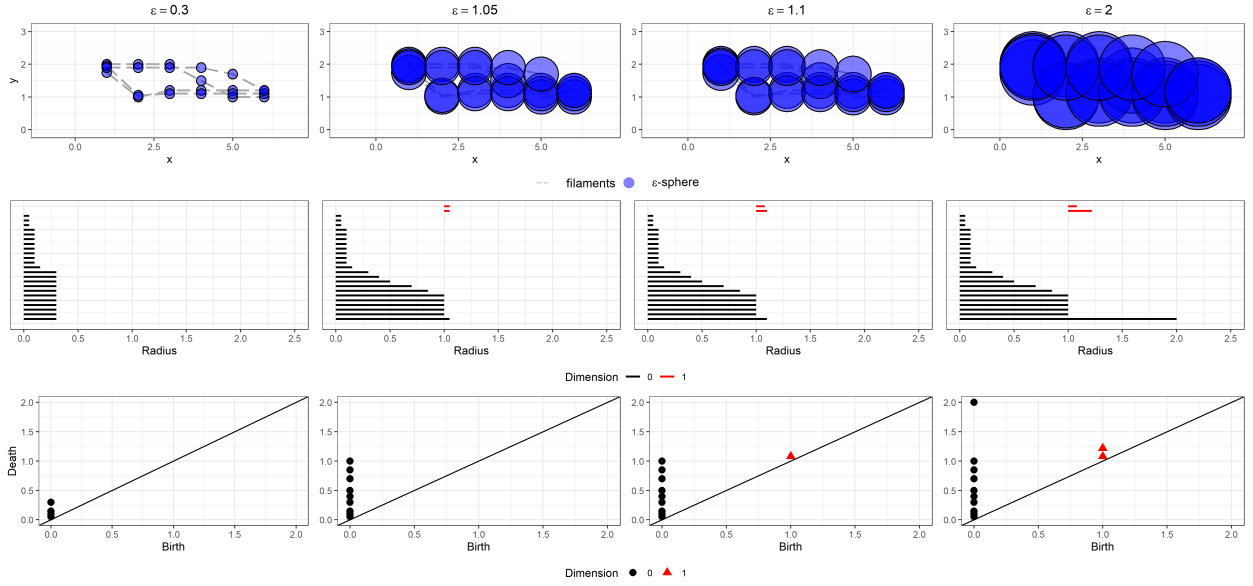


Figure 2.3: Investigation of the persistence homology of the demonstration filament network in Figure 2.2. The first row of figures, shows the growing ϵ -spheres about the sampled points of the fillaments. The second row shows the corresponding persistence barcode. The third row shows the corresponding persistence diagram. The columns progress with the algorithm, right-to-left.

in the Wasserstein distance. This method allows one to adjust the weight assigned to data that might be considered "topological noise" to fit the particular use case.

Definition 2.1. 1 Let D_X and D_Y be two persistence diagrams with cardinalities n and m respectively such that $n \leq m$ and denote $D_x = \{x_1, \dots, x_n\}$, $D_y = \{y_1, \dots, y_m\}$. Let $c > 0$ and $1 \leq p < \infty$ be fixed parameters. The d_p^c distance between two persistence diagrams D_x and D_y is

$$d_p^c(D_x, D_y) = \left(\frac{1}{m} \left(\min_{\pi \in \Pi_m} \sum_{l=1}^n \min(c, \|x_l - y_{\pi(l)}\|_\infty)^p + c^p |m - n| \right) \right)^{\frac{1}{p}},$$

where Π_m is the set of permutations of $(1, \dots, m)$. If $m < n$, define $d_p^c(D_x, D_y) := d_p^c(D_y, D_x)$.

The d_p^c distance not only calculates the distance of points in two persistence diagrams without the simulated points on the diagonal, it also adds a penalty term on the difference in cardinalities of the two sets of points. The parameter c in eq. (2.1) is a constant that controls the weight of penalization to be added in the d_p^c distance. Larger values of c will yield a larger penalization. The parameter p is typically chosen as 2 since this corresponds to the Euclidean distance. We tend to evaluate c between 0 and 1 as these have been empirically found to be appropriate options in real-world applications [21].

Since persistence diagrams can summarize the homological features of multiple dimensions in one diagram, such as in the last panel of Fig. 2.3, the persistence diagram contains both 0-dim features (connected components) with cardinality 5 and 1-dim features (holes) with cardinality 1. We can further define the d_p^c distance of a certain dimensional feature between a persistence diagram and a group of persistence diagrams.

Definition 2.2. 2 Denote $\mathcal{D}$ as a collection of persistence diagrams in the same class. For a specific β -dim homological feature, $\beta = 0, 1, 2, \dots$, the d_p^c distance between a persistence diagram D_x and the set of persistence diagrams $\mathcal{D}$ is

$$d_\beta(D_x, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} d_p^c(D_x, D),$$

where $|\mathcal{D}|$ represents the size of class $\mathcal{D}$.

With all preparations above complete, we can build the d_p^c -based network classifier. For K classes of filament networks, every network in a class is generated under a unique set of constraints. Therefore, we have K sets of persistence diagrams, where each set corresponds to a class of network generated under unique constraints. Given a new filament network with its persistence diagram D' , our goal is to classify under which constraints the network was most likely generated, i.e. to which class k it most likely belongs. We can estimate this membership by calculating the distance between D' and each class of persistence diagrams. We then assign the new network to the class with the smallest distance. Additionally, we parameterize the relative weights for different dimensions of homological features in the

calculation of the distance and force the weights' sum to 1. The classifier is summarized in Algorithm 1.

Algorithm 1: d_p^c -based network classifier

Let B is highest dimension of homological features under consideration.

1. Take the training set $T_1, T_2, \dots, T_K$ from each class of diagrams $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$,
2. For a new network with its corresponding persistence diagram D' , compute

$$d(D', T_k) = \sum_{\beta=0}^B w_\beta d_\beta(D', T_k),$$

where $\sum_{\beta=0}^B w_\beta = 1$, and w_β determine how much β -dim homological feature is considered,

3. Assign D' a class label c' such that,

$$c' = \arg \min_{1 \leq k \leq K} d(D', T_k),$$

2.3.2 Non-distance-based network classifier

Sub-sampling approach

In our method, rather than work on the coordinates of the actin beads directly, we took several pre-processing steps in order to extract more information from the provided network samples. The actin bead data included both spatial coordinates as well as an index indicating the filament sampled. That is, for our networks $\{X_1, X_2 \dots X_{150}\}$, we had 100 filaments in each network $\{F_1, F_2 \dots F_{100}\}$ and therefor our sampled points have features x, y , and f , where x and y are the coordinates in the plane and f is the index in the identity map of F .

Using R's spatial package, each filament was constructed by connecting the (x, y) bead-coordinates in each filament. A single filament network was then constructed as a multiline object from the corresponding collection of lines formed from the bead-coordinates. These multiline objects were then projected onto square grids of 200×200 cells. Then, for each network we sampled the multiline object into a grid with the identity function (i.e. the grid cells have value 0 unless a line and grid cell intersect then the value of the grid cell goes to 1). We now had a grid of 4000 cell values taking $\{0, 1\}$. We filtered these 4000 values to only those of value 1 and used the corresponding planar coordinates to build our new simplicial complexes and persistence diagrams. All re-sampling steps are summarized in Algorithm 2.

Due to the exponential growth of the computation of the Vietoris-Rips complex, rather than computing a Vietoris-Rips complex per network, 1000 points were randomly sampled from each network 3 times without replacement. This gives us 3 sets of 1000 points per network.

Algorithm 2: Re-sampling Algorithm

Let $X_i \in \{X_1 \dots X_{150}\}$ the set of simulated actin networks.

Let $a \in \{A_{i,(x,y,f)} \dots A_{i,(x,y,f)}\}$ the set of points in X_i . a is a pair, $(i, (x, y, f))$, where i is an index to X_i , x and y are planar coordinates and f is an index to a filament of X_i , $F_{i,f} \in \{F_{i,1} \dots F_{i,100}\}$, along which a lies. See Panel A of Fig. 2.4 for a graphical depiction of one X_i . For simplicity, we will describe how our algorithm is applied to X_1 :

1. Reconstruct the paths of each $F_{1,f}$ by connecting $A_{1,(x,y,f)}$. See Panel B, Fig. 2.4.
 2. We project F_1 onto a 200x200 square lattice with the same bounds as the given by the data. The cells, $c_{x,y}$ of the lattice take values $\{0, 1\}$, where $c_{x,y} = 1$ if it is intersected by one of F_1 and 0 otherwise. See Panel C of Fig. 2.4.
 3. Retain only $c_{x,y}$ where $c_{x,y} = 1$.
 4. Draw 1000 random $c_{x,y}$ without replacement and call these the elements of $R_{1,r}$. Repeat this twice so that r indexes three new samples drawn from X_1 . The three colors in Panel D of Fig. 2.4 denote the index r .
 5. Generate three separate persistence diagrams, $D_{1,r}$, for the new samples $R_{1,r}$. Shown in Panel E of Fig. 2.4.
-

This increased the sample size from 150 to 450 networks. We computed persistence diagrams for these 450 networks and could use these to construct a classifier using the same procedure as outlined in Algorithm 1.

Persistence vectorization/classification

In addition to distance based classification, a classification was performed on vectorized features of the persistence diagrams. A matrix was generated from the persistence diagrams which had row entries for each diagram and column features with the mean and standard error of persistence diagrams considering only the 0th or 1th persistence features alone and all topological features. That is, for each filament network, there is one persistence diagram which corresponds to one row of the vectorized matrix with mean and standard error of birth or death times for 0th, 1th, or all features. This gives us 12 columnar features. However, the mean and standard error of births for 0th features are constant 0 and are not considered. Therefore, we are left with 10 entries per vector. The vectorization procedure is shown in Algorithm 3.

2.4 Classification result

2.4.1 d_p^c -based classifier

In our data set, we are provided three classes of filament networks. Each class of filament network is generated with a different number of cross-linking proteins. Each class contains 50 samples. Therefore, there are total of 150 individual provided filament networks.

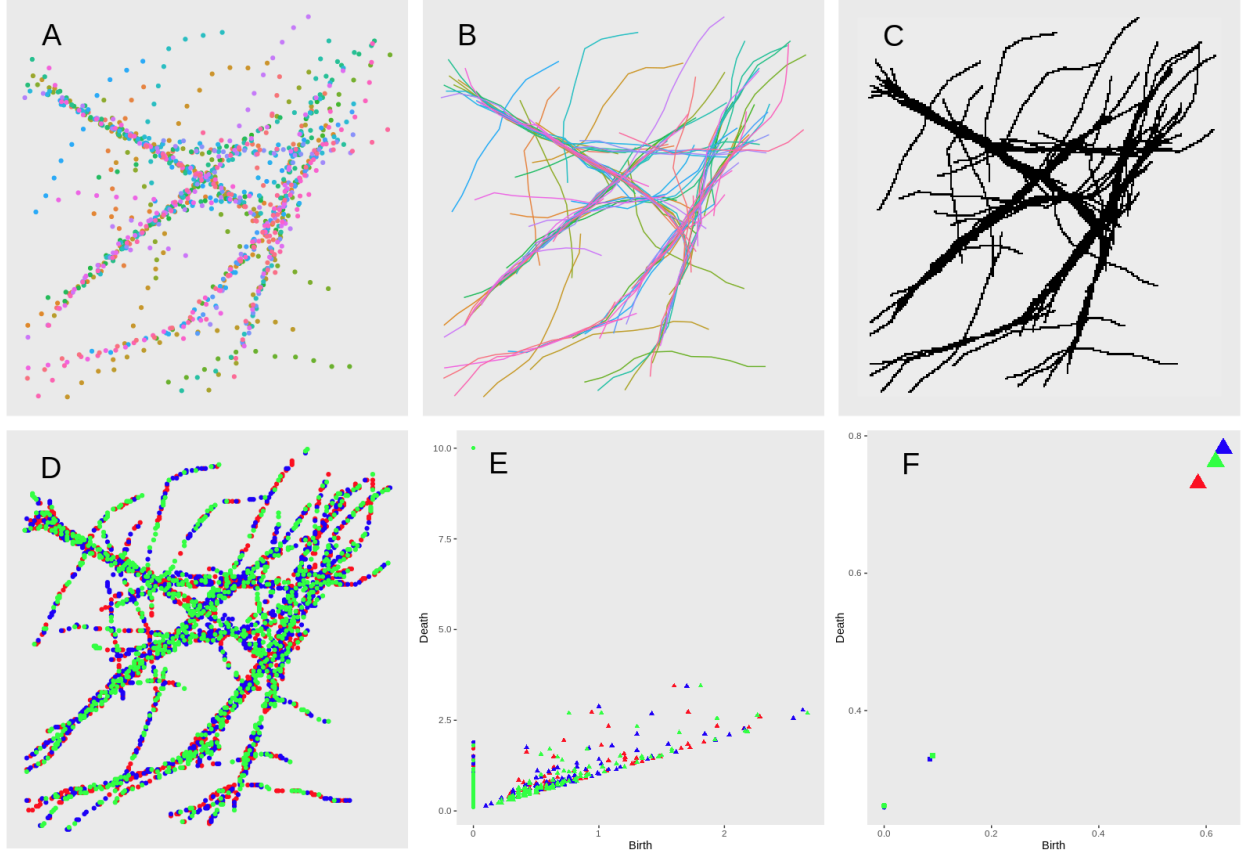


Figure 2.4: Example of one filament construction (B) from the supplied point cloud (A), rasterization (C), re-sampling (D), construction of persistence diagram (E) and vectorization of persistence diagram (F). The point cloud in (A) was supplied with an index that allowed for the construction of the filaments in (B); note the coloring. This was projected onto a 200x200 raster (C). The raster was sampled into 3 sets (colored) of 1000 points (D) without replacement and the x,y coordinates are used to generate a VR complex. The persistence diagrams and vectorizations in (E) and (F) are colored the same as (D), to show the variability that arises in the vectorized persistence diagram features when this approach is taken.

Algorithm 3: Vectorization of persistence diagrams

We vectorize $D_{1,r}$ from Algorithm 2 by taking the mean *birth* and mean *death* of connected components and holes. These data are plotted in Panel F of Figure 3.

For $D_{1,r}$ we have 3 row-vectors in the form:

$$v_{1,r} = \begin{bmatrix} \bar{x}(\text{Death of Connected components}) \\ s(\text{Death of Connected components}) \\ \bar{x}(\text{Birth of holes}) \\ s(\text{Birth of holes}) \\ \bar{x}(\text{Death of holes}) \\ s(\text{Death of holes}) \\ \bar{x}(\text{Birth of all features}) \\ s(\text{Birth of all features}) \\ \bar{x}(\text{Death of all features}) \\ s(\text{Death of all features}) \end{bmatrix}$$

Note: the mean $\bar{x}$ and standard error s of the birth of connected components would be 0 for all vectors, so these are not found in $v_{1,r}$. Repeat for all X_i . Our complete matrix of data has a final dimension 450×10 .

In order to compare classifiers, we employed 10-fold cross validation to estimate overall classification accuracy. All of the networks are randomly partitioned into 10 mutually exclusive sets. 9 partitions are selected as a training set, while the remaining 1 partition is used for testing. We repeat the classification 10 times such that every partition acts as a testing set exactly once. We consider the overall classification accuracy rate as the mean accuracy across all partitions.

After generating the persistence diagram for each filament network from the locations of actin beads, we tested our classifier on this data set using the d_p^c -based classifier in Algorithm 1. We consider only 0-dim and 1-dim topological homological features. We chose $p = 2$ to imitate Euclidean distance. The choices of parameter w_0, w_1, c were found to be optimal based on cross validation. When $w_0 = 0.5, w_1 = 0.5$, connected components and holes are weighted equally, and $c = 0.2$ gives a relatively small contribution from the cardinality difference in the d_p^c distances, the best classification accuracy rate is 89%.

2.4.2 non-distance based

A support vector machine (SVM) was used to develop a network classifier based on the 3-times re-sampled and vectorized persistence diagrams. 10-fold cross-validation was again used to assess the accuracy of the SVM classifier. Accuracy was measured only on one of the three sub-sampled networks $R_{i,1}$ per filament network X_i so as to fairly compare to the methods without re-sampling. The mean accuracy of the classifier was 96.3%. This was the highest accuracy attained on this set of simulated networks.

2.4.3 Comparison to other classifiers

We also list the accuracy rate by using other classifier in Table 2.1.

Our findings show that re-sampling of data can produce a more reliable filament network classifier even when data are mapped to a topological space and summarized in persistence diagrams. The d_p^c -based classifier performed much better than Wasserstein-based classifier. Our re-sampled and vectorized persistence diagram based classifier and d_p^c -based classifier are superior than any other common classifiers. These findings suggest that future with with real data microscopy data may be classified with similar methods.

2.4.4 Application to classifying myosin mutants

We are given labeled images of cells belonging to one of two classes: myosin mutant (MM) or wild type (WT). The MM cells have a genetic knockout such that they do not produce one of their myosin proteins. The WT cells are the control with no knockout. Myosin motor proteins are known, via empirical observation, to influence the connectivity and shape of the actin filament network. This experiment differs from our synthetic experiment in that our synthetic experiment involved the direct manipulation of the actin network through changes in cross-linker abundance. Whereas cross-linking is fundamental to the structure of the actin network, the linkage between myosin motors and actin networks is more complex and likely modulated by several interacting factors.

We perform classification of MMs and WTs using our top classifier, the re-sampled SVM. The data are provided as tifs from 22 MM and 20 WT cells. Each tif is a a very high resolution (roughly 2500x500 pixels with $0.043 \mu\text{m}^2$ pixels) grayscale image, wherein the pixel values are an 8-bit intensity. We perform preprocessing on the images simply by performing a gaussian blur and then thresholding to the top $\frac{1}{3}$ of values. The thresholded image is then a grid of values taking only 0 or 1, where 1 indicates the presense of a filament. We then take the processed images and proceed in classification from step (C) in Figure 2.4, resampling, computing PDs, vectorizing the PDs, and finally classifying with an SVM. We find a cross-validated (k=5) accuracy of roughly 85%.

Interpretation of these results is complicated by our uncertainty of the true, cumulative impact of the myosin knockout on the actin network. While the accuracy in this experiment is significantly lower than found in the analysis of the synthetic data, this could easily be explained by the single motor protein knockout having a relatively minor impact on actin organization when compared with up or down regulation of cross-linkers. Out of curiosity, we conducted a small survey ($n=6$), wherein we asked respondents who were familiar with the experiment to separate the cells into two classes. We found that respondents had an average accuracy of 63%, with a minimum accuracy of random chance and a maximum accuracy of 75%. Since our algorithm is more effective than manual classification by an expert, we hypothesize that there is a weak relationship between this myosin knockout and the topology of the actin network, and our classification algorithm is likely running into the ceiling of discernible difference in classes.

2.5 Clustering of filament networks

In our classification results, it is apparent that conducting supervised classification on a small sample and without a state-of-the-art accuracy for comparison can make diagnostics and inference problematic. When conducting cross-validation, each round’s classification result has only one misclassification to examine. Therefore, it is nearly impossible for an expert to visually examine the misclassification and establish a pattern. We can provide an alternative framework via unsupervised clustering, which will be helpful to biologists interested in visually examining the persistence information mapped back to the data space. In this section we take the approach of k-means clustering in order to examine the outcome clusters and try to learn from the mistaken cluster assignments.

Rather than using the subsampling approach, in this clustering application we choose a value of r , so that each cell has one corresponding vector $v_{i,1}$. Rather than clustering directly on the v_i s, we first transform our data into principal components and use the first two principal components. We can plot these principal components as in Figure 2.6 with color indicating myosin mutant (M) vs wildtype (WT) cells. In this figure, we see that there is clearly a separation of classes, but the classes are not well clustered about obvious centers.

K-Means clustering aims to find the partitioning of our data which minimizes the within groups sum of squares (WGSS) given k a number of partitions [19]. Determining the optimal k can be done through a heuristic known as the Elbow Method. The Elbow Method computes the outcome WGSS from k partitions of our data where k ranges from 1 to K ($K \ll n$). One selects the smallest k before returns on a larger k are considered too small. This heuristic approach seeks to optimize against overfitting and for minimizing the WGSS. In Figure 2.7, we show a plot of outcome WGSS for $k \in \{1, 2 \dots 15\}$. In this figure, we do not see one substantial decline in benefit increasing in k . This can be considered due to a lack of evident partitioning in the data.

Since $k = 2$ is a natural choice for k , given our experiment and we do not find strong evidence for another choice of k , we conduct the clustering partitioning the data into two groups. This method does not result in our highest accuracy outcome (80.9%); i.e the outcome groups are a mixture of classes. We plot the data showing labels in color and cluster assignments as shapes in Figure 2.8.

While this method does not provide the highest accurate result in this work, the unsupervised algorithm combined with using only the principle components of the persistence information and avoiding bootstrapping should all help to avoid overfitting. This method could be seen as preferable then for diagnostic and inferential purposes. We plot the outcome clusters in their original data format as we would return to the biologists in order for them to investigate and make further inferences. This output is shown in Figure 2.9, where the two columns are each a group of cells and the red Xs indicate incorrectly grouped cells. In this case, the first column is mostly mutant cells and the second column is mostly wildtype cells, so the Xs in the first column indicated wiltype cells and the Xs in the second column indicate mutant cells.

2.6 Conclusion

In this work, we successfully classified simulated actin filament networks, generated under three different initial conditions, with very high accuracy. We combined a machine learning framework with TDA for our classification. We compared with several distance-based classifiers. Our method most successful method uses a resampling technique, a vectorization of the persistence diagram and an SVM as the classifier. Each of these methods were built on the foundation of topological homology by encoding geometric features hidden in the data into the topological features and summarizing those into persistence diagrams.

From these results, we are able to confidently classify synthetic actin filament networks based on unique initial conditions. Accordingly, these methods could reveal key factors in the filament network organizing process and provide biologists with opportunities to study the interaction of motor proteins, actin networks and cytoplasmic streaming. We demonstrate that it is simple to bridge our method to classify cells by microscopy images of their actin networks. We also demonstrate a method of clustering on the principal components of the vectorized persistence diagrams and show how these clustering can be used to examine the unsupervised cluster assignments in the original data space of microscopy images. Overall, our work is the first time that actin filament networks have been studied by direct topological classification. Researchers could further advance their understanding of cell physiology through this work and future research.

Table 2.1: Accuracy rate of classifiers

classifier	accuracy
SVM, re-sampled	96%
d_p^c -based	89%
Wasserstein-based, re-sampled	87%
Wasserstein-based	83%
Bayes Factor [20]	83%
SVM PI [41]	75%
Neural Net PI	71%
SVM Raster	65%
Random Forest Raster	55%

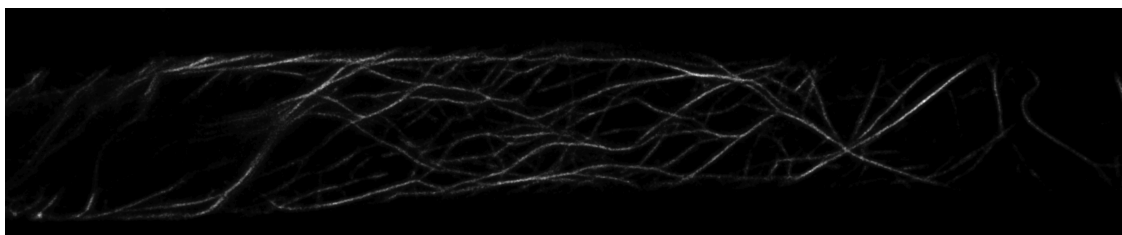


Figure 2.5: Example of a raw WT cell image.

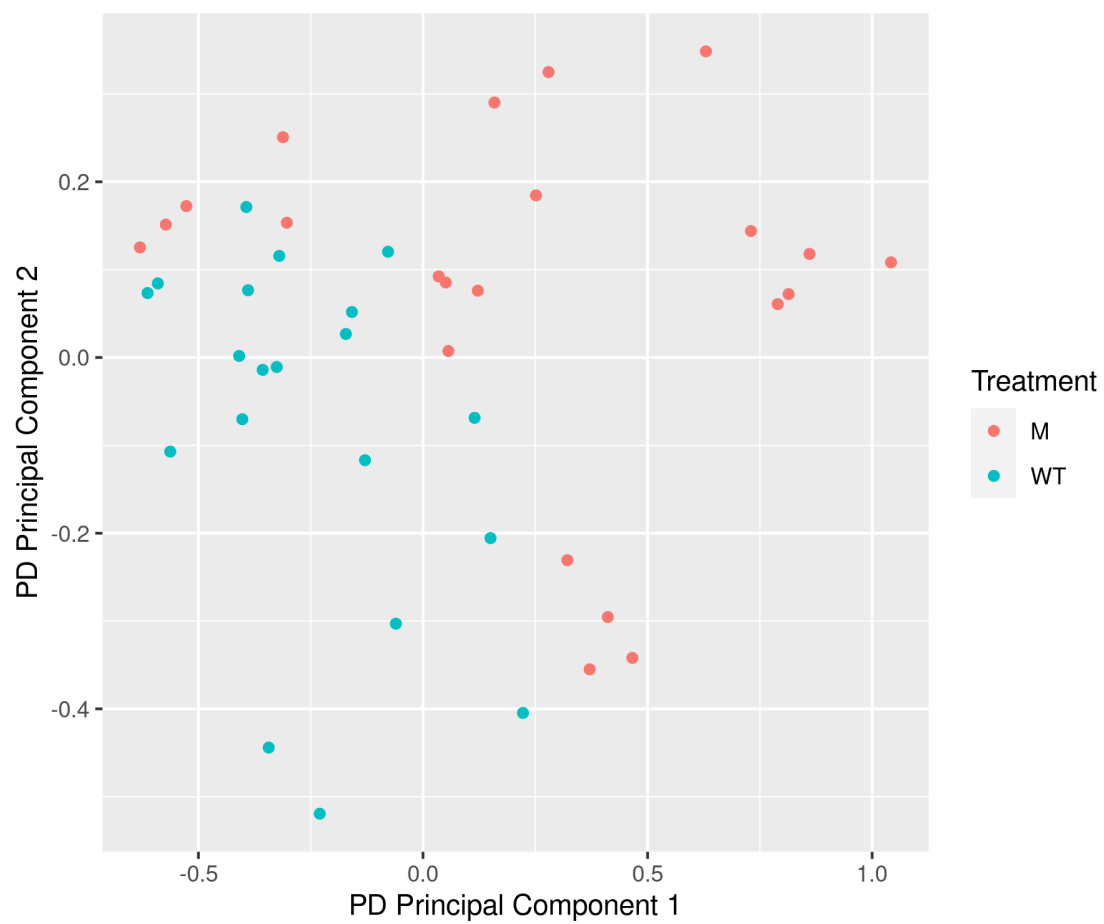


Figure 2.6: Plot of cell's persistence information principal components. Points are colored by treatment.

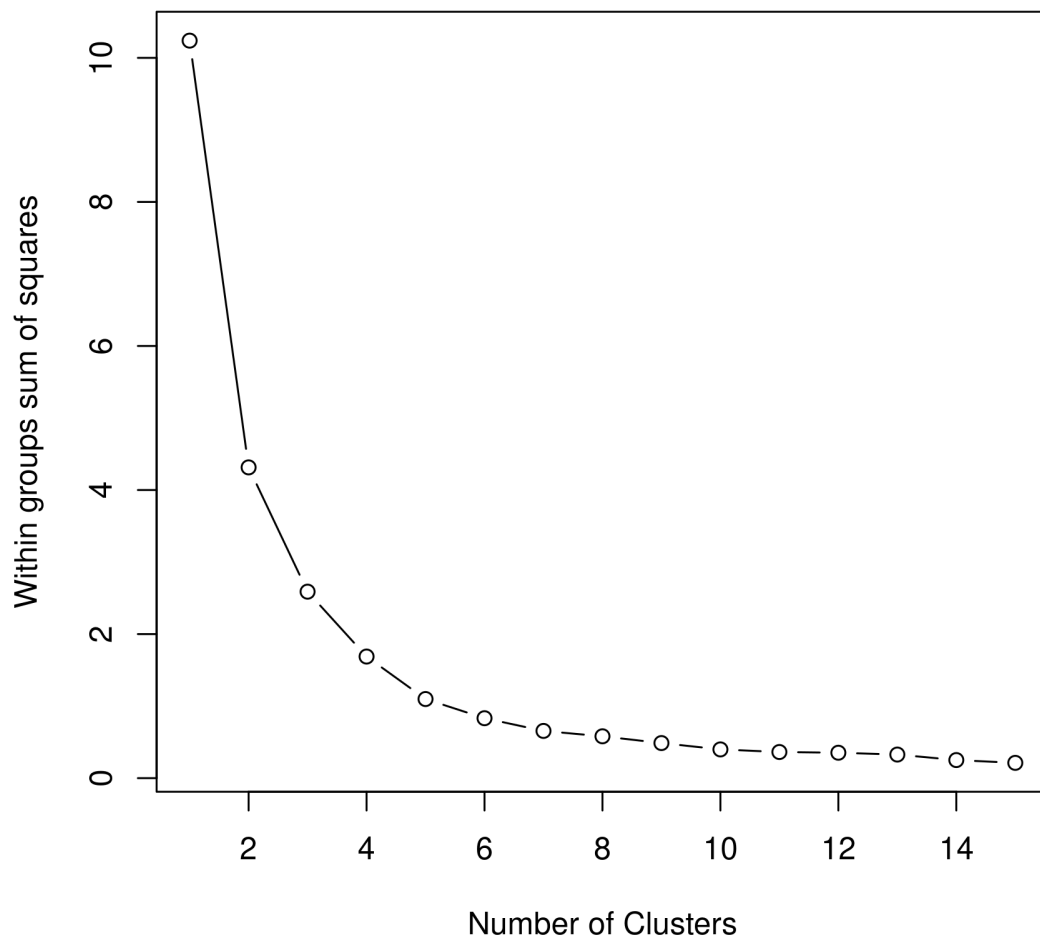


Figure 2.7: WGSS with several numbers of clusters (values of k).

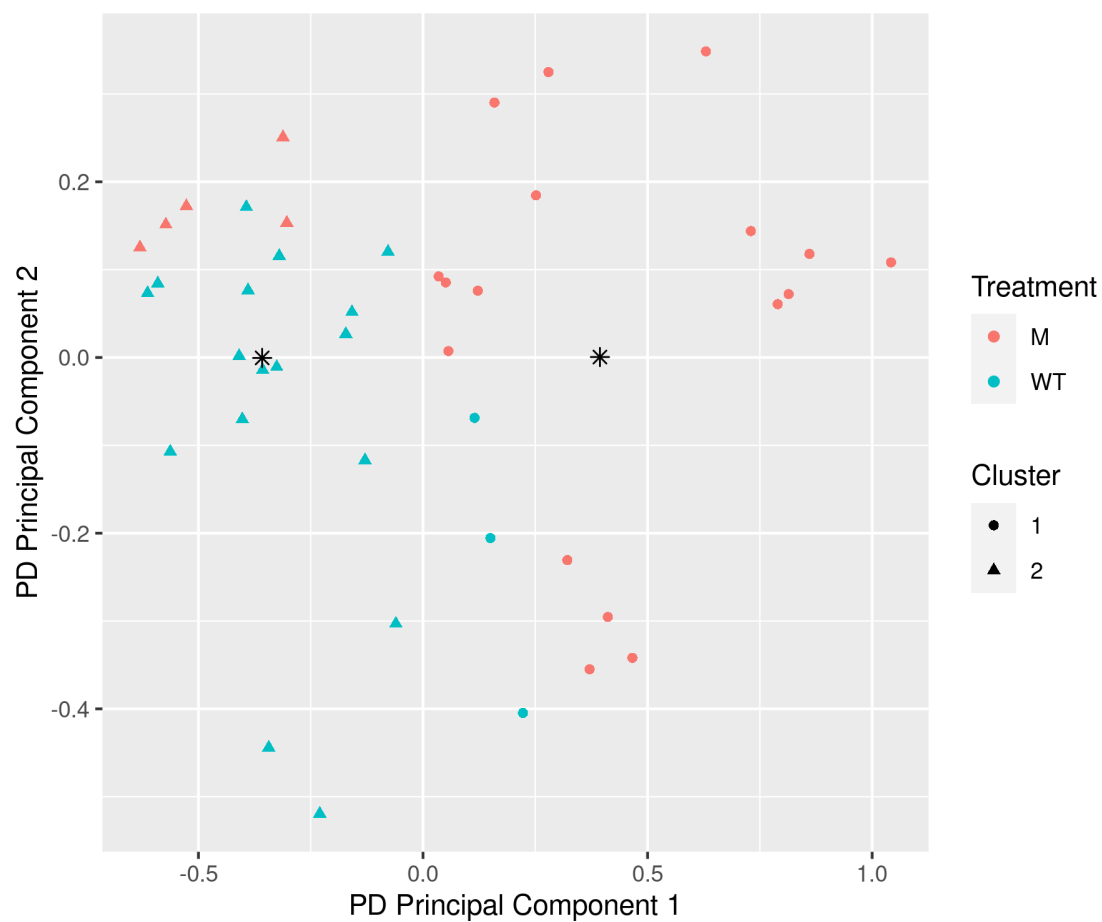


Figure 2.8: Plot of cell's persistence information principle components. Points are colored by treatment and class assignments from k-means clustering are indicated by shape. The asterisks indicate the cluster centers.

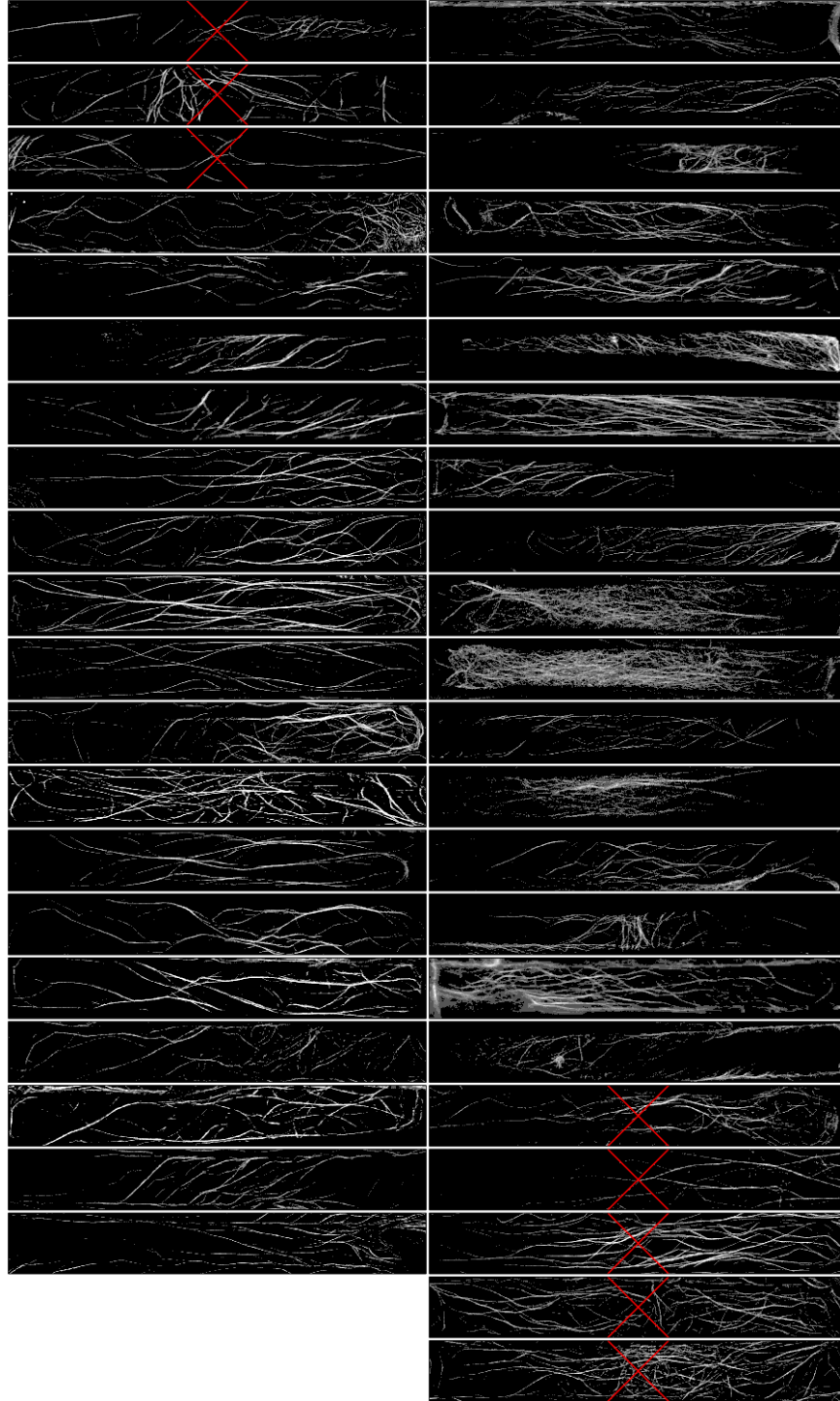


Figure 2.9: Diagram showing misclassified cells based on k-means clustering with $k = 2$. The groups are separated by column and the Xs indicate cells that do not agree with the majority in the cluster. The left column is mostly mutant, so the X's indicate wildtypes. The right side is mostly wildtype, so the Xs indicate mutants.

Chapter 3

Topological CNNs

In this chapter, I propose a novel generalization of the convolutional neural network (CNN) based on topological manifolds. Deep neural network (NN) architectures are the preeminent tools for many image classification tasks, since they are simple to implement and capable of distinguishing between a large number of classes with a high degree of precision and accuracy [63]. The modern graphic processing unit (GPU) has brought the ability to run tremendously large and powerful network classifiers to a reasonable cost. Quantum algorithms promise even faster NNs with novel structures [64]. A common NN architecture used for image classification is the CNN [65, 6]. The characterizing principle of CNNs is the use of convolutional layers which take advantage of the grid-space of a digitized natural image to sparsify a fully connected network and to build spatial features for later classification. CNNs typically fit a very large number of parameters, on the order of 10^5 , which are trained through a stochastic optimization routine. CNNs balance the learning of spatially local and global features —i.e. they must simultaneously learn the textures, lines, edges, etc— and also where in an image the subject is most prominent and how to assemble complex features. It is desirable for CNNs to produce models which are invariant under rotation or scaling [66].

Topological data analysis (TDA) is a natural method of investigating CNNs, as scaling and rotation invariance are critical to CNN performance and are also tenets of topology. TDA has been shown to be a valuable tool for investigating the hidden topological structure of data [67, 26, 18] and has seen rapid development in machine learning [68, 69, 21, 20, 29, 70]. Despite the immense parameter space of a CNN, it has been shown through topological clustering that CNNs often arrive at spatially meaningful configurations within their convolutional layers [38]. Perhaps this is unsurprising, given that earlier work showed that natural images themselves cluster about grids which can be embedded on a manifold called the Klein bottle [71]. This intuition has also been demonstrated in findings that show how a small number of natural images can be considered a very strong prior for the learning of the local spatial topology of the image [72].

CNNs typically have several major drawbacks including that the models are difficult to interpret, require large, labeled datasets, use non-trivial optimization procedures, and often do not generalize well [73]. It has been demonstrated that as CNNs grow in size and complexity they often do not result in a proportional increase in utility [65]. This suggests that bigger/deeper models alone will not continue to advance image classification. We should instead utilize our growing understanding of the empirical, topological structures of CNNs to develop novel tools that address contemporary CNNs’ shortcomings. Improvements to interpretability, generalization, training set size and training speed are all of great value.

This work seeks to demonstrate a framework to leverage the known topology of CNN convolutional layer-weights. I develop and extend work on the incorporation of topological features to NNs. I introduce two new types of convolutional layers which construct topological features or restrict convolutions based on functions on topological manifolds. Through topological feature generation and continuous topological convolution I develop a new CNN framework I call the topological-CNN (TCNN). TCNNs are inherently easier to interpret than CNNs, as the local image convolutions have clear spatial interpretations. I show synthetic and empirical results which indicate that these models can also have superior generalizability and are faster to train. Many studies have incorporated topological information into NNs via preprocessing or adaptations to the loss function [74, 75, 76, 77]. This work takes the view that the weights of the convolutional layers themselves can be viewed as sets of functions in a topological space. From this vantage, I propose a highly adaptable framework within the convolutional structure of the network which is instantiated with desirable topological properties.

My methods have direct analogues of the techniques used with convolutional neural networks, such as pooling and holdouts. In addition, they are readily extensible in a number of ways. For example, as we discover types of patches that are important but are not yet included within the Klein bottle geometry, they can be added to create new geometries in a systematic way. The methods are also extensible in the sense that other modalities, such as video and 3D imaging, have their own geometries and corresponding neural network architectures, and can thus be treated in the same way.

3.1 Introduction

It was observed in [78] that one motivation for the construction of Convolutional Neural Networks (CNNs) was that they permit a sparsification based on the geometry of the space of features. In the case of convolutional neural networks for images, the geometry used was that of a two-dimensional grid of features, in this case pixels. M. Robinson has also pointed out the importance of geometries or topologies on spaces of features, coining the term *topological signal processing* to describe this notion [79]. In this paper, we study a space of image filters closely related to a subfamily of the Gabor filters, whose geometry is that of a well known geometric object, the *Klein bottle*¹. These filters augment the feature

¹See Figure 3.2 for an image of a Klein Bottle immersed in $\mathbb{R}^3$.

set in the data, but they can also be used to construct analogues of convolutional neural networks with improved performance on a number of measures of performance. The method uses a discretization (in the form of a graph) of the Klein bottle as a template for new layers, which produce additional sparsification.

We implement the use of the Klein bottle geometry and its image filters via additional structure on convolutional layers. We call neural networks with these layers **Topological Convolutional Neural Networks (TCNNs)**. We perform experiments on image and video data. The results show significant improvement in TCNNs compared to conventional CNNs with respect to various metrics.

Deep neural network (NN) architectures are the preeminent tools for many image classification tasks, since they are capable of distinguishing between a large number of classes with a high degree of precision and accuracy [63]. CNNs are components of the most commonly used neural network architecture for image classification, e.g. see [65, 6, 80]. The characterizing property of CNNs is their use of convolutional layers which take advantage of the 2-dimensional topology of an image to sparsify a fully connected network and employ weight sharing across slices. Each convolutional layer in a CNN assembles spatially local features, e.g. textures, lines, and edges, into complex global features, e.g. the location and classification of objects. CNNs are also used to classify videos; see e.g. [81], [82], and [83]. CNNs have several major drawbacks including that the models are difficult to interpret, require large datasets, and often do not generalize well to new data [73]. It has been demonstrated that as CNNs grow in size and complexity they often do not enjoy a proportional increase in utility [65]. This suggests that bigger, deeper models alone will not continue to advance image classification.

The topological structure in the layers of a TCNN is inspired from prior research on natural image data with topological data analysis (TDA). Small patches in natural images cluster around a Klein bottle embedded in the space of patches [71]. The patches corresponding to this Klein bottle are essentially edges, lines, and interpolations between those; see top panels of Figure 3.3(b)(c). It has been shown through TDA that CNN weights often arrive at these same Klein bottle patches after training [38]. The key idea of TCNNs is to use the empirically discovered Klein bottle, and the image patches it represents, directly in the structure of the convolutional layers. For example, the Klein bottle image patches are used in TCNNs as convolutional filters that are fixed during training. There is an analogue of the Klein bottle for video data. Because of symmetries present in the geometric models of feature spaces, there are generalized notions of weight sharing which encode invariances under rotation and black-white reversal.

The method is not simply an addition of features to particular data sets, but is in fact a general methodology that can be applied to all image or video data sets. In the case of images, it builds in the notion that edges and lines are important features for any image data set, and that this notion should be included and accounted for in the architecture for any kind of image data analysis. TDA contains a powerful set of methods that can be used

to discover these latent manifolds on which data sit [22, 84, 70, 29]. These methods can then be used to inform the parametrization of the TCNN.

TCNNs are composed of two new types of convolutional layers which construct topological features and restrict convolutions based on embeddings of topological manifolds into the space of images (and similarly for video). TCNNs are inherently easier to interpret than CNNs, since in one type of layer (Circle Features layer and Klein Features layer; Definition 9) the local convolutional kernels are easily interpreted as points on the Klein bottle, all of which have clear visual meaning (e.g. edges and lines). Following a Klein Features layer it makes sense to insert our other type of topological layer (Circle One Layer and Klein One Layer; Definitions 7 and 8), in which the 2D slices in the input and output are parameterized by a discretization of the Klein bottle and all weights between slices that are farther away than a fixed threshold distance on the Klein bottle are held at zero throughout training. This has the effect of localizing connections in this convolutional layer to slices that are nearby to each other as points on the Klein bottle. The idea of this ‘pruned’ convolutional layer is that it aggregates the output of Klein bottle filters from the first layer that are all nearby each other. *Both of these new types of topological convolutional layers can be viewed as a strict form of regularization, which explains the ability of the TCNN to more effectively generalize to new data.* We also provide versions of these layers for video and 3D convolutions.

The main points of the paper are as follows.

1. The method provides improved performance on measures of accuracy, speed of learning and data requirements, and generalization over standard CNNs.
2. Our method can be substituted for standard convolutional neural network layers that occur in other approaches, and one should expect improvements in these other situations as well. In this paper, we compare TCNNs to traditional CNNs with the same architecture except for the modified convolutional layers. The comparatively better performance of the TCNNs provides evidence that TCNNs improve the general methodology of CNNs, with the expectation that these advantages can be combined with other methods known to be useful. For example, we use TCNN layers in a ResNet for video classification; see Section 3.5. As another example, we expect state-of-the-art transfer learning results to improve when some convolutional layers in the network are replaced by TCNN layers.
3. Our approach suggests a general methodology for building analogues of neural networks in contexts other than static images. We carry this out for the study of video data. In this case, the Klein bottle $\mathcal{K}$ is replaced by a different manifold, the so-called *tangent bundle* to $\mathcal{K}$, denoted by $T(\mathcal{K})$. There are also straightforward extensions to 3D imaging and video. The improvement afforded by these methods is expected to increase as the data complexity increases, and we find this to be the case in the passage from static images to video.

4. The simple geometries of the feature spaces enable reasoning about the features to use. In the video situation, we found that using the entire manifold was computationally infeasible, and selected certain natural submanifolds within $T(\mathcal{K})$, which allowed for the improved performance. Even in the case of static images, there are natural submanifolds of $\mathcal{K}$ which might be sufficient for the study of specific classes of images, such as line drawings, and that restricting to them would produce more efficient computation.
5. There are more general methods of obtaining geometries on feature spaces, which do not require that the feature space occur among the family of manifolds already studied by mathematicians. One such method is the so-called Mapper construction [85]. A way to use such structures to produce a sparsified neural network structure has been developed [38, Section 5.3]. The manifold methods described in this paper are encapsulated in the general framework described in Remark 6 which can be applied to other domains.
6. Because of the intuitive geometric nature of the approach, it permits additional transparency into the performance of the neural network.
7. There are no pretrained components in any of the models considered in this paper. This is significant in particular because the vast majority of state-of-the-art models for classification of the UCF-101 video dataset do use pretrained components [86], [87], [88].

The structure of the paper is as follows. In Section 3.2 we recall the basic structure of convolutional neural networks and we set up notation. TCNNs are introduced in Section 3.3; the version for image data is in Section 3.3.1, the video version is in Section 3.3.2, and a connection with Gabor filters is explained in Section 3.3.3. Then in Section 3.4 we describe experiments and results comparing TCNNs to standard CNNs on image data. In Section 3.5 we do similar experiments on video data.

3.2 Background: CNNs

In this section we describe in detail the components of the CNN critical to the construction of the TCNN. One of the central motivations of the structure of a CNN is the desire to balance the learning of spatially local and global features. A convolutional layer can be thought of as a sparsified version of the fully connected, feed-forward (FF) layer, with additional homogeneity enforced across subsets of weights. The traditional CNN makes use of the latent image space by creating a small perceptive field with respect to the L^∞ -distance in which weights can be nonzero, thus sparsifying the parameter space by enforcing locality in the image. Additionally, homogeneity of weights across different local patches is enforced, further reducing the parameter space by using the global structure of the grid.

To describe the structure of CNNs and TCNNs, we adopt the language in [38], which we summarize as needed. To aid the reader, we provide an accompanying visual guide in Figure 3.1.

We describe a feed forward neural network (FFNN) as a directed, acyclic graph (Definition 1).

Definition 1. *A Feed Forward Neural Network (FFNN) is a directed acyclic graph Γ with a vertex set $V(\Gamma)$ satisfying the following properties:*

1. $V(\Gamma)$ is decomposed as the disjoint union of its **layers**

$$V(\Gamma) = V_0(\Gamma) \sqcup V_1(\Gamma) \dots \sqcup V_r(\Gamma).$$

2. If $v \in V_i(\Gamma)$, then every edge (v, w) of Γ satisfies $w \in V_{i+1}(\Gamma)$.
3. For every non-initial node $w \in V_i(\Gamma) : i > 0$, there is at least one $v \in V_{i-1}(\Gamma)$ such that (v, w) is an edge of Γ .

The vertices in $V(\Gamma)$ are also called **nodes**. For all i , $V_i(\Gamma)$ consists of the **nodes in layer- i** . The 0^{th} layer $V_0(\Gamma)$ consists of the **inputs** to the neural network (Figure 3.1 (a)). The last layer $V_r(\Gamma)$ consists of the **outputs**.

Notation 2. Let Γ be a FFNN with vertex set $V(\Gamma)$.

1. Often we suppress Γ in the notation, writing $V = V(\Gamma)$ for the set of all nodes and $V_i = V_i(\Gamma)$ for the set of nodes in layer i .
2. For any $v \in V$, the set $\Gamma(v)$ consists of all $w \in V$ such that (v, w) is an edge in Γ , and the set $\Gamma^{-1}(v)$ consists of all w such that (w, v) is an edge in Γ .

To describe the edges between nodes in successive layers, we use the notion of a **correspondence** between V_i and V_{i+1} , which is simply a subset of the product $C \subset V_i \times V_{i+1}$. For $v_0 \in V_i$ and $w_0 \in V_{i+1}$, we define the subsets

$$\begin{aligned} C(v_0) &:= \{w \in V_{i+1} \mid (v_0, w) \in C\} \subset V_{i+1}, \\ C^{-1}(w_0) &:= \{v \in V_i \mid (v, w_0) \in C\} \subset V_i. \end{aligned}$$

Note that C is determined by the subsets $C(v_0) \subset V_{i+1}$ for $v_0 \in V_i$, and it is also determined by $C(w_0)$ for $w_0 \in V_{i+1}$. In this way, a correspondence is a generalization of a map from V_i to V_{i+1} ; Given an element in V_i the correspondence provides a subset of V_{i+1} , and we denote correspondences by $V_i \xrightarrow{C} V_{i+1}$.

We adopt the convention that given nodes $v \in V_i, w \in V_{i+1}$, the edge (v, w) is in Γ if and only if $(v, w) \in C$. Note that this implies $C(v) = \Gamma(v)$. We call C the **edge-defining correspondence of the layer**. In this way, a FFNN specifies an edge-defining correspondence between each pair of successive layers, and conversely, choices of edge-defining

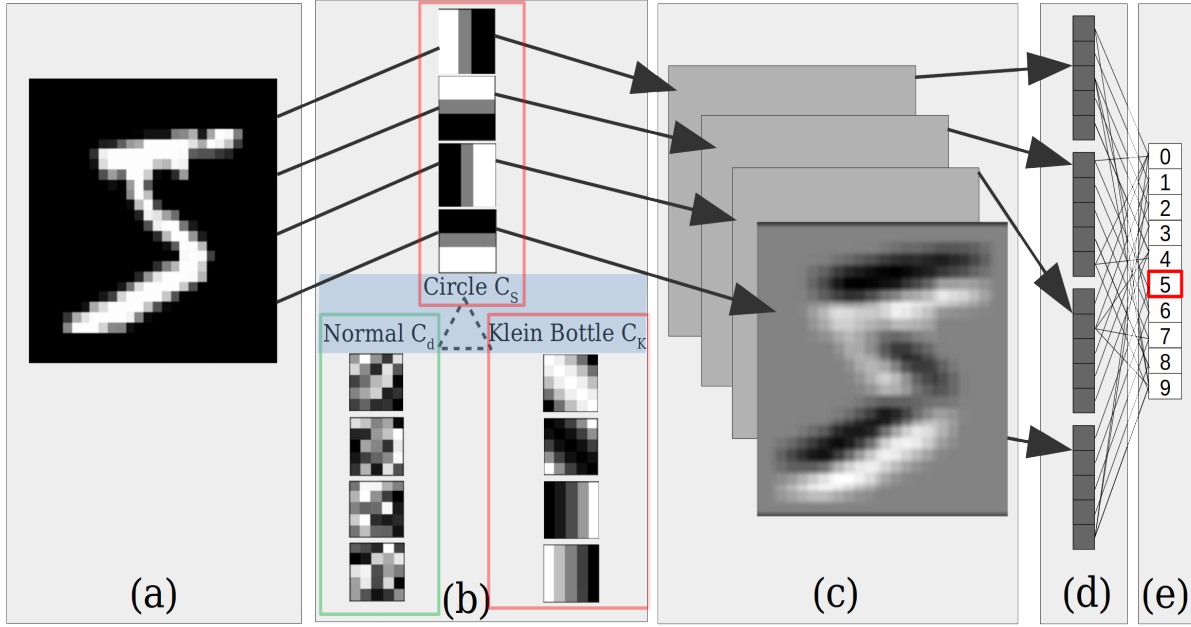


Figure 3.1: Visual guide to CNN (green rectangle) and TCNN (red rectangles) architectures. A typical CNN layer for image classification takes in a 2-dimensional array (a) as an input image and convolves this image through a system of spatially localized filters (b) to produce multiple slices of processed data (c) called feature maps. The feature maps in (c) are flattened to a column vector in (d) and passed through a fully-connected layer to the output nodes (e) which are of cardinality equal to the number of classes in the prediction space. The TCNN modifies the typical CNN framework by specifying the weights in (b) to lie on a topological manifold such as the circle or Klein bottle. We indicate this choice by the dashed triangle in (b) and red rectangles indicating the selection of circle or Klein weights. This shows the behavior of CF and KF type TCNN layers. The COL and KOL type TCNN layers also modify the weights of a traditional CNN layer but in a different way: They localize weights with respect to the circle and Klein bottle topologies yet, as usual, instantiate them randomly and allow them to vary during training.

correspondences between successive layers specify the edges in a FFNN. The simplest type of layer in a neural network is as follows.

Definition 3. Let V_{i+1} be a layer in a FFNN. We call V_{i+1} a **fully connected layer** if the edge-defining correspondence $C \subset V_i \times V_{i+1}$ is the entire set. In that case, we denote this correspondence $C_c = V_i \times V_{i+1}$.

We proceed to describe convolutional layers. We model digital images as grids indexed by $\mathbb{Z}^2$. Modifications of our constructions to finite size images will be clear. The values of the grid specifying a grayscale image are then triples $(x, y, i) : x, y \in \mathbb{Z}$ where $i \in [0, 1]$ is the intensity at (x, y) . Equivalently, an image is a map $\mathbb{Z}^2 \rightarrow [0, 1]$. Similarly, videos are modeled as grids indexed by $\mathbb{Z}^3 = \mathbb{Z}^2 \times \mathbb{Z}$, where the $\mathbb{Z}^2$ are the spacial dimensions and the third dimension is time. This generalizes to grids indexed by $\mathbb{Z}^N$ for any positive integer N .

In a CNN, the nodes in each convolutional layer form multiple grids of the same size. We model the nodes in a convolutional layer as the product of a finite index set $\mathcal{X}$ with a grid, $V_i = \mathcal{X} \times \mathbb{Z}^N$. With this notation, the graph structure of a CNN is specified as in the following definition. A convolutional layer is a sparsification of a fully connected layer which enforces locality in the grids $\mathbb{Z}^N$. For further detail, see [38]. We explain the homogeneity restrictions on the weights in (3.1).

Definition 4. Let V_{i+1} be a layer in a FFNN. We call V_{i+1} a **convolutional layer** or a **normal one layer (NOL)** if $V_i = \mathcal{X} \times \mathbb{Z}^N$ and $V_{i+1} = \mathcal{X}' \times \mathbb{Z}^N$ for some finite sets $\mathcal{X}$ and $\mathcal{X}'$ and a positive integer N , and if for some fixed **threshold** $s \geq 0$ the edge-defining correspondence $C \subset V_i \times V_{i+1}$ is of the form

$$C = C_c \times C_{d,N}(s),$$

where $C_c = \mathcal{X} \times \mathcal{X}'$ is the fully connected correspondence and $C_{d,N}(s) \subset \mathbb{Z}^N \times \mathbb{Z}^N$ is the correspondence given by

$$C_{d,N}(s)^{-1}(\underline{x}') := \{\underline{x} \in \mathbb{Z}^N \mid d_{\mathbb{Z}^N}(\underline{x}, \underline{x}') \leq s\}$$

for all $\underline{x}' \in \mathbb{Z}^N$. Here, $d_{\mathbb{Z}^N}$ is the L^∞ -metric on $\mathbb{Z}^N$ defined by

$$d_{\mathbb{Z}^N}(\underline{x}, \underline{x}') = \max\{|x_1 - x'_1|, \dots, |x_N - x'_N|\}.$$

A **Convolutional Neural Network (CNN)** is a FFNN such that the first layers $V_0, \dots, V_i$ are convolutional layers and the final² layers $V_{i+1}, \dots, V_{r-1}$ are fully connected.

Remark 5. It is common to have pooling layers following convolutional layers in a CNN. Pooling layers downsample the size of the grids. They can be used in TCNNs in the same way and for the same purposes as traditionally used in CNNs. We do not discuss pooling in this paper for simplicity.

²Usually, there are 1 to 3 fully connected layers.

The correspondence $C_{d,N}(s)^{-1}$ maps a vertex to a vertex set that is localized with respect to the threshold s . In a 2-dimensional image, the above definition gives the typical square convolutional filter. This constructions results in spatially meaningful graph edges.

The graph structures given in Definitions 1, 3, and 4 yield the skeleton of a CNN. To pass data through the network, we need a system of functions on the nodes and edges of Γ that pass data from layer $i - 1$ to layer i . These functions are called **activations** and **weights**. The weights are real numbers $\lambda_{v,w}$ associated to each edge,

$$\Lambda = \{\lambda_{v,w} \mid v \in V_{i-1}, w \in V_i, \text{ and } (v, w) \in \Gamma\}.$$

Let $V_{i-1} = \mathcal{X} \times \mathbb{Z}^N$ and $V_i = \mathcal{X}' \times \mathbb{Z}^N$, as in a CNN. Denote $v = (\kappa, \underline{x}) \in V_{i-1}$ and $w = (\kappa', \underline{x}') \in V_i$. Then the homogeneity of the weights, a characteristic of a CNN, is the translational invariance

$$\lambda_{(\kappa, \underline{x}), (\kappa', \underline{x}')} = \lambda_{(\kappa, \underline{x} + \underline{z}), (\kappa', \underline{x}' + \underline{z})}. \quad (3.1)$$

The activations $\mathcal{A} = \{(u_v, f_v) \mid v \in V(\Gamma)\}$ associate to each vertex v a real number u_v and a function $f_v : \mathbb{R} \rightarrow \mathbb{R}$. To pass data from layer $i - 1$ to layer i means to determine u_w for $w \in V_i$ from the values $u_v, v \in V_{i-1}$ via the formula

$$u_w := f_w \left(\sum_{v \in \Gamma^{-1}(w)} \lambda_{v,w} \cdot u_v \right).$$

The **activation functions** f_v in neural networks usually map $x \in \mathbb{R}$ to $0 < x < 1$ or $0 \leq x$. The output activations are a probability distribution on the output nodes of Γ , so they are non-negative real numbers. We use the activation function ReLU, defined as $f(x) = \max(0, x)$, for all non-terminal layers, and the softmax $\sigma(x_i) = e^{x_i} (\sum_j e^{x_j})^{-1} : i, j \in \{1 \dots n\}$, $x_i \in \mathbb{R}^+$ as the terminal layer activation function. We choose a common optimization method, adaptive moment estimation (Adam), to determine the back-propagation of our changes based on the computed gradients.

Figure 3.1 (a,b,c) displays an example of a weight and activation system for a convolutional layer from $V_0 = \mathcal{X}(1) \times \mathbb{Z}^2$ to $V_1 = \mathcal{X}(4) \times \mathbb{Z}^2$, where $\mathcal{X}(\eta)$ denotes a finite set of cardinality η . The correspondence is $C_c \times C_{d,2}(2) \subset V_0 \times V_1$, where $C_c = \mathcal{X}(1) \times \mathcal{X}(4)$ is fully connected and $C_{d,2}(2) \subset \mathbb{Z}^2 \times \mathbb{Z}^2$ localizes connections at L^∞ -distance 2. Panel (a) shows the input V_0 , each weight-matrix in (b) is a vector of coefficients from $\{\lambda_{v,w}\}$, (c) shows the resulting activations in V_1 .

3.3 Topological Convolutional Layers

In Section 3.3.1 we introduce new types of neural network layers that form our TCNNs for 2D image classification. In Section 3.3.2 we introduce layers for TCNNs used for video

classification. Section 3.3.3 demonstrates the connection between our constructions and Gabor filters.

3.3.1 2D Images

Locality in a typical convolutional neural network is a function of the L^∞ –distance between cells, which is specified by the correspondence $C_{d,2}$ (see Definition 4) in the case of a 2–dimensional image. We add novel, topological criteria to this notion of locality through metrics on topological manifolds. The general technique is described in the following remark.

Remark 6. *The layers defined in Definitions 8, 7, 10 11, are all given by an edge-defining correspondence C of the following general form.*

Let M be a manifold and let $\mathcal{X}, \mathcal{X}' \subset M$ be two discretizations of M , meaning finite sets of points. Let $V_i = \mathcal{X} \times \mathbb{Z}^N$ and $V_{i+1} = \mathcal{X}' \times \mathbb{Z}^N$ be successive layers in a FFNN. Fix a threshold $s \geq 0$. Let d be a metric on M . Define a correspondence $C(s) \subset \mathcal{X} \times \mathcal{X}'$ by

$$C(s)^{-1}(\kappa') = \{\kappa \in \mathcal{X} \mid d(\kappa, \kappa') \leq s\}$$

for all $\kappa' \in \mathcal{X}'$. Together with another threshold $s' \geq 0$, this defines a correspondence $C \subset V_i \times V_{i+1}$ by

$$C = C(s) \times C_{d,N}(s'),$$

where $C_{d,N}(s')$ is the convolutional correspondence from Definition 4. This means that

$$\begin{aligned} C^{-1}(\kappa', \underline{x}') &= C_S(s)^{-1}(\kappa') \times C_{d,2}(s')^{-1}(\underline{x}') \\ &= \{(\kappa, x) \in \mathcal{X} \times \mathbb{Z}^N \mid d_S(\kappa, \kappa') \leq s \text{ and } d_{\mathbb{Z}^N}(\underline{x}, \underline{x}') \leq s'\} \end{aligned}$$

for all $(\kappa', \underline{x}') \in \mathcal{X}' \times \mathbb{Z}^N$.

The first example we give is a layer that localizes with respect to position on a circle in addition to the usual L^∞ –locality in a convolutional layer. Let $S^1 = \{\kappa \in \mathbb{R}^2 \mid |\kappa| = 1\}$ be the unit circle in the plane $\mathbb{R}^2$. A typical discretization of S^1 is the set of n -th roots of unity $\mathcal{X} = \{e^{2\pi i k/n} \mid 0 \leq k \leq n-1\}$ for some $n \geq 1$.

Definition 7. *Let $\mathcal{X}, \mathcal{X}' \subset S^1$ be two discretizations of the circle. Let $V_i = \mathcal{X} \times \mathbb{Z}^2$ and $V_{i+1} = \mathcal{X}' \times \mathbb{Z}^2$ be successive layers in a FFNN. Fix a threshold $s \geq 0$.*

*The **circle correspondence** $C_S(s) \subset \mathcal{X} \times \mathcal{X}'$ is defined by*

$$C_S(s)^{-1}(\kappa') = \{\kappa \in \mathcal{X} \mid d_S(\kappa, \kappa') \leq s\}$$

for all $\kappa' \in \mathcal{X}'$, where the metric d_S is given by

$$d_S(\kappa, \kappa') = \cos^{-1}(\kappa \cdot \kappa') \text{ for } \kappa, \kappa' \in S^1.$$

We call V_{i+1} a **circle one layer (COL)** if, for some other threshold $s' \geq 0$, the edge defining correspondence $C \subset V_i \times V_{i+1}$ is of the form

$$C = C_S(s) \times C_{d,2}(s'),$$

where $C_{d,2}(s')$ is the convolutional correspondence from Definition 4. This means that

$$\begin{aligned} C^{-1}(\kappa', x', y') &= C_S(s)^{-1}(\kappa') \times C_{d,2}(s')^{-1}(x', y') \\ &= \{(\kappa, x, y) \in \mathcal{X} \times \mathbb{Z}^2 \mid d_S(\kappa, \kappa') \leq s \text{ and } d_{\mathbb{Z}^2}((x, y), (x', y')) \leq s'\} \end{aligned}$$

for all $(\kappa', x', y') \in \mathcal{X} \times \mathbb{Z}^2$.

Next, we define a layer that localizes weights with respect to a metric on the Klein bottle $\mathcal{K}$. See Figure 3.2 for a visualization of the nodes and weights. Recall that $\mathcal{K}$ is the 2-dimensional manifold obtained from $\mathbb{R}^2$ as a quotient by the relations $(\theta_1, \theta_2) \sim (\theta_1 + 2k\pi, \theta_2 + 2l\pi)$ for $k, l \in \mathbb{Z}$ and $(\theta_1, \theta_2) \sim (\theta_1 + \pi, -\theta_2)$. The construction uses an embedding $F_{\mathcal{K}}$ of $\mathcal{K}$ into the vector space of quadratic functions on the square $[-1, 1]^2$, motivated by the embedded Klein bottle observed in [71] and its appearance in the weights of CNNs as observed in [38]. An image patch in the embedded Klein bottle $F_{\mathcal{K}}(\theta_1, \theta_2)$ has a natural ‘orientation’ given by the angle θ_1 . Visually, one sees lines through the center of the image at angle $\theta_1 + \pi/2$; see the top right image in Figure 3.3. The embedding is given by

$$F_{\mathcal{K}}(\theta_1, \theta_2)(x, y) = \sin(\theta_2)(\cos(\theta_1)x + \sin(\theta_1)y) + \cos(\theta_2)Q(\cos(\theta_1)x + \sin(\theta_1)y), \quad (3.2)$$

where $Q(t) = 2t^2 - 1$. As given, $F_{\mathcal{K}}$ is a function on the torus, which is parameterized by the two angles θ_1 and θ_2 . It actually defines a function on $\mathcal{K}$ since it satisfies $F_{\mathcal{K}}(\theta_1, \theta_2) = F_{\mathcal{K}}(\theta_1 + 2k\pi, \theta_2 + 2l\pi)$ and $F_{\mathcal{K}}(\theta_1 + \pi, -\theta_2) = F_{\mathcal{K}}(\theta_1, \theta_2)$.

Definition 8. Let $\mathcal{X}, \mathcal{X}' \subset \mathcal{K}$ be two finite subsets of the Klein bottle. Let $V_i = \mathcal{X} \times \mathbb{Z}^2$ and $V_{i+1} = \mathcal{X}' \times \mathbb{Z}^2$ be successive layers in a FFNN. Fix a threshold $s \geq 0$.

The **Klein correspondence** $C_{\mathcal{K}}(s) \subset \mathcal{X} \times \mathcal{X}'$ is defined by

$$C_{\mathcal{K}}(s)^{-1}(\kappa') = \{\kappa \in \mathcal{X} \mid d_{\mathcal{K}}(\kappa, \kappa') \leq s\}$$

for all $\kappa' \in \mathcal{X}'$, where the metric $d_{\mathcal{K}}$ is defined by

$$d_{\mathcal{K}}(\kappa, \kappa') = \left(\int_{[-1,1]^2} (F_{\mathcal{K}}(\kappa)(x, y) - F_{\mathcal{K}}(\kappa')(x, y))^2 dx dy \right)^{\frac{1}{2}}$$

for $\kappa, \kappa' \in \mathcal{K}$.

We call V_{i+1} a **Klein one layer (KOL)** if, for some other threshold $s' \geq 0$, the edge defining correspondence $C \subset V_i \times V_{i+1}$ is of the form

$$C = C_{\mathcal{K}}(s) \times C_{d,2}(s'),$$

which means that

$$\begin{aligned} C^{-1}(\kappa', x', y') &= C_{\mathcal{K}}(s)^{-1}(\kappa') \times C_{d,2}(s')^{-1}(x', y') \\ &= \{(\kappa, x, y) \in \mathcal{X} \times \mathbb{Z}^2 \mid d_{\mathcal{K}}(\kappa, \kappa') \leq s \text{ and } d_{\mathbb{Z}^2}((x, y), (x', y')) \leq s'\} \end{aligned}$$

for all $(\kappa', x', y') \in \mathcal{X}' \times \mathbb{Z}^2$.

We define two other layers based on the circle and the Klein bottle (Definition 9). First, we define an embedding of the circle S^1 into the space of functions on $[-1, 1]^2$ by composing $F_{\mathcal{K}}$ with the embedding $S^1 \hookrightarrow \mathcal{K}$, $\theta \mapsto (\theta, \pi/2)$, i.e.

$$F_{S^1}(\theta)(x, y) := F_{\mathcal{K}}(\theta, \pi/2)(x, y) = \cos(\theta)x + \sin(\theta)y.$$

Now the idea is to build convolutional layers with fixed weights λ given by discretizations of $F_{S^1}(\theta)(x, y)$ and $F_{\mathcal{K}}(\theta_1, \theta_2)(x, y)$. This is motivated by [38] which showed that convolutional neural networks (in particular VGG16) learn the filters $F_{S^1}(\theta)$. Instead of forcing the neural networks to learn these weights, we initialize the network with these weights. Intuitively, this should cause the network to train more quickly to high accuracy. Moreover, we choose to fix these weights during training (gradient = 0) to prevent overfitting, which we conjecture contributes to our observed improvement in generalization to new data. We also use discretizations of the images $F_{\mathcal{K}}(\theta_1, \theta_2)$ given by the full Klein bottle as weights, motivated by the reasoning that the trained weights observed in [38] are exactly the high-density image patches found in [71] which cluster around the Klein bottle. These layers with fixed weights can be thought of as a typical pretrained convolutional layer in a network.

Definition 9. Let $M = S^1$ or $\mathcal{K}$ and let $\mathcal{X} \subset M$ be a finite subset. Let $V_i = \mathbb{Z}^2$ and $V_{i+1} = \mathcal{X} \times \mathbb{Z}^2$ be successive layers in a FFNN. Suppose V_{i+1} is a convolutional layer with threshold $s \geq 0$ (Definition 4). Then V_{i+1} is called a **Circle Features (CF) layer** or a **Klein Features (KF) layer**, respectively, if the weights $\lambda_{-, (\kappa, -, -)}$ are given for $\kappa \in \mathcal{X}$ by a convolution over V_i of the filter of size $(2s+1) \times (2s+1)$ with values

$$\text{Filter}(\kappa)(n, m) = \int_{-1+\frac{2m}{2s+1}}^{-1+\frac{2(m+1)}{2s+1}} \int_{-1+\frac{2n}{2s+1}}^{-1+\frac{2(n+1)}{2s+1}} F_M(\kappa)(x, y) dx dy$$

for integers $0 \leq n, m \leq 2s$.

In summary, we have the following. Both $\mathcal{K}$ and S^1 can be discretized into a finite subset $\mathcal{X}$ by specifying evenly spaced values of the angles. Given such a discretization $\mathcal{X}$, the convolutional layers in a TCNN have slices indexed by $\mathcal{X}$. Note that $\mathcal{X}$ has a metric induced by the embedding $F_{\mathcal{K}}$ and the L^2 -metric on functions on $[-1, 1]^2$. The **COL** and **KOL** layers are convolutional layers with slices indexed by $\mathcal{X}$ where all weights between slices whose distances in $\mathcal{X}$ are greater than some fixed threshold are forced to be zero. The **CF** and **KF** layers are convolutional layers with slices indexed by $\mathcal{X}$ and such that the weights are instantiated on the slice corresponding to $(\theta_1, \theta_2) \in \mathcal{K}$ to be the image $F_{\mathcal{K}}(\theta_1, \theta_2)$ discretized

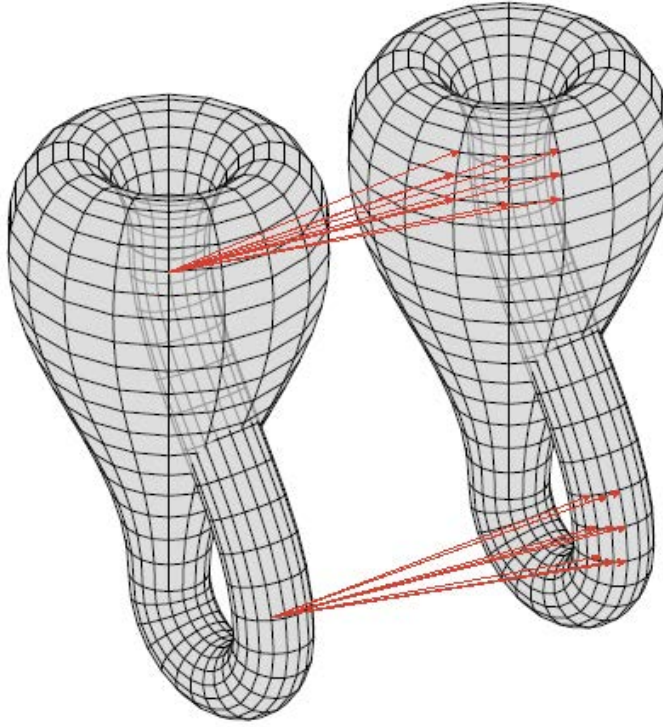


Figure 3.2: A visual representation of the neurons and weights in a KOL with threshold $s = 2$. Each intersection of grid lines on the two Klein bottles is a neuron. The red arrows depict the nonzero weights corresponding to the given input neuron.

to the appropriate grid shape; examples of these weights are shown in Figure 3.3. These weights are fixed during training. See also the visual guide in Figure 3.1.

3.3.2 Video

For video, the space of features of interest is parameterized by the tangent bundle of the translational Klein bottle $\mathcal{K}^t$, denoted $T(\mathcal{K}^t)$. The translational Klein bottle $\mathcal{K}^t$ is a 3-dimensional manifold and its tangent bundle $T(\mathcal{K}^t)$ is 6-dimensional. These manifolds parameterize video patches (3.3) in a manner related to the Klein bottle parameterization $F_{\mathcal{K}}$ of 2D images patches from (3.2).

Before providing the precise definitions of $\mathcal{K}^t$ and $T(\mathcal{K}^t)$ as well as the formulas for the parameterizations, we describe the idea roughly. An image patch in the embedded Klein bottle $F_{\mathcal{K}}(\theta_1, \theta_2)$ has a natural ‘orientation’ given by the angle θ_1 . Visually, one sees lines through the center of the image at angle $\theta_1 + \pi/2$. Given a real number r , there is a 2D image patch $F_{\mathcal{K}^t}(\theta_1, \theta_2, r)$ given by the translation of $F_{\mathcal{K}}(\theta_1, \theta_2)$ by r units along the line through the origin at angle θ_1 , i.e. along the line perpendicular to the lines in the image. One can extend this image to a video that is constant in time. Videos that change in time are obtained by enlarging $\mathcal{K}^t$ to its tangent bundle $T(\mathcal{K}^t)$. The tangent bundle consists of pairs of a point $(\theta_1, \theta_2, r) \in \mathcal{K}^t$ and a vector (u, v, w) tangent to (θ_1, θ_2, r) . The embedding $F_{T(\mathcal{K}^t)}$ sends such a pair to a video patch $F_{T(\mathcal{K}^t)}(\theta_1, \theta_2, r, u, v, w)$ that at time t is the image $F_{\mathcal{K}^t}(\theta_1 + tu, \theta_2 + tv, r + tw)$. For example, $F_{T(\mathcal{K}^t)}(\theta_1, \theta_2, 0, 0, 0, 1)$ is the video patch that translates $F_{\mathcal{K}}(\theta_1, \theta_2)$ at unit speed along the line through the origin at angle θ_1 . Similarly, $F_{T(\mathcal{K}^t)}(\theta_1, \theta_2, 0, 1, 0, 0)$ is the video patch that rotates $F_{\mathcal{K}}(\theta_1, \theta_2)$ at unit speed.

Precisely, we use the following construction. Denote the coordinates on $\mathbb{R}^3 \times \mathbb{R}^3$ by the variables $(\theta_1, \theta_2, r, u, v, w)$. The variables (θ_1, θ_2, r) parameterize $\mathcal{K}^t$ and the variables (u, v, w) parameterize the tangent spaces. To be precise, $\mathcal{K}^t$ is given as the quotient of $\mathbb{R}^3$ by the relations $(\theta_1, \theta_2, r) \sim (\theta_1 + 2k\pi, \theta_2 + 2l\pi, r)$ for all $k, l \in \mathbb{Z}$ and $(\theta_1, \theta_2, r) \sim (\theta_1 + \pi, -\theta_2, -r)$, and similarly $T(\mathcal{K}^t)$ can be described as a quotient of $\mathbb{R}^3 \times \mathbb{R}^3$. We suppress further discussion of these relations because they are only significant to this work in that they are respected by the embeddings $F_{\mathcal{K}^t}$ and $F_{T(\mathcal{K}^t)}$.

Let $I = [-1, 1]$. Denote by $C(I^2, I)$ the space of continuous functions $I^2 \rightarrow I$, which represent image patches at infinite resolution, and similarly denote the space of video patches by $C(I^2 \times I, I)$. The embeddings

$$F_{\mathcal{K}^t} : \mathcal{K}^t \rightarrow C(I^2, I)$$

and

$$F_{T(\mathcal{K}^t)} : T(\mathcal{K}^t) \rightarrow C(I^2 \times I, I)$$

are given by

$$F_{\mathcal{K}^t}(\theta_1, \theta_2, r)(x, y) = \sin(\theta_2)(\cos(\theta_1)(x + r \cos(\theta_1)) + \sin(\theta_1)(y + r \sin(\theta_1))) \\ + \cos(\theta_2)Q(\cos(\theta_1)(x + r \cos(\theta_1)) + \sin(\theta_1)(y + r \sin(\theta_1)))$$

and

$$F_{T(\mathcal{K}^t)}(\theta_1, \theta_2, r, u, v, w)(x, y, t) := F_{\mathcal{K}^t}(\theta_1 + tu, \theta_2 + tv, r + tw), \quad (3.3)$$

where $Q(z) = 2z^2 - 1$.

Using the embedding $F_{T(\mathcal{K}^t)}$, we define a metric on $T(\mathcal{K}^t)$ by pulling back the L^2 metric on $C(I^2 \times I, I)$,

$$d_{T(\mathcal{K}^t)}(\kappa, \kappa') := \left(\int_{I^2 \times I} (F_{T(\mathcal{K}^t)}(\kappa)(x, y, t) - F_{T(\mathcal{K}^t)}(\kappa')(x, y, t))^2 dx dy dt \right)^{\frac{1}{2}} \text{ for } \kappa, \kappa' \in T(\mathcal{K}^t). \quad (3.4)$$

The metric $d_{T(\mathcal{K}^t)}$ allows us to define a new type of layer in a neural network. Recall the 3D version of the convolutional correspondence $C_{d,3}(s)$ from Definition 4.

Definition 10. (6D Moving Klein Correspondence) *Let $\mathcal{X}, \mathcal{X}' \subset T(\mathcal{K}^t)$ be two finite subsets. Let $V_i = \mathcal{X} \times \mathbb{Z}^3$ and $V_{i+1} = \mathcal{X}' \times \mathbb{Z}^3$ be successive layers in a FFNN. Fix a threshold $s \geq 0$.*

*The **6D Moving Klein correspondence** $C_{T(\mathcal{K}^t)}(s) \subset \mathcal{X} \times \mathcal{X}'$ is defined by*

$$C_{T(\mathcal{K}^t)}(s)^{-1}(\kappa') = \{\kappa \in \mathcal{X} \mid d_{T(\mathcal{K}^t)}(\kappa, \kappa') \leq s\}$$

for all $\kappa' \in \mathcal{X}$, where the metric $d_{T(\mathcal{K}^t)}$ is defined in (3.4).

*We call V_{i+1} a **6D Moving Klein one layer (6MKOL)** if, for some other threshold $s' \geq 0$, the edge defining correspondence $C \subset V_i \times V_{i+1}$ is of the form*

$$C = C_{T(\mathcal{K}^t)}(s) \times C_{d,3}(s'),$$

which means that

$$C^{-1}(\kappa', x', y', t') = C_{T(\mathcal{K}^t)}(s)^{-1}(\kappa') \times C_{d,3}(s')^{-1}(x', y', t') \\ = \{(\kappa, x, y, t) \in \mathcal{X} \times \mathbb{Z}^3 \mid d_{T(\mathcal{K}^t)}(\kappa, \kappa') \leq s \text{ and } d_{\mathbb{Z}^3}((x, y, t), (x', y', t')) \leq s'\}$$

for all $(\kappa', x', y', t') \in \mathcal{X} \times \mathbb{Z}^3$.

There are particular submanifolds of $T(\mathcal{K}^t)$ whose corresponding video patches we conjecture to be most relevant for video classification. In 6MKOL layers, we often choose $\mathcal{X}$ and $\mathcal{X}'$ to be subsets of these submanifolds. One reason to do this is that discretizing the 6-dimensional manifold $T(\mathcal{K}^t)$ results in a large number of filters, significantly bloating the size of the neural network. Indeed, discretizing θ_1 and θ_2 into 4 values each, as in our Klein bottle experiments,

and discretizing the other dimensions into only 3 values produces $4^2 * 3^4 = 1296$ points in $\mathcal{X}$. Moreover, these are videos rather than static images, so they contain many pixels: 5^3 pixels for 5×5 video with 5 time steps. Another reason to specialize to submanifolds is one general philosophy of this paper: Well-chosen features, rather than an abundance of features, provide better generalization due to less over-fitting.

The five 2-dimensional submanifolds of $T(\mathcal{K}^t)$ that we choose to work with are

$$\begin{aligned}\tilde{\mathcal{K}} &:= \{(\theta_1, \theta_2, 0, 0, 0, 0) \in T(\mathcal{K}^t)\}, \\ S_\tau^\pm &:= \{(\theta_1, \theta_2, 0, 0, 0, \pm 1) \in T(\mathcal{K}^t)\}, \\ S_\rho^\pm &:= \{(\theta_1, \theta_2, 0, \pm 1, 0, 0) \in T(\mathcal{K}^t)\}.\end{aligned}\tag{3.5}$$

Under the embedding $F_{T(\mathcal{K}^t)}$, $\tilde{\mathcal{K}}$ corresponds to the Klein bottle images held stationary in time, $S_\tau^\pm$ corresponds to the Klein bottle images translating in time perpendicular to their center line, as described above, where the sign $\pm$ controls the direction of translation, and $S_\rho^\pm$ corresponds to the Klein bottle images rotating either clockwise or counterclockwise depending on the sign $\pm$.

We now define convolutional layers with fixed weights given by discretizations of the video patches corresponding to the manifolds $T(\mathcal{K}^t)$, $\tilde{\mathcal{K}}$, $S_\tau^\pm$, and $S_\rho^\pm$. These can be viewed as pretrained convolutional layers that would typically appear as the first layer in a pretrained video classifier. The motivation is the same as for the analogous CF and KF layers defined for images in Definition 9 – faster training and better generalization to new data due to initializing the network with meaningful filters that are fixed during training.

Definition 11. *Let $M \subset T(\mathcal{K}^t)$ be any subset, for example a submanifold such as $\tilde{\mathcal{K}}$, $S_\tau^\pm$, or $S_\rho^\pm$ (see (3.5)), or unions of such submanifolds. Let $\mathcal{X} \subset M$ be a finite subset.*

*Let $V_i = \mathbb{Z}^3$ and $V_{i+1} = \mathcal{X} \times \mathbb{Z}^3$ be successive layers in a FFNN. Suppose V_i is a convolutional layer with threshold $s \geq 0$ (i.e., the edge-defining correspondence is $C_c \times C_{d,3}(s)$ as in Definition 4). Then V_{i+1} is called a **M-Features (M-F) layer** if the weights $\lambda_{-,(\kappa,-,-,-)}$ are given for $\kappa \in \mathcal{X}$ by a convolution over V_i of the filter of size $(2s+1) \times (2s+1) \times (2s+1)$ with values*

$$\text{Filter}(\kappa)(n, m, p) = \int_{-1+\frac{2p}{2s+1}}^{-1+\frac{2(p+1)}{2s+1}} \int_{-1+\frac{2m}{2s+1}}^{-1+\frac{2(m+1)}{2s+1}} \int_{-1+\frac{2n}{2s+1}}^{-1+\frac{2(n+1)}{2s+1}} F_{T(\mathcal{K}^t)}(\kappa)(x, y, t) dx dy dt$$

for integers $0 \leq n, m, p \leq 2s$.

3.3.3 Gabor filters versus Klein bottle filters

The Klein bottle filters given by $F_{\mathcal{K}}(\theta_1, \theta_2)$ as in (3.2) are related to Gabor filters. In fact, besides a minor difference, they are a particular type of Gabor filters. The purpose of this section is to explain this relationship. The significance of this relationship is that,

while Gabor filters are commonly used in image recognition tasks, our constructions use a particular 2-parameter family of Gabor filters that is especially important, as identified by the analysis in [71]. Restricting to this 2-parameter family provides a compact set of filters that can be effectively used as pretrained weights in a neural network. One may use other families of Gabor filters for this purpose, but then the question is on what basis does one choose a particular family. The Klein bottle filters are a topologically justified choice. In Section 3.4.1, we compare the performance of some other choices of Gabor filters with the Klein bottle filters.

Recall that the Gabor filters are functions on the square $[-1, 1]^2$ given by

$$\begin{aligned} g(\lambda, \omega, \psi, \sigma, \gamma)(x, y) &= e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} \times \cos\left(\frac{2\pi x'}{\lambda} + \psi\right), \\ x' &= x \cos(\omega) + y \sin(\omega), \\ y' &= -x \sin(\omega) + y \cos(\omega). \end{aligned} \tag{3.6}$$

The Klein bottle filters $F_{\mathcal{K}}(\theta_1, \theta_2)$ do not taper off in intensity near the edges of the square, so we would like to remove this tapering from the Gabor filters too for a proper comparison. This is done by removing the exponential term from g , or equivalently, setting $\sigma = \infty$. This simultaneously removes the dependence on γ . So we have the restricted class of filters

$$g(\lambda, \omega, \psi, \infty, 0)(x, y) = \cos\left(\frac{2\pi}{\lambda} x'(\omega) + \psi\right).$$

Given θ_1, θ_2 parameterizing the Klein bottle filter $F_{\mathcal{K}}(\theta_1, \theta_2)$, we claim that a similar Gabor filter is given by

$$\begin{aligned} \tilde{g}(\theta_1, \theta_2) &:= g\left(2 + \frac{4}{\pi}\theta_2, \theta_1, \theta_2 + \pi, \infty, 0\right) \\ &= -\cos\left(\frac{\pi \cdot x'(\theta_1)}{1 + \frac{2}{\pi}\theta_2} + \theta_2\right). \end{aligned}$$

To see the similarity between $\tilde{g}(\theta_1, \theta_2)$ and $F_{\mathcal{K}}(\theta_1, \theta_2)$, we examine the ‘primary circle’ $\theta_2 = \pi/2$ and the ‘small circle’ $\theta_2 = 0$. The formula for other values of θ_2 interpolates along the Klein bottle between these two circles. On the primary circle, we have

$$\begin{aligned} \tilde{g}(\theta_1, \pi/2) &= \sin\left(\frac{\pi}{2} \cdot x'(\theta_1)\right) \\ F_{\mathcal{K}}(\theta_1, \pi/2) &= x'(\theta_1). \end{aligned}$$

These are both odd functions of $x'(\theta_1)$ that are equal to ± 1 at $x'(\theta_1) = \pm 1$ and are equal to 0 at $x'(\theta_1) = 0$. Similarly, on the small circle, we have

$$\begin{aligned}\tilde{g}(\theta_1, 0) &= -\cos\left(\pi \cdot x'(\theta_1)\right) \\ F_K(\theta_1, 0) &= 2\left(x'(\theta_1)\right)^2 - 1.\end{aligned}$$

These are both even functions of $x'(\theta_1)$ that are equal to 1 at $x'(\theta_1) = \pm 1$ and are equal to -1 at $x'(\theta_1) = 0$.

3.4 2D Images

3.4.1 Experiments and Results

We conduct several experiments on the image datasets described in Section 3.4.1. On individual datasets, we investigate the effect of Gaussian noise on training the TCNN Section 3.4.1, the interpretability of TCNN activations Section 3.4.1, and the learning rate of TCNNs in terms of testing accuracy over a number of batches in Section 3.4.1. Across different datasets, we investigate the generalization accuracy of TCNNs when trained on one dataset and tested on another in Section 3.4.1. We compare TCNNs to traditional CNNs in all of these domains. We also test other choices of Gabor filters versus Klein filters in the KF layers; see Section 3.4.1. All of these experiments use one or two convolutional layers, followed by a 3-layer fully connected network terminating in 10 or 2 nodes, depending on whether the network classifies digits or cats and dogs. For additional details on the neural network models, metaparameters, and train/test splits, see Section 3.4.2.

Description of Data

We perform digit classification on 3 datasets: MNIST [78], SVHN [89] and USPS [90]. These datasets are quite different from either other in style, while all consisting of images of digits 0 through 9. In particular, a human can easily identify to which dataset a particular image belongs. This makes generalization between the datasets a non-trivial task, because neural networks that train on one of the datasets will in general overfit to the particular style and idiosyncracies present in the given data, which will be inconsistent with different-looking digits from the other datasets. The datasets come at significantly different resolutions: 28^2 , 32^2 , and 16^2 , respectively. Additionally the sizes of the datasets vary widely: roughly $7e4$, $5e4$, and $7e3$, respectively. SVHN digits are typeset whereas MNIST and USPS are handwritten. MNIST and USPS are direct 2-D impressions with significant pre-processing already done, while the SVHN numbers are natural images of typeset digits with tilts, warping, presence of secondary digits, and other irregularities.

We also use two collections of labeled images of cats and dogs: Cats vs. Dogs [91] (which we call Kaggle), and the cat and dog labeled images from CIFAR-10, see [92]. Note that

the Kaggle Cats vs Dogs dataset, upon our download, included several images which were empty/corrupt and could not be loaded, so the size reported here is the total number of loadable images. This seems to be typical for this dataset. These datasets contain 2.5e4 and 1.2e4 images, respectively. The resolutions of the images in each dataset are 50^2 and 32^2 , respectively. Since we use these datasets to test the generalization accuracy of a neural network trained on one and tested on the other, we down-resolve the Kaggle data to 32^2 to be the same size as CIFAR-10.

Synthetic experiments

The central hypothesis of this work is that constraining CNNs to use the Klein bottle filters (see KF, Definition 9) and to train with respect to the Klein bottle topology on slices (see KOL, Definition 8) will provide the model with highly meaningful local features right away. So, we expect a model trained on our CF, KF, COL, and KOL layers to outperform conventional CNNs when global noise is added to the images. To test this idea, we add class-correlated Gaussian noise $N(\mu_k, \sigma_k^2)$ to our images, $\mathbf{z}_k = \mathbf{x}_k + \mathcal{E}_k$, $\mathcal{E}_k \sim N(\mu_k, \sigma_k^2)$, where $\mathbf{x}_k$ is an image of class k and $\mathcal{E}_k$ is a vector of the same dimension as the image, drawn from a normal distribution. The hyperparameters μ_k and σ_k^2 hyperparameters are drawn from $N(.2, .04)$ and $\chi^2(1) \times .04$ respectively. We test our classification models when trained on $\mathbf{z}$ and tested on $\mathbf{x}$ and we also test the converse. Also see the end of this section for some analysis of varying the mean of μ_k and distribution of σ_k^2 .

The results of this experiment on MNIST are displayed in Figure 3.4. A conventional two layer network’s performance is greatly deteriorated by the Gaussian noise added to the images in training or testing. In the case of noise added to the training set, we expect the CNN to learn the class-correlated μ_k, σ_k^2 , and then perform poorly on the test set without Gaussian noise. Note that Figure 3.4 includes model loss and it is clear that the 2-layer conventional CNN is classifying within the training set with high accuracy. We suspect that the TCNNs’ superior performance in the noisy training set experiment is due to the fixed filters in the CF and KF layers which force a smoothing of the noise prior to the learned layers and hence generate classifiers invariant under the global noise. A similar smoothing argument explains the TCNNs superior performance in the noisy test set experiment.

All four of the TCNNs outperform conventional CNNs on training and testing. Interestingly, while our CF based systems struggle to learn at first, after many epochs of training they are best at not fitting noise. Also it is an interesting observation that the KOL and COL systems seem to learn slower but add robustness to avoiding fitting noise in lengthy training. It is important to note that this class-correlated noise is a pathological example by design, and is meant primarily to show that the topological feature and convolution layers help to prevent fitting signals outside of the edges, angles, etc. that good models are expected to fit.

We now choose parameters τ and ω from 0 to .8 in increments of .2 and sample $\mu_k \sim N(\tau, .04)$ and $\sigma_k^2 \sim \chi^2(1) \times \omega^2$. In the first column of Figure 3.5, we simulate distributions $\mu_k \sim N(\tau, .04)$ and $\sigma_k^2 \sim \chi^2(1) \times .04$ testing the parameter τ from 0 to .8 in increments of .2.

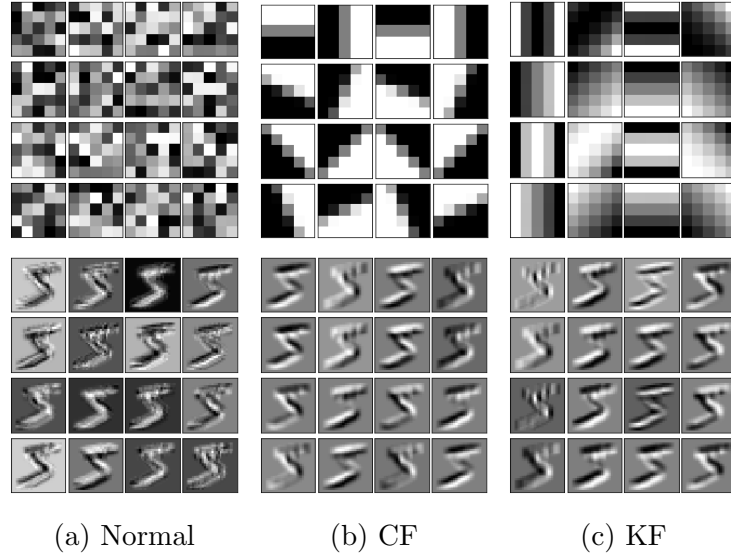


Figure 3.3: Table of weights (top row) and activations (bottom row) for 3 convolutional layers (a),(b),(c) evaluated on a handwritten 5 from MNIST (initial data shown in Figure 3.1). The first column (a) is a normal CNN layer (NOL) trained on one epoch of MNIST data followed by 2 fully connected linear layers. This network has a testing accuracy of approximately 99%. Second (b) is a Circle Filters (CF) layer. Third (c) is a Klein Filters (KF) layer.

Table 3.1: Size, dimensions and sources of all image datasets used in this work.

Dataset	Size	Dimensions	Link
SVHN	60000	32^2	http://ufldl.stanford.edu/housenumbers
USPS	9298	16^2	https://web.stanford.edu/~hastie/StatLearnSparsity_files/DATA/zipcode.html
MNIST	70000	28^2	http://yann.lecun.com/exdb/mnist/
CIFAR-10	12000	32^2	https://www.cs.toronto.edu/~kriz/cifar.html
Kaggle	24946	64^2	https://www.kaggle.com/c/dogs-vs-cats

In the second column, we simulate distributions $\mu_k \sim N(.2, \omega^2)$ and $\sigma_k^2 \sim \chi^2(1) \times \omega^2$ testing ω from 0 to .8 in increments of .2.

First we examine the top row of Figure 3.5 in which we train on data with Gaussian noise and test on the original data while varying τ and ω^2 . When τ is 0, the noise added has little class correlated signal (μ_k are small), but since ω^2 is still .04, there is an uncorrelated noise component. This noise deteriorates accuracy in all models. When τ grows, the TCNN trains significantly better than the normal CNN. When ω^2 is low, the models all perform similarly, and when ω^2 is large, all models perform poorly. However, there is a wide range of values of ω^2 for which the TCNN dramatically outperforms the CNN.

Next we examine the bottom row in which we train on the original MNIST training data and test on the Gaussian-noisy test data. Varying τ has little impact on the noisy test-set accuracy in the TCNN. As ω^2 grows in the noisy test-set experiment, it increases the random component of the noise, which degrades both classifiers as expected. Again, there is a significant portion of the range of ω^2 for which the TCNN is far superior to the CNN. These results are expected since we fix the convolutional weights in a sensible (topological) configuration in the KF layer. This forces the learned layers to classify on a smoothed version of the images which are more robust to the added noise.

Interpretability

We contrast the interpretability of CNNs and TCNNs by demonstrating the difference in activations between an NOL, CF, and KF (Figure 3.3) filtered image from MNIST. The activations shown in Figure 3.3 correspond to the output of the first layer of CF, KF and NOL models. The NOL filters are empirically derived by training on the MNIST training data ($n = 6e10$). The 16 CF filters correspond to 16 evenly spaced angles on a circle and the 16 KF filters correspond to 4 evenly spaced values for each of 2 angles on the Klein bottle.

Observing the activations in Figure 3.3, it is empirically clear that the activations of the CF and KF layers are easier to interpret. In the case of CF, the filters are edges at various angles, and the activations show where each such orientation of an edge appears in the image. In the case of KF, filters containing interior lines are also included, and the activations reveal the presence of these interior lines in the image. The NOL learned filters and activations are, in comparison, difficult to interpret. While some of the filters seem to be finding edges, others have idiosyncratic patterns that make it difficult to assess how the combined output features contribute to a classification. The NOL model has an accuracy of approximately 99%, so further training on MNIST is unlikely to significantly alter these filters.

Rate of learning

We find significant jumps in accuracy in the KF + KOL and KF + COL networks compared with NOL + NOL, and importantly, the networks with a KF layer achieve high accuracy earlier in training than those without, suggesting potential applications to smaller datasets for which less training can be done. See the bar chart in Figure 3.6 for a comparison of

testing accuracy attained from training on only 1,000 images in each dataset. The SVHN dataset, which is much richer than MNIST, and the USPS dataset, which has much lower resolution than MNIST, received larger benefits from the use of a TCNN. This suggests that the benefit of the feature engineering in TCNNs is greatest when the local spatial priors are relatively weak or hidden.

Generalizability

We compare model generalizability between several models trained on either SVHN or MNIST and tested on the other, and similarly across the Kaggle and CIFAR cats vs. dogs datasets. See Figures 3.7, 3.8 for a comparison of testing accuracies throughout training. The TCNNs achieve decent generalization in both the digit and cat vs dog domains. Using the KF + KOL TCNN, 30% accuracy is achieved generalizing from MNIST to SVHN, and over 60% accuracy is achieved generalizing from SVHN to MNIST. Contrast this with the 10% generalization accuracy of the NOL + NOL conventional CNNs from MNIST to SVHN – the same as random guessing. Note that the addition of a pooling layer had negligible effect on these results. Similarly, TCNNs are better than CNNs at generalizing between the Kaggle and CIFAR cats vs. dogs datasets, however the difference is less dramatic than it is for digit classification. Here, the addition of pooling layers has no impact on the generalizability of CNNs, but provides further improvement on the generalizability of TCNNs.

Gabor filters versus Klein bottle filters

As described in Section 3.3.3, the Klein bottle filters given by the embedding F_K in (3.2) can be viewed roughly as a subset of the 5-parameter family of Gabor filters given by (3.6). Given any subset of Gabor filters, one can make a convolutional layer analogous to the KF layer (Definition 9) where the filters are initialized to the chosen Gabor filters and frozen throughout training. In this section, we compare a KF layer with a layer instantiated with another choice of Gabor filters, given by the parameters below in (3.7) and pictured in Figure 3.9. While the 2-parameter family of Klein bottle filters is topologically justified as important for image analysis, we do not have any other means for selecting a different family of Gabor filters, and hence the 2-parameter family in (3.7) is necessarily somewhat arbitrary. Our heuristic motivation for this choice is that these are high contrast filters that are sparsely sampled from the two angles ψ and ω . The fixed parameters σ , λ , and γ are experimentally chosen so that the resulting 3×3 filters are high contrast. The variable parameter ω plays the same role of rotating the filter as the parameter θ_1 in the parameterization F_K of the Klein bottle filters.

$$\sigma = 2\pi, \lambda = \pi, \gamma = \frac{\pi}{8}, \psi \in \left\{ \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4} \right\}, \omega \in \left\{ \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4} \right\}. \quad (3.7)$$

Using these specific Gabor filters we create a ‘Gabor’ convolutional layer in the same fashion as the KF topological layer except with these Gabor filters as weights instead of the Klein

filters. We train a Gabor + NOL convolutional network on the datasets considered in this paper and compare its accuracy to a KF + NOL network and a standard NOL + NOL network. See Figure 3.10. We find that the Klein bottle filters perform similarly or better than the Gabor filters. This suggests that the Klein bottle filters are a good choice of Gabor filters to use in the topological layers of a TCNN, as hypothesized. Note again that some highly restrictive choice of Gabor filters must be made for this type of construction since they come in a large 5-parameter family. Further, several of the parameters are difficult to interpret in the context of the typically small kernels of CNNs.

3.4.2 Details of methods

Train/test splits

With regard to train/test splits, we perform two different types of experiment: (1) accuracy of TCNNs and CNNs are compared on a fixed dataset, and (2) either a TCNN or a CNN is trained on one dataset and validated on a second dataset to measure the capacity of the model for generalization. For type (1), we split the dataset into training and testing sets. For type (2), we train and test on the entire datasets. For type (1), the train/test splits are as follows:

For type (2), the dimension of the images is determined by the lowest resolution in the comparison, i.e., we down-resolve the higher resolution images to the lower resolution to be able to simply test generalizability.

Metaparameter selection

We choose metaparameters for each experiment that allow us to test TCNN performance against traditional CNN performance and do not seem to favor any particular model in the experiment with regard to the question at hand. Our strategy in comparing methods is to simply pick a relatively conventional, simple set of network specifications. We have selected an optimizer, batch size, learning rate, etc. on the criteria that all models are able to reasonably traverse the loss function on which they are optimized. We select a single configuration for all models and apply this consistently throughout the experiments. While of course the precise results fluctuate depending on these choices, our findings are generally consistent no matter the configurations we chose. We expect that, since our adaptations are within the typical CNN construction, meta-parameter selection has little effect on the relative impact of choosing a TCNN over a conventional CNN.

We provide the precise metaparameter specifications and some additional detail on our reasoning below. Within each Figure in the paper, the metaparameters are the same across all models presented. The following table lists various metaparameters by Figure number.

Conv-layers. The number of convolutional layers was chosen to be 2 for all experiments outside of Figure 3.3. Using 2 convolutional layers allows us to incorporate a feature layer

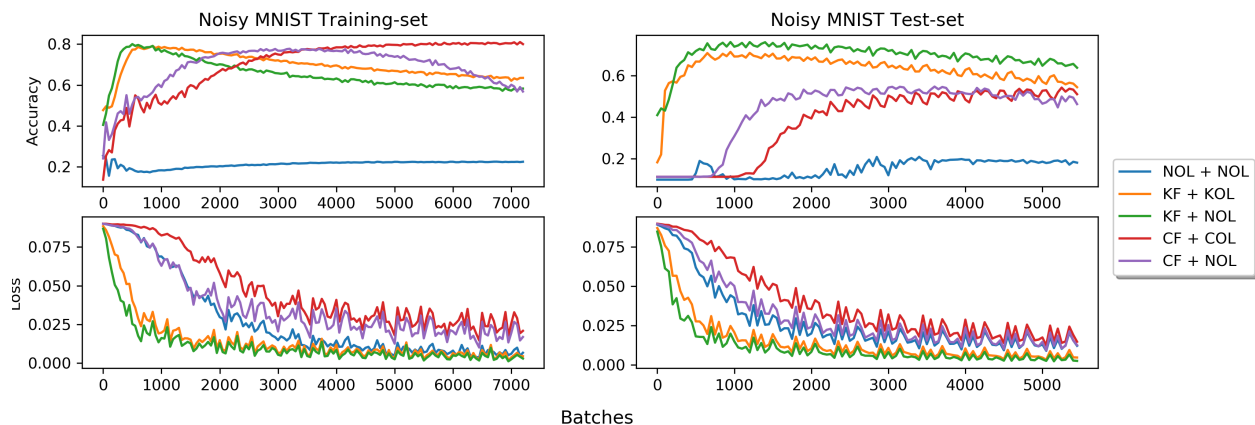


Figure 3.4: Two synthetic experiments on noisy MNIST data, designed to test model assumptions. The first column displays the results of the experiment where Gaussian noise is added to the training data but not the testing data. The second column displays the results of the experiment where the training data is the original MNIST data and the testing data are corrupted by Gaussian noise. The first row is testing accuracy and the second row is training loss.

Table 3.2: Test and traing splits for numeric image datasets.

Dataset	Train	Test
MNIST	85%	15%
SVHN	80%	20%
USPS	80%	20%

Table 3.3: Metaparameters corespondent to figures in this work.

Figure	Conv-layers	Conv-slices	Kernel size	LR	Batch size	Epochs
3.3	1	16	5	1e−4	100	1
3.4 and 3.5	2	64	3	1e−5	100	5
3.6	2	64	3	1e−4	100	1
3.7 and 3.8	2	64	3	1e−5	100	5

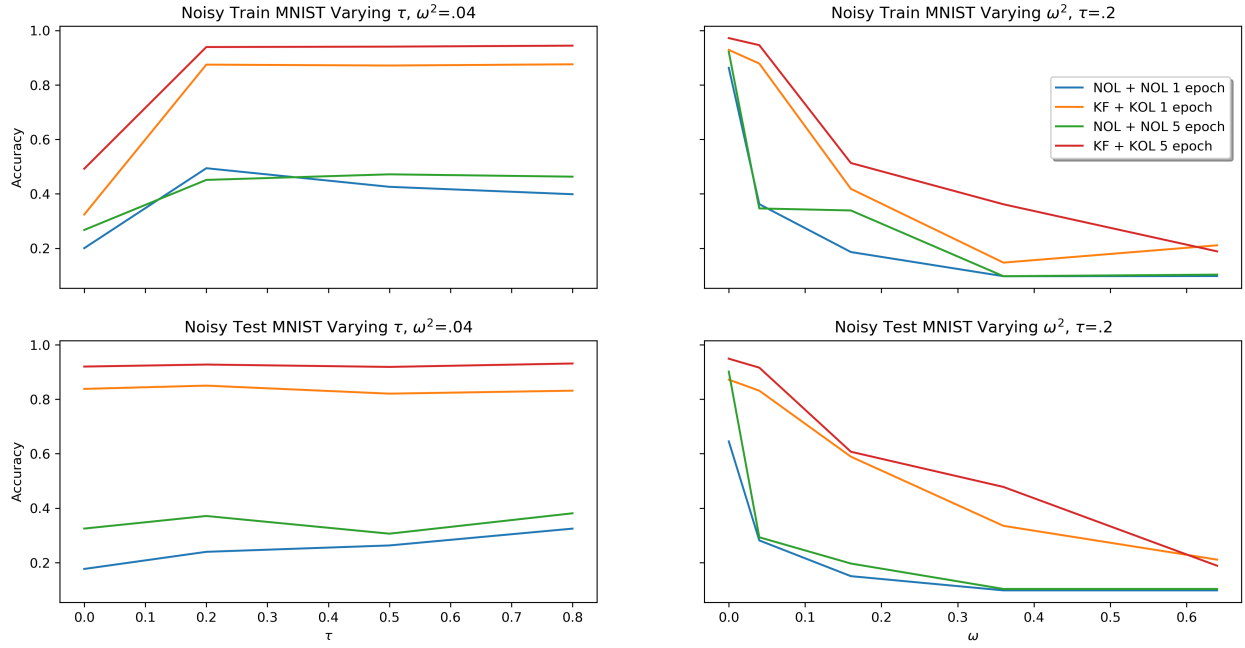


Figure 3.5: Sweep of distributions for synthetic MNIST data varying the mean and variance from which class-noise distributions are drawn. The first column shows the tested values of τ and the second column shows the tested values of ω^2 . The first row shows accuracies when training on data with Gaussian noise and testing on unaltered test set. The second row shows accuracies when training on unaltered training data and testing on noise-added test data. We show results after 1 epoch and 5 epochs of training.

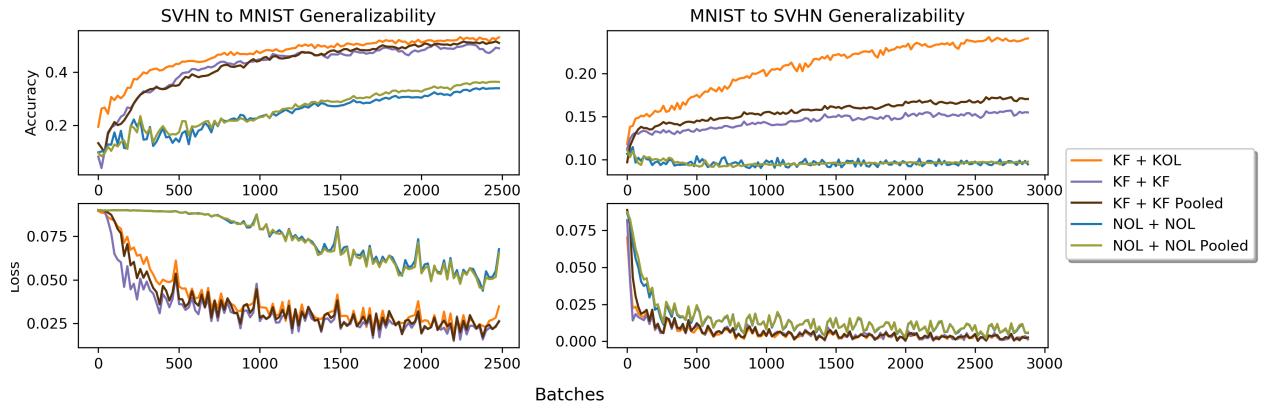
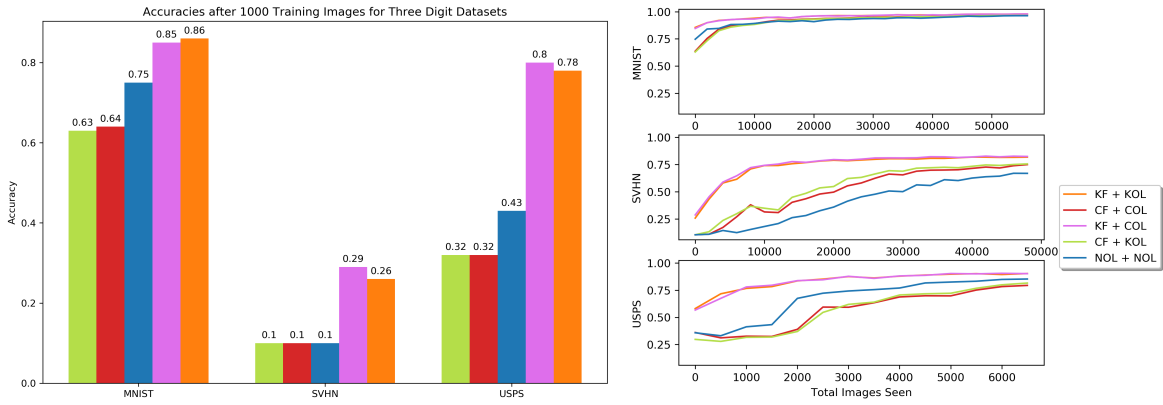


Figure 3.7: Comparisons of testing accuracy and validation loss when generalizing from SVHN to MNIST and vice versa.

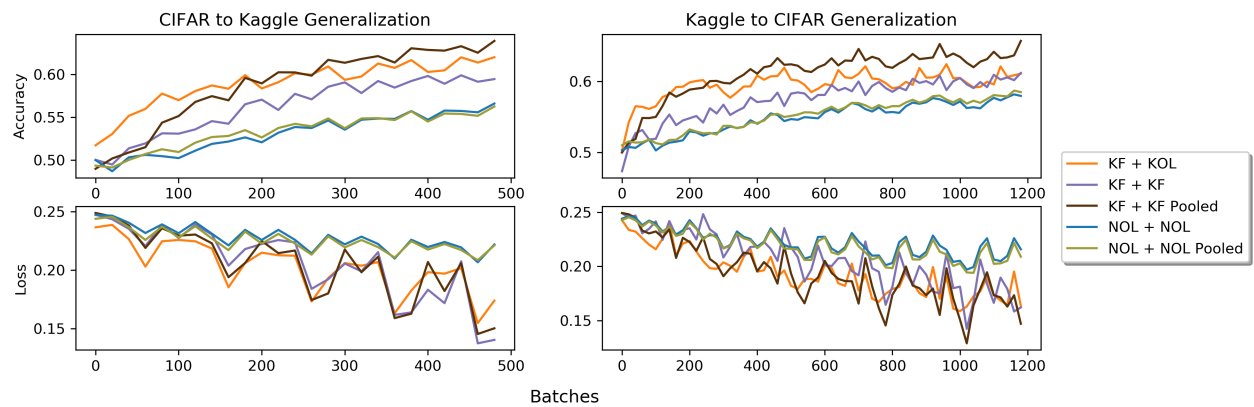


Figure 3.8: Comparisons of testing accuracy and validation loss when generalizing from CIFAR to Kaggle and vice versa.

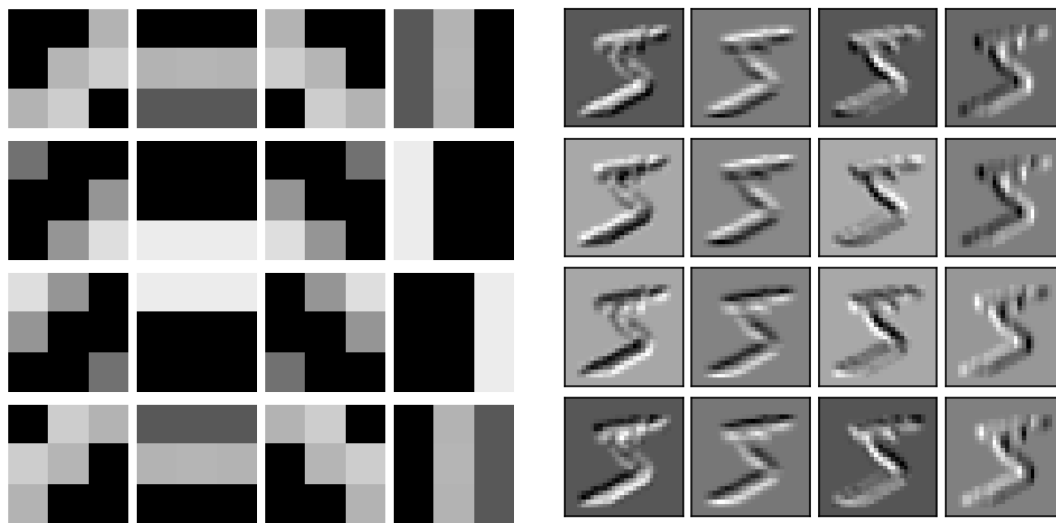


Figure 3.9: Example of a small set of gabor filters and activations on a handwritten 5 from MNIST.

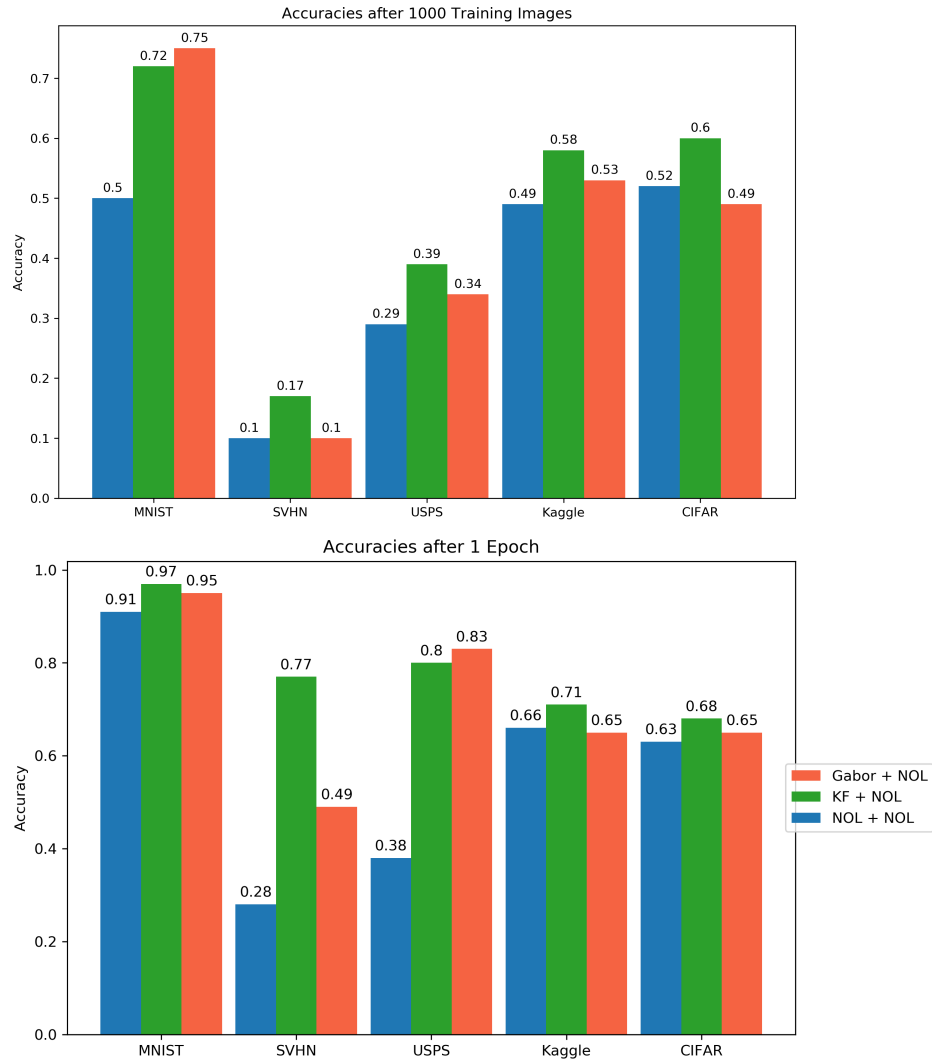


Figure 3.10: Top panel: testing accuracies on 3 digit datasets and 2 cats and dogs datasets after 100 training images. Bottom panel: testing accuracies after 1 full epoch of training. The KF model consistently performs similarly or better than the Gabor filters.

and a correspondence layer together, e.g. KF+KOL. We consistently use 2 convolutional layers for uniformity throughout the paper.

Conv-slices. The number of slices in each convolutional layer was chosen to be 64 for all experiments outside of Figure 3.3.

Kernel size. Figure 3.3 contains the only experiments in which we use a kernel size of 5. This is strictly for visualization purposes, as it allows the reader to see a more nuanced picture of the circle and Klein features. All other experiments use a kernel size of 3 in each convolutional layer.

Learning rate (LR). An LR of $1e-5$ or $1e-4$ is used in each experiment. We simply choose the highest power of ten where none of the models exhibit pathological behavior.

Batch size. We use a batch size of 100 throughout all experiments.

Epochs. We do 5 epochs of training for questions of generalization and 1 epoch for questions of training speed. We choose 5 epochs for questions of generalization because training over many epochs is common in applications and often results in greater potential for over-fitting. Over-fitting should reduce generalizability, hence we trained long enough to allow for this possibility. 5 epochs allows us to compare the rate of over-fitting at different stages of training. In the cases of training speed, 1 epoch is sufficient to illustrate the comparisons since the primary interest is in the accuracy and loss over the first few batches. 1 epoch is used to create Figure 3.3 out of convenience.

Fully connected layers. All experiments use 2 fully connected layers following a flattening layer. The flattening layer simply flattens the outputs of the final convolutional layer into a 1D vector. The first fully connected layer has 512 nodes, and the second has nodes of cardinality equal to the number of classes in the output.

Training

We processed images in 100 image batches. Computing was performed on an AMD 2990WX CPU with 128GB RAM and an NVIDIA RTX 2080TI GPU.

3.5 Video

3.5.1 Experiments and Results

We conduct several experiments on the datasets described in Section 3.5.1. On the UCF-101 dataset, we compare the learning rate of a TCNN in terms of testing accuracy over a number of batches in Section 3.5.1 and we compare it to the learning rate of a CNN. We also note that the TCNN achieves 70% accuracy after 100 epochs compared to 55% accuracy of the CNN. We also investigate the generalization accuracy of TCNNs when trained on the KTH dataset and tested on the Weizmann data in Section 3.5.1.

Description of Data

We use three datasets of videos: UCF-101 [81], KTH [82], and Weizmann [83]. The UCF-101 dataset consists of 13320 videos in 101 classes of human actions, such as Baby Crawling, Playing Cello, Tennis Swing, etc. There are a large variety of camera motions, object scales and viewpoint, background, and lighting conditions. The videos are between 2 and 15 seconds long. The KTH dataset consists of 2391 videos of 6 types of human actions (walking, jogging, running, boxing, hand waving, and clapping). Each video is of one of 25 different human subjects in one of a few different settings, e.g. outdoors and indoors, for an average length of 4 seconds. This is a significantly simpler dataset than UCF-101 due to the controlled nature of the videos e.g. backgrounds are homogeneous and the human actor is the only moving object.

The Weizmann dataset is used to test of generalization of a network trained on the KTH dataset. Only 3 Weizmann classes have a perfect analog in KTH: handwaving, running and walking. We test generalization on these 3 classes. Example frames from Weizmann and KTH, where their classes intersect, are provided in Figure 3.11. There are 29 Weizmann videos in the three classes. Both KTH and Weizmann have variable image lengths and KTH has half the frame rate of Weizmann. We choose to take about 1 second of each video at 25fps on which to perform classification.

Accuracy and rate of learning on a ResNet

We train 12 layer ResNet classifiers as in [65] on the UCF-101 dataset. One of these classifiers is a conventional CNN whose convolutional layers operate on all 3 dimensions of the video. The other classifier is a TCNN that is identical to the CNN except for the first layer in which we use a M-F layer (Definition 11) instead of a NOL. The M-F layer is of type $M = \tilde{\mathcal{K}} \cup S_\tau^\pm \cup S_\rho^\pm$ (see (3.5)), that is, we use the features given by translating the Klein bottle in time, rotating the Klein bottle in time, and holding the Klein bottle still in time – see the discussion above Definition 11.

Our main result is that, when using the M-F layer as the first layer instead of NOL, we find a significant increase in final accuracy as well as a much faster learning rate in the initial epochs; see Figure 3.12. In Figure 3.13, we plot the best class accuracy, best five, and best ten, as a function of training epochs.

We also perform the same experiment on the KTH dataset. The results are plotted in blue and green in Figure 3.14. Again, the TCNN achieves high accuracy more quickly and remains higher throughout all 200 epochs of training. The difference in final accuracy between the TCNN and the CNN is not as dramatic as we observe on the UCF-101 dataset, because the KTH dataset experiences less variation between videos, so it is easier for the CNN to learn meaningful features without the help of the M-F layer.

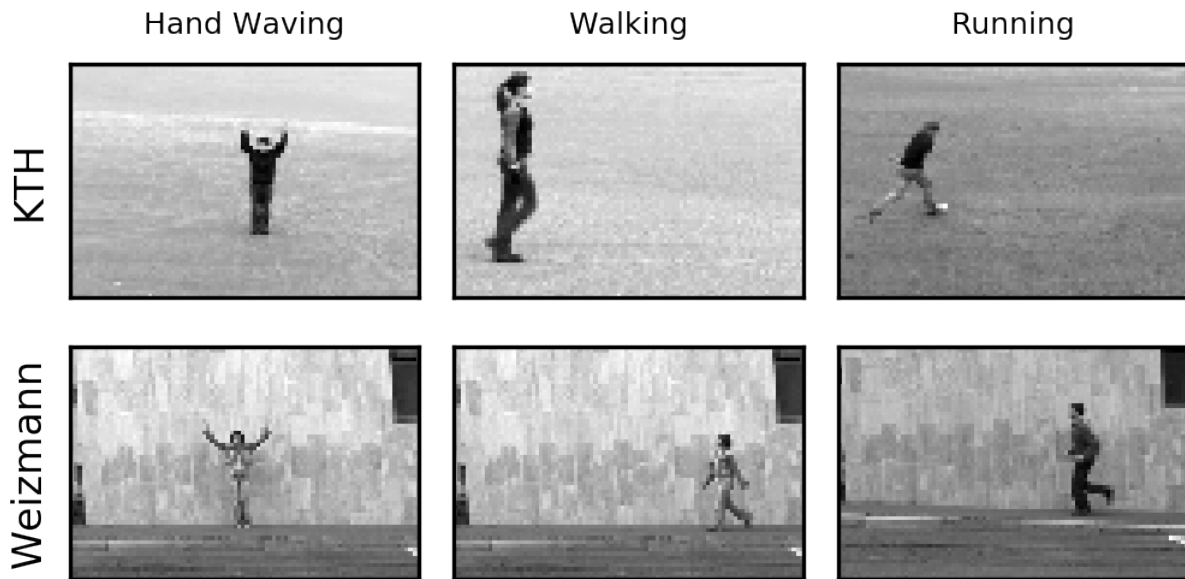


Figure 3.11: Examples of one frame from Weizmann (top row) and KTH (bottom row) videos from the same classes (columns).

Table 3.4: Video dataset metadata.

Dataset	Size	Dimensions	Link
UCF-101	13320	320x240x25fps	https://www.crcv.ucf.edu/data/UCF101.php
KTH	2391	160x120x25fps	https://www.csc.kth.se/cvap/actions/
Weizmann	29	180x144x50fps	http://www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html

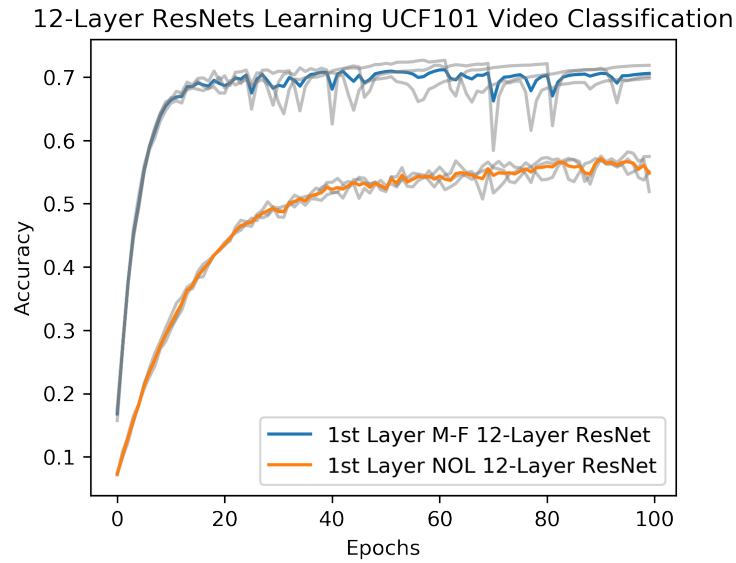


Figure 3.12: Testing accuracy of a 12 layer ResNet trained on UCF-101.

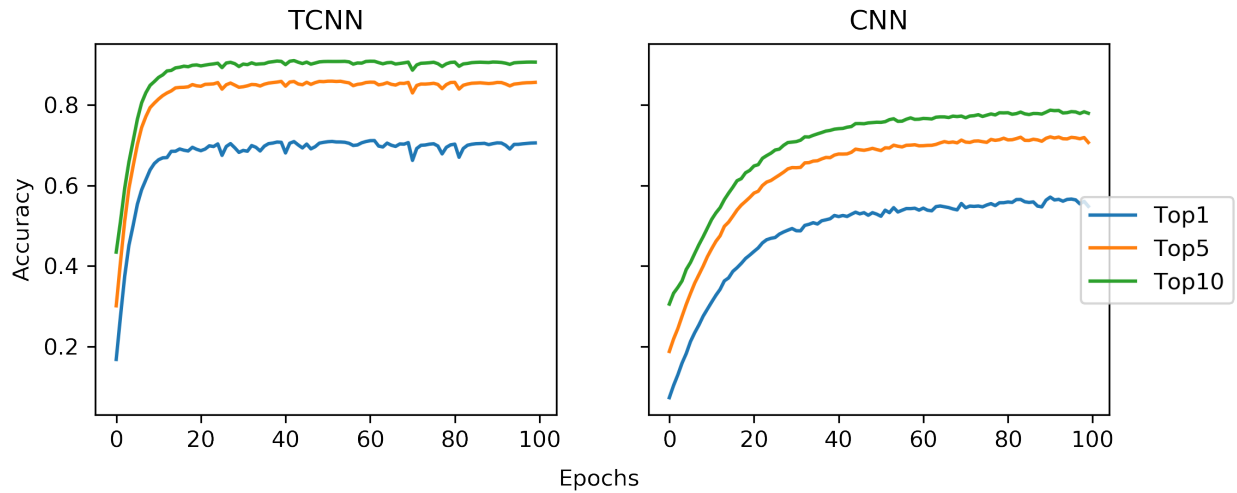


Figure 3.13: Top 1, 5, and 10 testing overall classification accuracies as a function of epochs for both the TCNN ResNet (left) and the CNN ResNet (right).

Generalizability

We compare generalizability of a TCNN and a CNN model trained on the KTH dataset and tested on the Weizmann dataset. Both of these models are ResNets, and the TCNN is the same as the CNN except with the first layer a M-F layer instead of a NOL layer, as described in Section 3.5.1. The TCNN generalizes from KTH to Weizmann with a testing accuracy of about 65%, whereas the CNN achieves only about 52% generalization accuracy.

The full results are shown in Figure 3.14. Observe that the generalization accuracy of the CNN takes a large dive at around 50 epochs, which suggests that it is initially learning features specific to the KTH dataset that do not generalize well, and then as it continues to train it learns more meaningful features that do generalize. We conjecture that the dip in accuracy on the Weizmann dataset occurs during the transition from learning KTH-specific artifacts to learning meaningful features. On the other hand, the TCNN immediately begins learning meaningful features that generalize well. This is the desired effect of the M-F layer – it biases the learner to use significant features in the video right away, instead of initially learning artifacts particular to the dataset.

3.5.2 Details of methods

Train/test splits

In our video experiments, we only have a train/test split for UCF-101. KTH and Weizmann videos were only used in the test of generalization, so for these datasets we use the entire set of KTH in training and the entire set of Weizmann to test (in the three common classes). The UCF-101 split is as follows:

Metaparameter selection

Our video experiments differ from image experiments in that we use kernels which are not square (cubic). To indicated the size of a 3-dimensional kernel, we use the notation (t, y, x) , where t is the size in the time dimension, and x, y are spatial sizes in their respective dimensions.

Notably, the ResNet12 residual network which produced our highest accuracy classifiers for UCF-101 (Figures 3.12 and 3.13), has a significantly different form from all other models in this work, both in depth and in the use of a residual block. We choose a simple block architecture: where each block has two convolutional layers with a kernel size ν . In order to add the input to the output, we need tensors of matching dimension, so we use $\text{Upsample}(\cdot)$ which is a trilinear upsampler and $\text{Batch}(\cdot)$ which is performs batch normalization. We indicate a 3-dimensional pooling layer by $\text{Pool}(\cdot)$. Our residual network then has the following structure:

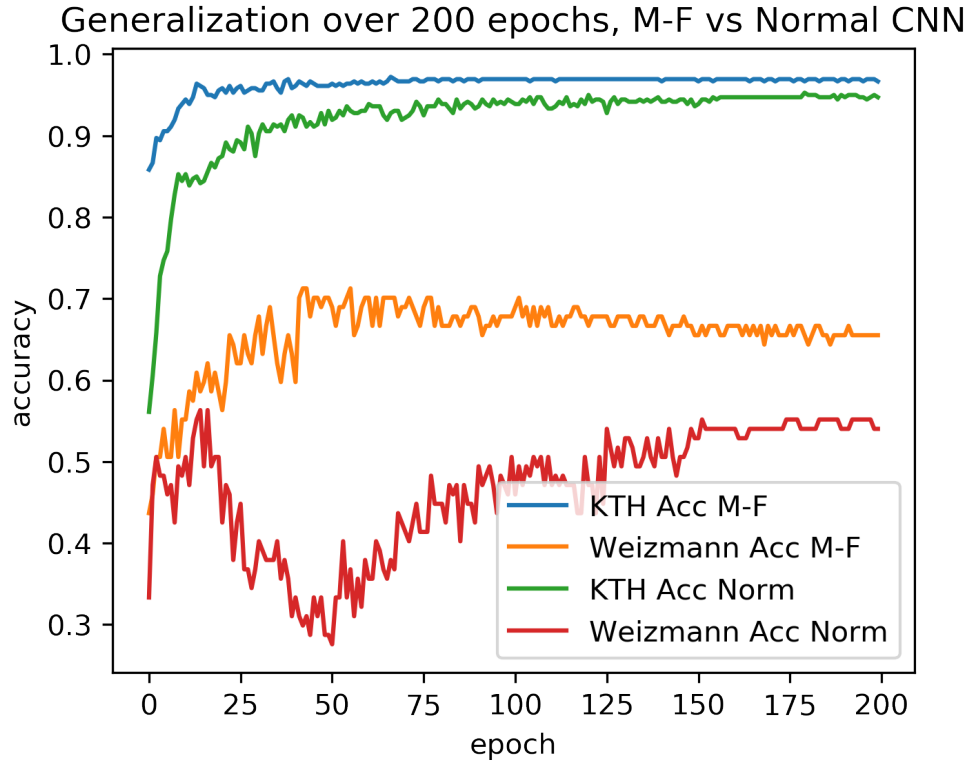


Figure 3.14: Testing accuracy of a TCNN and a CNN trained on the KTH dataset and tested on the Weizmann dataset.

Table 3.5: Train/test split for UCF-101.

Dataset	Train	Test
UCF-101	67%	33%

Table 3.6: Metaparameters for video experiment figures.

Figure	Conv-layers	Conv-slices	Kernel size	LR	Batch size	Epochs
3.14	3	20, 80, 160	$(5 \times 11 \times 11), (5 \times 11 \times 11), (3 \times 9 \times 4)$	$3e-3$	100	200
3.12 & 3.13	12	$180 \times 9, 360 \times 3$	$(5, 11, 11), (1, 5, 5), (1, 3, 3), (3, 3, 3)$	$1e-5$	100	5

Table 3.7: Residual block for our 12-convolutional layer ResNet where x is an input tensor, data passes from top to bottom, so $x + y$ is the returned tensor.

Block(ν, x):
$y = \text{Convolution}(\nu)(x)$
$y = \text{Convolution}(\nu)(y)$
$y = \text{Batch}(y)$
$y = \text{Upsample}(y)$
$x + y$

Table 3.8: Structure of our 12-convolutional layer ResNet with order of operations indicated by descending through the table. The first column of the table indicates the kernel size of a convolutional layer or layers in a residual block.

ν	ResNet12(x):
	$y = M - F_1(x)$
(1, 5, 5)	$y = \text{Block}(\nu, y)$
(1, 3, 3)	$y = \text{Block}(\nu, y)$
(3, 3, 3)	$y = \text{Pool}(C_d(\nu)(y))$
	$y = \text{Batch}(y)$
(3, 3, 3)	$y = C_d(\nu)(y)$
	$y = \text{Batch}(y)$
(3, 3, 3)	$y = \text{Pool}(C_d(\nu)(y))$
	$y = \text{Batch}(y)$
	$y = \text{FC}(y)$
	$y = \text{Batch}(y)$
	$y = \text{softmax}(y)$

Training

We processed videos in 50 (UCF-101) or 100 (KTH vs Weizmann) image batches. Computing was performed on an AMD 2990WX CPU with 128GB RAM and an NVIDIA RTX 2080TI GPU.

Conclusion

The work presented in this dissertation reviews, generalizes and extends methods of TDA. TDA is a promising field with several disparate branches, two of which are explored in this research. I present a hybrid, machine learned classifier for microbiological data which uses a persistence vectorization and bootstrapping. I also present a generalization of the CNN, which takes a manifold prior to partially specify its topology. Both of these studies are novel additions to the literature on TDA.

In Chapter 1, I provided a review of key background information for my work as well as a review of current state-of-the-art methods. I drew a rough arc from topological features to their persistent representations in data. I demonstrated general methods to create topological filtrations through the construction of simplicial complexes. I then covered the more specific background to the development of the methods presented in the following two chapters. Finally, I reviewed widely used contemporary applications of TDA as well as applications of TDA for neural networks, which is a new and rapidly developing corpus. I plan to include this work in a upcoming, larger review paper of contemporary applications of TDA.

In Chapter 2, I presented a typical application of TDA, to study the connectivity of intracellular structures. The state-of-the-art methods for studying the topology of actin networks are time consuming, and rely on significant preprocessing and human interjections, and result only in data loosely correlated with dependent variables [10]. I demonstrated the utility of TDA to discern between several classes of simulated cells. I demonstrated a novel method of data augmentation for TDA to improve these results. I demonstrated how to bridge the methods developed on synthetic data to the microscopy domain and analyze experimental data comparing cells with and without a genetic knockout. This work is the first time that actin networks have been studied using persistence homology. This is also the first fully automated algorithm for the study of filament networks. This work will be incorporated into a larger, collaborative submission with our partners in biology and engineering.

In Chapter 3, I proposed a theoretically and empirically motivated, novel generalization on the convolutional neural network (CNN). This framework draws on findings about the long-run topologies of well trained CNNs as well as theoretical knowledge about the space of natural images. I contrasted the empirical results of a variety of such formulations with each other and a typical CNN. This work serves as a template for the development of limitless models with augmented topologies based on empirical knowledge and/or theoretical

connections to a logical manifold. I demonstrated the efficacy of this generalization in several image and video classification experiments. An earlier version of this work was accepted as a conference paper at the Neural Information Processing Seminar, and the full paper is currently under review at the Journal of Machine Learning Research.

Bibliography

- [1] Frédéric Chazal and Bertrand Michel. An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists. *arXiv preprint arXiv:1710.04019*, 2017. [x](#), [5](#), [8](#), [16](#), [18](#)
- [2] Pek Y Lum, Gurjeet Singh, Alan Lehman, Tigran Ishkanov, Mikael Vejdemo-Johansson, Muthu Alagappan, John Carlsson, and Gunnar Carlsson. Extracting insights from the shape of complex data using topology. *Scientific reports*, 3:1236, 2013. [1](#)
- [3] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005. [1](#)
- [4] Gunnar Carlsson, Afra Zomorodian, Anne Collins, and Leonidas J Guibas. Persistence barcodes for shapes. *International Journal of Shape Modeling*, 11(02):149–187, 2005. [1](#)
- [5] Gunnar Carlsson, Gurjeet Singh, and Afra Zomorodian. Computing multidimensional persistence. In *International Symposium on Algorithms and Computation*, pages 730–739. Springer, 2009. [1](#)
- [6] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017. [1](#), [44](#), [46](#)
- [7] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*, 2018. [2](#)
- [8] Larry Wasserman. Topological data analysis. *Annual Review of Statistics and Its Application*, 5:501–532, 2018. [2](#), [4](#), [12](#)
- [9] Simon L Freedman, Shiladitya Banerjee, Glen M Hocky, and Aaron R Dinner. A versatile framework for simulating the dynamic mechanical structure of cytoskeletal networks. *Biophysical journal*, 113(2):448–460, 2017. [2](#), [24](#), [26](#), [27](#)
- [10] Takumi Higaki, Natsumaro Kutsuna, Toshio Sano, Noriaki Kondo, and Seiichiro Hasezawa. Quantification and cluster analysis of actin cytoskeletal structures in plant cells: role of actin bundling in stomatal movement during diurnal cycles in arabidopsis guard cells. *The Plant Journal*, 61(1):156–165, 2010. [2](#), [80](#)

- [11] Vasileios Maroulas and Andreas Nebenführ. Tracking rapid intracellular movements: a Bayesian random set approach. *The Annals of Applied Statistics*, 9(2):926–949, 2015. [2](#), [16](#)
- [12] Marian Gidea and Yuri Katz. Topological data analysis of financial time series: Landscapes of crashes. *Physica A: Statistical Mechanics and its Applications*, 491:820–834, 2018. [4](#)
- [13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and others. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. [4](#)
- [14] John Stillwell. *Classical topology and combinatorial group theory*, volume 72. Springer Science & Business Media, 2 edition, 1991. [4](#), [5](#)
- [15] Definition of persistence | Dictionary.com. [8](#)
- [16] Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008. [11](#)
- [17] Peter Bubenik, Vin De Silva, and Jonathan Scott. Metrics for generalized persistence modules. *Foundations of Computational Mathematics*, 15(6):1501–1531, 2015. [11](#), [22](#)
- [18] Andrew Marchese and Vasileios Maroulas. Signal classification with a point process distance on the space of persistence diagrams. *Advances in Data Analysis and Classification*, 12(3):657–682, 2018. [12](#), [14](#), [29](#), [44](#)
- [19] Andrew Marchese, Vasileios Maroulas, and Josh Mike. K- means clustering on the space of persistence diagrams. volume 10394, page 103940W. International Society for Optics and Photonics, 2017. [12](#), [37](#)
- [20] Vasileios Maroulas, Farzana Nasrin, and Christopher Oballe. A bayesian framework for persistent homology. *SIAM Journal on Mathematics of Data Science*, 2(1):48–74, 2020. [12](#), [14](#), [25](#), [39](#), [44](#)
- [21] Vasileios Maroulas, Cassie Putman Micucci, and Adam Spannaus. A stable cardinality distance for topological classification. *Advances in Data Analysis and Classification*, pages 1–18, 2019. [13](#), [22](#), [29](#), [31](#), [44](#)
- [22] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015. [13](#), [47](#)
- [23] Jacob Townsend, Cassie Putman Micucci, John H Hymel, Vasileios Maroulas, and Konstantinos D Vogiatzis. Representation of molecular structures with persistent homology for machine learning applications in chemistry. *Nature communications*, 11(1):1–9, 2020. [14](#)
- [24] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier.

- Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252, 2017. [14](#)
- [25] Marian Gidea. Topological data analysis of critical transitions in financial networks. pages 47–59. Springer, 2017. [14](#)
 - [26] Yuan Wang, Hernando Ombao, and Moo K Chung. Topological data analysis of single-trial electroencephalographic signals. *The annals of applied statistics*, 12(3):1506, 2018. [14](#), [44](#)
 - [27] Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. pages 4741–4748, 2015. [14](#), [16](#)
 - [28] Yuhei Umeda. Time series classification via topological data analysis. *Information and Media Technologies*, 12:228–239, 2017. [14](#), [22](#)
 - [29] Ioannis Sgouralis, Andreas Nebenfuühr, and Vasileios Maroulas. A bayesian topological framework for the identification and reconstruction of subcellular motion. *SIAM Journal on Imaging Sciences*, 10(2):871–899, 2017. [14](#), [44](#), [47](#)
 - [30] Le Yin, Ioannis Sgouralis, and Vasileios Maroulas. Topological reconstruction of subcellular motion with ensemble kalman velocimetry. *Foundations of Data Science*, 2019. [16](#)
 - [31] Jose A Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15(3):799–838, 2015. [16](#)
 - [32] Clément Thomas, Stéphane Tholl, Danièle Moes, Monika Dieterle, Jessica Papuga, Flora Moreau, and André Steinmetz. Actin bundling in plants. *Cell motility and the cytoskeleton*, 66(11):940–957, 2009. [17](#)
 - [33] Simon L Freedman, Shiladitya Banerjee, Glen M Hocky, and Aaron R Dinner. A versatile framework for simulating the dynamic mechanical structure of cytoskeletal networks. *Biophysical journal*, 113(2):448–460, 2017. [17](#), [19](#)
 - [34] Simon L Freedman, Glen M Hocky, Shiladitya Banerjee, and Aaron R Dinner. Nonequilibrium phase diagrams for actomyosin networks. *Soft matter*, 14(37):7740–7747, 2018. [19](#)
 - [35] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. [19](#)
 - [36] Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. Image processing with neural networks—a review. *Pattern recognition*, 35(10):2279–2301, 2002. [19](#)
 - [37] Zhenni Wang, Christine Di Massimo, Ming T Tham, and A Julian Morris. A procedure for determining the topology of multilayer feedforward neural networks. *Neural Networks*, 7(2):291–300, 1994. [19](#)

- [38] Gunnar Carlsson and Rickard Brüel Gabrielsson. Topological Approaches to Deep Learning. *arXiv:1811.01122 [cs, math, stat]*, November 2018. arXiv: 1811.01122. [19](#), [44](#), [46](#), [48](#), [49](#), [51](#), [54](#), [55](#)
- [39] Mathieu Carriere, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. A general neural network architecture for persistence diagrams and graph classification. 2019. [19](#)
- [40] Zixuan Cang and Guo-Wei Wei. Topologynet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions. *PLoS computational biology*, 13(7):e1005690, 2017. [22](#)
- [41] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252, 2017. [22](#), [29](#), [39](#)
- [42] Jen-Yu Liu, Shyh-Kang Jeng, and Yi-Hsuan Yang. Applying topological persistence in convolutional neural network for music audio signals. *arXiv preprint arXiv:1608.07373*, 2016. [22](#)
- [43] Jen-Yu Liu, Yi-Hsuan Yang, and Shyh-Kang Jeng. Weakly-supervised visual instrument-playing action detection in videos. *IEEE Transactions on Multimedia*, 21(4):887–901, 2018. [22](#)
- [44] Talha Qaiser, Yee-Wah Tsang, Daiki Taniyama, Naoya Sakamoto, Kazuaki Nakane, David Epstein, and Nasir Rajpoot. Fast and accurate tumor segmentation of histology images using persistent homology and deep convolutional features. *Medical image analysis*, 55:1–14, 2019. [23](#)
- [45] William H Guss and Ruslan Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *arXiv preprint arXiv:1802.04443*, 2018. [23](#)
- [46] Clément Thomas, Stéphane Tholl, Danièle Moes, Monika Dieterle, Jessica Papuga, Flora Moreau, and André Steinmetz. Actin bundling in plants. *Cell motility and the cytoskeleton*, 66(11):940–957, 2009. [24](#)
- [47] Andrew Marchese and Vasileios Maroulas. Topological learning for acoustic signal identification. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1377–1381. IEEE, 2016. [25](#)
- [48] Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G Escolar, Kaname Matsue, and Yasumasa Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016. [25](#)

- [49] Thomas Bonis, Maks Ovsjanikov, Steve Oudot, and Frédéric Chazal. Persistence-based pooling for shape pose recognition. In *International Workshop on Computational Topology in Image Context*, pages 19–29. Springer, 2016. [25](#)
- [50] Chunyuan Li, Maks Ovsjanikov, and Frederic Chazal. Persistence-based structural recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1995–2002, 2014. [25](#)
- [51] Francisco Belchi, Mariam Pirashvili, Joy Conway, Michael Bennett, Ratko Djukanovic, and Jacek Brodzki. Lung topology characteristics in patients with chronic obstructive pulmonary disease. *Scientific reports*, 8(1):5341, 2018. [25](#)
- [52] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011. [25](#)
- [53] Nikhil Singh, Heather D Couture, JS Marron, Charles Perou, and Marc Niethammer. Topological descriptors of histology images. In *International Workshop on Machine Learning in Medical Imaging*, pages 231–239. Springer, 2014. [25](#)
- [54] Yen-Chi Chen and Adrian Dobra. Measuring human activity spaces from gps data with density ranking and summary curves. *arXiv preprint arXiv:1708.05017*, 2017. [25](#)
- [55] Yen-Chi Chen et al. Generalized cluster trees and singular measures. *The Annals of Statistics*, 47(4):2174–2203, 2019. [25](#)
- [56] Thierry Sousbie, Christophe Pichon, and Hajime Kawahara. The persistent cosmic web and its filamentary structure–ii. illustrations. *Monthly Notices of the Royal Astronomical Society*, 414(1):384–403, 2011. [25](#)
- [57] Rien van de Weygaert, Erwin Platen, Gert Vegter, Bob Eldering, and Nico Kruithof. Alpha shape topology of the cosmic web. In *2010 International Symposium on Voronoi Diagrams in Science and Engineering*, pages 224–234. IEEE, 2010. [25](#)
- [58] Larry Wasserman. Topological data analysis. *Annual Review of Statistics and Its Application*, 5:501–532, 2018. [25](#), [29](#)
- [59] Simon L Freedman, Glen M Hocky, Shiladitya Banerjee, and Aaron R Dinner. Nonequilibrium phase diagrams for actomyosin networks. *Soft matter*, 14(37):7740–7747, 2018. [26](#), [27](#)
- [60] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010. [26](#), [29](#)
- [61] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015. [29](#)
- [62] Michael Kerber, Dmitriy Morozov, and Arnur Nigmatov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics (JEA)*, 22:1–4, 2017. [29](#)

- [63] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. [44](#), [46](#)
- [64] Dong-Ling Deng, Xiaopeng Li, and S Das Sarma. Quantum entanglement in neural network states. *Physical Review X*, 7(2):021021, 2017. [44](#)
- [65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. [44](#), [45](#), [46](#), [73](#)
- [66] Holger Schwenk and Maurice Milgram. Transformation invariant autoassociation with application to handwritten character recognition. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 992–998. MIT Press, 1995. [44](#)
- [67] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2224–2232. Curran Associates, Inc., 2015. [44](#)
- [68] Theo Lacombe, Marco Cuturi, and Steve OUDOT. Large scale computation of means and clusters for persistence diagrams using optimal transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9770–9780. Curran Associates, Inc., 2018. [44](#)
- [69] Roland Kwitt, Stefan Huber, Marc Niethammer, Weili Lin, and Ulrich Bauer. Statistical topological data analysis - a kernel perspective. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3070–3078. Curran Associates, Inc., 2015. [44](#)
- [70] Vasileios Maroulas, Joshua L Mike, and Christopher Oballe. Nonparametric estimation of probability density functions of random persistence diagrams. *Journal of Machine Learning Research*, 20(151):1–49, 2019. [44](#), [47](#)
- [71] Gunnar Carlsson, Tigran Ishkhanov, Vin de Silva, and Afra Zomorodian. On the Local Behavior of Spaces of Natural Images. *International Journal of Computer Vision*, 76(1):1–12, January 2008. [44](#), [46](#), [54](#), [55](#), [60](#)
- [72] Nicolas L. Roux, Yoshua Bengio, Pascal Lamblin, Marc Joliveau, and Balázs Kégl. Learning the 2-d topology of images. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 841–848. Curran Associates, Inc., 2008. [44](#)

- [73] Qinghe Zheng, Mingqiang Yang, Jiajie Yang, Qingrui Zhang, and Xinxin Zhang. Improvement of generalization ability of deep cnn via implicit regularization in two-stage training process. *IEEE Access*, 6:15844–15869, 2018. [45](#), [46](#)
- [74] Kwangho Kim, Jisu Kim, Joon Sik Kim, Frederic Chazal, and Larry Wasserman. Efficient topological layer based on persistent landscapes. *arXiv preprint arXiv:2002.02778*, 2020. [45](#)
- [75] Aïcha BenTaieb and Ghassan Hamarneh. Topology aware fully convolutional networks for histology gland segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 460–468. Springer, 2016. [45](#)
- [76] Rickard Brüel-Gabrielsson, Bradley J. Nelson, Anjan Dwaraknath, Primož Skraba, Leonidas J. Guibas, and Gunnar Carlsson. A Topology Layer for Machine Learning. *arXiv:1905.12200 [cs, math, stat]*, May 2019. arXiv: 1905.12200. [45](#)
- [77] Chi Seng Pun, Kelin Xia, and Si Xian Lee. Persistent-Homology-based Machine Learning and its Applications – A Survey. *arXiv:1811.00252 [math]*, November 2018. arXiv: 1811.00252. [45](#)
- [78] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. [45](#), [61](#)
- [79] M. Robinson. *Topological Signal Processing*. Springer, 2014. [45](#)
- [80] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. [46](#)
- [81] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild. *CRCV-TR-12-01*, November, 2012. [46](#), [73](#)
- [82] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing Human Actions: A Local SVM Approach. In *Proceedings of the 17th International Conference on Pattern Recognition, ICPR*, volume 3, pages 32–36, 2004. [46](#), [73](#)
- [83] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007. [46](#), [73](#)
- [84] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, Alessandro Rinaldo, and Larry Wasserman. Robust topological inference: Distance to a measure and kernel distance. *The Journal of Machine Learning Research*, 18(1):5845–5884, 2017. [47](#)

- [85] G. Singh, F. Mémoli, and G. Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In *Eurographics Symposium on Point-Based Graphics*, 2007. 48
- [86] M Kalfaoglu, Sinan Kalkan, and A Aydin Alatan. Late temporal modeling in 3d cnn architectures with bert for action recognition. *arXiv preprint arXiv:2008.01232*, 2020. 48
- [87] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12056–12065, 2019. 48
- [88] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 48
- [89] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 61
- [90] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994. 61
- [91] Dogs vs. cats. <https://www.kaggle.com/c/dogs-vs-cats/overview>. Accessed: 2019-10-01. 61
- [92] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 61

Vita

Ephraim “Ephy” Robert Love was born in Ypsilanti, Michigan on April 25th, 1991. Growing up in Michigan, Ephy attended the Hebrew Day School of Ann Arbor and then Dexter Public Schools, each outstanding institutions, which fostered his love of learning. Ephy spent a considerable part of his youth participating in Young Judaea, which taught him the importance of pluralism, social action and a strong personal identity.

Ephy learned to truly appreciate the outdoors first from his sister, Shoshana Love, and from his brother-in-law, Karl Malcolm. Ephy is an avid rock climber, an amateur hunter-angler and a member of a search and rescue team. He loves to balance times of hard, physical work in the outdoors, with times of careful thought behind a desk. He owes this rewarding balance to his siblings.

Ephy’s career in academia began in East Hall, at the University of Michigan, where undergraduate, graduate and postdoctoral students of the Psychology Department showed him the joy of a life of scientific pursuit. He learned about the layers of hard work behind experimental science from Jim Donner, Buda Martonyi, and Doug Dopp; three men who quietly contributed their lives to science. Ephy’s parents, Terri and Jack, made sure that Ephy had every opportunity to be on Michigan’s campus, learning from scientists and engineers alike.

Ephy attended the University of Michigan, where he studied political economy, Hebrew and Judaic culture studies, and complex systems analysis. He graduated with a Bachelor of Arts degree in 2013 and moved to Knoxville, where he began studies in the Department of Geography at the University of Tennessee. Ephy graduated with a Master of Science degree in 2015, studying geographic information systems with Dr. Nicholas Nagle. Ephy’s work on spatial statistics and analysis.

For three years Ephy worked in health care consulting, writing software and working on statistical and Machine learning research projects. With the encouragement of his family, he took time away from his career to return to the University of Tennessee and begin his doctoral studies in 2018, in the Bredesen Center’s Data Science and Engineering doctoral program. Ephy completed his studies under the advisement of Dr. Vasileios Maroulas and presented his dissertation in April of 2021.