Cubical homology-based Image Classification - A Comparative Study

by

Seungho Choe

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Applied Computer Science

University of Winnipeg

Cubical homology-based Image Classification - A Comparative Study

by

Seungho Choe

Supervisory Committee

Dr. Sheela Ramanna, Supervisor
(Department of Applied Computer Science)

Dr. Talal Halabi, Departmental Member
(Department of Applied Computer Science)

Dr. Ketan V. Kotecha, External Member
(Symbiosis Centre for Applied Artificial Intelligence (SCAAI), Symbiosis International, Pune, India)

Supervisory Committee

Dr. Sheela Ramanna, Supervisor
(Department of Applied Computer Science)

Dr. Talal Halabi, Departmental Member
(Department of Applied Computer Science)

Dr. Ketan V. Kotecha, External Member
(Symbiosis Centre for Applied Artificial Intelligence (SCAAI), Symbiosis International, Pune, India)

ABSTRACT

Persistent homology is a powerful tool in topological data analysis (TDA) to compute, study and encode efficiently multi-scale topological features and is being increasingly used in digital image classification. The topological features represent number of connected components, cycles, and voids that describe the shape of data. Persistent homology extracts the birth and death of these topological features through a filtration process. The lifespan of these features can represented using persistent diagrams (topological signatures). Cubical homology is a more efficient method for extracting topological features from a 2D image and uses a collection of cubes to compute the homology, which fits the digital image structure of grids. In this research, we propose a cubical homology-based algorithm for extracting topological features from 2D images to generate their topological signatures. Additionally, we propose a score, which measures the significance of each of the sub-simplices in terms of persistence. Also, gray level co-occurrence matrix (GLCM) and contrast limited adapting histogram equalization (CLAHE) are used as a supplementary method for extracting features. Machine learning techniques are then employed to classify images using the topological signatures. Among the eight tested algorithms with six published image datasets with varying pixel sizes, classes, and distributions, our experiments demonstrate that cubical homology-based machine learning with deep residual network (ResNet 1D) and Light Gradient Boosting Machine (lightGBM) shows promise with the extracted topological features.

Keywords: Cubical complex, Cubical homology, Image classification, Machine learning, Persistent homology

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First of all, I'd like to thank Prof. Sheela Ramanna who has been more than a supervisor to me. From the beginning to the end, she has been a great mentor; always very supportive, caring and enabling in my high and low moments. She gave me the opportunity to come and study in this wonderful country. I'm very grateful to her for everything and I consider it a privilege to be one of her students. I'm also thankful for the University of Winnipeg Research grants which have made this study financially possible on my end. I pass my gratitudes to my former instructors Prof. Christopher J. Henry, Prof. Simon Liao, Prof. Talal Halabi and to all the other valued members of the Applied Computer Science Department including Mr. Ferdinand Borillo and Ms. Connie Arnhold. Special thanks to our Faculty Dean Prof. Mavis Reimer, the Graduate Studies Officers Mr. Eric Benson, Ms. Deanna England, Ms. Dagmawit Habtemariam, the Faculty of Graduate Studies and the U of W family. I appreciate the presence and the support of all my friends and colleagues including Vrushang Patel, Negin Ashrafi, Rajesh Jaiswal, and especially Cenker Sengoz who has always helped me when I have difficulty in studying. I'd also like to thank the examining committee including Prof. Ramanna, Dr. Talal Halabi and Dr. Ketann V. Kotecha for all the time and effort they will donate examining this work. Last but not least, I'm grateful to my wife Ms. Kim, dear parents and my brother who have always been there for me. Without their support, I could not have made it this far in life. Thanks to everyone else whom I haven't named that has been with me throughout this journey. Thank you Manitoba and Canada for being so wonderful and welcoming. I love you all!

Seungho Choe

DEDICATION

Hyein Kim...

To my mother, my father and my brother...

Chapter 1

Introduction

The origin of topological data analysis (TDA) and persistent homology can be traced back to H. Edelsbrunner, D. Letscher and A. Zomorodian [23, 24]. More recently, TDA has emerged as a growing field in applied algebraic topology to infer relevant features for complex data [12]. One of the fundamental methods in computational topology is persistent homology [70, 10], which is a powerful tool to compute, study and encode efficiently multiscale topological features of nested families of simplicial complexes and topological spaces [22]. Simplices are building blocks used to study the shape of data and a simplicial complex is its higher level counterpart. The process of shape construction is commonly referred to as a filtration [71]. There are many forms of filtrations and a good survey is presented in [2]. Persistent homology extracts the birth and death of topological features throughout a filtration built from a dataset [28]. In other words, persistent homology is a concise summary representation of topological features in data and is represented in a persistent diagram or barcode. This is important since it tracks changes and makes it possible to analyze data at multiple scales since data structure associated with topological features is a multiset which makes learning harder. Persistent diagrams are then mapped into metric spaces with additional structure useful for machine learning tasks [1]. Application of TDA in machine learning (also known as TDA pipeline) in several fields is well-documented [12]. The TDA pipeline consists of using data (e.g., images, signals) as input and then filtration operations are applied to obtain persistence diagrams. Subsequently, ML methods such as support vector machines, tree classifiers are applied to the persistent diagrams.

In [28], a random forest classifier was used to classify the well-known MNIST image dataset using the voxel structure to obtain topological features. However, it has been shown that mapping of topological signatures to a representation necessary for machine learning is pre-defined which is a limitation [32]. The success of deep learning [43] in computer vision problems, has led to its use in deep networks that can handle barcodes [33]. Hofer et. al. used a persistence diagram as a topological signature and compute a parametrized projection from the persistence diagram, then leverage it during training of the network. The output of this process is stable when using the 1-Wasserstein distance. Classification of 2D object shapes and social network graphs were successfully demonstrated by the authors. In [11], persistent diagrams were used with neural network classifiers in graph classification problems. Persistent barcodes were used to classify brain activation patterns in rs-fMRI video frames [19, 20]. The topological and geometric structures underlying data are often represented as point clouds. More recently, multiclass classification of point cloud datasets was discussed in [40].

However, it has been shown that the implementations of persistent homology (of simplicial complexes) is inefficient for computer vision since it requires excessive computational resources [3] due to the formulations based on triangulations. To mitigate the problem of complexity, cubical homology was introduced which allows direct application of its structure [41, 63]. Simply, cubical homology uses a collection of cubes to compute the homology, which fits the digital image structure of grids. Since there is neither skeletonization nor triangulation in the computation of cubical homology, it has advantages in the fast segmentation of images for extracting features. This aspect of cubical homology is the motivation for its application in the feature engineering process of this thesis.

1.1 Problem Definition and Proposed Approach

Since machine learning models rely on accurate feature representations, multiscale representation of features are becoming increasingly important in applications involving computer vision and image analysis. Persistence homology is able to bridge the gap between geometry and topology and persistent-homology based machine learning models have been used in various areas including image classification and analysis [60]. In this thesis, we address the problem of construction of feature vectors based on cubical homology for different types of 2D images with varying pixel sizes, classes and distributions. We then study the effect of these topological features on selected machine learning models. The comparative study is meant to give insights into the application of cubical homology for classification of different types of images as well as the computational challenges with the topological signatures for the selected image datasets. Fig. 1.1 illustrates our proposed approach. The first two steps form the core of this thesis, namely, the generation of 1D topological signatures using a threshold score. This score allows us to filter out low persistence features (or noise).

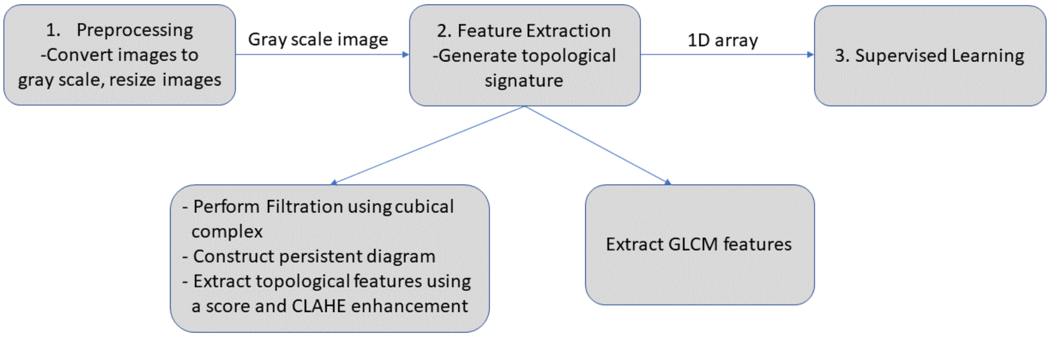

1. Preprocessing
-Convert images to gray scale, resize images
Gray scale image
2. Feature Extraction
-Generate topological signature
1D array
3. Supervised Learning
- Perform Filtration using cubical complex
- Construct persistent diagram
- Extract topological features using a score and CLAHE enhancement
Extract GLCM features

Figure 1.1: Classification pipeline

1.2 Contributions

Our contributions are as follows:

- We propose a cubical homology-based algorithm for extracting topological features from 2D images to generate their topological signatures.

- We propose a score, which is used as measure of the significance of the subcomplex calculated from the persistence diagram. Also, we use gray level co-occurrence matrix (GLCM) and contrast limited adapting histogram equalization (CLAHE) for getting additional image features, in an effort to improve the classification performance.

- We provide a comparative study of eight well-known machine learning algorithms using the extracted topological features from six different published image datasets.

1.3 Thesis Layout

The rest of this thesis organized as follows:

Chapter 2 provides a background of other topological approaches in image and graph classification problems.

Chapter 3 introduces the mathematical framework for understanding persistent homology. Basic definitions for graph theory, simplicial homology, and cubical homology are given.

Chapter 4 discusses the process of feature engineering. Specifically, we introduce the notion of score in this chapter.

Chapter 5 introduces six image datasets which is used as benchmark for evaluating the performance of proposed method.

Chapter 6 introduces machine learning algorithms briefly. Also, we provide the evaluation criteria, and discuss the classification results.

Chapter 7 concludes the thesis, summarizes the work done and provides possible future research directions.

Chapter 2

Related Works

Topological approach is widely used in machine learning for image classification and graph classification. In particular, persistence diagrams (PD) are common tools to extract features and are used in a number of different ways. In this chapter, we discuss a few papers that use persistence diagrams in image classification as well as in science, to model and analyze geometric structures.

2.1 Direct Use of Persistence Diagrams

2.1.1 Analyzing force networks

In [42], PDs are used in quantifying differences between force networks derived from particulate systems. Here, force networks are designed to model interactions between particles typically derived from either experiments or simulations. For the problem of analyzing network, it is important to find a connection between particles and backbone of the force chains. Such networks are characterized by their critical parameters, which are difficult to use directly because these parameters contain important geometric structures of force distributions between particles. PDs of the network are able to extract concise information of network components using Betti numbers. PDs define a set of points $\mathrm{FN}(f, \theta)$ that exceeds the threshold θ of the force field f . The threshold moves from high to low to calculate persistent homology thereby revealing the geometric features.

2.1.2 Analyzing Molecular Dynamics

Persistence diagrams are also used in the case of analyzing polyatomic structure such as molecular liquid, granular materials, and metallic glasses. The authors in [50] discuss the relationship between medium-range order and short-range order of amorphous materials. Here, PDs are used to extract structures of these materials such as size and shape of crystalline materials. In this paper, topological properties(*birth* and *death*) are computed from a persistence diagram $\mathcal{A}$, where $D_n(\mathcal{A})$ is constructed which represents a collection of pairs (*birth*, *death*) of the many body atomic structure in the glass. A normalized distribution for D_n is then calculated. PDs have an advantage because they reduce the size of data by summarizing their geometric features.

2.2 Machine Learning from Persistence Diagrams

2.2.1 Classical Methods

In [21], microvascular patterns in endoscopy images can be categorized as *regular* and *irregular*. Furthermore, there are three types in regular surface of microvascular; *oval*, *tubular*, and *villous*. To classify these patterns, persistent homology plays a important role by deriving topological features with persistence diagrams. In this paper, per *q-th norm* of *p*-th diagram is computed as,

$$N_q = \left[\sum_{A \in \mathrm{Dgm}_p(f)} \mathrm{pers}(A)^q \right]^{\frac{1}{q}},$$

where $\mathrm{Dgm}_p(f)$ denotes the p-th diagram of f and $\mathrm{pers}(A)$ is persistence of a point A in $\mathrm{Dgm}_p(f)$. Since N_q is a norm of p-th Betti number with restriction (or threshold) s, it will get p-th Betti number of $\mathbb{M}_s$ where $\mathbb{M}$ is the rectangle covered by pixels. Then, $\mathbb{M}$ is mapped to $\mathbb{R}$ by signed distance function. A naive Bayesian learning method which combines the results of several Adaboost classifiers is then used to classify the images. The analysis also concludes that only a few of geometric and topological features are responsible for a large majority of decisions.

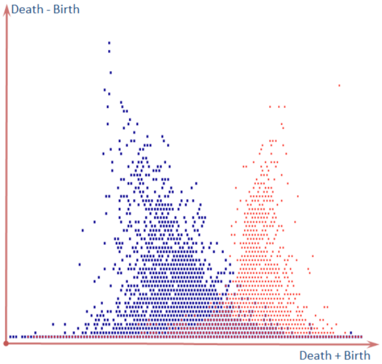


Figure 2.1: The superposition of two persistence diagrams. Figure retrieved from [21]

2.2.2 Multi-Scale Kernels for Machine Learning

The authors in [61] point out that persistence diagrams are hard to use directly in a class of learning techniques such as Support Vector Machines or Principal Component Analysis which use kernel functions to redefine the Hilbert space structure. This paper introduces a multi-scale kernel for persistence diagrams which is based on scale space theory [34]. This kernel is defined on L_2-valued feature map and satisfies *Lipschitz continuous* which implies that it maintains the stability property of persistent homology. (See Figure 2.2.) The focus is on stability of persistent homology since any occurrence of small changes in the input, affects both the 1-Wasserstein distance and persistent diagrams. Experiments on two benchmark datasets for 3D shape classification/retrieval and texture recognition are discussed.

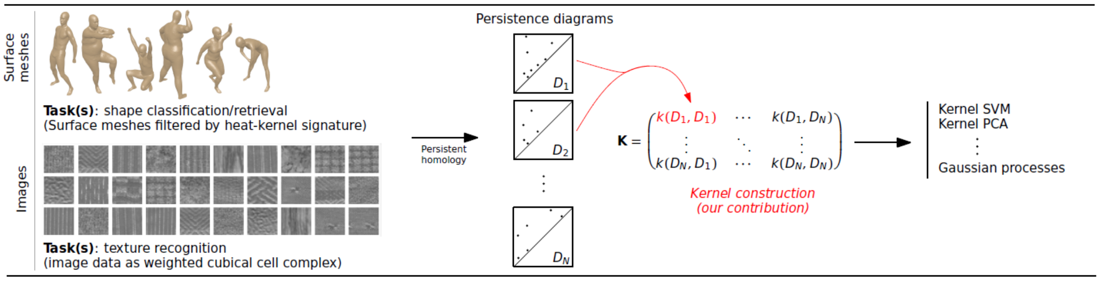

Surface meshes
Task(s): shape classification/retrieval
(Surface meshes filtered by heat-kernel signature)
Images
Task(s): texture recognition
(image data as weighted cubical cell complex)
Persistent homology
Persistence diagrams
D_1
D_2
D_N
$\mathbf{K} = \begin{pmatrix} k(D_1, D_1) & \cdots & k(D_1, D_N) \\ \vdots & \ddots & \vdots \\ k(D_N, D_1) & \cdots & k(D_N, D_N) \end{pmatrix}$
Kernel construction
(our contribution)
Kernel SVM
Kernel PCA
Gaussian processes

Figure 2.2: Construction of kernel from sample data. Figure retrieved from [61].

2.2.3 Filtration Methods for the TDA pipleline

Among the multiple filtration methods for computing persistent homology, some filtration methods including Vietoris-Rips filtration regard pixels as point cloud and others use the structure of pixels directly [28]. This work discuss how to use topological features with different filtration methods in the context of the MNIST digits image dataset using the random forest classifier (See Figure 2.3).

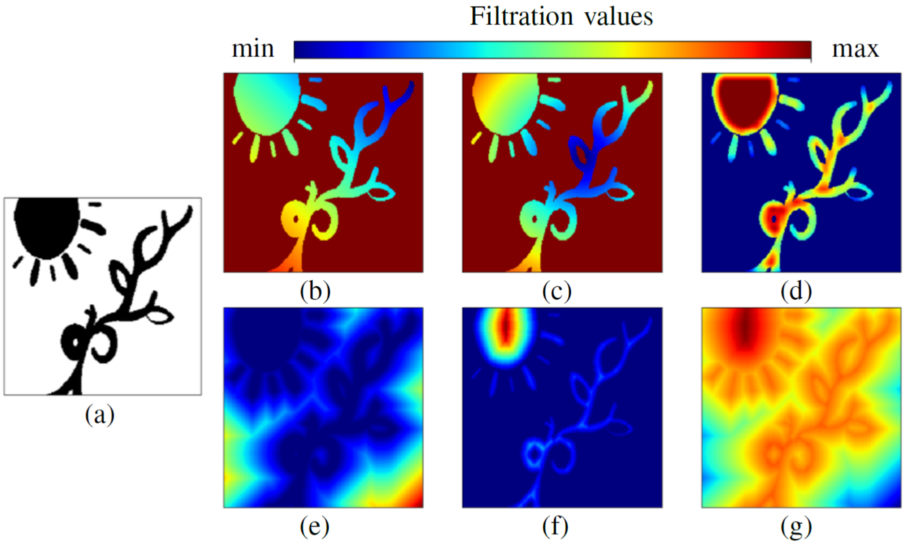


Figure 2.3: Grayscale image obtained from different filtration methods. Figure retrieved from [28]

2.2.4 Deep Learning Methods

Persistent homology is widely used in topological data analysis, however, its structure of a multiset makes learning harder. Hofer et. al [33] construct a kernel for a persistence diagram as a topological signature. The characteristics of the kernel is that it satisfies some properties that are *Lipschitz continuous* and *differentiable*. A parametrized projection from the persistence diagram is computed, and then leveraged during the training of the deep network. The output of this process is stable when it comes to the 1-Wasserstein distance. This is demonstrated in the classification of 2D object shapes and social network graphs.

2.2.5 Persistence as Feature Vector

Although image classification tasks involve grayscaling of images in preprocessing, it is possible to utilize the RGB channel without grayscaling. In [17], RGB intensity values of each pixel of an image is mapped to the point cloud $P \in \mathbb{R}^5$ and then deriving a feature vector. Computing and arranging the persistence of point cloud data by descending order makes it possible to understand persistence of features(See Figure 2.4). The extracted topological features and the traditional image processing features are used in both vector based supervised classification and deep network based classification experiments on the CIFAR-10 image data set.

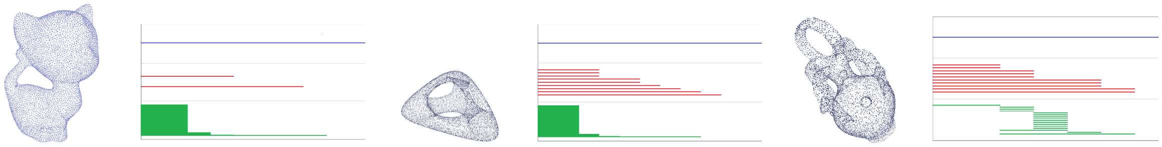


Figure 2.4: Point cloud and its persistence diagrams. Figure retrieved from [17]

2.2.6 Betti Curves for classification of chaotic time series datasets

In [67], the authors apply topological data analysis to the classification of time series data. A 1D convolutional neural network is used where the input data is a *Betti sequence*. Persistent homology is used to generate Betti sequences from what are known as quasi-attractors. A quasi-attractor represents the set of delay vectors and encodes transition rules of the underlying system.

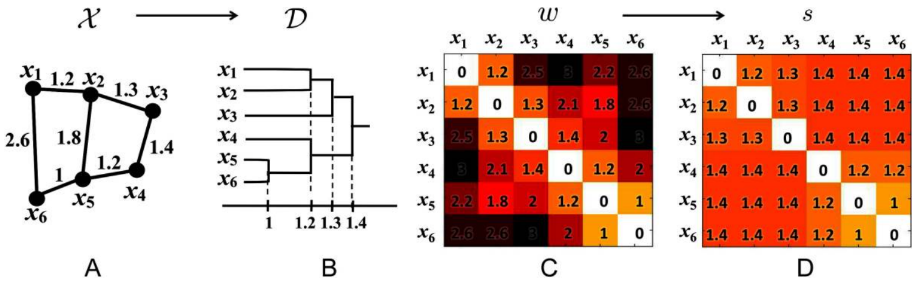
2.3 Other Approaches

2.3.1 Vector Summaries of Persistence Diagram

Although persistent homology is useful tool to identify geometric signatures in many cases, it is hard to handle vector spaces in terms of distances [7]. Vector summaries of persistence diagram is a technique that transforms a persistence diagram into vectors and summarizes a function by its minimum through a pooling technique. The authors present a novel pooling within the bag-of-words approach that shows significant improvement in shape classification and recognition problems with the Non-Rigid 3D Human Models SHREC 2014 dataset.

2.3.2 Betti Numbers in resting state brain network analysis

In topological data analysis, Betti numbers represent counts of the number of homology groups such as points, cycles, and so on. In [15], similarity of brain networks of twins are measured using Betti numbers. Specifically, the first Betti number represents the count of the number of cycles and is significant in network analysis. Figure 2.5 is an example of network, by the authors. The significance of the number of cycles is evaluated using the Kolmogorov–Smirnov (KS) distance.


$\mathcal{X} \longrightarrow \mathcal{D}$
$w \longrightarrow s$
x_1 1.2 x_2 1.3 x_3
2.6 1.8 1.4
1 1.2
x_6 x_5 x_4
x_1 x_2 x_3 x_4 x_5 x_6
1 1.2 1.3 1.4
A
B
C
D

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	1.2	2.5	3	2.2	2.6
x_2	1.2	0	1.3	2.1	1.8	2.6
x_3	2.5	1.3	0	1.4	2	3
x_4	3	2.1	1.4	0	1.2	2
x_5	2.2	1.8	2	1.2	0	1
x_6	2.6	2.6	3	2	1	0

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	1.2	1.3	1.4	1.4	1.4
x_2	1.2	0	1.3	1.4	1.4	1.4
x_3	1.3	1.3	0	1.4	1.4	1.4
x_4	1.4	1.4	1.4	0	1.2	1.2
x_5	1.4	1.4	1.4	1.2	0	1
x_6	1.4	1.4	1.4	1.2	1	0

Figure 2.5: Example of network and its topological analysis. Figure retrieved from [15]

2.3.3 Betti Numbers in resting state (rs-fMRI) videos

In [19, 20] persistent barcodes were used to visualize brain activation patterns in resting state functional magnetic resonance imaging(rs-fMRI) video frames. The authors use a geometric Betti number that counts the total number of connected cycles forming a vortex (nested, usually non concentric, connected cycles) derived from the triangulation of brain activation regions. The vortexes correspond to the changing activation areas in the video frames. These activation areas represent intrinsic brain activity that contains reproducible temporal sequences (lag structures). The authors found that persistent, recurring blood oxygen level dependent (BOLD) signals in triangulated rs-fMRI video frames display previously undetected topological findings,

Chapter 3

Mathematical Foundations

In this chapter, we give basic definitions for simplicial, cubical and persistent homology. A simplicial complex is a space or an object that is built from a union of points, edges, triangles, tetrahedra, and higher-dimensional polytopes. Homology theory is in the domain of algebraic topology related to the connectivity in multi-dimensional shapes [3].

3.1 Simplicial Homology

Graphs are mathematical structures used to study pairwise relationships between objects and entities.

Definition 1. A *graph* [6] is a pair of sets, $G = (V,\ E)$, where V is the set of vertices (or nodes) and E is a set of edges.

Let S be a subset of a group G. Then the subgroup generated by S, denoted $\langle S\rangle$, is the subgroup of all elements of G that can be expressed as the finite operation of elements in S and their inverses. For example, the set of all integers, $\mathbb{Z}$ can be expressed by operation of elements $\{1\}$ so $\mathbb{Z}$ is the subgroup generated by $\{1\}$.

Definition 2. A *rank* [6] of a group G is the size of the smallest subset that generates G.

For instance, since $\mathbb{Z}$ is the subgroup generated by $\{1\}$, rank($\mathbb{Z}$)=1.

Definition 3. A *simplex complex* [62] on a set V is a family of arbitrary-cardinality subsets of V closed under the subset operation, which means if a set S is in the family, all subsets of S are also in the family. An element of the family is called a *simplex* or *face*.

Definition 4. Also, $p - simplex$ [62] can be defined to the convex hull of $p + 1$ affinely independent points $x_0, x_1, \cdots, x_p \in \mathbb{R}^d$.

For example, in a graph, 0-simplex is a point, 1-simplex is an edge, 2-simplex is a triangle, 3-simplex is a tetrahedron and so on. (See Figure 3.1. [26])

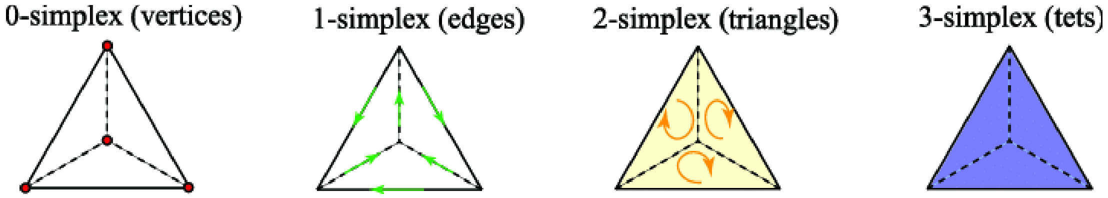


Figure 3.1: Examples of p-simplex for $p = 0, 1, 2, 3$ in tetrahedron. A 0-simplex is a point, a 1-simplex is an edge which convex hull of two points, a 2-simplex is a triangle which convex hull of three distinct points and a 3-simplex is tetrahedron where the convex hull of four points.

3.1.1 Chain, Boundary, and Cycle

To extend simplicial homology to persistent homology, the notion of *chain*, *boundary*, and *cycle* is necessary [54].

Definition 5. A p-chain [62] is a subset of p-simplices in a simplicial complex K. Assume K is a triangle. Then, a 1-chain is a subset of 1-simplices, in other words, a subset of the three edges.

Definition 6. A boundary [62], generally denoted ∂, of p-simplex is the set of $(p - 1)$-simplices faces.

For example, a triangle is a 2-simplex, so the boundary of a triangle is a set of 1-simplices which are the edges. Therefore, the boundary of the triangle is the three edges.

Definition 7. A cycle [62] can be defined using the definitions of chain and boundary. A p-cycle c is a p-chain with empty boundary. Put it simply, it is a path where the starting point and destination point is the same.

3.2 Cubical Homology

Cubical homology [41] is efficient since it allows direct use of the cubical structure of the image whereas simplicial theory requires increasing the complexity of data. While the simplicial homology is built with the triangle and its higher-dimensional structure such as tetrahedron, cubical homology consists of *cubes.* In cubical homology, each cube has a unit size and the n-cube represents its dimension. For example, 0-cubes are points, 1-cubes are lines with unit length, 2-cubes are unit squares, and so on.

Definition 8. 0-cubes [41, 35, 36] can be defined as an interval,

$$[m] = [m,\ m],\ m \in \mathbb{Z},$$

which generate subsets $I \in \mathbb{R}$, such that

$$I = [m,\ m+1],\ m \in \mathbb{Z}.$$

Therefore, I is called a 1-cube, or *elementary interval*.

Definition 9. A n-cube [41, 35, 36] can be expressed as a product of elementary intervals as

$$Q = I_1 \times I_2 \times \cdots \times I_n \subseteq \mathbb{R}^n,$$

where Q indicates n-cube, $I_i (i = 1,\ 2,\ \cdots,\ n)$ is an elementary interval.

A *d-dimensional image* is a map $\mathcal{I} : I \subseteq \mathbb{Z}^d \to \mathbb{R}$.

Definition 10. A *pixel* [41, 35, 36] can be defined an element $v \in I$, where $d = 2$. If $d > 2$, v is called a *voxel.*

Definition 11. [41, 35, 36] Let $\mathcal{I}(v)$ be *intensity* or *greyscale* value. Also, in the case of *binary images*, we consider a map $\mathcal{B} : I \subseteq \mathbb{Z}^d \to \{0,\ 1\}$.

A voxel is represented by a d-cube and with all of its faces added, we have

$$\mathcal{I}'(\sigma) := \min_{\sigma \text{ face of } \tau} \mathcal{I}(\tau).$$

Let K be the cubical complex built from the image I, and let

$$K_i := \{\sigma \in K | \mathcal{I}'(\sigma) \leq i\},$$

be the i-th *sublevel set* of K. Then, the set $\{K_i\}_{i\in \mathrm{Im}(I)}$ defines a filtration of the cubical complexes. So, the pipeline to filtration from image with cubical complex is as follows:

$$\text{Image} \rightarrow \text{Cubical complex} \rightarrow \text{Sublevel sets} \rightarrow \text{Filtration}$$

Also, *chain*, *boundary*, and *cycle* in cubical homology can be defined by the same manner as in section 3.1.1.

3.3 Persistent Homology

In topology, there are subcomplices of complex K and *cubes* are created (*birth*) and destroyed (*death*) by filtration. Assume that K^i $(0 \leq i \leq,\ i \in \mathbb{Z})$ is a subcomplex of filtered complex K such that

$$\emptyset \subseteq K^0 \subseteq K^1 \subseteq \cdots \subseteq K^n = K,$$

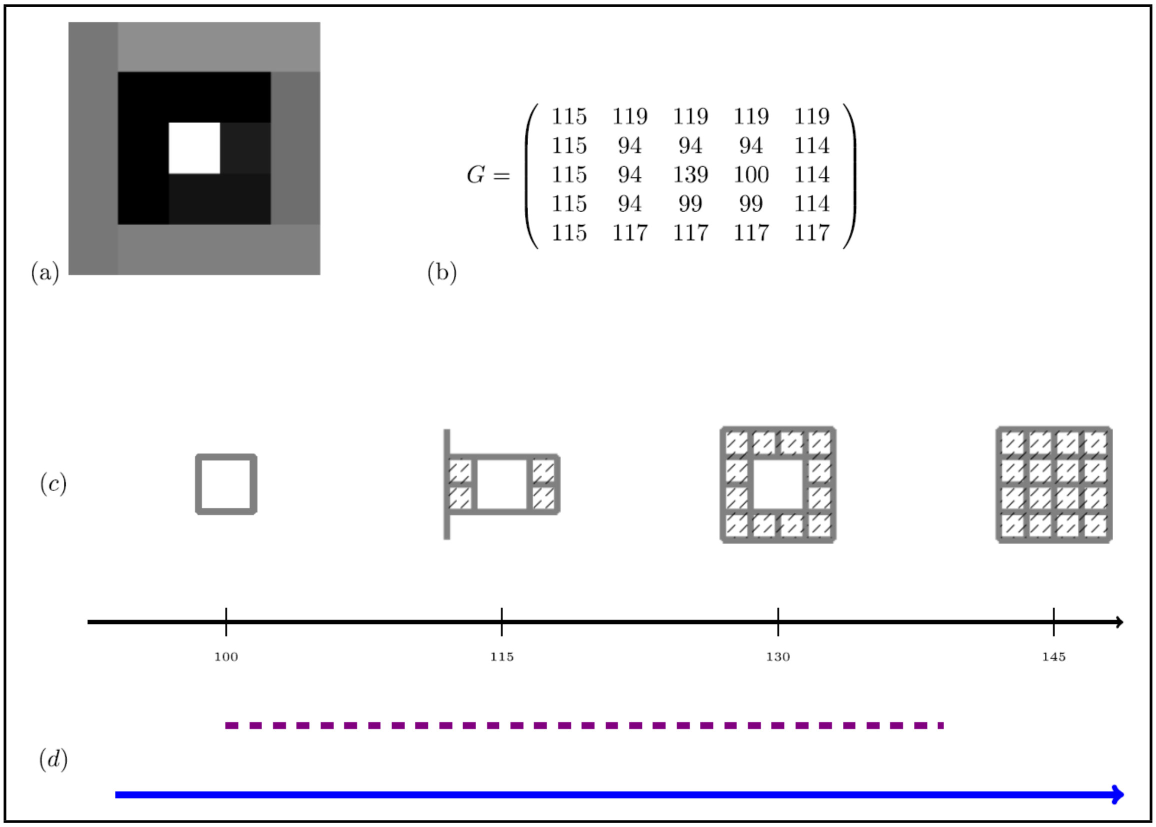
and $\mathcal{Z}_k^i,\ \mathcal{B}_k^i$ are its corresponding cycle group and boundary group.

Definition 12. *Persistent homology* [25] can be defined as

$$\mathcal{H}_k = \mathcal{Z}_k / \mathcal{B}_k \tag{3.1}$$

Definition 13. A *persistence* [25] is a lifetime of these attributes based on the filtration method used.

One can plot the birth and death times of the topological features as a barcode also known as *persistence barcode* shown in Figure 3.2. This diagram graphically represents the topological signature of the data. Illustration of persistence is useful when detecting change in terms of topology and geometry, which plays a crucial role in supervised machine learning [46].



$$G = \begin{pmatrix} 115 & 119 & 119 & 119 & 119 \\ 115 & 94 & 94 & 94 & 114 \\ 115 & 94 & 139 & 100 & 114 \\ 115 & 94 & 99 & 99 & 114 \\ 115 & 117 & 117 & 117 & 117 \end{pmatrix}$$

Figure 3.2: An example of persistent homology for grayscale image. **(a)** A given image, **(b)** A matrix of gray level of given image, (c) the filtered cubical complex of the image, (d) the persistence barcode according to (c). This figure is taken from [54]

Chapter 4

Feature Engineering

In this chapter, we describe the feature engineering process that was used in this thesis. The main purpose of this process is to obtain a 1-dimensional array from each image in the dataset. Each point from the persistence diagram plays a significant role in the extraction of the topological features. Also, the *Gray level co-occurrence matrix* (GLCM) supports these topological features as additional signatures. Because every image dataset is not identical in size and some images have very high resolution, resizing every image to 200x200 and converting them to gray-scale guarantees a relatively constant duration of extraction (about 4 seconds) regardless of its original size.

Algorithm 1 gives the method for extracting topological features from a dataset. In this algorithm, β_0 and β_1 are Betti numbers derived from Eqn. 3.1 where the dimension of i^{th} homology is called the i^{th} Betti number of K. β_0 gives the number of connected components and β_1 gives the number of holes. Betti numbers represent the count of the number of topological features. The number of these features in each dimension is captured by the corresponding Betti number.

Algorithm 1: Extraction of Topological Features

```
$N \leftarrow$ number of dataset;
for $i = 1,\ 2,\ \cdots,\ N$ do
    $img \leftarrow$ load $i^{th}$ image from dataset;
    $img \leftarrow$ resize $img$ to (200, 200) and convert to grayscale;
    $PD_0 \leftarrow$ set of points of $\beta_0$ in persistence diagram of $img$ with cubical
      complex;
    $PD_1 \leftarrow$ set of points of $\beta_1$ in persistence diagram of $img$ with cubical
      complex;
    $PD_0 \leftarrow$ sort $PD_0$ in descending order of $persistence$;
    $PD_1 \leftarrow$ sort $PD_1$ in descending order of $persistence$;
    $d_i \leftarrow$ project each point in $PD_0$ to [0, 1];
    $d_i \leftarrow d_i$ + project each point in $PD_1$ to [1, 2];
    $fimg \leftarrow$ adapt CLAHE filter to $img$;
    $fPD_0 \leftarrow$ set of points of $\beta_0$ in persistence diagram of $fimg$ with cubical
      complex;
    $fPD_1 \leftarrow$ set of points of $\beta_1$ in persistence diagram of $fimg$ with cubical
      complex;
    $fPD_0 \leftarrow$ sort $fPD_0$ in descending order of $persistence$;
    $fPD_1 \leftarrow$ sort $fPD_1$ in descending order of $persistence$;
    $d_i \leftarrow d_i$ + project each point in $fPD_0$ to [0, 1];
    $d_i \leftarrow d_i$ + project each point in $fPD_1$ to [1, 2];
    $d_i \leftarrow d_i$ + convert $img$ to GLCM with distances (1, 2, 3), directions (0°,
      45°, 90°, 135°), and properties (energy, homogeneity);
Output: $D(d_1,\ d_2,\ \cdots,\ d_N)$
```

4.1 Projection of Persistence Diagrams

After filtration by cubical complex, we are ready to construct a persistence diagram. The dth persistence diagram, $\mathcal{D}_d$ contains all of the d-dimensional topological information. These are series of points with a pair of (*birth*, *death*), where *birth* indicates the time at which the topological features were created and the *death* gives the time at which these features are destroyed. From here, *persistence* is defined using the definition of *birth* and *death* as,

$$pers(birth, death) := death - birth,\ where\ (birth, death) \in \mathcal{D}_d. \tag{4.1}$$

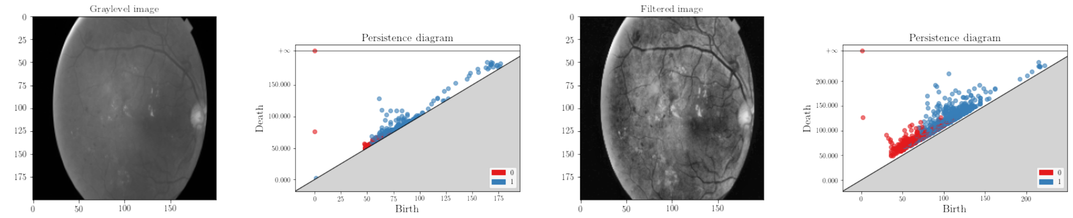
Then, a low-persistence feature is treated as having a low importance, or 'noise' whereas high-persistence features are regarded as 'real' features [25]. However, using *persistence* as a result of projection of a topological feature to a 1-dimensional value is inadequate, because it is impossible to distinguish the features which have the same *persistence* but different values for *birth*. Therefore, we propose a metric (*score*), to compensate for this limitation of *persistence* as

$$score_d(birth, death) := \begin{cases} 0 & \text{if } persistence < threshold \\ d + \left(\frac{e^{sin\frac{death}{255\cdot 2}\pi} - 1}{e-1}\right)^3 - \left(\frac{e^{sin\frac{birth}{255\cdot 2}\pi} - 1}{e-1}\right)^3 & \text{if } persistence \geq threshold \end{cases} \tag{4.2}$$

A threshold is a value that allows us to ignore noise. Therefore, the *score* takes into account not only the *persistence*, but also other aspects such as *dimension, birth, and death* of topological features.

4.2 Contrast Limited Adapting Histogram Equalization (CLAHE)

When pixel values are concentrated in a narrow range, it is hard to perceive features visually. Histogram equalization makes the distribution of pixel values in the image balanced, thereby enhancing the image. However, this method often results in degrading the content of the image and also amplifying the noise. Therefore, it produces undesirable results. Contrast limited adapting histogram equalization (CLAHE) is a well-known method for compensating the weakness of histogram equalization by dividing an image into small sized blocks and performing histogram equalization for each block [59]. After completing histogram equalization in all blocks, bi-linear interpolation makes the boundary of the tiles (blocks) smooth. In this thesis, we apply CLAHE to the extraction of topological features process in an effort to finding features efficiently. An illustration of the CLAHE method on the APTOS data is given



(a) (b) (c) (d)

Figure 4.1: Comparison of the original image and the CLAHE filtered image **(a)** Original image. **(b)** Persistence diagram of the original image (a). (c) CLAHE Filtered image, **(d)** Persistence diagram of the filtered image (c).

in Figure 4.1.

For extracting textual features, we use the well-known Gray Level Co-occurrence Matrix (GLCM) [49]. We used three distances (1, 2, 3) and four directions (0°, 45°, 90°, 135°) to get the GLCM features. From each of co-occurrence matrices, two global statistics were extracted: energy and homogeneity resulting in $3 \times 4 \times 2 = 24$ textual features for each image.

Table 4.1 gives a sample list of extracted features from the APTOS dataset. From the CLAHE filtered image, 144 features are extracted for each dimension. Similary, 100 topological features for each dimension and 24 GLCM features are extracted from the original-gray level image.

Table 4.1: Results of feature engineering process applied to the APTOS dataset

img	label	glcm1	glcm2	⋯	glcm24	dim0_0	dim0_1	⋯	dim0_99	dim1_0	dim1_1	⋯	dim1_99	fdim0_0	fdim0_1	⋯	fdim0_ 143	fdim1_ 0	fdim1_ 1	⋯	fdim1_ 143
0	2	0.1603	0.1571	⋯	0.4639	1	0.0366	⋯	0	1.2054	1.1815	⋯	0	0.9999	0.2060	⋯	0.0319	1.7698	1.6339	⋯	1.1067
⋮																					
3661	2	0.1196	0.1160	⋯	0.5387	0.9999	0.0020	⋯	0	1.4787	1.0636	⋯	0	0.9999	0.1295	⋯	0.0042	1.9493	1.3658	⋯	1.10478

Chapter 5

Image Datasets

In this chapter, we give a brief description of the six published image datasets used in this work, which are collected from *Mendelay*, *Tensorflow Dataset*, and *Kaggle competition*. Also, Table 5.1 gives summarized information of datasets.

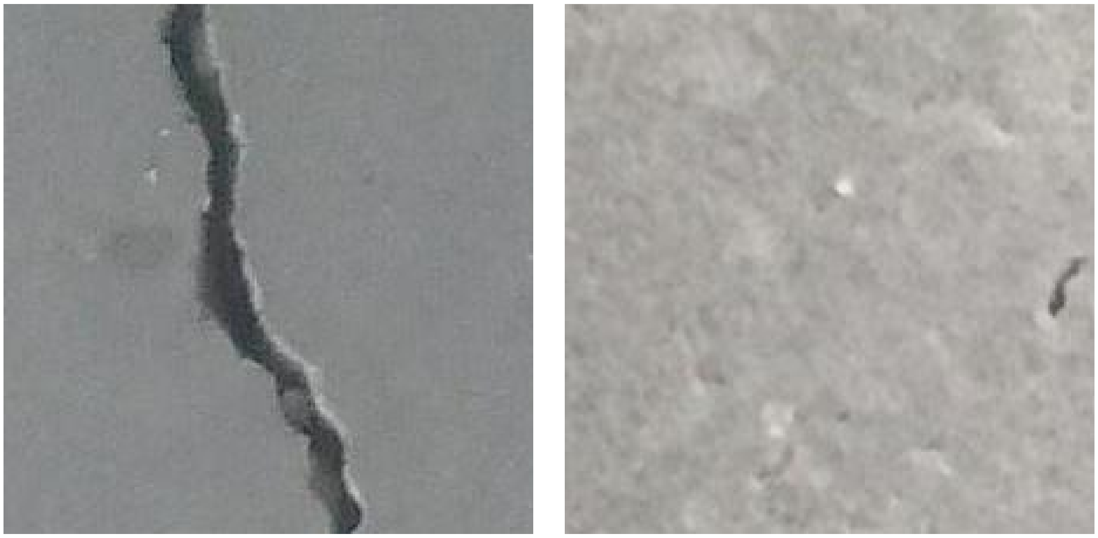
5.1 Concrete Crack Images for Classification

The *concrete crack images for classification* dataset [55], contains a total of 40,000 images where each image consists of 227×227 pixels. It was collected from the METU campus building and consists of 2 states; 20,000 images of positive crack and 20,000 images of negative crack. A crack on an outer wall occurs as time goes on or due to natural aging. It is important to detect these cracks in terms of evaluating and predicting structural deterioration and reliability of buildings. Samples of the two types of images are shown in Figure 5.1.

5.2 APTOS Blindness Detection

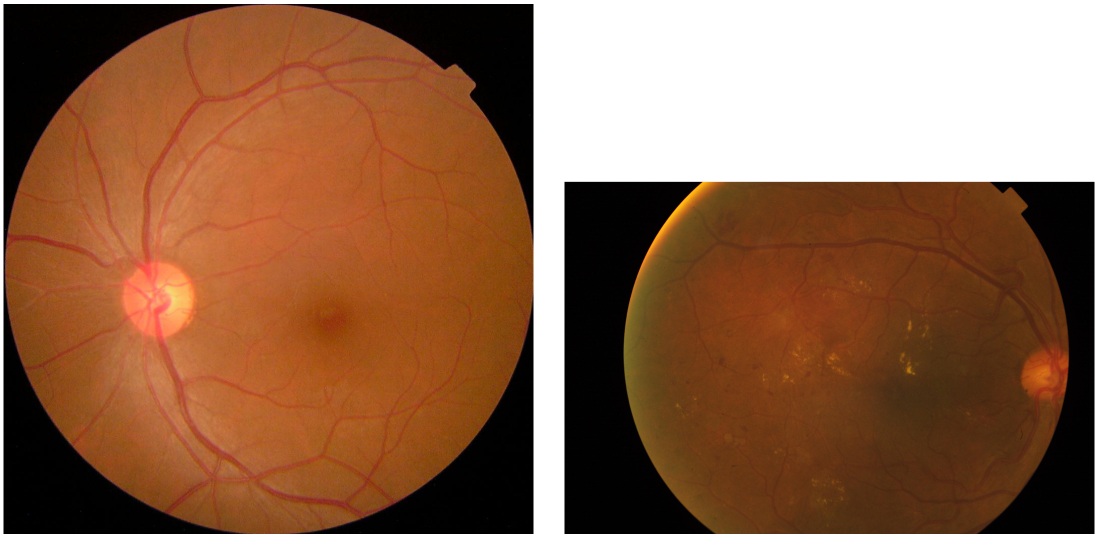
APTOS blindness detection dataset is a set of retina images taken by fundus photography for detecting and preventing diabetic retinopathy from causing blindness[1]. This dataset has 3,662 images and consists of 1,805 images diagnosed as non-diabetic retinopathy and 1,857 images diagnosed as diabetic retinopathy as shown in Figure 5.2. Figure 5.3 shows the distribution of examples in the four classes using a severity range from 1 to 4.

[1] https://www.kaggle.com/c/aptos2019-blindness-detection/overview



(**a**) Negative crack image (**b**) Positive crack image

Figure 5.1: Sample images of the Concrete Crack Dataset.



(**a**) Non diabetic retinopathy (**b**) Diabetic retinopathy

Figure 5.2: Sample images of the APTOS dataset.

5.3 Pest Classification in Mango Farms

Pest classification in Mango farms dataset [44] is a collection of 46,500 images of mango leaves affected by 15 different types of pests and one normal (unaffected) mango leaf as shown in Figure 5.5. Some of these pests can be detected visually.

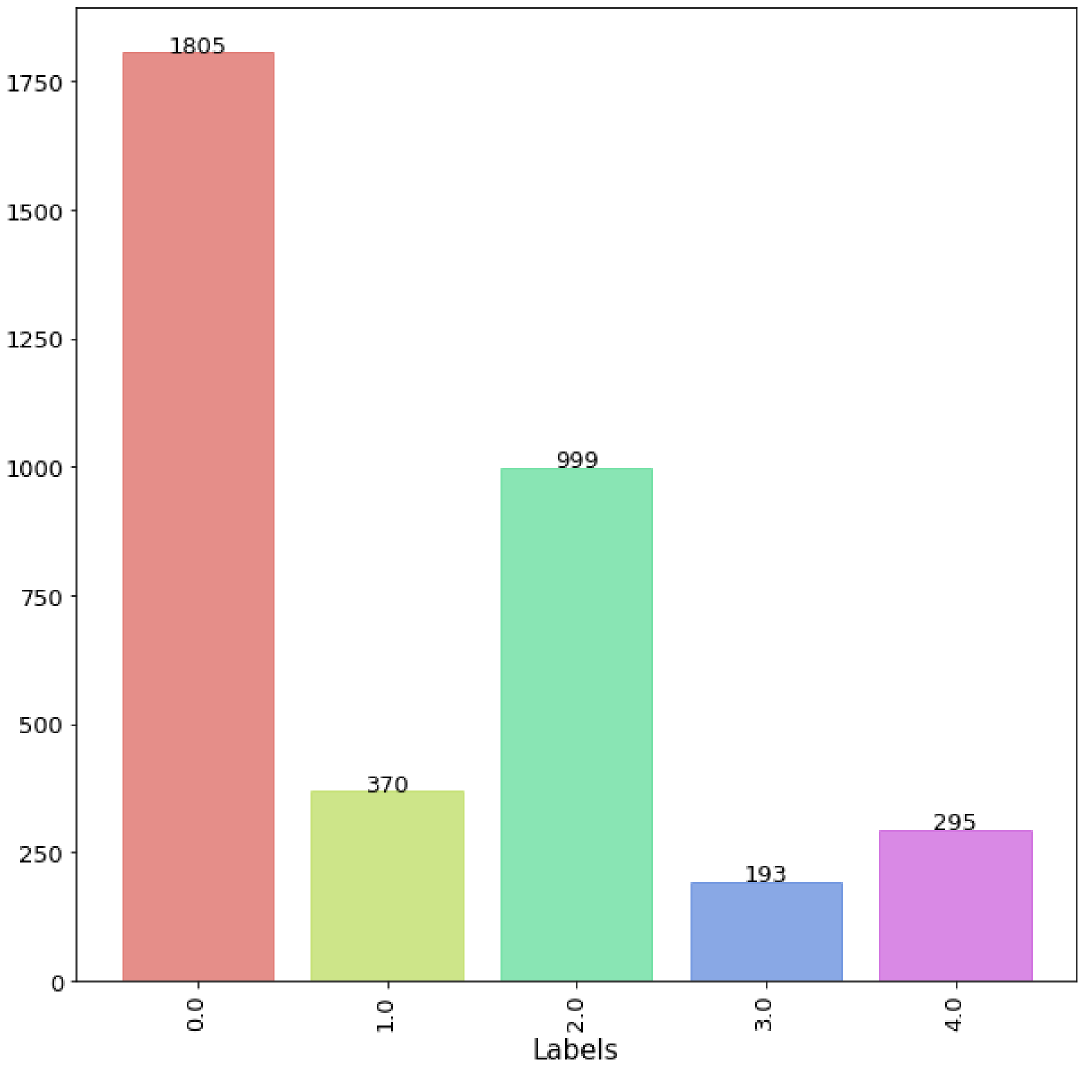


Figure 5.3: Data distribution for the APTOS 2019 Blindness Detection dataset.

Figure 5.4 shows the data distribution of examples in the 15 classes of pests and one normal class.

5.4 Indian Fruits

The *Indian fruits* dataset [5] contains 23,848 images that cover five popular fruits in India; apple, orange, mango, pomegranate, and tomato. This dataset includes variation of each fruit resulting in 40 classes. This dataset was already separated into training and testing sets as shown in Figure 5.6. Note, that this dataset has an

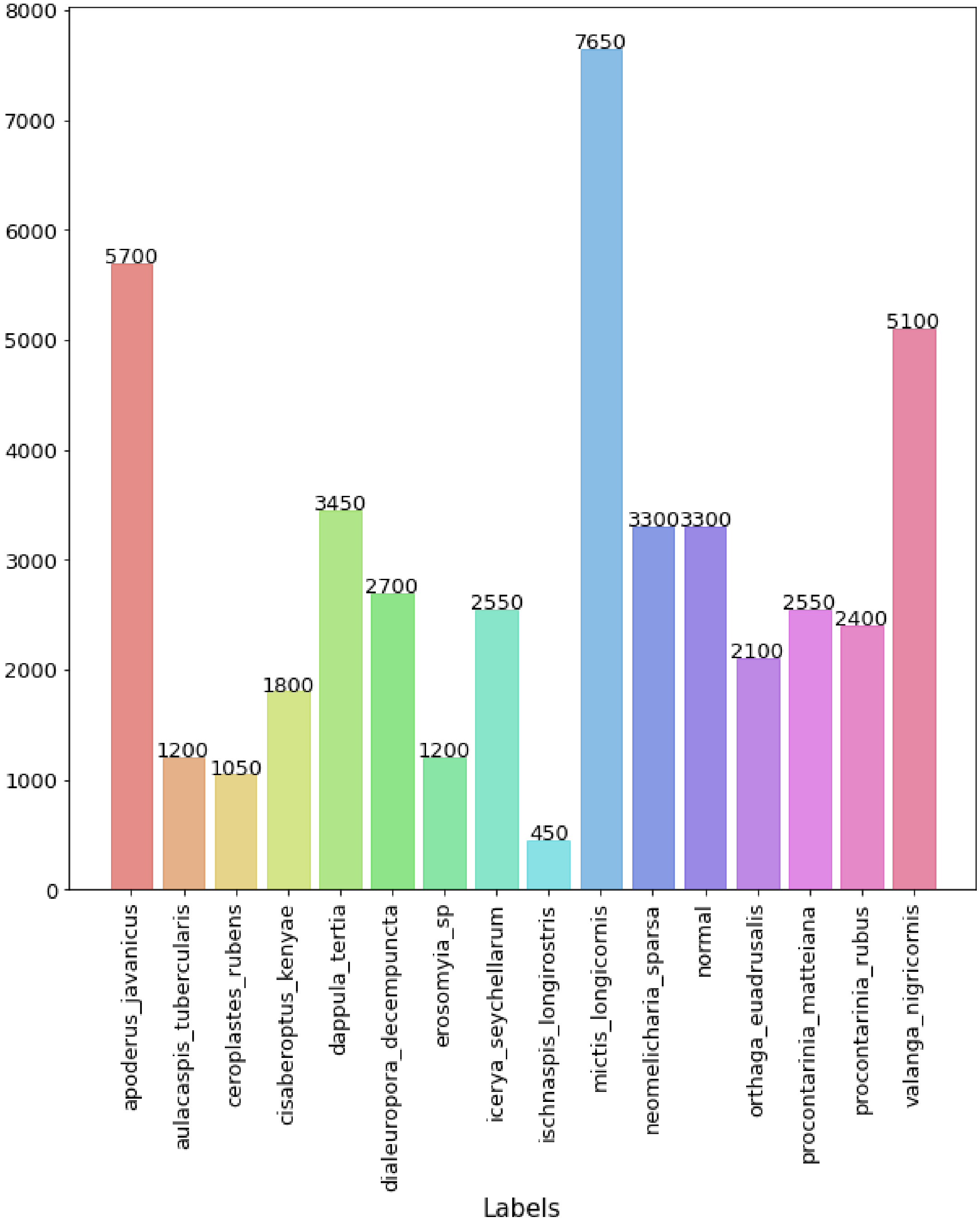
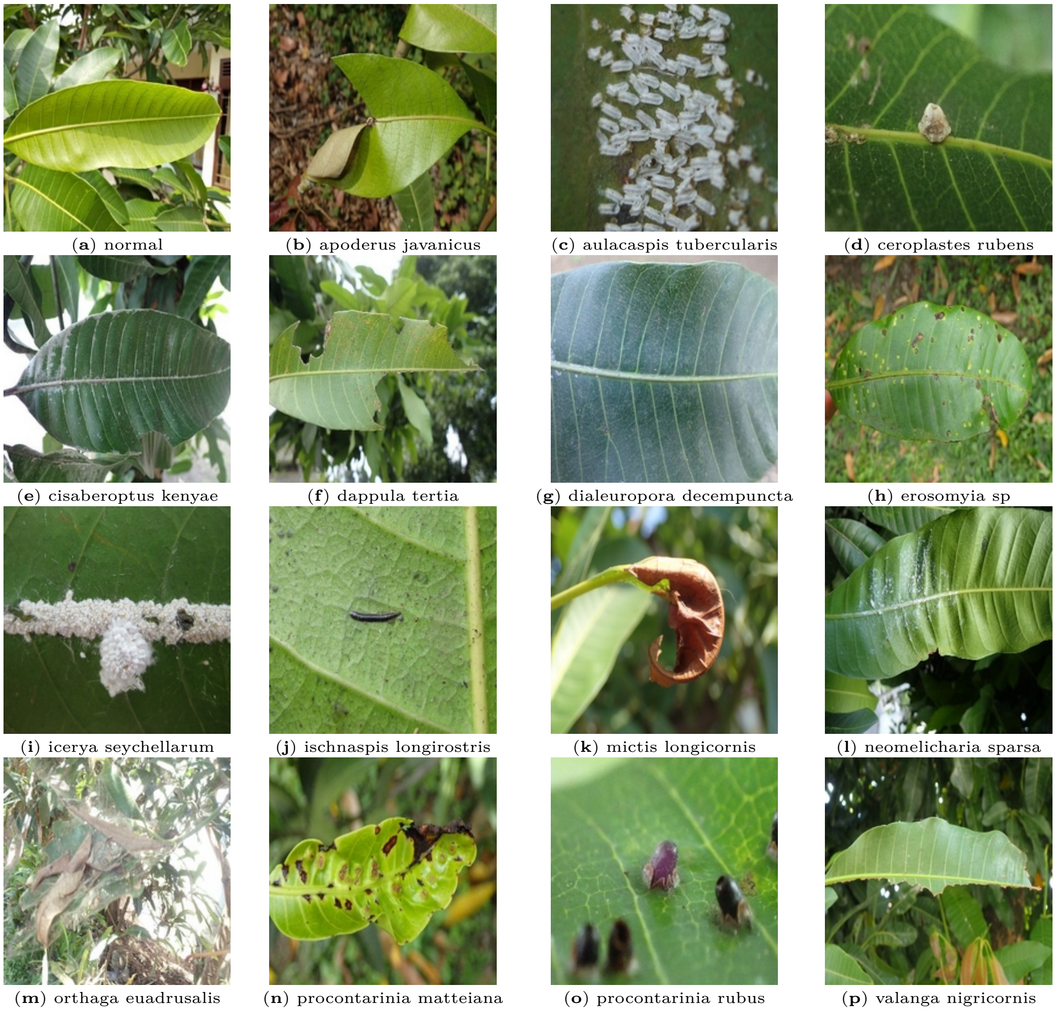


Figure 5.4: Data distribution for Pest classification in Mango farms dataset.

imbalanced class distribution.



(**a**) normal (**b**) apoderus javanicus (**c**) aulacaspis tubercularis (**d**) ceroplastes rubens

(**e**) cisaberoptus kenyae (**f**) dappula tertia (**g**) dialeuropora decempuncta (**h**) erosomyia sp

(**i**) icerya seychellarum (**j**) ischnaspis longirostris (**k**) mictis longicornis (**l**) neomelicharia sparsa

(**m**) orthaga euadrusalis (**n**) procontarinia matteiana (**o**) procontarinia rubus (**p**) valanga nigricornis

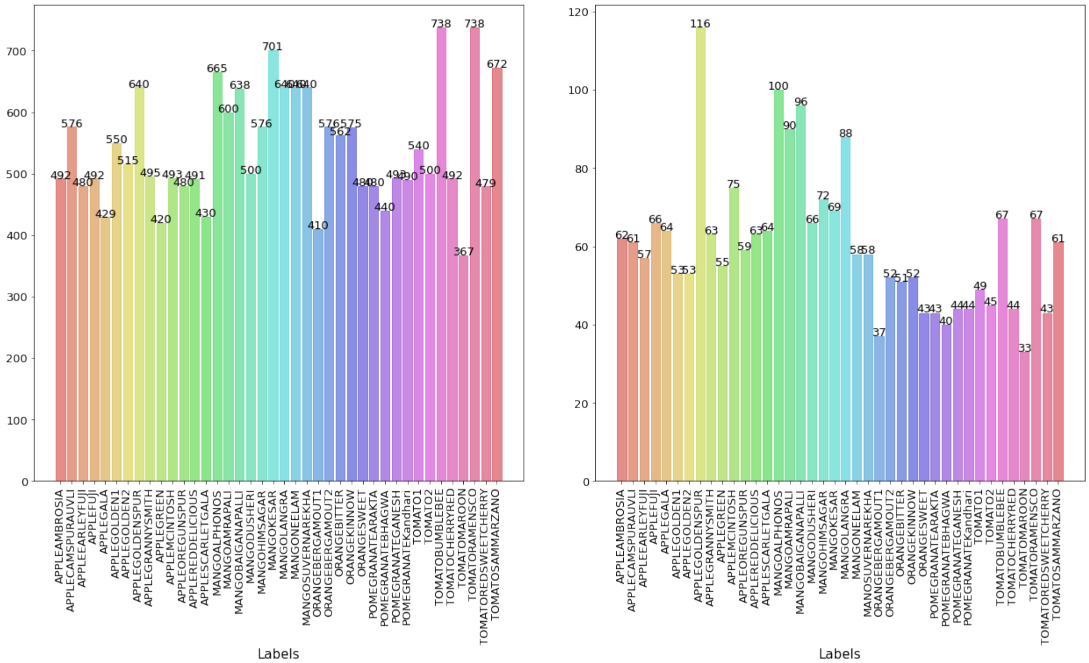
Figure 5.5: Sample images of Pest classification in Mango farms.

5.5 Colorectal Histology

The *colorectal histology* dataset [37] contains 5,000 histological images of different tissue types of colorectal cancer. It consists of 8 classes of tissue types with 625 images for each class as shown in Figure 5.7.

5.6 Fashion MNIST

The *Fashion MNIST* dataset [69] is a collection 60,000 training images of fashion products as shown in Figure 5.8. It consists of 28×28 grayscale images labeled by one of 10 classes. Since the dataset contains an equal number of images for each class,



(**a**) Training

(**b**) Testing

Figure 5.6: Data distribution for the Indian Fruits dataset.

there are 6,000 test images in each class resulting in a balanced dataset.

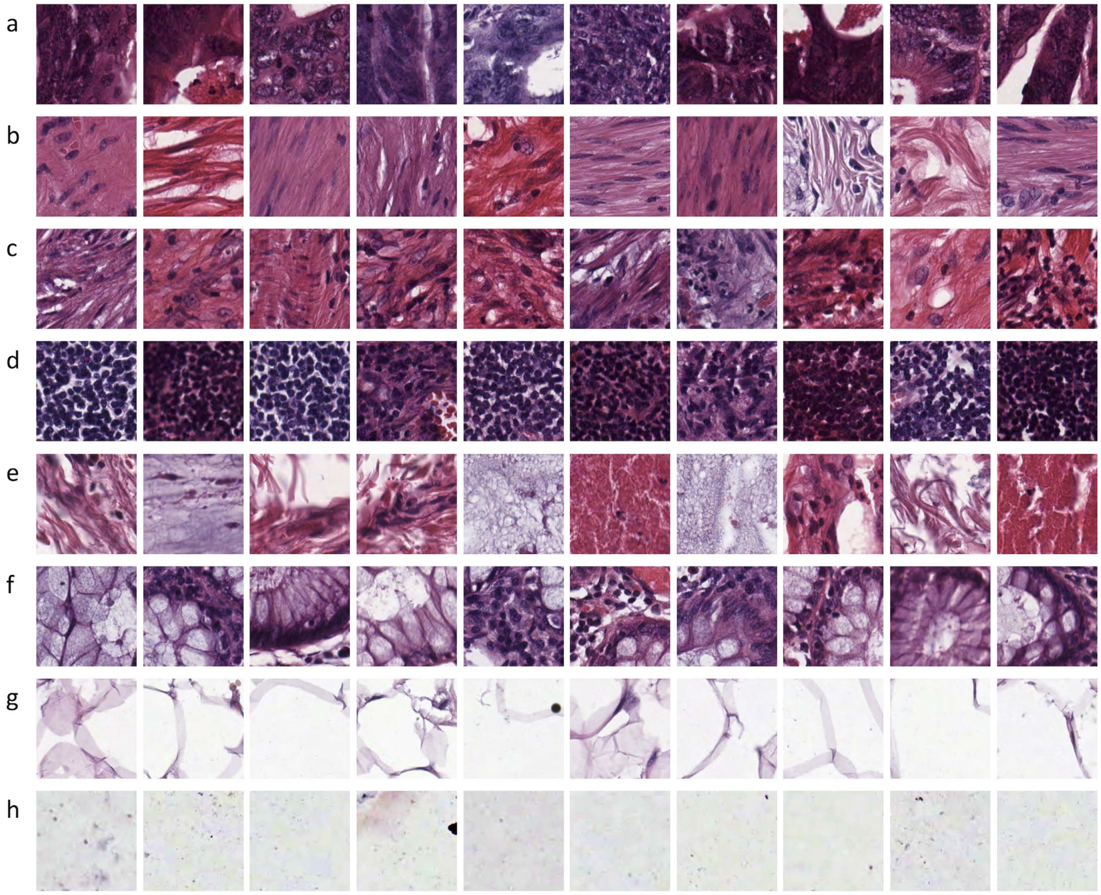
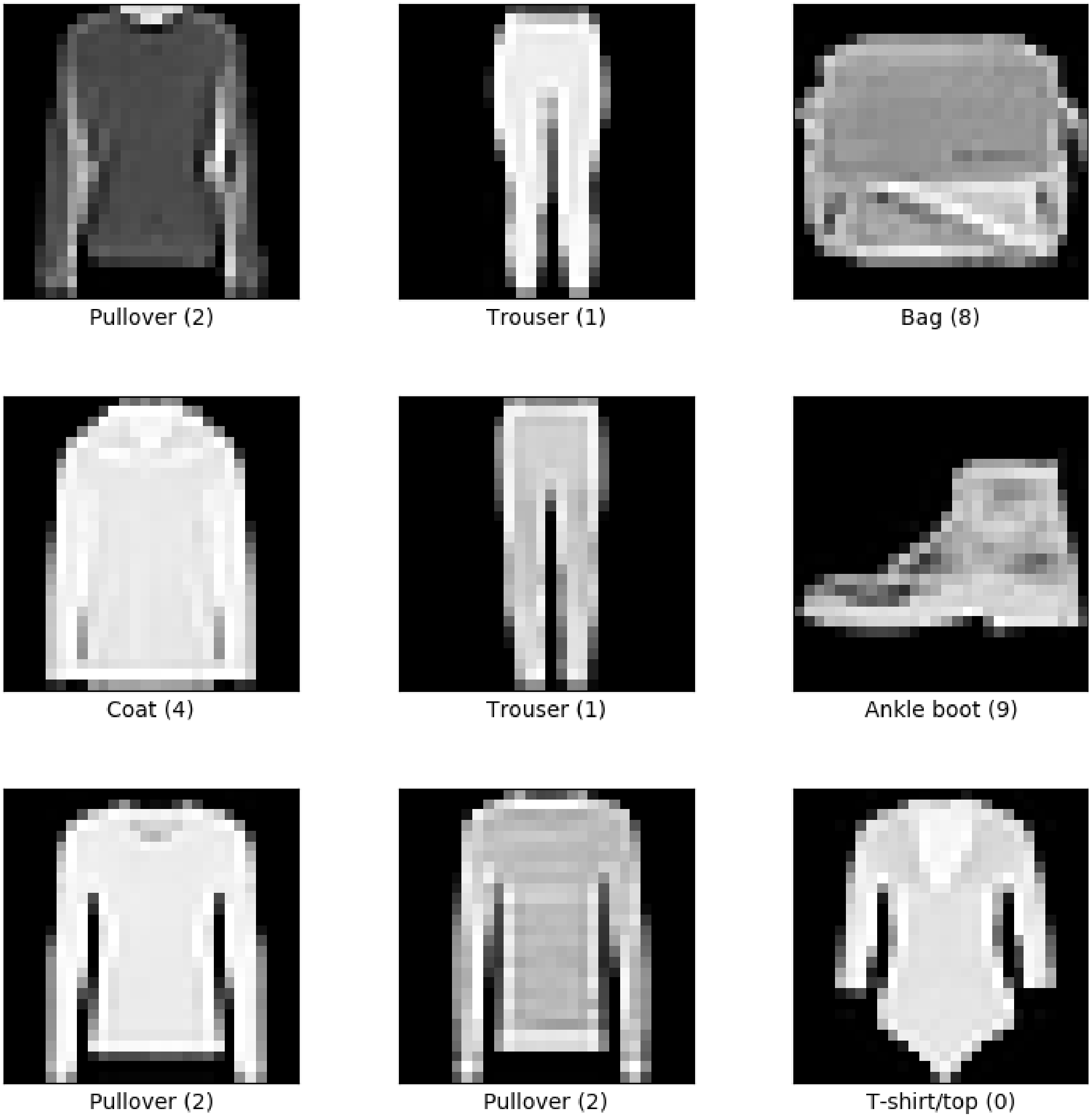


Figure 5.7: Example of colorectal cancer histology. (**a**) tumour epithelium, (**b**) simple stroma, (**c**) complex stroma, (**d**) immune cell conglomerates, (**e**) debris and mucus, (**f**) mucosal glands, (**g**) adipose tissue, (**h**) background

Table 5.1 gives the dataset characteristics in terms of various image datasets used in this work. Also, we provide the preprocessing time per each image. For example, the feature extraction time for the concrete dataset was 5 hours 12 minutes.



Pullover (2)

Trouser (1)

Bag (8)

Coat (4)

Trouser (1)

Ankle boot (9)

Pullover (2)

Pullover (2)

T-shirt/top (0)

Figure 5.8: Example of the Fashion MNIST dataset

Table 5.1: Datasets details with preprocessing times

Dataset	Size	Num of classes	Pixel Size	Balanced	time in secs/image
Concrete	40,000	2	227×227	Yes	0.4713
Mangopest	46,000	16	from 500×333 to 1280×853	No	0.5394
Indian fruits	23,848	40	100×100	No	0.4422
Fashion MNIST	60,000	10	28×28	Yes	0.4297
APTOS	3,662	5	227×227	No	0.5393
Colorectal histology	5,000	8	150×150	Yes	0.3218

Chapter 6

Machine Learning Implementations and Results

6.1 Brief Description

In this thesis, the following machine learning algorithms were implemented: Deep Residual Network, decision tree, random forest, k-nearest neighbours, support vector machine, XGBoost, and light GBM.

6.1.1 Deep Residual Network (ResNet)

Deep Residual Network suggested by [31] is an ensemble of VGG-19 [47], plain network, and residual network as a solution to the network depth-accuracy degradation problem. This is done by a residual learning framework which is a feedforward network with a shortcut. Multi-scale 1D ResNet is used in this thesis where multiscale refers to flexible convolutional kernels rather than flexible strides [45]. The authors use different sizes of kernels so that the network can learn features from original signals with different views with multiple scales.The structure of the model is described in Figure 6.1. The 1D ResNet model [45] consists of a number of subblock of the basic CNN blocks. A basic CNN block computes batch normalization after convolution for

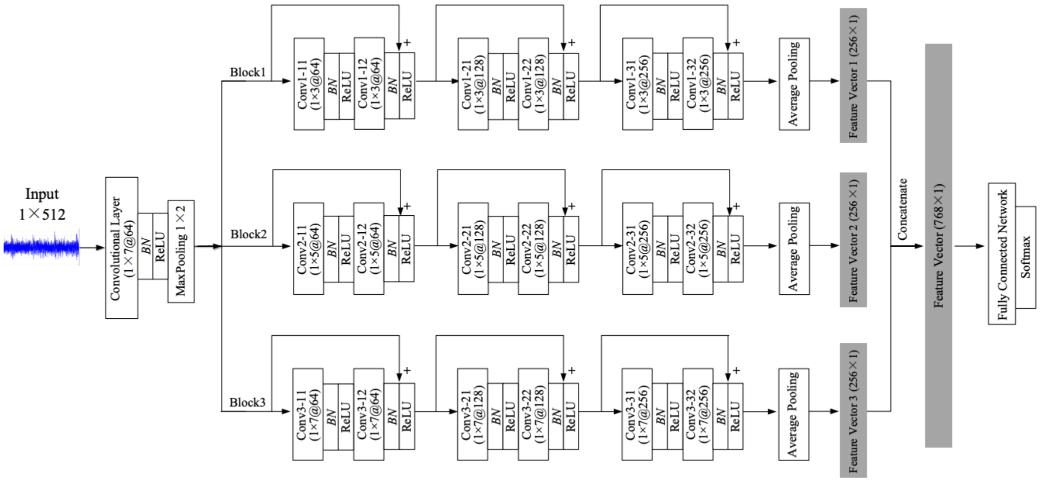


Figure 6.1: Structure of the Multi scale 1D ResNet [45]

input as,

$$\begin{aligned} y &= W \otimes x + b \\ s &= \mathrm{BN}(y) \\ h &= \mathrm{ReLU}(s) \end{aligned} \tag{6.1}$$

where $\otimes$ denotes convolution operator and BN is a batch normalization operator. Also, stacking two basic CNN blocks forms subblock of the basic CNN blocks as,

$$\begin{aligned} h_1 &= \mathrm{Basic}(x) \\ h_2 &= \mathrm{Basic}(h_1) \\ y &= h_2 + x \\ \hat{h} &= \mathrm{ReLU}(y) \end{aligned} \tag{6.2}$$

where Basic operator denotes the basic block as in 6.1. Following these process, it is possible to construct multiple subblocks of CNN with different kernel sizes.

For our experiments, for training the network, 100 epochs was used with a 0.01 learning rate. In addition, we used an early stopping option if there is no improvement in the validation loss after 20 epochs. Therefore, a number of epochs for each training experiment is different.

6.1.2 Classification and Regression Tree

Typically, the decision tree learns from top to down recursively, choosing the best attribute to construct the tree by partitioning the training data. There are several versions of decision tree algorithms such as $ID3$, $C4.5$, and $CART$ [53, 29]. The Classification and Regression Tree algorithm (CART) implementation was used in this study with the following parameters: *Gini* index as the criterion, *best* as splitter and unlimited depth.

6.1.3 Random Forest

Random forest is a widely used learning method and is an ensemble of multiple decision trees where the training set is drawn at random from distributions sampled independently and meant to reduce the impact of overfitting with a single tree [8]. 200 trees in the forest were used with *gini* as a criterion, and unlimited depth so nodes can be expanded until all leaves are pure.

6.1.4 k-Nearest Neighbors (kNN)

$k-$*nearest neighbors* is a distance-based non-parametric supervised learning [4] used for classification and regression problems. However, since it is sensitive to data with large dimensions, a proper choice of k becomes important. In our experiments, we use $k = 5$ and $Minkowski$ as a metric.

6.1.5 Support Vector Machines (SVM)

Support vector machines map input vectors into a hyperplane which implies high-dimensional space and construct an optimal separating hyperplane [68, 16]. When the hyperplane splits the data, it computes the distance, $margin$, between the hyperplane and its nearest attribute. SVM implements kernel functions which allows for attributes with higher dimensions to be separated linearly. Besides the linear case, SVM's based on polynomials, splines, radial basis function networks and multilayer perceptrons [64], have been successfully applied in several areas for example in life-sciences [52]. $Radial\ bias\ function$ (rbf), and a regularization parameter $C = 10$ was used in our experiments.

6.1.6 Gradient Boosting Machine (XGBoost, lightGBM)

While the random forest method is an ensemble bagging method using decision trees, *gradient boosting machine* (GBM) uses boosting [27] to train the model (decision trees) by adding new weak models consecutively with the negative gradient from the loss function and is one of the most successful machine learning models in recent years. XGBoost, extreme gradient boosting [14, 13], is widely applied in many fields due to its accuracy and rapid learning compare to the original GBM. Both XGBoost and lightGBM [39] are advanced models of gradient boosting machines. The main idea of this model is to make accurate predictions by combining some weak models so that it makes the model robust to outliers and flexible to customize [51]. XGBoost is a histogram-based algorithm that uses bins to split the features into a discrete, therefore, it is more efficient than the pre-sorted method of the conventional GBM. lightGBM combines two techniques: Gradient-based One-Side Sampling and Exclusive Feature Bundling [57]. Therefore, the main difference between LightGBM and XGBoost is that LightGBM uses a leaf-wise growth algorithm, whereas XGBoost is a level-wise growth algorithm. Because the leaf-wise growth algorithm compares nodes and uses leaves of higher gradient only, LightGBM is much faster than XGBoost. Both of algorithms utilize 1000 estimators ($n\_estimators = 1000$) with a base score of 0.5.

6.2 Implementation details

In this thesis, a variety of well-known supervised learning algorithms using packages from the Python ecosystem supported by scikit-learn [9] were used. All tests are conducted using a desktop workstation with Intel i7-9700K at 3.6 GHz, 8 CPU cores, 16GB RAM and Gigabyte GeForce RTX 2020 GPU. To a large extent, the implementation follows pipeline shown below:

$$\text{Data Collection} \rightarrow \text{Feature Engineering}$$
$$\rightarrow \text{Training the models} \rightarrow \text{Evaluating the model performance}$$

Data sets used for benchmarking were collected from various sources that include *Mendelay*, *Tensorflow dataset*, and *Kaggle competition*. Feature engineering and learning algorithms were implemented with Python libraries: Gudhi [65, 18] for calculating persistent homology, PyTorch [56] for modeling and execution of ResNet

1D, and scikit-learn [58] for implementation of other machine learning algorithms. Also, libraries such as NumPy [30] and pandas [48] were used for computing matrices and analyzing the data structure.

6.2.1 Evaluation Criteria

The performance of the machine learning models was evaluated using *accuracy* and *weighted F1 score*. After the model is trained, predicted labels are compared with true labels and these are separated as *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), and *False Negative* (FN).

Definition 14. *Accuracy* is the proportion of correctly predicted samples from all samples.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6.3}$$

The accuracy metric gives us intuitive performance of model, however, this metric has weakness in the case of imbalanced data. Therefore, we use F1 score.

Definition 15. *Precision* is the proportion of true positives from those samples predicted as true

$$Precision = \frac{TP}{TP + FP} \tag{6.4}$$

Definition 16. *Recall* is the proportion of true positive from those samples that are actually true.

$$Recall = \frac{TP}{TP + FN} \tag{6.5}$$

Definition 17. *F1 score* is the harmonic mean of the precision and recall.

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6.6}$$

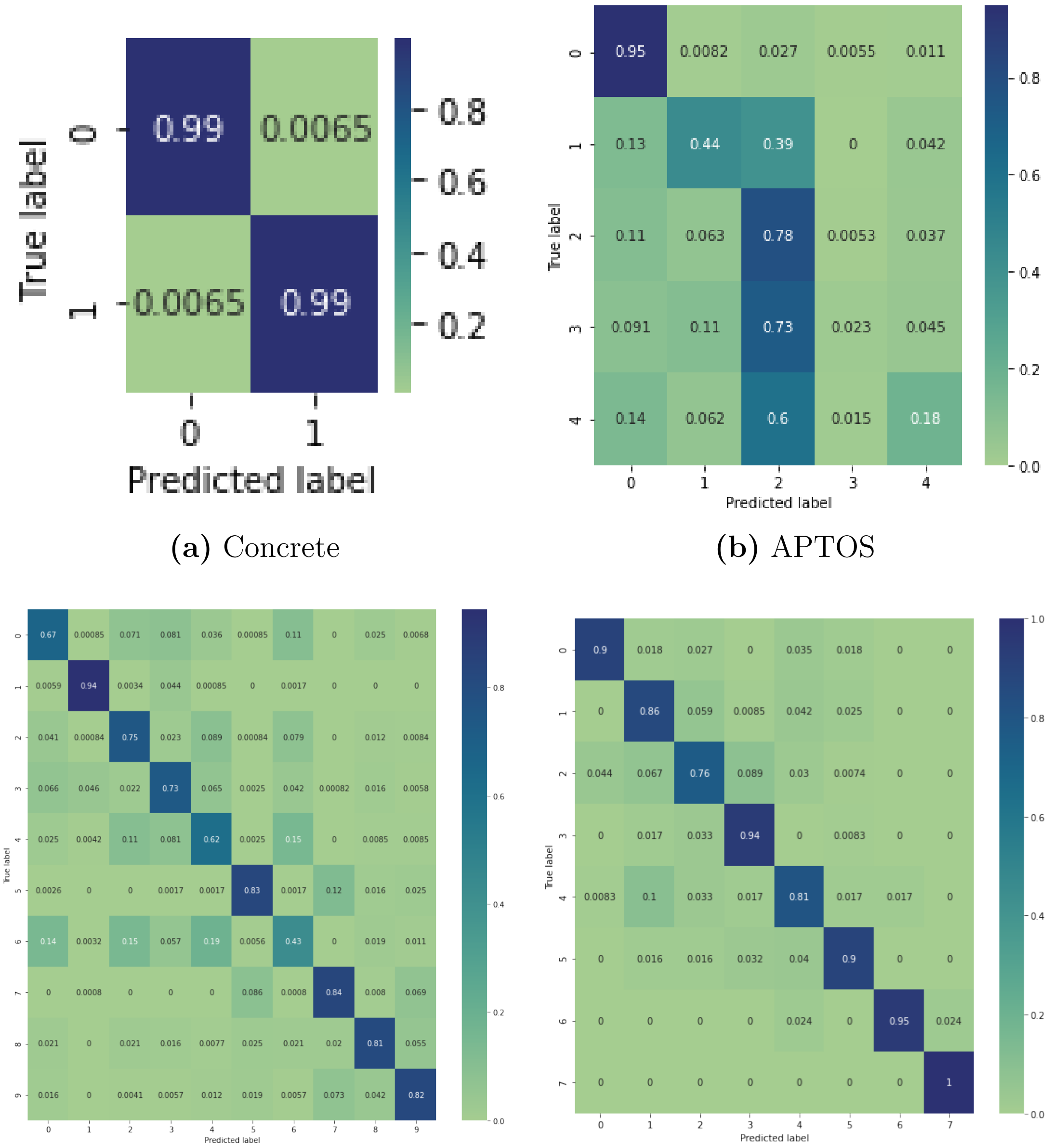
When $F1$ *score* is calculated for each label, it is called *weighted* $F1$ *score* which is suitable for imbalanced data. Since scikit-learn supports $F1$ *score* metric, we evaluate the performance using this library.

6.3 Analysis of results

Table 6.1 gives the accuracy, *weighted $F1$ score*, and run-time information for each of the datasets. In addition, the accuracy score reported with the benchmark datasets is given in the related works column. The best result is indicated in blue. We also revisit the characteristics of the datasets which is summarized in Table 5.1. Overall, ResNet 1D outperforms other ML algorithms while different types of gradient boosting machines show fairly good accuracy and weighted F1 score. When it comes to the binary classification problems, as in the Concrete dataset, most of the algorithms achieve 0.99 accuracy and F1 score. However, for the multi-class problem, the performance of SVM and kNN gets worse, mainly due to the difficulty of parameter setting. The proposed methods perform significantly worse than the benchmark with the Fashion MNIST and APTOS datasets. This is because it is hard to obtain good trainable topological signatures from the images that have low resolution even though Fashion MNIST was resized. In the case of the APTOS dataset, imbalanced training data is the main cause of poor results. Label 0 indicates the absence of diabetic retinopathy and has the highest number of images (See Figure 5.3). However, the presence of diabetic retinopathy can be found in 4 classes of which label 2 (severity level 2.0) has the most number of cases. As a result, more than half of examples were classified as label 2 (See Figure 6.2.(b)). Imbalanced data such as Mangopest and Indian fruits were classified well because there were sufficient training examples. In summary, the best classification performance using cubical homology with the ResNet 1D classifier was obtained for 3 out of 6 datasets. The topological signatures were not helpful in the classification of the Fashion MNIST and APTOS images. With the concrete dataset, the result is comparable with only slight difference ($\leq .005$) with the benchmark result. Confusion matrices that shows hard to classify types are given in Figures 6.2, 6.3 and 6.4. It is noteworthy that for these datasets, application of cubical homology has led to meaningful results in 4 out of 6 datasets.

Table 6.1: Accuracy and weighted F1 score for each dataset.

		ResNet 1D	Decision Tree	Gradient Boost	LightGBM	Random Forest	SVM	XGBoost	kNN	Related works
concrete	Accuracy	0.994	0.989	0.991	**0.9945**	0.993	0.956	0.9935	0.890	**0.999** with CNN [55]
	Weighted F1	0.994	0.988	0.989	0.994	0.992	0.955	0.993	0.884	
	run time	465.87	9.08	252.15	11.25	7.63	214.05	59.15	1.93	
mangopest	Accuracy	**0.931**	0.764	0.681	0.898	0.869	0.474	0.889	0.666	0.76 with CNN[44]
	Weighted F1	0.931	0.764	0.676	0.898	0.869	0.439	0.889	0.663	
	run time	760.94	17.17	5562.09	260.62	13.94	662.45	2041.22	2.33	
Indian fruits	Accuracy	**1.000**	0.9608	0.9608	0.9608	0.9608	0.7313	0.9608	0.676	0.999 SVM with deep features [5]
	Weighted F1 score	1.000	0.9608	0.9608	0.9608	0.9608	0.7236	0.9608	0.656	
	run time	271.21	4.44	4265.09	82.55	4.13	72.73	451.65	1.18	
Fashion MNIST	Accuracy	0.7427	0.567	0.696	**0.749**	0.693	0.535	0.746	0.397	**0.99** with CNN [38]
	Weighted F1	0.7414	0.569	0.694	0.749	0.692	0.529	0.746	0.390	
	run time	467.12	8.36	1808.07	89.37	7.66	935.21	1108.20	3.38	
APTOS	Accuracy	0.7326	0.698	0.760	**0.787**	0.782	0.674	0.775	0.655	**0.971** with CNN [66]
	Weighted F1	0.667	0.695	0.737	0.771	0.757	0.591	0.764	0.637	
	run time	61.81	0.63	86.02	13.16	0.70	3.49	42.34	0.08	
colorectal histology	Accuracy	**0.892**	0.75	0.842	0.869	0.855	0.679	0.874	0.759	0.874 with SVM[37]
	Weighted F1	0.89	0.727	0.832	0.850	0.834	0.686	0.843	0.743	
	run time	86.23	1.18	255.08	12.52	1.10	4.06	44.63	0.14	
	Accuracy	**0.882±0.109**	0.789±0.147	0.822±0.121	0.876±0.087	0.856±0.102	0.675±0.154	0.873±0.90	0.674±0.148	
	Weighted F1	**0.871±0.125**	0.784±0.148	0.815±0.124	0.870±0.091	0.851±0.105	0.654±0.164	0.866±0.092	0.662±0.147	



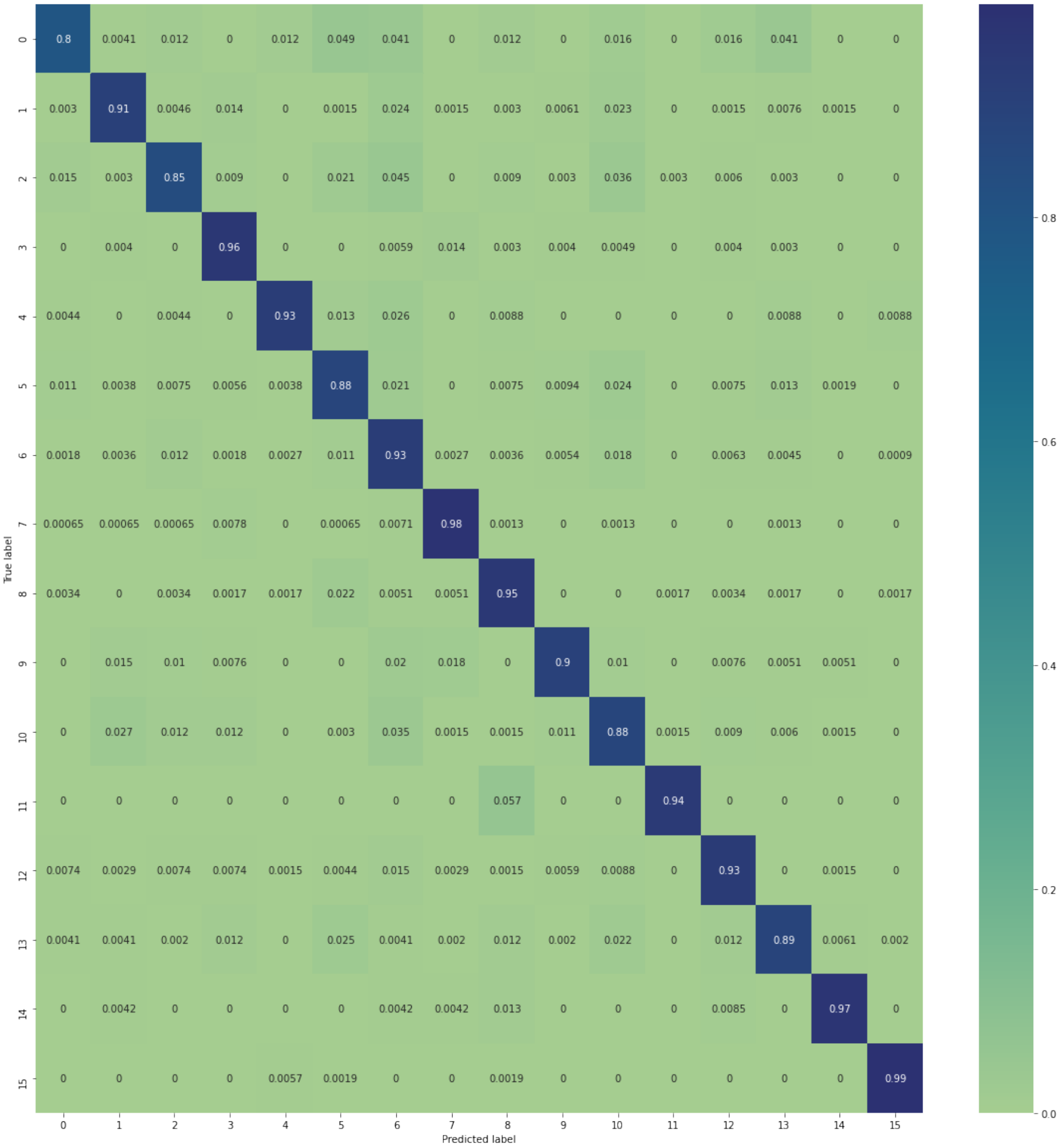
(a) Concrete

(b) APTOS

(c) Fashion MNSIT

(d) Colorectal histology

Figure 6.2: Confusion matrices with ResNet 1D



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.8	0.0041	0.012	0	0.012	0.049	0.041	0	0.012	0	0.016	0	0.016	0.041	0	0
1	0.003	0.91	0.0046	0.014	0	0.0015	0.024	0.0015	0.003	0.0061	0.023	0	0.0015	0.0076	0.0015	0
2	0.015	0.003	0.85	0.009	0	0.021	0.045	0	0.009	0.003	0.036	0.003	0.006	0.003	0	0
3	0	0.004	0	0.96	0	0	0.0059	0.014	0.003	0.004	0.0049	0	0.004	0.003	0	0
4	0.0044	0	0.0044	0	0.93	0.013	0.026	0	0.0088	0	0	0	0	0.0088	0	0.0088
5	0.011	0.0038	0.0075	0.0056	0.0038	0.88	0.021	0	0.0075	0.0094	0.024	0	0.0075	0.013	0.0019	0
6	0.0018	0.0036	0.012	0.0018	0.0027	0.011	0.93	0.0027	0.0036	0.0054	0.018	0	0.0063	0.0045	0	0.0009
7	0.00065	0.00065	0.00065	0.0078	0	0.00065	0.0071	0.98	0.0013	0	0.0013	0	0	0.0013	0	0
8	0.0034	0	0.0034	0.0017	0.0017	0.022	0.0051	0.0051	0.95	0	0	0.0017	0.0034	0.0017	0	0.0017
9	0	0.015	0.01	0.0076	0	0	0.02	0.018	0	0.9	0.01	0	0.0076	0.0051	0.0051	0
10	0	0.027	0.012	0.012	0	0.003	0.035	0.0015	0.0015	0.011	0.88	0.0015	0.009	0.006	0.0015	0
11	0	0	0	0	0	0	0	0	0.057	0	0	0.94	0	0	0	0
12	0.0074	0.0029	0.0074	0.0074	0.0015	0.0044	0.015	0.0029	0.0015	0.0059	0.0088	0	0.93	0	0.0015	0
13	0.0041	0.0041	0.002	0.012	0	0.025	0.0041	0.002	0.012	0.002	0.022	0	0.012	0.89	0.0061	0.002
14	0	0.0042	0	0	0	0	0.0042	0.0042	0.013	0	0	0	0.0085	0	0.97	0
15	0	0	0	0	0.0057	0.0019	0	0	0.0019	0	0	0	0	0	0	0.99

Figure 6.3: Confusion matrix for the Mango pest dataset with ResNet 1D implementation

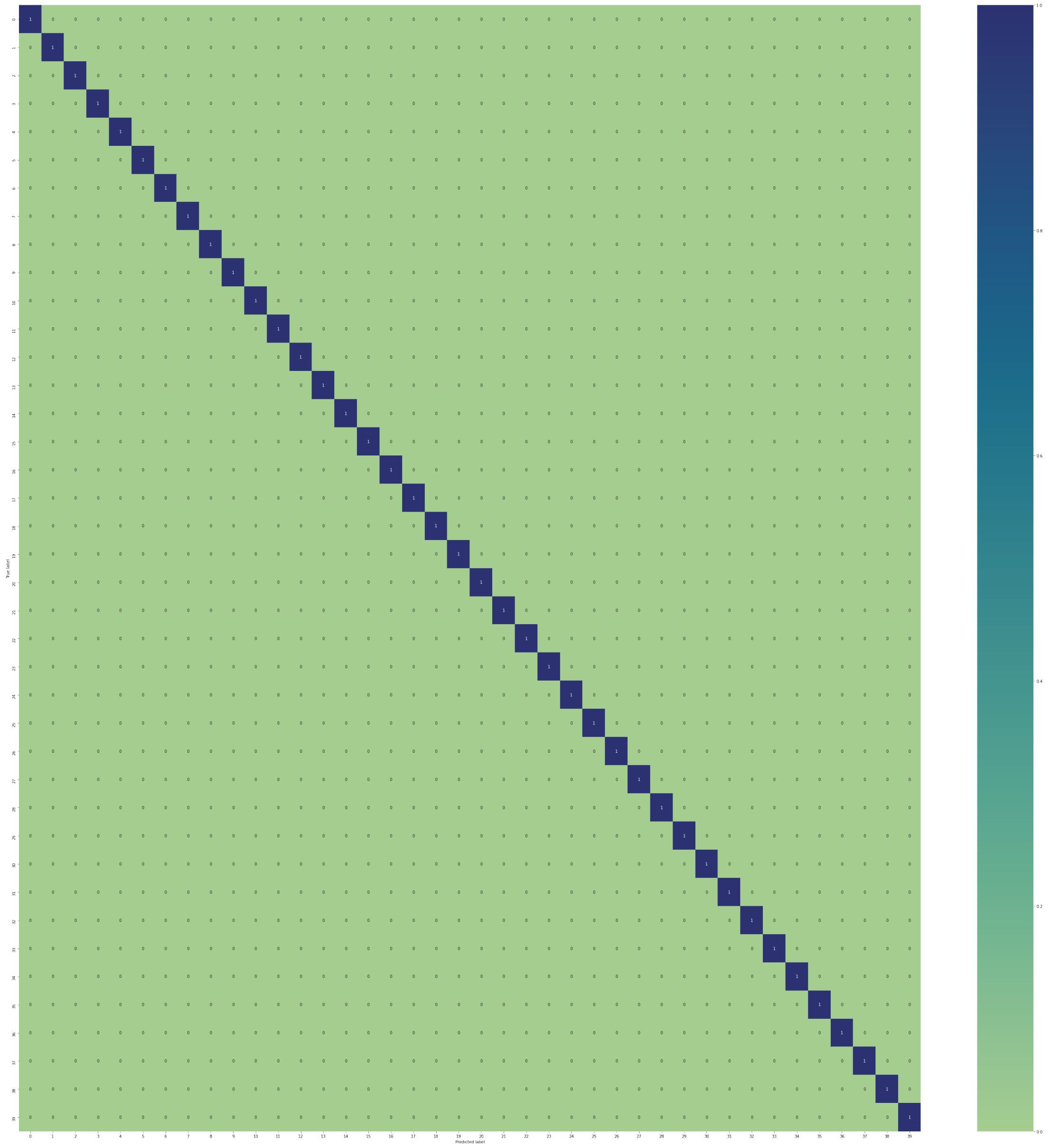


Figure 6.4: Confusion matrix for the Indian fruits dataset with ResNet 1D implementation

Chapter 7

Conclusion

In this thesis, we address the problem of construction of feature vectors based on cubical homology for different types of 2D images with varying pixel sizes, classes and distributions. We then study the effect of these topological features on selected machine learning models. In the process of extracting features, we proposed a *score* that filters out low peristence features and transforms the input image into a 1-dimensional array. We implemented ResNet 1D, lightGBM, XGBoost, and other well-known ML methods with the data obtained from our proposed feature engineering process. We used the accuracy, weighted F1 score, and execution time to compare the classification performance of 8 algorithms. Our experiments demonstrate that in three out of six datasets, our proposed method outperforms the results from the benchmark methods. However, with two datasets, the performance of our proposed method is poor, due to low resolution and imbalanced dataset respectively. This work reveals that application of cubical homology to image classification shows promise especially with ResNetID deep learning algorithm.

7.1 Future Research Directions

Since conversion of input images to 2D data, is very time consuming, future work will involve seeking more efficient ways to reduce the time for pre-processing.

- **Score** Although the proposed scoring method works well for most image datasets, we may consider how this score handles the situation when an image has low-resolution and data distribution is imbalanced. In addition, optimization of the threshold value is also part of the future work.

- **Pre-processing** The run time for pre-processing is significant for the entire dataset. For example, in the worst case, the pre-processing time for the $Fashion$ $MNIST$ dataset was 7.1 hours. Decreasing the pre-processing time will contribute to the overall performance of the pipeline.

- **Deep learning model** ResNet 1D model is designed to use 512 features as input. However, we get far less than 512 topological features for each image so that the data contains dispensable features. Therefore, we may improve the ResNet 1D model by designing the model to require fewer features.

- **Feature Engineering** Future work would also involve performing correlation analysis to evaluate the importance of the derived features.

Appendix A

Snapshots from the Experiments

In this appendix, we present some snapshots from our experiments especially feature engineering in order to provide the reader a better illustration of our experimental environment.

A.1 Feature Engineering

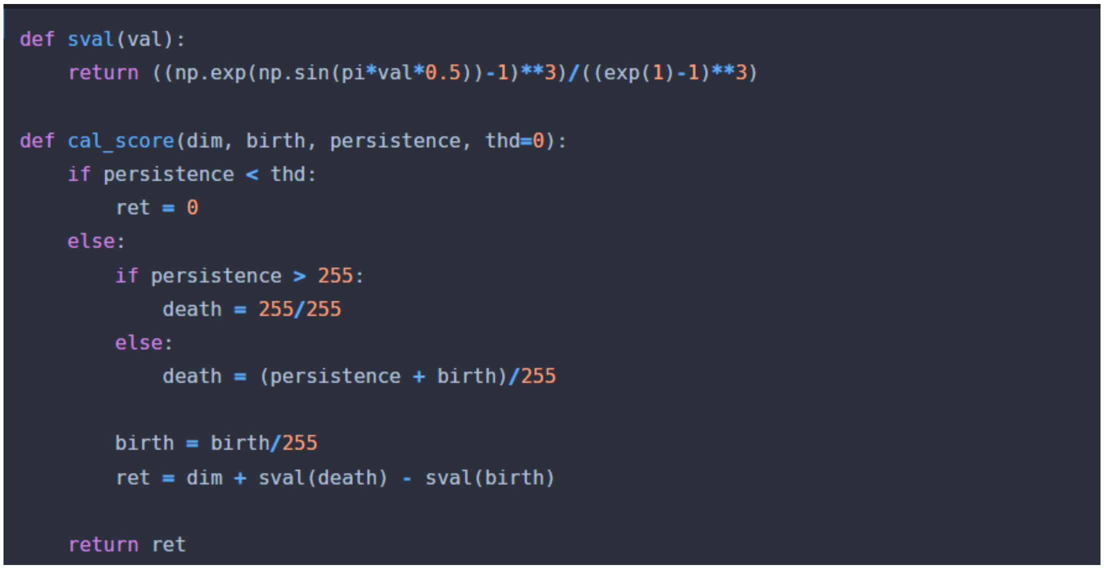
A.1.1 Score Metric

In Figure A.1 we give a snapshot from definition of the *score* which we proposed in Section 4.1. To ignore noise, we set threshold for persistence less than 10. Also, for convenient calculation, death will be 1 if a component persists infinitely.

A.1.2 Extraction of Topological Feature

Persistent homology is computed by `GUDHI` library and this library requires perseus format for cubical homology. Perseus format consists of multiple lines of single number where the first line indicates the dimension of the image (2 in this case), second & third lines indicate number of cubes each dimension, and following lines are birth time of each cube from bottom left to top right. Figure A.2 shows the process of converting the image to perseus format. We regard each pixel as cube and its value as its time of birth.

Figure A.3 shows part of the process of extraction of topological features. We get `p_data` after computing persistent homology, score will be calculated for each dimension of subcomplex. Then, we arrange scores in descending orders.

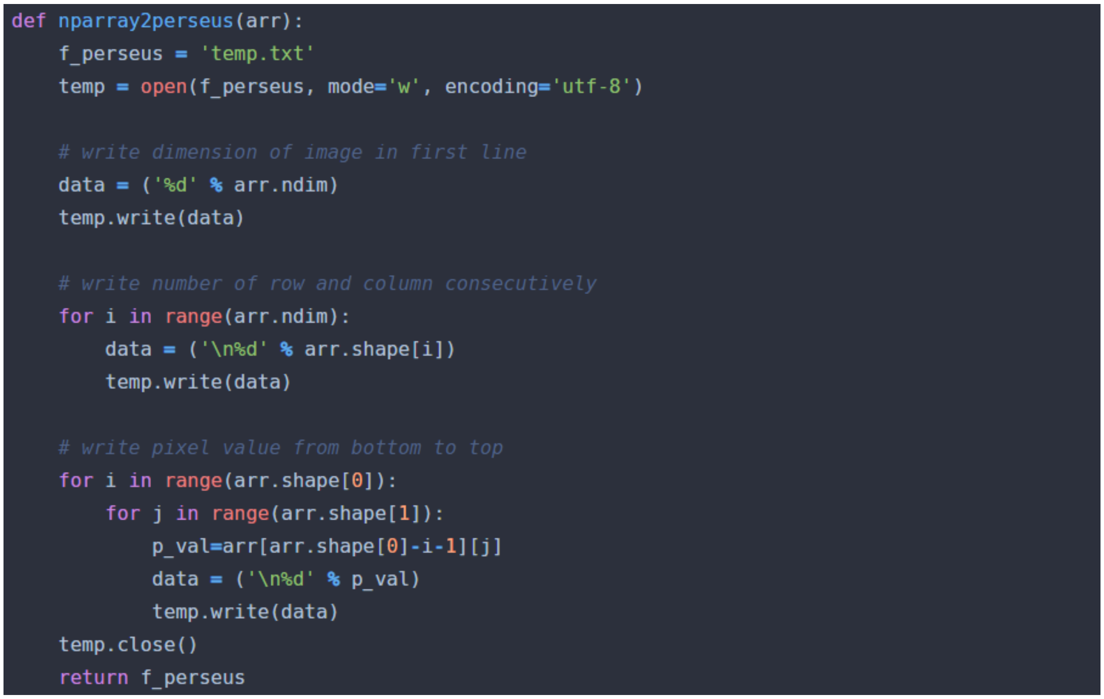


```
def sval(val):
    return ((np.exp(np.sin(pi*val*0.5))-1)**3)/((exp(1)-1)**3)

def cal_score(dim, birth, persistence, thd=0):
    if persistence < thd:
        ret = 0
    else:
        if persistence > 255:
            death = 255/255
        else:
            death = (persistence + birth)/255

        birth = birth/255
        ret = dim + sval(death) - sval(birth)

    return ret
```

Figure A.1: Definition of score function.

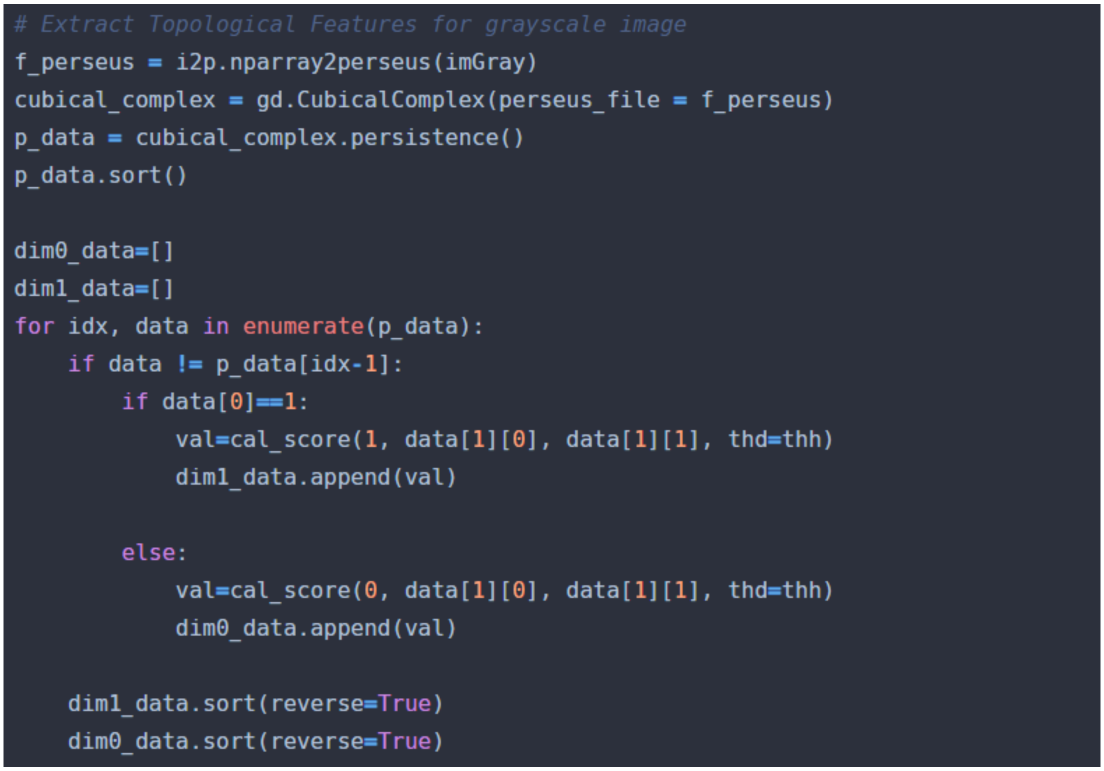
A.1.3 GLCM computation

In Figure A.4 we show a snapshot of GLCM computation. We define 3 distances, 4 angles, and 2 properties. Therefore, we get 24 GLCM features after computation.



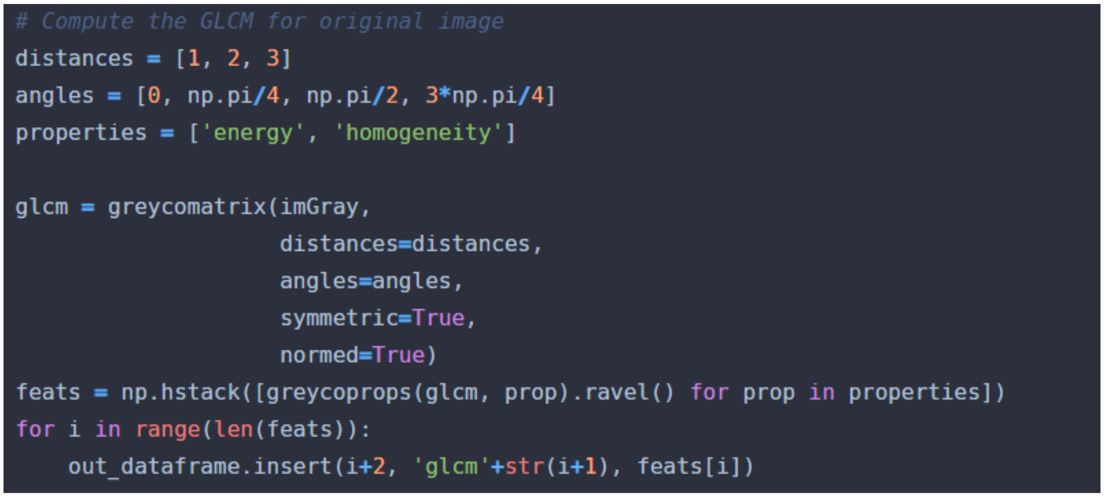
```
def nparray2perseus(arr):
    f_perseus = 'temp.txt'
    temp = open(f_perseus, mode='w', encoding='utf-8')

    # write dimension of image in first line
    data = ('%d' % arr.ndim)
    temp.write(data)

    # write number of row and column consecutively
    for i in range(arr.ndim):
        data = ('\n%d' % arr.shape[i])
        temp.write(data)

    # write pixel value from bottom to top
    for i in range(arr.shape[0]):
        for j in range(arr.shape[1]):
            p_val=arr[arr.shape[0]-i-1][j]
            data = ('\n%d' % p_val)
            temp.write(data)
    temp.close()
    return f_perseus
```

Figure A.2: Converting image to perseus format.



```
# Extract Topological Features for grayscale image
f_perseus = i2p.nparray2perseus(imGray)
cubical_complex = gd.CubicalComplex(perseus_file = f_perseus)
p_data = cubical_complex.persistence()
p_data.sort()

dim0_data=[]
dim1_data=[]
for idx, data in enumerate(p_data):
    if data != p_data[idx-1]:
        if data[0]==1:
            val=cal_score(1, data[1][0], data[1][1], thd=thh)
            dim1_data.append(val)

        else:
            val=cal_score(0, data[1][0], data[1][1], thd=thh)
            dim0_data.append(val)

    dim1_data.sort(reverse=True)
    dim0_data.sort(reverse=True)
```

Figure A.3: Extraction of topological features.



```
# Compute the GLCM for original image
distances = [1, 2, 3]
angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]
properties = ['energy', 'homogeneity']

glcm = greycomatrix(imGray,
                    distances=distances,
                    angles=angles,
                    symmetric=True,
                    normed=True)
feats = np.hstack([greycoprops(glcm, prop).ravel() for prop in properties])
for i in range(len(feats)):
    out_dataframe.insert(i+2, 'glcm'+str(i+1), feats[i])
```

Figure A.4: Extraction of GLCM features.

Bibliography

[1] Henry Adams, Sofya Chepushtanova, Tegan Emerson, Eric Hanson, Michael Kirby, Francis Motta, Rachel Neville, Chris Peterson, Patrick Shipman, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology, 2016.

[2] Mehmet Emin Aktas, Esra Akbas, and Ahmed El Fatmaoui. Persistence homology of networks: Methods and applications, 2019.

[3] Madjid Allili, Konstantin Mischaikow, and Allen Tannenbaum. Cubical homology and the topological classification of 2d and 3d imagery. In *Proceedings 2001 international conference on image processing (Cat. No. 01CH37205)*, volume 2, pages 173–176. IEEE, 2001.

[4] N.S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] Santi Kumari Behera, Amiya Kumar Rath, and Prabira Kumar Sethy. Fruit recognition using support vector machine based on deep features. *Karbala International Journal of Modern Science*, 6(2):16, 2020.

[6] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[7] 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.

[8] Leo Breiman. Random forests. *Journal of Machine Learning*, 45(126):5–32, 2001.

[9] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort,

Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[10] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[11] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics*, pages 2786–2796. PMLR, 2020.

[12] F. Chazal and B. Michel. An Introduction to Topological Data Analysis: Fundamental and Practical aspects for Data Scientists. *arXiv*, 1710(04019):1–38, 2017.

[13] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. 2016.

[14] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[15] Moo K Chung, Hyekyoung Lee, Alex DiChristofano, Hernando Ombao, and Victor Solo. Exact topological inference of the resting-state brain networks in twins. *Network Neuroscience*, 3(3):674–694, 2019.

[16] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[17] Tamal Dey, Sayan Mandal, and William Varcho. Improved image classification using topological persistence. In *Proceedings of the conference on Vision, Modeling and Visualization*, pages 161–168, 2017.

[18] Pawel Dlotko. Cubical complex. In *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.4.1 edition, 2021.

[19] Arjuna P. H. Don, James F. Peters, Sheela Ramanna, and Arturo Tozzi. Topological view of flows inside the bold spontaneous activity of the human brain. *Frontiers in Computational Neuroscience*, 14:34, 2020.

[20] Arjuna P.H. Don, James F. Peters, Sheela Ramanna, and Arturo Tozzi. Quaternionic views of rs-fmri hierarchical brain activation regions. discovery of multi-level brain activation region intensities in rs-fmri video frames. *Chaos, Solitons and Fractals*, 152:111351, 2021.

[21] Olga Dunaeva, Herbert Edelsbrunner, Anton Lukyanov, Michael Machin, Daria Malkova, Roman Kuvaev, and Sergey Kashin. The classification of endoscopy images with persistent homology. *Pattern Recognition Letters*, 83:13–22, 2016.

[22] H. Edelsbrunner and J.L. Harer. Persistent homology. a survey. *Contemporary Mathematics*, 453:257–282, 2008.

[23] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *41st Annual Symposium on Foundations of Computer Science*, pages 454–463. IEEE Comput. Soc. Press, Los Alamitos, California, 2000.

[24] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Springer Discrete and Computational Geometry*, 28(4):511–533, 2001.

[25] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000.

[26] Matthew Fisher, Boris Springborn, Peter Schröder, and Alexander I Bobenko. An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing. *Computing*, 81(2-3):199–213, 2007.

[27] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[28] Adélie Garin and Guillaume Tauzin. A topological" reading" lesson: Classification of mnist using tda. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1551–1556. IEEE, 2019.

[29] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining concepts and techniques, third edition.* Morgan Kaufmann Publishers, Waltham, Mass., 2012.

[30] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[31] 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.

[32] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. *arXiv preprint arXiv:1707.04041*, 2017.

[33] Christoph D. Hofer, Roland Kwitt, and Marc Niethammer. Learning representations of persistence barcodes. *Journal of Machine Learning Research*, 20(126):1–45, 2019.

[34] Taizo Iijima. Basic theory on the normalization of pattern (in case of typical one-dimensional pattern). *Bulletin of Electro-technical Laboratory*, 26:368–388, 1962.

[35] Tomasz Kaczynski, Konstantin Michael Mischaikow, and Marian Mrozek. *Computational homology*, volume 3. Springer, 2004.

[36] William D Kalies, Konstantin Mischaikow, and Greg Watson. Cubical approximation and computation of homology. *Banach Center Publications*, 47:115–131, 1999.

[37] Jakob Nikolas Kather, Cleo-Aron Weis, Francesco Bianconi, Susanne M Melchers, Lothar R Schad, Timo Gaiser, Alexander Marx, and Frank Gerrit Z"ollner. Multi-class texture analysis in colorectal cancer histology. *Scientific reports*, 6:27988, 2016.

[38] Mohammed Kayed, Ahmed Anter, and Hadeer Mohamed. Classification of garments from fashion mnist dataset using cnn lenet-5 architecture. In *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, pages 238–243. IEEE, 2020.

[39] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[40] Rolando Kindelan, José Frías, Mauricio Cerda, and Nancy Hitschfeld. Classification based on topological data analysis, 2021.

[41] Piotr Kot. Homology calculation of cubical complexes in rn. *Computational Methods In Science And Technology*, 12(2):115–121, 2006.

[42] Miroslav Kramár, Arnaud Goullet, Lou Kondic, and Konstantin Mischaikow. Persistence of force networks in compressed granular media. *Physical Review E*, 87(4):042207, 2013.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[44] Kusrini Kusrini, Suputa Suputa, Arief Setyanto, I Made Artha Agastya, Herlambang Priantoro, Krishna Chandramouli, and Ebroul Izquierdo. Data augmentation for automated pest classification in mango farms. *Computers and Electronics in Agriculture*, 179:105842, 2020.

[45] Ruonan Liu, Fei Wang, Boyuan Yang, and S Joe Qin. Multiscale kernel based residual convolutional neural network for motor fault diagnosis under nonstationary conditions. *IEEE Transactions on Industrial Informatics*, 16(6):3797–3806, 2019.

[46] Andrew Marchese. Data analysis methods using persistence diagrams. 2017.

[47] Muhammad Mateen, Junhao Wen, Sun Song, Zhouping Huang, et al. Fundus image classification using vgg-19 architecture with pca and svd. *Symmetry*, 11(1):1, 2019.

[48] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[49] P Mohanaiah, P Sathyanarayana, and L GuruKumar. Image texture feature extraction using glcm approach. *International journal of scientific and research publications*, 3(5):1–5, 2013.

[50] Takenobu Nakamura, Yasuaki Hiraoka, Akihiko Hirata, Emerson G Escolar, and Yasumasa Nishiura. Persistent homology and many-body atomic structure for medium-range order in the glass. *Nanotechnology*, 26(30):304001, 2015.

[51] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[52] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[53] Armin Ott and Alexander Hapfelmeier. Nonparametric subgroup identification by prim and cart: A simulation and application study. *Computational and mathematical methods in medicine*, 2017, 2017.

[54] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6:1–38, 2017.

[55] Ç F Özgenel and A Gönenç Sorguç. Performance comparison of pretrained convolutional neural networks on crack detection in buildings. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 35, pages 1–8. IAARC Publications, 2018.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[57] Vrushang Patel, Seungho Choe, and Talal Halabi. Predicting future malware attacks on cloud systems using machine learning. In *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 151–156. IEEE, 2020.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[59] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[60] Chi Seng Pun, Kelin Xia, and Si Xian Lee. Persistent-homology-based machine learning and its applications – a survey, 2018.

[61] Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4741–4748, 2015.

[62] Joseph J Rotman. *An introduction to algebraic topology*, volume 119. Springer Science & Business Media, 2013.

[63] Daniel Strömbom. Persistent homology in the cubical setting: theory, implementations and applications, 2007.

[64] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[65] The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.4.1 edition, 2021.

[66] Borys Tymchenko, Philip Marchenko, and Dmitry Spodarets. Deep learning approach to diabetic retinopathy detection. *arXiv preprint arXiv:2003.02261*, 2020.

[67] Yuhei Umeda. Time series classification via topological data analysis. *Information and Media Technologies*, 12:228–239, 2017.

[68] Vapnik V. *The nature of statistical learning theory.* Springer-Verlag, New-York, 1995.

[69] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[70] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33(2):249–274, 2005.

[71] A. F. Zomorodian. *Computing and Comprehending Topology: Persistence and Hierarchical Morse Complexes.* PhD thesis, University of Illinois at Urbana-Champaign, Department of Computer Science, 2001.