Data Reduction Algorithms in Machine Learning and Data Science

by

Benyamin Ghojogh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Author's declaration

Statement of Contributions

The papers published out of this paper are listed in Appendix A. In this appendix, I have listed the papers published during my PhD, supervised by my supervisors. There are also some published papers, which were left from my master's and I do not list them in the appendix.

As can be seen in this list, there are some coauthors for papers, which I list them here. I only list the coauthors of papers whose materials are provided in this thesis:

- Professor coauthors: These professors have supervised me and my coauthors in writing the corresponding papers.
  - In most papers: Prof. Mark Crowley, Prof. Fakhri Karray. In the histopathology papers (applications of data reduction for medical image analysis): Prof. H.R. Tizhoosh. In some tutorial papers on manifold learning: Prof. Ali Ghodsi
- Student coauthors: There are some papers/sections on which I was the sole contributor. On some other papers/sections, these coauthors have contributed almost equally in doing the corresponding projects. Note that in these papers, the names are starred and the order of names on papers are completely random. All the starred names on papers are the first authors.
  - Paper [60] (Sections 3.3.2.1 and 5.1.3): Idea designing by B. Ghojogh. Equally coding by B. Ghojogh, M. Sikaroudi, and S. Shafiei. Paper writing by B. Ghojogh.
  - Paper [122] (Section 5.1.2): Idea designing by A. Safarpoor and S. Shafiei. Equally coding by B. Ghojogh, M. Sikaroudi. Equally paper writing by B. Ghojogh, M. Sikaroudi, A. Safarpoor, and S. Shafiei.
  - Paper [55] (Section 3.1.1): Idea designing and paper writing by B. Ghojogh. Equally coding by B. Ghojogh and M. Sikaroudi.
  - Paper [93] (Section 4.3.1): Idea designing by Prof. Mark Crowley and M. N. Samad. Equally coding by B. Ghojogh, H. Ma, M. N. Samad, and D. Zheng. Equally paper writing by B. Ghojogh, H. Ma, M. N. Samad, and Prof. Mark Crowley. Note that the iMondrian method was not my main project and I was a collaborator/coauthor in proposing this method.
  - Paper [120] (Section 5.1.4): Equally idea designing by Prof. H. R. Tizhoosh, M. Sikaroudi, B. Ghojogh, and A. Safarpoor. Equally coding by B. Ghojogh

  and M. Sikaroudi. Equally paper writing by B. Ghojogh, M. Sikaroudi, and A. Safarpoor.

  - Paper [119] (Sections 3.3.2.2 and 5.1.5): Equally idea designing, coding and paper writing by B. Ghojogh and M. Sikaroudi.

- There are some other coauthors, but in the papers whose materials are not provided in this thesis. To see other coauthors, please refer to Appendix A.

Abstract

Raw data are usually required to be pre-processed for better representation or discrimination of classes. This pre-processing can be done by data reduction, i.e., either reduction in dimensionality or numerosity (cardinality). Dimensionality reduction can be used for feature extraction or data visualization. Numerosity reduction is useful for ranking data points or finding the most and least important data points. This thesis proposes several algorithms for data reduction, known as dimensionality and numerosity reduction, in machine learning and data science. Dimensionality reduction tackles feature extraction and feature selection methods while numerosity reduction includes prototype selection and prototype generation approaches. This thesis focuses on feature extraction and prototype selection for data reduction. Dimensionality reduction methods can be divided into three categories, i.e., spectral, probabilistic, and neural network-based methods. The spectral methods have a geometrical point of view and are mostly reduced to the generalized eigenvalue problem. Probabilistic and network-based methods have stochastic and information theoretic foundations, respectively. Numerosity reduction methods can be divided into methods based on variance, geometry, and isolation.

For dimensionality reduction, under the spectral category, I propose weighted Fisher discriminant analysis, Roweis discriminant analysis, and image quality aware embedding. I also propose quantile-quantile embedding as a probabilistic method where the distribution of embedding is chosen by the user. Backprojection, Fisher losses, and dynamic triplet sampling using Bayesian updating are other proposed methods in the neural network-based category. Backprojection is for training shallow networks with a projection-based perspective in manifold learning. Two Fisher losses are proposed for training Siamese triplet networks for increasing and decreasing the inter- and intra-class variances, respectively. Two dynamic triplet mining methods, which are based on Bayesian updating to draw triplet samples stochastically, are proposed. For numerosity reduction, principal sample analysis and instance ranking by matrix decomposition are the proposed variance-based methods; these methods rank instances using inter-/intra-class variances and matrix factorization, respectively. Curvature anomaly detection, in which the points are assumed to be the vertices of polyhedron, and isolation Mondrian forest are the proposed methods based on geometry and isolation, respectively.

To assess the proposed tools developed for data reduction, I apply them to some applications in medical image analysis, image processing, and computer vision. Data reduction, used as a pre-processing tool, has different applications because it provides various ways of feature extraction and prototype selection for applying to different types of data. Dimensionality reduction extracts informative features and prototype selection selects the most informative data instances. For example, for medical image analysis, I use Fisher losses and dynamic triplet sampling for embedding histopathology image patches and demonstrating how different the tumorous cancer tissue types are from the normal ones. I also propose offline/online triplet mining using extreme distances for this embedding. In image processing and computer vision application, I propose Roweisfaces and Roweisposes for face recognition and 3D action recognition, respectively, using my proposed Roweis discriminant analysis method. I also introduce the concepts of anomaly landscape and anomaly path using the proposed curvature anomaly detection and use them to denoise images and video frames. I report extensive experiments, on different datasets, to show the effectiveness of the proposed algorithms. By experiments, I demonstrate that the proposed methods are useful for extracting informative features and instances for better accuracy, representation, prediction, class separation, data reduction, and embedding. I show that the proposed dimensionality reduction methods can extract informative features for better separation of classes. An example is obtaining an embedding space for separating cancer histopathology patches from the normal patches which helps hospitals diagnose cancers more easily in an automatic way. I also show that the proposed numerosity reduction methods are useful for ranking data instances based on their importance and reducing data volumes without a significant drop in performance of machine learning and data science algorithms.

Acknowledgements

I thank everybody who helped me in both life and education; I also thank God. I thank my PhD supervisors, Prof. Mark Crowley and Prof. Fakhri Karray, who helped and guided me kindly and carefully in my PhD. I thank all my teachers, professors, and staff who helped me, in education and life, since kindergarten until now. I thank all my current and previous academic committee members and other advisors, including Prof. Zhou Wang, Prof. Hamid R. Tizhoosh, Prof. Javad Alirezaie, Prof. Allaa Hilal, Prof. Wayne Oldford, Prof. Ali Ghodsi, Prof. Saber Salehkaleybar, Prof. Hoda Mohammadzade, Prof. Saeed Sharifian, Prof. Amir Kashi, Prof. Tahereh Seyedena, and other professors/instructors for their helpful comments and guidance. I also thank my dear parents, Shokuh Azam Zolfaghari and Yousef Ghojogh, and my brother, Aydin. I thank my dear family members, friends, lab-mates, co-authors, and landlords who helped me significantly in hard times of life. I do not name them so that not any name is missed; I truly thank them all.

Dedication

This is dedicated to my lovely parents, Shokuh Azam Zolfaghari and Yousef Ghojogh, who put a lot of efforts for me and my brother, Aydin.

World is in the Hilbert space,
And is vast and all-encompassing.
But it is so simple,
And falls on a low-dimensional submanifold.

Benyamin Ghojogh, August 2018

Table of Contents

List of Figures

List of Tables

Abbreviations

***k*NN** k Nearest Neighbors xv, xvi, 35, 46, 50, 73, 75, 99, 100, 104–106, 117

ADMM Alternating Direction Method of Multipliers 43–45, 47, 48, 150

APAC Approximate Pairwise Accuracy Criterion xv, xvi, 35, 104–106

AW-FDA Automatically Weighted FDA xv, xvi, 32, 33, 35, 104–106

AW-KFDA Automatically Weighted Kernel FDA xvi, 32, 35, 104–106

BA Batch ALL 12, 90, 93, 132, 133

BH Batch Hard 12, 88, 90, 93, 132, 133

BSH Batch Semi-Hard 12, 90, 93, 132, 133

BUNCA Bayesian Updating with NCA Loss xv, xviii, xx, xxi, 2, 7, 8, 20, 27, 29, 31, 55, 59, 64, 82, 83, 90–93, 102, 103, 126, 130–133, 145–148, 151, 154, 155

BUT Bayesian Updating with Triplet Loss xv, xviii, xx, xxi, 2, 7, 8, 20, 27, 29, 31, 55, 59, 64, 82, 83, 90–93, 102, 103, 126, 130–133, 145–148, 151, 154, 155

CAD Curvature Anomaly Detection xxi, 2, 7, 8, 22, 27–29, 65, 73, 75, 81, 93, 99, 100, 103, 137, 138, 140, 145, 147–149, 151, 154, 155

CDF Cumulative Distribution Function xvii, 16, 51, 120, 122, 124

CDM Confused Distance Maximization xv, xvi, 35, 104–106

CRC1 Colorectal Cancer 1 xv, xx, 83–87

CRC2 Colorectal Cancer 2 xv, xx, 83, 91, 93

CW-FDA Cosine Weighted FDA xv, xvi, 32, 33, 35, 104–106

CW-KFDA Cosine Weighted Kernel FDA 32, 35

DL Dictionary Learning 22, 71, 72, 136, 137

DROP Decremental Reduction Optimization Procedure xxi, 13, 27, 76, 135, 136, 139–141

DSDA Double Supervised Discriminant Analysis 37, 39, 97, 99, 107

DWS Distance Weighted Sampling 12, 90, 93, 132, 133

EE Elliptic Envelope 13, 138, 140

ENN Edited Nearest Neighbor xxi, 13, 27, 76, 135, 136, 139, 141

EP Easiest Positive 12, 90, 93, 132, 133

EPEN Easiest Positive Easiest Negative 88–90

EPHN Easiest Positive Hardest Negative 88–90

FDA Fisher Discriminant Analysis xv, 5, 7, 10, 11, 14, 27–29, 32–34, 36, 37, 39, 54, 59, 61, 66, 72, 97, 99, 104, 105, 107, 125, 136, 148–152

FDC Fisher Discriminant Contrastive xviii, 2, 7, 8, 27, 31, 59–62, 64, 83, 86, 87, 126, 129–131, 146–148, 154

FDT Fisher Discriminant Triplet xviii, 2, 7, 8, 27, 31, 59–62, 64, 83, 86, 87, 126, 129–131, 146–148, 154

HPEN Hardest Positive Easiest Negative 88–90

HPHN Hardest Positive Hardest Negative 88–90

HSIC Hilbert-Schmidt Independence Criterion 10, 14

iCAD Inverse Curvature Anomaly Detection xxi, 7, 28, 73, 76, 137–141, 149, 151, 154

iMondrian isolation Mondrian xii, xix, xxi, 7, 23, 28, 65, 77–81, 103, 140–145, 147, 149, 152, 154, 155

IRMD Instance Ranking by Matrix Decomposition xxi, 7, 22, 27, 28, 65, 70, 76, 81, 103, 133, 135–137, 139–141, 145, 147–149, 151, 154, 155

ISCA Image Structure Component Analysis xvii, xx, 5, 26, 31, 32, 39, 40, 43, 44, 46, 64, 103, 108, 113–117, 145, 146, 148, 150, 153, 154

K-CAD Kernel Curvature Anomaly Detection 73, 75, 100, 137, 138, 140

K-iCAD Kernel Inverse Curvature Anomaly Detection 73, 76, 137, 139–141

KFDA Kernel Fisher Discriminant Analysis xvi, 104–106

LE Laplacian Eigenmap xvi, 10, 42, 111–113

LLE Locally Linear Embedding xvii, xviii, 10, 26, 47, 49, 54, 118, 119, 122, 123, 125, 147, 150

LLISE Locally Linear Image Structure Embedding xvii, xx, 5, 26, 31, 32, 39, 40, 46, 49, 50, 64, 103, 108, 113, 117–119, 145, 146, 148, 150, 153, 154

LOF Local Outlier Factor 13, 138, 140, 144, 145

MDS Mltidimensional Scaling xvi, 10, 41, 42, 54, 110

MSE Mean Squared Error xvi, 15, 26, 40, 56, 100, 107–109, 113, 116, 126, 147, 148

MVU Maximum Variance Unfolding 10, 26, 148

NCA Neighborhood Component Analysis 12, 90, 93, 132, 133

NMF Nonnegative Matrix Factorization 22, 71, 72, 135–137

NR No Reduction 135–137, 139, 141

PCA Principal Component Analysis xvi–xviii, 10, 11, 14, 26, 28, 36, 37, 39–43, 54, 94, 96, 97, 107, 109–113, 115–117, 124, 125, 134, 145, 147, 149, 150, 154

PNCA Proxy-Neighborhood Component Analysis 12, 90, 93, 132, 133

POW Powered Distance xv, xvi, 35, 104–106

PSA Principal Sample Analysis xviii, xxi, 7, 27, 28, 65, 66, 70, 76, 81, 103, 133–135, 139–141, 145, 147–149, 151, 154

QQE Quantile-Quantile Embedding xvii, xviii, 6, 16, 26, 31, 51, 54, 64, 103, 119–126, 145, 148, 150, 154, 155

RBF Radial Basis Function xvi, xvii, 75, 76, 107–112, 115–119, 128, 138, 140, 145

RDA Roweis Discriminant Analysis xv, xvi, 5, 8, 14, 26, 29, 31, 36–39, 64, 93, 94, 96, 97, 99, 102, 103, 107, 108, 145–150, 153

SDM Sorted by Distance from Mean xviii, 133–137, 139–141

SE Shell Extraction 13, 27, 76, 135–137, 140, 141

SNE Stochastic Neighbor Embedding 11, 26, 147, 148

SOS Stratified Ordered Sampling 13, 27, 28, 76, 135–137, 139–141

SPCA Supervised Principal Component Analysis 10, 14, 26, 36, 37, 39, 72, 97, 107, 136, 137, 149, 154

SRS Simple Random Sampling xviii, 133–135

SSIM Structural Similarity Index xvi, 5, 15, 26, 31, 39–42, 64, 103, 107–114, 118, 145, 146, 148–150, 153, 154

SVD Singular Value Decomposition 22, 71, 72, 136, 137

SVM Support Vector Machine 13, 134–136, 138–141, 144, 145

t-SNE Student-t distributed Stochastic Neighbor Embedding xviii, 11, 26, 54, 124, 125, 148

TCGA The Cancer Genome Atlas xiv, xv, xx, 83–87

UMAP Uniform Manifold Approximation and Projection 11, 85, 86, 91, 126, 129, 131

W-KFDA Weighted Kernel FDA 32, 34, 35

WFDA Weighted Fisher Discriminant Analysis xv, xvi, xx, 5, 25, 31, 32, 34, 64, 103–106, 145, 146, 148, 150, 153

WSI Whole Slide Image xiv, 83, 84, 86, 153

Chapter 1

Introduction

1.1 Problem Definition: Data Reduction

It is common for the popular deep learning approaches to use data augmentation to satisfy the need to train huge number of parameters without overfitting, the increasing amount of data requires some crucial data reduction methods for various motivations. In general, data reduction is useful for:

- better storage efficiency,
- improving time of computation,
- better representation of data or discrimination of classes,
- removing outliers,
- even better recognition performance.

Data reduction approaches fall into two types, i.e., dimensionality reduction and numerosity reduction which reduce the dimensionality and the sample size of data, respectively (see Fig. 1.1). Numerosity reduction falls into prototype selection [28] and prototype generation [129] where instances are selected in the former and selected or generated as new instances in the latter. Dimensionality reduction approaches can be divided into feature extraction and feature selection [59] where the features of data are completely changed to another space with lower dimensionality in the former and the dimensionality of the transformed data is a subset of the original dimensionality in the latter. This thesis focuses on prototype selection in numerosity reduction and feature extraction in dimensionality reduction. In the following, we detail numerosity and dimensionality reduction further.

Assume there exists a set of n instances $\{\boldsymbol{x}_i\}_{i=1}^n$ in a d-dimensional Euclidean space, $\forall i \in \{1, \dots, n\} : \boldsymbol{x}_i \in \mathbb{R}^d$. The instances together form a matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$. In supervised learning, there are $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ meaning that every instance $\boldsymbol{x}_i$ has a corresponding label $\boldsymbol{y}_i \in \mathbb{R}^\ell$, where ℓ is the dimensionality of the label. We can then form the label matrix $\boldsymbol{Y} = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_n] \in \mathbb{R}^{\ell \times n}$. In classification, each instance belongs to one of $|\mathcal{C}|$ classes where $\mathcal{C}$ is the set including labels of classes. The cardinality of the set of instances in class c is denoted by n_c.

In pattern recognition, data can be either expanded or reduced. This reduction or expansion can be either in sample size (numerosity) or dimensionality. Table 1.1 shows the nine possible permutations of changing data in terms of numerosity and dimensionality. Numerosity reduction and expansion are addressed by numerosity reduction and data augmentation algorithms, respectively. On the other hand, dimensionality reduction and expansion deal with dimensionality reduction and kernels, respectively. This thesis tackles the two important fields of numerosity and dimensionality reduction for pattern recognition and machine learning.

Table 1.1: The landscape of data reduction/expansion. The bold topics are tackled in this proposal.

		numerosity		
		increase	decrease	no change
dimensionality	increase	data augmentation + kernels	**numerosity reduction** + kernels	kernels
	decrease	data augmentation + **dimensionality reduction**	**numerosity reduction** + **dimensionality reduction**	**dimensionality reduction**
	no change	data augmentation	**numerosity reduction**	original

As Table 1.1 shows, this thesis contributes to five cells of this table:

- In the case of dimension increase and numerosity reduction, we have kernel numerosity reduction. My proposed kernel CAD has this case.
- In the case of dimension decrease and numerosity increase, we have dimensionality reduction with data augmentation. Using data augmentation with my proposed FDT, FDC, BUT, and BUNCA cover this case.

- In the case of dimension decrease and numerosity decrease, we have both dimensionality reduction and numerosity reduction. A combination of my proposed methods of dimensionality reduction and numerosity reduction falls in this category.
- In the case of dimension decrease and no change in numerosity, we have dimensionality reduction. My proposed dimensionality reduction methods fall in this category.
- In the case of no change in dimension and numerosity reduction, we have numerosity reduction. My proposed numerosity reduction methods fall in this category.

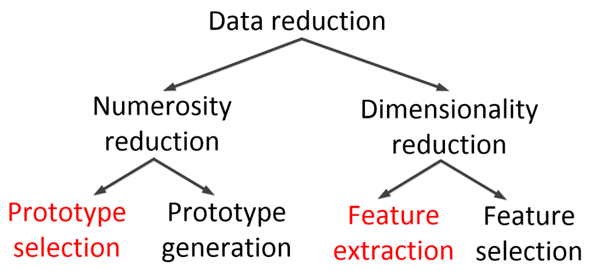

Figure 1.1: The taxonomy of data reduction. The red parts are tackled in this thesis.

1.2 Dimensionality Reduction and Manifold Learning

The task of dimensionality reduction has the goal to reduce the dimensionality of a dataset from d to $p \in (0, d]$ by mapping to a lower dimensional subspace or manifold [3]. In other words, we want to have $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{p \times n}$ from $\boldsymbol{X} \in \mathbb{R}^{d \times n}$. In feature extraction for dimensionality reduction, which I focus on, a new set of features is found for better representation or discrimination of data. There are different names for dimensionality reduction in the literature, i.e., *feature extraction*, *embedding*, *encoding*, *subspace learning*, *manifold learning*, and *representation learning* [59].

There exist several motivations for dimensionality reduction:

- According to the manifold hypothesis [24], data usually exist on a subspace or submanifold unless it is random noise. Therefore, the whole d-dimensional space is not required and a huge amount of it is unnecessary information. We can find the best p-dimensional subspace to represent the data with the least possible reconstruction error. This makes dimensionality reduction a data reduction method.

- Dimensionality reduction is useful for feature extraction. The extracted features are useful for classification, representation, clustering, or revealing patterns in data.
- Dimensionality reduction is one of the useful methods for data visualization. Data visualization can be used to reveal patterns by human visual system.

Note that previously in basic machine learning and dimensionality reduction, people often used to extract features using dimensionality reduction and then apply the classification, regression, or clustering task. However, modern deep learning is usually end-to-end. Although modern deep learning is end-to-end, it extracts features and learns embedding spaces in the layers of network; hence, deep learning can also be seen as dimensionality reduction. Researchers usually visualize the extracted features of neural network to interpret and analyze why deep learning is working properly on their data. The problem of end-to-end models is their harder and more limited possibility of troubleshooting if the performance is not satisfactory on some data. The insights and meaning of data coming from representation learning are critical to understand model performance more fully. Some of these insights can end up being useful for improving or understanding deep neural networks and why they worked the way they did.

1.3 Numerosity Reduction and Prototype Selection

In numerosity reduction, the goal is to reduce the cardinality of a dataset from n to $m \in (0, n]$ by ranking the instances from the most to least important in terms of representation, discrimination, etc. In other words, we want to have $\hat{\boldsymbol{X}} \in \mathbb{R}^{d \times m}$ and $\hat{\boldsymbol{Y}} \in \mathbb{R}^{\ell \times m}$ (if supervised), where $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$ include the best m instances and their labels in terms of representation of data and/or discrimination of classes. In numerosity reduction, a subset of data points is found to best represent the underlying manifold or topology of data. The selected subset of instances in numerosity reduction are called *instances*, *prototypes*, or *samples*. It is noteworthy that dimensionality reduction is more well-known and probably more effective than numerosity reduction in the literature and practical projects.

There exist several motivations for numerosity reduction:

- There usually exists some dummy information in data which is not completely useful (or is sometimes even destructive) for discrimination or representation of data. In other words, we usually have too much data and the data instances do not contribute equal amounts of information to learning a discriminative or representative model and thus could be sorted by this information if it can be quantified.

- In some cases, a reduced set of data points can actually better represent the underlying patterns or distributions in the data leading to better discrimination for classification or prediction for regression. It can improve the signal to noise ratio.

- In some applications in edge computing, low-battery embedded systems, or space exploration, there is limited possibility for storage or energy. In these domains, it is qualitatively and quantitatively useful to store or transmit a portion of data which is its best representation in terms of either more fitted regression or more discriminative classification performance. The amount of portion can be decided according to the amount of data reduction or drop in the representation performance.

Note that numerosity reduction, as well as dimensionality reduction, can be either unsupervised or supervised. In unsupervised numerosity reduction, the most informative instances for better representation of data cloud are important while in supervised numerosity reduction, the instances most contributing to the labels are more important to be selected.

1.4 Organization of the Thesis

As was introduced earlier, data reduction can be divided into dimensionality reduction and numerosity reduction (see Fig. 1.1). In the following, I detail the organization of thesis in each of these categories.

1.4.1 Dimensionality Reduction

The dimensionality reduction methods, focused on feature extraction, can be grouped into three categories which are spectral dimensionality reduction, probabilistic dimensionality reduction, and neural network-based dimensionality reduction. These categories are based on the generalized eigenvalue problem, latent variables, and neural networks, respectively. I propose different algorithms in each of these categories (see Fig. 1.2). In the next chapter, I will explain why these methods are proposed and what problems they are tackling.

In the area of spectral dimensionality reduction, I propose WFDA, RDA, and some methods in image quality aware embedding (see Fig. 1.2). WFDA weights the pairs of classes in FDA to consider the different confusion of classes. RDA is a generalization of several different subspace learning methods based on generalized eigenvalue problem. SSIM kernel, ISCA, and LLISE are the proposed methods for image quality aware embedding.

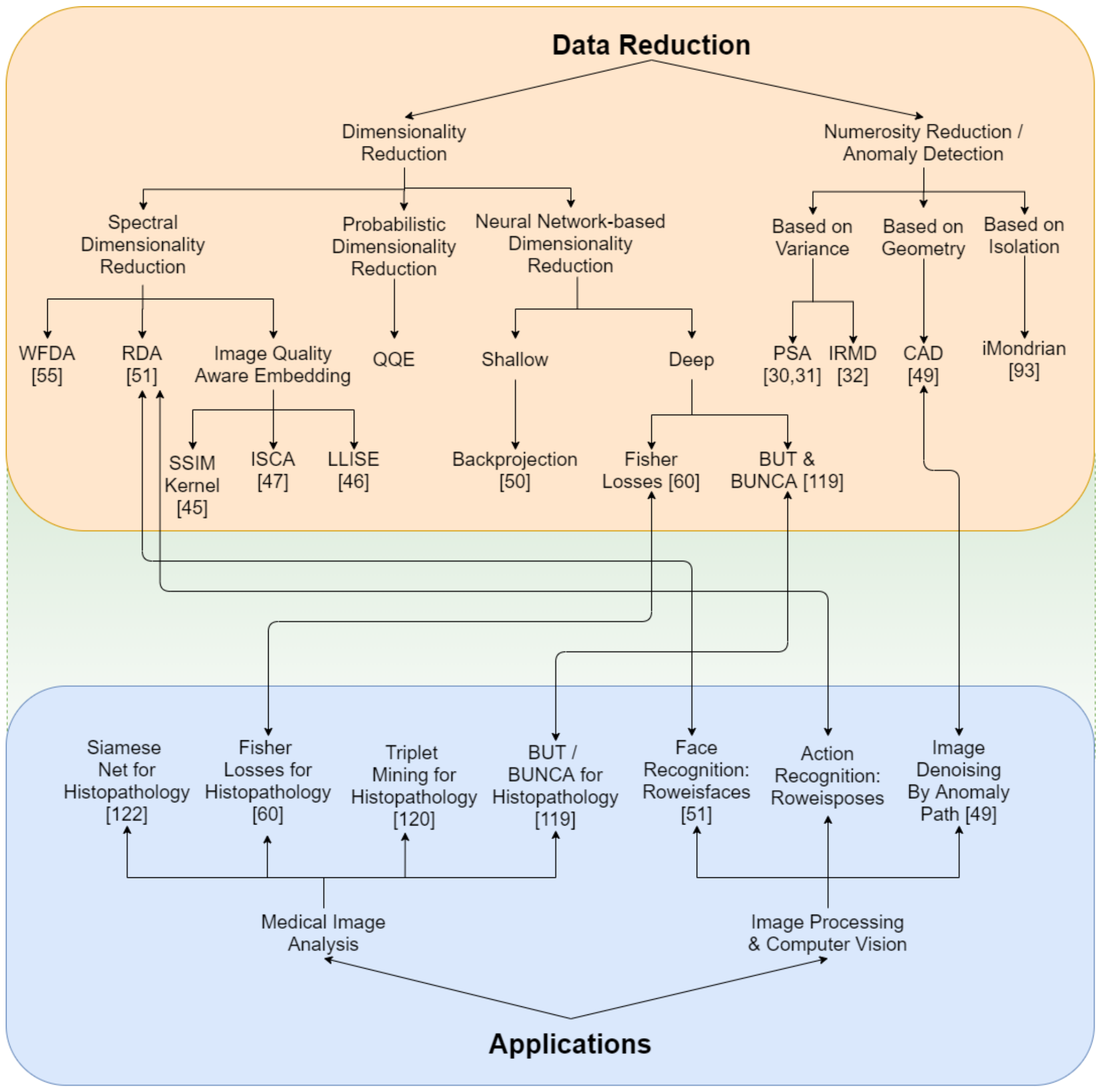
Data Reduction
Dimensionality Reduction
Numerosity Reduction / Anomaly Detection
Spectral Dimensionality Reduction
Probabilistic Dimensionality Reduction
Neural Network-based Dimensionality Reduction
Based on Variance
Based on Geometry
Based on Isolation
WFDA [55]
RDA [51]
Image Quality Aware Embedding
QQE
Shallow
Deep
PSA [30,31]
IRMD [32]
CAD [49]
iMondrian [93]
SSIM Kernel [45]
ISCA [47]
LLISE [46]
Backprojection [50]
Fisher Losses [60]
BUT & BUNCA [119]
Siamese Net for Histopathology [122]
Fisher Losses for Histopathology [60]
Triplet Mining for Histopathology [120]
BUT / BUNCA for Histopathology [119]
Face Recognition: Roweisfaces [51]
Action Recognition: Roweisposes
Image Denoising By Anomaly Path [49]
Medical Image Analysis
Image Processing & Computer Vision
Applications

Figure 1.2: The overall structure of the thesis.

In the category of probabilistic dimensionality reduction, I propose QQE which deals with the distribution of data and embedding. Neural network-based dimensionality reduction methods can be either shallow or deep networks. For shallow networks, I propose the backprojection algorithm for training the feedforward networks which can be used for discriminating classes. For deep networks, I propose several methods for Siamese triplet networks [118]. Because of similarity of the goals of FDA, triplet loss [118], and contrastive loss [65], I propose FDT and FDC losses for training triplet nets. Moreover, for the sake of dynamic triplet sampling, I propose BUT and BUNCA which use Bayesian updating theorem [101].

Note that in some of the proposed dimensionality reduction methods, I also propose the kernel version of method. Here, I explain the intuitive reason for the proposal of kernel variants. In the input space, the pattern of data may be nonlinear or complicated. Using a pulling function, which maps data from the input space to a usually high dimensional feature space, the pattern of data hopefully gets simpler or more linear. Therefore, using the kernel variants of methods, I hope to deal with simpler data.

1.4.2 Numerosity Reduction

The numerosity reduction methods, focused on prototype selection, can be based on variance, geometry, or isolation (see Fig. 1.2). Based on variance, I propose PSA and IRMD. PSA deals with the inter-class and intra-class variances of data for ranking the data instances. IRMD decomposes the matrix of data and uses the informative bases of data which can be the most variant directions of data, for example. As a method based on geometry, I propose iCAD and kernel iCAD which make use of the polyhedron curvature by looking at every data point as a vertex of polyhedron. Note that an opposite view to numerosity reduction can help for the task of anomaly detection. Hence, I also propose CAD which uses the concept of polyhedron curvature for anomaly detection. iMondrian forest, which is a novel hybrid of isolation forest and Mondrian forest, is proposed for an isolation-based anomaly detection method. Note that the iMondrian forest project is not my main project and I was just a collaborator/coauthor in that project. In the next chapter, I will explain why these methods are proposed and what problems they are tackling. It is noteworthy that I also propose the kernel variants of some of the proposed numerosity reduction methods. The reason for this is the same as the explained reason for kernel variants of dimensionality reduction methods.

1.4.3 Applications

In addition to proposing methods in dimensionality reduction and numerosity reduction, I also propose or use some data reduction methods for different applications. My application methods can be divided into two main categories, i.e., medical image analysis (focused on histopathology data [77]) and image processing & computer vision (see Fig. 1.2). I apply the proposed theoretical methods in some applications. In medical image analysis, in collaboration with the KIMIA lab in the University of Waterloo, I worked on digital histopathology image embedding, where histopathology refers to the diagnosis and study of diseases of the body tissues. In the fields of image processing and computer vision, I propose some methods for face and action recognition based on the proposed RDA, as well as image denoising using the proposed CAD method.

In medical image analysis, I use Fisher losses, i.e. FDT and FDC losses, for embedding of histopathology data. I also mine triplets based on extreme distances of histopathology patches in both offline and online manners. The proposed BUT and BUNCA losses are also applied to embed the histopathology patches. For image processing and computer vision, I apply the proposed RDA subspace learning method in the fields of face recognition and 3D action recognition, by proposing Roweisfaces and Roweisposes, respectively. Finally, the concepts of an anomaly landscape and anomaly path are proposed, using CAD, for making an anomaly normal or vice versa. These concepts can be used for image denoising. In the next chapter, I will explain why these methods are proposed and what problems they are tackling.

1.4.4 Outline of Thesis

Chapter 1 introduces the data reduction problem, its taxonomy, its related work, and the open problems addressed in this thesis. Chapter 2 reviews the related work, introduces the technical background, and reports the open problems in data reduction for machine learning. Chapter 3 proposes several algorithms in the three categories of dimensionality reduction, i.e., spectral dimensionality reduction, probabilistic dimensionality reduction, and neural network-based dimensionality reduction. I also propose several algorithms in the three categories of numerosity reduction, i.e. variance based, geometry based, and ensemble methods, in Chapter 4. Some proposed applications of data reduction are demonstrated in Chapter 5. The experimental results of the thesis are reported in Chapter 6. Finally, Chapter 7 summarizes the thesis, concludes the thesis with some discussions, and highlights the other open problems to indicate the possible future directions.

Chapter 2

Related Work, Background, and Open Problems

This chapter reviews the related work and explains the technical background. It also explains the open problems, in data reduction, which are tackled in this thesis and enumerates the contributions of this thesis. In Section 2.1, the related work for dimensionality reduction and numerosity reduction are introduced. The technical background are explained in Section 2.2. Then, Section 2.3 reports some open problems in dimensionality reduction and numerosity reduction. The proposed algorithms in this thesis tackle these open problems; hence, the contributions of this thesis are introduced.

2.1 Related Work

In this section, I review the related work for both dimensionality reduction and numerosity reduction.

2.1.1 Related Work for Dimensionality Reduction

Dimensionality reduction methods can be divided into three categories, i.e., spectral dimensionality reduction, probabilistic dimensionality reduction, and neural network-based dimensionality reduction which have geometric, probabilistic, and information-theoretic points of view to dimensionality reduction, respectively.

2.1.1.1 Spectral Dimensionality Reduction

The methods in spectral dimensionality reduction reduce to eigenvalue decomposition and generalized eigenvalue problem [42]. They use a geometric approach and they unfold the manifold in a lower dimensional subspace.

Some unsupervised methods within the spectral approach are PCA [35], MDS [16, 38], Sammon mapping [113], LE [7, 87], Isomap [126], LLE [110, 114], and graph embedding [63]. PCA [35] is a linear method for feature extraction. It captures the directions of data with most variance. Dual PCA is also useful in case the dimensionality of data is high where PCA might face some computational problems. Moreover, dual PCA provides inner products required for the kernel trick used in kernel PCA. MDS [16] is also a linear method if we use Euclidean distance for it. MDS tries to preserve the similarities of data points in the embedded space and it is equivalent to PCA [85].

Sammon mapping [113], which is closely related to MDS [85], tries to preserve the distances rather than similarities. Sammon mapping is a non-linear method [113]. Isomap [126] is also related to MDS. It uses geodesic distance rather than Euclidean distance in the formulation of kernel [16, 66]. Because of geodesic distance, Isomap is a non-linear method. LLE [110, 114], however, has another point of view to non-linear manifold learning. It tries to linearly reconstruct every data point by its neighbors in the original space and it uses the same weights of linear reconstruction for embedding in the low dimensional space. Kernel LLE [148] performs the stages of finding neighbors and linear reconstruction in the feature space. LE [7], as another non-linear method, tries to preserve the similarities of data points in the embedded space.

It can be shown that PCA, MDS, and Isomap are all kernel PCA with their own kernels [66]. Therefore, the idea of Maximum Variance Unfolding (MVU) or Semi-Definite Embedding (SDE) [136] proceeds from the question "why don't we find the best possible kernel which finds the kernel capturing the most variant direction of data to unfold it?" MVU uses semi-definite embedding optimization and thus it is iterative and slow to train relative to many spectral dimensionality reduction methods.

We can also have supervised dimensionality reduction methods which take the class labels into account. A promising subspace learning method is Supervised Principal Component Analysis (SPCA) [4] which uses the Hilbert-Schmidt Independence Criterion (HSIC) [64]. HSIC is a measure whose idea is to calculate the dependence of two random variables by measuring their correlation in Hilbert space. SPCA reduces to PCA if we do not consider the relation of the class labels. Another supervised method is FDA [43] which uses the within- and between-class scatters of data. FDA is equivalent to linear discriminant analysis [33]. Kernel FDA [99] performs FDA in the feature space by this rule that any solution in the feature space must lie in the span of all the training vectors mapped to the feature space. A problem with the spectral methods is not handling relatively large number of instances because of the overhead of eigenvalue problems.

2.1.1.2 Probabilistic Dimensionality Reduction

The probabilistic dimensionality reduction methods have a probabilistic approach where it is assumed that there is low dimensional latent variable which has caused the high dimensional variable and we need to infer and discover that latent variable. Some of the methods in this category use graphical models. The benefit of a probabilistic approach over the spectral methods is handling missing data. Please note that I explain the neural network-based probabilistic methods in the category of neural network-based methods for the sake of better organization.

Some examples of probabilistic dimensionality reduction are factor analysis [27], whose nonlinear extension is the variational autoencoder [81], probabilistic PCA, probabilistic linear discriminant analysis, and probabilistic FDA which cast the spectral methods to the probabilistic approach. Some other examples are Stochastic Neighbor Embedding (SNE) [69] and t-SNE [94, 131] where Gaussian and Student-t distributions are considered for the embedded space, respectively. A recent successful method is the Uniform Manifold Approximation and Projection (UMAP) [98] which optimizes over the probability of closeness of graphs in the input and embedded spaces. An advantage of the probabilistic methods is being relatively robust to noise because of their stochastic behaviour.

2.1.1.3 Neural Network-Based Dimensionality Reduction

The neural network-based dimensionality reduction category has an information theoretic approach where the middle of the neural network or autoencoder is seen as a bottleneck of information which keeps only the useful and important information.

Some examples are restricted Boltzmann machine and deep belief network [70] which are fundamental dimensionality reduction methods in a network structure. They were proposed in order to make the networks deep without the problem of vanishing gradients. Another example is autoencoder where the latent embedding space is encoded by a middle layer of a possibly deep autoencoder. There is also deep metric learning [80] which encodes data in an embedding space trained by deep neural network. It tries to increase and decrease the inter- and intra-class variances of data, respectively [60]. Note that metric learning can be seen as dimensionality reduction because it can be seen as linear or nonlinear projection onto the embedding space and then applying Euclidean distance in that space. In variational autoencoder [81], the latent space is tended to have a specific distribution such as Gaussian. Another example is adversarial autoencoder [95] which uses a game-theoretic optimization for encoding.

Recently, some deep metric learning methods have been proposed which focus on maximizing and minimizing the inter-class and intra-class variances of data [60]. A Siamese network is a set of several (typically two or three) networks which share weights with each other [118]. The weights are trained using a loss based on anchor, neighbor (positive), and distant (negative) samples, where anchor and neighbor belong to the same class, but the anchor and distant instances are in different classes. The loss functions used to train a Siamese network usually make use of the anchor, neighbor, and distant samples, trying to pull the anchor and neighbor towards one another and simultaneously push the anchor and distant tiles away from each other. Two different loss functions, which are used for training Siamese networks, are triplet loss [118] and contrastive loss [65] for networks with three and two sub-networks, respectively. Neighborhood Component Analysis (NCA) [61] and Proxy-Neighborhood Component Analysis (PNCA) [100] are also softmax forms of the triplet loss which are used for training Siamese nets. Sampling triplets from data points can be done using triplet mining. Some of the existing triplet mining methods are Batch ALL (BA) [19], Batch Semi-Hard (BSH) [118], Batch Hard (BH) [68], Easiest Positive (EP) [141], and Distance Weighted Sampling (DWS) [139]. A problem with the neural network-based methods is their slower training pace compared to other categories of dimensionality reduction. An advantage of this category is handling data with large number of instances.

2.1.2 Related Work for Numerosity Reduction

Numerosity reduction methods can be seen in two perspectives. If the important, informative, and usual points are aimed to be found, the task is named prototype selection. In contrary, if the anomaly or unusual points are aimed to be found, we deal with the task of anomaly detection. In this section, I review the related work for both of these approaches.

2.1.2.1 Prototype Selection

One of the methods of numerosity reduction is prototype selection [28] in which the informative instances from a dataset are selected and the rest is discarded. Prototype selection is also known by other names such as *instance selection*, *instance ranking*, and *numerosity* *reduction* [76, 28]. In ENN [138], instances surrounded by a majority of neighbors from other classes are removed. The DROP [137] removes instances one by one if the number of neighbor instances which are correctly classified improves after omitting the instance. Among DROP1 to DROP5 algorithms [137], DROP3 has the best accuracy-time trade-off; however, its time complexity is not good. There are also some heuristic prototype selection methods such as Random Mutation Hill Climbing [123] which finds the best instances using mutations and testing the accuracy fitness. Heuristic methods usually take a noticeable amount of time to run. Some methods focus on the boundary and median points, such as Stratified Ordered Sampling (SOS) [76] which concentrates on selecting boundary instances and recursively finds the median instances. Shell Extraction (SE) [89] defines a reduction sphere and removes the instances in it, resulting in a shell of boundary points.

2.1.2.2 Anomaly Detection

Local Outlier Factor (LOF) [11] is one of the well-known anomaly detection algorithms. It defines a measure for local density for every data point according to its neighbors. It compares the local density of every point with its neighbors and finds the anomalies. One-class Support Vector Machine (SVM) [117] is another method which estimates a function which is positive on the regions of data with high density and negative elsewhere. Therefore, the points with negative values of that function are considered as anomalies. If the data are assumed to have Gaussian distribution as the most common distribution, an Elliptic Envelope (EE) can be fit to the data [109] and the points having low probability in the fitted envelope are considered to be anomalies. Isolation forest [90] is an isolation-based anomaly detection method which isolates the anomalies using an ensemble approach. The algorithm takes advantage of the observation that many anomalous points will be very easily isolated to a leaf of size one by a by a very simple tree splittings algorithm. In this method, the ensemble includes isolation trees where the greater the depth of tree needed to isolate a point, the more normal the point is judged to be.

2.2 Technical Background

In this section, I review technical background for the proposed methods in this thesis.

2.2.1 Linear Subspace Learning Based on Generalized Eigenvalue Problem

In linear subspace learning, the d-dimensional data $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ are projected onto the p-dimensional (where $p \leq d$) column space of a projection matrix denoted by $\boldsymbol{U} \in \mathbb{R}^{d\times p}$. The low dimensional projected data are $\boldsymbol{U}^\top \boldsymbol{X}$. Many of the linear subspace learning methods have solutions based on generalized eigenvalue problem. These methods are in the spectral dimensionality reduction category.

PCA rotates data to align with the most variant directions. In its subspace, the most variant directions of data are preserved [74, 35]. The optimization problem in PCA is expressed as [74, 35]:

$$\underset{\boldsymbol{U}}{\text{maximize}} \quad \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_T \boldsymbol{U}), \qquad \text{subject to} \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}, \tag{2.1}$$

where $\boldsymbol{S}_T \in \mathbb{R}^{n\times n}$ is the total scatter or the covariance matrix:

$$\mathbb{R}^{n\times n} \ni \boldsymbol{S}_T := \breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top = \boldsymbol{X}\boldsymbol{H}\boldsymbol{H}\boldsymbol{X}^\top = \boldsymbol{X}\boldsymbol{H}\boldsymbol{X}^\top, \tag{2.2}$$

where $\mathbb{R}^{n\times n} \ni \boldsymbol{H} := \boldsymbol{I} - (1/n)\mathbf{1}\mathbf{1}^\top$ is the idempotent centering matrix and $\breve{\boldsymbol{X}} := \boldsymbol{X}\boldsymbol{H}$.

On the other hand, FDA [25] maximizes the inter-class variance and minimizes the intra-class variance in order to reduce the confusion of classes. For this, it maximizes the Fisher criterion [43]:

$$\underset{\boldsymbol{U}}{\text{maximize}} \quad \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_B \boldsymbol{U}), \qquad \text{subject to} \quad \boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U} = \boldsymbol{I}, \tag{2.3}$$

where $\boldsymbol{S}_B$ and $\boldsymbol{S}_W$ are the between (inter-class) and within (intra-class) scatters, respectively. The total scatter can be considered as the summation of the between and within scatters [144]:

$$\boldsymbol{S}_T = \boldsymbol{S}_B + \boldsymbol{S}_W \implies \boldsymbol{S}_B = \boldsymbol{S}_T - \boldsymbol{S}_W. \tag{2.4}$$

Therefore, the optimization in FDA can be:

$$\underset{\boldsymbol{U}}{\text{maximize}} \quad \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_T \boldsymbol{U}), \qquad \text{subject to} \quad \boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U} = \boldsymbol{I}. \tag{2.5}$$

SPCA [4] uses the empirical estimation of the HSIC [64]. It uses HSIC for the projected data $\boldsymbol{U}^\top \boldsymbol{X}$ and the labels $\boldsymbol{Y}$ and maximizes the dependence of them. Its optimization is [4, 35]:

$$\underset{\boldsymbol{U}}{\text{maximize}} \quad \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top \boldsymbol{U}), \qquad \text{subject to} \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}, \tag{2.6}$$

where $\boldsymbol{K}_y$ is the kernel matrix over the labels. Comparing Eqs. (2.1), (2.5), and (2.6) shows that these methods belong to a family of methods based on eigenvalue and generalized eigenvalue problems. This gave me a motivation to propose RDA.

2.2.2 Structural Similarity Index

SSIM is useful for image quality assessment. It has been shown to be more effective than MSE for image fidelity measures [134]. I require this for the proposed image quality aware embedding methods (see Section 3.1.3). SSIM between two reshaped image patches $\breve{\boldsymbol{x}}_1 = [x_1^{(1)}, \dots, x_1^{(q)}]^\top \in \mathbb{R}^q$ and $\breve{\boldsymbol{x}}_2 = [x_2^{(1)}, \dots, x_2^{(q)}]^\top \in \mathbb{R}^q$, in color intensity range $[0, 255]$, is [135, 133]:

$$\mathbb{R} \ni \text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := \Big(\frac{2\mu_{x_1}\mu_{x_2} + c_1}{\mu_{x_1}^2 + \mu_{x_2}^2 + c_1}\Big)\Big(\frac{2\sigma_{x_1}\sigma_{x_2} + c_2}{\sigma_{x_1}^2 + \sigma_{x_2}^2 + c_2}\Big)\Big(\frac{\sigma_{x_1,x_2} + c_3}{\sigma_{x_1}\sigma_{x_2} + c_3}\Big), \tag{2.7}$$

where $\mu_{x_1} = (1/q)\sum_{i=1}^q x_1^{(i)}$, $\sigma_{x_1} = \Big[\big(1/(q-1)\big)\sum_{i=1}^q (x_1^{(i)} - \mu_{x_1})^2\Big]^{0.5}$, $\sigma_{x_1,x_2} = \big(1/(q-1)\big)\sum_{i=1}^q (x_1^{(i)} - \mu_{x_1})(x_2^{(i)} - \mu_{x_2})$, $c_1 = (0.01 \times 255)^2$, $c_2 = 2\,c_3 = (0.03 \times 255)^2$, and μ_{x_2} and σ_{x_2} are defined similarly for $\breve{\boldsymbol{x}}_2$.

Because of $c_2 = 2\,c_3$, the SSIM is simplified to $\text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = s_1(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) \times s_2(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$, where $s_1(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := (2\mu_{x_1}\mu_{x_2} + c_1)/(\mu_{x_1}^2 + \mu_{x_2}^2 + c_1)$ and $s_2(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := (2\sigma_{x_1,x_2} + c_2)/(\sigma_{x_1}^2 + \sigma_{x_2}^2 + c_2)$. Because of spatial variety of image statistics, the SSIM is usually computed for patches of an image. A sliding window moves pixel by pixel on the two images and calculates the $\text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$ for every patch. I denote the reshaped vectors of the two images by $\boldsymbol{x}_1 \in \mathbb{R}^d$ and $\boldsymbol{x}_2 \in \mathbb{R}^d$, and a reshaped patch in the two images by $\breve{\boldsymbol{x}}_1 \in \mathbb{R}^q$ and $\breve{\boldsymbol{x}}_2 \in \mathbb{R}^q$. Therefore, an SSIM vector denoted by $\boldsymbol{s}(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{R}^d$ is obtained whose i-th element is SSIM for the patch around the i-th pixel.

Note that since $c_2 = 2\,c_3$, SSIM can be simplified to $\text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = s_1(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) \times s_2(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$, where:

$$s_1(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := (2\mu_{x_1}\mu_{x_2} + c_1)/(\mu_{x_1}^2 + \mu_{x_2}^2 + c_1), \tag{2.8}$$

$$s_2(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := (2\sigma_{x_1,x_2} + c_2)/(\sigma_{x_1}^2 + \sigma_{x_2}^2 + c_2) \tag{2.9}$$

If the vectors $\breve{\boldsymbol{x}}_1$ and $\breve{\boldsymbol{x}}_2$ have zero mean, i.e., $\mu_{x_1} = \mu_{x_2} = 0$, the SSIM becomes $\mathbb{R} \ni \text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = (2\breve{\boldsymbol{x}}_1^\top \breve{\boldsymbol{x}}_2 + c)/(||\breve{\boldsymbol{x}}_1||_2^2 + ||\breve{\boldsymbol{x}}_2||_2^2 + c)$, where $c = (q-1)\,c_2$ [105]. I denote the reshaped vectors of the two images by $\boldsymbol{x}_1 \in \mathbb{R}^d$ and $\boldsymbol{x}_2 \in \mathbb{R}^d$, and a reshaped block in the two images by $\breve{\boldsymbol{x}}_1 \in \mathbb{R}^q$ and $\breve{\boldsymbol{x}}_2 \in \mathbb{R}^q$. The distance based on SSIM, which I denote by $||.||_S$, is [105, 12]:

$$\mathbb{R} \ni ||\breve{\boldsymbol{x}}_1 - \breve{\boldsymbol{x}}_2||_S^2 := 1 - \text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = \frac{||\breve{\boldsymbol{x}}_1 - \breve{\boldsymbol{x}}_2||_2^2}{||\breve{\boldsymbol{x}}_1||_2^2 + ||\breve{\boldsymbol{x}}_2||_2^2 + c}, \tag{2.10}$$

where $\mu_{x_1} = \mu_{x_2} = 0$. There exists another SSIM distance, defined as [12]:

$$\mathbb{R} \ni ||\breve{\boldsymbol{x}}_i - \breve{\boldsymbol{x}}_j||_S := \sqrt{2 - s_1(\breve{\boldsymbol{x}}_i, \breve{\boldsymbol{x}}_j) - s_2(\breve{\boldsymbol{x}}_i, \breve{\boldsymbol{x}}_j)}. \tag{2.11}$$

2.2.3 Quantile and Quantile-Quantile Plots

Quantile and quantile-quantile plots are used for visual statistical tests to see how different from or similar to a references distribution the distribution of data is [103]. I require them for my proposed QQE (see Section 3.2.1). In the following, I review the technical background for quantile and quantile-quantile plots.

2.2.3.1 Quantile Function and Quantile Plot

The *quantile function* for a distribution is defined as [103]:

$$Q(p) := F^{-1}(p) := \inf\{x \mid F(x) \geq p\}, \tag{2.12}$$

where $p \in [0, 1]$ is called *position* and $F(x)$ is the CDF. The two-dimensional plot $(p, Q(p))$ is called the *quantile plot*. If we have a drawn sample, with sample size n, from a distribution, the quantile plot is a *sample (or empirical) quantile*. The sample quantile plot is $(p_i, Q(p_i)), \forall i \in \{1, \dots, n\}$. For the sample quantile, we can determine the i-th position, denoted by p_i, as:

$$p_i := \frac{i - \alpha}{n - \alpha - \beta + 1}, \tag{2.13}$$

where different values for α and β result in different positions.

For the multivariate quantile plot, *spatial rank* fulfills the role played by position in the univariate case. Spatial rank $\boldsymbol{u}_i \in \mathbb{R}^d$ of $\boldsymbol{x}_i \in \mathbb{R}^d$ with respect to the sample $\{\boldsymbol{x}_j\}_{j=1}^n$ is defined as [18]:

$$\boldsymbol{u}_i := \frac{1}{n} \sum_{j=1, j\neq i}^{n} \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2}, \tag{2.14}$$

whose term in the summation is a generalization of the sign function for the multivariate vector [96]. The multivariate *spatial quantile* (or *geometrical quantile*) for the multivariate spatial rank $\boldsymbol{u} \in \mathbb{R}^d$ is defined as:

$$\boldsymbol{Q}(\boldsymbol{u}) := \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathbb{E}(\Phi(\boldsymbol{u}, \boldsymbol{x} - \boldsymbol{\theta}) - \Phi(\boldsymbol{u}, \boldsymbol{x})), \tag{2.15}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is a random vector, $\Phi(\boldsymbol{u}, \boldsymbol{t}) := \|\boldsymbol{t}\|_2 + \boldsymbol{u}^\top \boldsymbol{t}$, and $\boldsymbol{u}$ is a vector in unit ball, i.e., $\boldsymbol{u} \in \{\boldsymbol{v} \mid \boldsymbol{v} \in \mathbb{R}^d, \|\boldsymbol{v}\|_2 < 1\}$ [18].

2.2.3.2 Quantile-Quantile Plot

Assume we have two quantile functions for two univariate distributions. If we match their positions and plot $(Q_1(p), Q_2(p)), \forall p \in [0,1]$, we will have *quantile-quantile plot* or *qq-plot* in short [103]. Again, this plot can be an empirical plot, i.e., $(Q_1(p_i), Q_2(p_i)), \forall i \in \{1, \dots, n\}$. Usually, as a statistical test, we want to see whether the first distribution is similar to the second empirical or theoretical distribution [103]. Note that if the qq-plot of two distributions is a line with slope 1 (angle $\pi/4$) and intercept 0, the two distributions have the same distributions [103]. The slope and the intercept of the line show the difference of spread and location of the two distributions, respectively.

In order to extend the qq-plot to multivariate distributions, we can consider the marginal quantiles. However, this fails to take the dependence of marginals into account [18]. There exist different methods for a promising generalization. One of these methods is *fuzzy qq-plot* [21] (note that it is not related to fuzzy logic). In a fuzzy qq-plot, a sample of size n is drawn from the reference distribution and the data points of the two samples are matched using optimization. An affine transformation is also applied to the observed sample in order to have an invariant comparison to the affine transformation. In the multivariate qq-plot, the matched data points are used to plot the qq-plots for every component; therefore, we will have d qq-plots where d is the dimensionality of data. Note that these plots are different from the d qq-plots for the marginal distributions. The technical detail of fuzzy qq-plot is explained in the following.

2.2.3.3 Multivariate Fuzzy Quantile-Quantile Plot

Assume we have a dataset with size n and dimensionality d, i.e., $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$. We want to transform its distribution as $\boldsymbol{x}_i \mapsto \boldsymbol{y}_i, \forall i \in \{1, \dots, n\}$. We draw a sample $\{\boldsymbol{y}_i \in \mathbb{R}^d\}_{i=1}^n$ of size n from the desired (reference) distribution. Note that in case we already have a reference sample $\{\boldsymbol{y}_i \in \mathbb{R}^d\}_{i=1}^m$ rather than the reference distribution, we can employ bootstrapping or oversampling if $m > n$ and $m < n$, respectively, to have $m = n$. The data points $\{\boldsymbol{x}_i\}_{i=1}^n$ and $\{\boldsymbol{y}_i\}_{i=1}^n$ are matched by [21]:

$$\underset{\boldsymbol{A}, \boldsymbol{b}, \sigma}{\text{minimize}} \quad \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{y}_{\sigma(i)} - \boldsymbol{b}\|_2^2, \tag{2.16}$$

where $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b} \in \mathbb{R}^d$ are used to make the matching problem invariant to affine transformation. If $\mathcal{P}$ is the set of all possible permutations of integers $\{1, \dots, n\}$, we have $\sigma \in \mathcal{P}$. This optimization problem finds the best permutation regardless of any affine transformation.

In order to solve this problem, one can iteratively switch between solving for $\boldsymbol{A}$, $\boldsymbol{b}$, and σ until there is no change in σ [21]. Given $\boldsymbol{A}$ and $\boldsymbol{b}$, one can solve:

$$\underset{\sigma}{\min.} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{y}_{\sigma(i)} - \boldsymbol{b}\|_2^2 \equiv \underset{\boldsymbol{\Psi}}{\min.} \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{C}(i,j)\boldsymbol{\Psi}(i,j), \tag{2.17}$$

which is an assignment problem and can be solved using the Hungarian method [82]. The $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{n\times n}$ are the cost matrix and a matrix with only one 1 in every row, respectively. Note that $\boldsymbol{\Psi}(i,j) = 1$ means that the $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ are matched. The $\boldsymbol{C}$ should be computed before solving the optimization where $\boldsymbol{C}(i,j) := \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{y}_j - \boldsymbol{b}\|_2^2$.

According to the 1's in the obtained $\boldsymbol{\Psi}$, we have σ. Then given σ, one can solve:

$$\underset{\boldsymbol{A},\boldsymbol{b}}{\text{minimize}} \quad \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{y}_{\sigma(i)} - \boldsymbol{b}\|_2^2, \tag{2.18}$$

which is a multivariate regression problem. The solution is [26]:

$$\mathbb{R}^{(d+1)\times d} \ni \boldsymbol{\beta} := (\breve{\boldsymbol{Y}}^\top \breve{\boldsymbol{Y}})^{-1} \breve{\boldsymbol{Y}}^\top \breve{\boldsymbol{X}}, \tag{2.19}$$

where $\mathbb{R}^{n\times(d+1)} \ni \breve{\boldsymbol{Y}} := \big[[\boldsymbol{y}_{\sigma(1)}, \dots, \boldsymbol{y}_{\sigma(n)}]^\top, \boldsymbol{1}_{n\times 1}\big]$ and $\mathbb{R}^{n\times d} \ni \breve{\boldsymbol{X}} := \boldsymbol{X}^\top = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^\top$. We will have $\boldsymbol{\beta} = [\boldsymbol{A}, \boldsymbol{b}]^\top$. Therefore, $\boldsymbol{A}$ and $\boldsymbol{b}$ are found where $\boldsymbol{A}^\top$ is the top $d \times d$ sub-matrix of $\boldsymbol{\beta}$ and $\boldsymbol{b}^\top$ is the last row of $\boldsymbol{\beta}$.

Note that it is better to set the initial rotation matrix to the identity matrix, i.e. $\boldsymbol{A}^{(0)} = \boldsymbol{I}$, for not having much rotation in assignment. In this way, only few iterations suffice to solve the matching problem. This iterative optimization gives us the matching σ and the samples $\{\boldsymbol{x}_i\}_{i=1}^n$ and $\{\boldsymbol{y}_i\}_{i=1}^n$ are matched. Then, we have d qq-plots, one for every dimension. These qq-plots are named fuzzy qq-plots [21]. Considering the spatial ranks, the quantiles are [18]:

$$\boldsymbol{Q}_X(\boldsymbol{u}_i) = \boldsymbol{x}_i, \qquad \forall i \in \{1, \dots, n\}, \tag{2.20}$$

$$\boldsymbol{Q}_Y(\boldsymbol{u}_i) = \boldsymbol{y}_{\sigma(i)}, \quad \forall i \in \{1, \dots, n\}. \tag{2.21}$$

2.2.4 Siamese Network and Its Losses

Siamese network is composed of two or three networks sharing their weights which are used for increasing and decreasing the inter- and intra-class variances of data in the embedding space [60]. I require this for our proposed Fisher losses (see Section 3.3.2.1) and usage of Siamese network for histopathology embedding (see Sections 5.1.2, 5.1.3, and 5.1.4). In the following, I review the technical background for Siamese network and its popular loss functions.

2.2.4.1 Siamese Network

Siamese network is a set of several (typically two or three) networks which share weights with each other [118]. The weights are trained using a loss based on anchor, neighbor (positive), and distant (negative) samples, where anchor and neighbor belong to the same class, but the anchor and distant tiles are in different classes. I denote the anchor, neighbor, and distant samples by $\boldsymbol{x}_a$, $\boldsymbol{x}_n$, and $\boldsymbol{x}_d$, respectively. The loss functions used to train a Siamese network usually make use of the anchor, neighbor, and distant samples, trying to pull the anchor and neighbor towards one another and simultaneously push the anchor and distant tiles away from each other. In the following, two different loss functions are introduced for training Siamese networks.

2.2.4.2 Triplet Loss for Siamese Network

The triplet loss uses anchor, neighbor, and distant. Let $\mathbf{f}(\boldsymbol{x})$ be the output (i.e., embedding) of the network for the input $\boldsymbol{x}$. The triplet loss tries to reduce the distance of anchor and neighbor embeddings and desires to increase the distance of anchor and distant embeddings. As long as the distances of anchor-distant pairs get larger than the distances of anchor-neighbor pairs by a margin $\alpha \geq 0$, the desired embedding is obtained. The triplet loss, to be minimized, is defined as [118]:

$$\ell_{\mathrm{t}} = \sum_{i=1}^{b} \Big[\|\mathbf{f}(\boldsymbol{x}_a^i) - \mathbf{f}(\boldsymbol{x}_n^i)\|_2^2 - \|\mathbf{f}(\mathbf{x}_a^i) - \mathbf{f}(\boldsymbol{x}_d^i)\|_2^2 + \alpha \Big]_+, \tag{2.22}$$

where $\boldsymbol{x}^i$ is the i-th triplet sample in the mini-batch, b is the mini-batch size, $[z]_+ := \max(z, 0)$ is the standard Hinge loss, and $||\cdot||_2$ denotes the ℓ_2 norm.

2.2.4.3 Contrastive Loss for Siamese Network

The contrastive loss uses pairs of samples which can be anchor and neighbor or anchor and distant. If the samples are anchor and neighbor, they are pulled towards each other; otherwise, their distance is increased. In other words, the contrastive loss performs like the triplet loss but one by one rather than simultaneously. The desired embedding is obtained when the anchor-distant distances get larger than the anchor-neighbor distances by a margin of α. This loss, to be minimized, is defined as [65]:

$$\ell_{\mathrm{c}} = \sum_{i=1}^{b} \Big[(1-y)||\mathbf{f}(\boldsymbol{x}_1^i) - \mathbf{f}(\boldsymbol{x}_2^i)||_2^2 + y\left[-||\mathbf{f}(\boldsymbol{x}_1^i) - \mathbf{f}(\boldsymbol{x}_2^i)||_2^2 + \alpha\right]_+ \Big], \tag{2.23}$$

where y is zero and one when the pair $\{\boldsymbol{x}_1^i, \boldsymbol{x}_2^i\}$ is anchor-neighbor and anchor-distant, respectively.

2.2.5 Bayesian Updating Theorem

Bayesian updating theorem is used for updating the parameters of distribution after receiving some new data; hence, it is useful for online streaming data. I require this for our proposed BUT and BUNCA (see Section 3.3.2.2) and usage of BUT and BUNCA for histopathology embedding (see Sections 5.1.5). In the following, I review the technical background for Bayesian updating and conjugate priors.

2.2.5.1 Bayesian Updating

Let X and θ be two random variables where θ is a parameter of the distribution of X. According to Bayes' rule, we have:

$$\mathbb{P}(\theta|X) = \frac{\mathbb{P}(X|\theta)\,\mathbb{P}(\theta)}{\mathbb{P}(X)} \implies \mathbb{P}(\theta|X) \propto \mathbb{P}(X|\theta)\,\mathbb{P}(\theta), \tag{2.24}$$

which shows the relation of the posterior $\mathbb{P}(\theta|X)$, likelihood $\mathbb{P}(X|\theta)$, and prior $\mathbb{P}(\theta)$. Given some data X and the prior over the parameter of interest θ, we want to find the posterior using Eq. (2.24). This is the basic idea behind *Bayesian updating* in which the posterior over the parameter of interest is updated after receiving some new data, i.e., using the new data X, we have $\mathbb{P}(\theta) \mapsto \mathbb{P}(\theta|X)$ [75].

2.2.5.2 Conjugate Priors

If the posterior distribution $\mathbb{P}(\theta|X)$ and the prior distribution $\mathbb{P}(\theta)$ are in the same probability distribution family, they are called *conjugate distributions* and the prior is the *conjugate prior* for the likelihood $\mathbb{P}(X|\theta)$. Assume there already exist some data, denoted by X^0, and some new data, X', are received. The existing data X^0 has a distribution with some parameter(s) θ. The posterior of the parameter of interest, i.e., $\mathbb{P}(\theta|X)$, can be updated using the new data. Hence, this can be used to update the parameter(s) of the distribution of X using the newly received data [75].

Let the data X have a multivariate normal (or Gaussian) distribution, so their likelihood is $\mathbb{P}(X|\theta)$. Assume both the mean and covariance of likelihood are considered as random variables, so θ includes mean and covariance. Using the new data X', we want to update the parameters, mean and covariance, of the normal distribution. In this case, the likelihood $\mathbb{P}(X|\theta)$ has a multivariate normal distribution, and for updating the posterior, we should use the conjugate prior for the likelihood. The conjugate prior distribution for the multivariate normal distribution with both random mean and covariance is the normal-inverse-Wishart distribution [101]. In my analysis, I also require the skewed generalized Student-t distribution. For the theory of relevant distributions for dynamic sampling, please refer to [101, 119].

2.2.5.3 Updates of Parameters for Multivariate Normal Distribution

Assume the distribution of batch of n_0 data instances, which we already have, is the multivariate normal distribution with the mean $\boldsymbol{\mu}^0$ and covariance matrix $\boldsymbol{\Sigma}^0$. If we have n' new data instances with mean $\boldsymbol{\mu}'$ and covariance matrix $\boldsymbol{\Sigma}'$.

According to Bayesian updating, the mean and covariance of distribution of data can be updated (see [119]). The mean and covariance matrix of the distribution of data can be updated by the expectation of marginal distributions for the mean and covariance. According to the expectations of these two distributions which can be found in [101, 119], the updates of mean and covariance of the j-th class can be given as:

$$\boldsymbol{\mu}^0 \leftarrow \mathbb{E}(\boldsymbol{\mu}\,|\,\boldsymbol{x}^0) = \frac{n'\boldsymbol{\mu}' + n_0\boldsymbol{\mu}^0}{n' + n_0}, \tag{2.25}$$

$$\boldsymbol{\Sigma}^0 \leftarrow \mathbb{E}(\boldsymbol{\Sigma}\,|\,\boldsymbol{x}^0) = \frac{\boldsymbol{\Upsilon}^{-1}}{n'+n_0-d-1}, \quad \forall\, n'+n_0>d+1, \tag{2.26}$$

where:

$$\mathbb{R}^{d\times d} \ni \boldsymbol{\Upsilon} := n'\boldsymbol{\Sigma}' + n_0\boldsymbol{\Sigma}^0 + \frac{n_1' n_0}{n_1' + n_0}(\boldsymbol{\mu}^0 - \boldsymbol{\mu}')(\boldsymbol{\mu}^0 - \boldsymbol{\mu}')^\top, \tag{2.27}$$

and $\boldsymbol{\mu}'$, $\boldsymbol{\mu}^0$, $\boldsymbol{\Sigma}'$, and $\boldsymbol{\Sigma}^0$ can be calculated by sample mean and sample covariance matrix using the new batch of data. Note that for $n' + n_0 \leq d + 1$, the covariance matrix can be updated by Maximum Likelihood Estimation (MLE).

2.2.6 Matrix Decomposition and Factorization

Matrix decomposition factorizes the matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ into multiplication of two matrices $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^\top$ where the columns of $\boldsymbol{U} \in \mathbb{R}^{d\times k}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times k}$ can be interpreted as bases and corresponding coefficients, respectively, and $k \in \mathbb{Z}_+$, usually $k := \min(d, n)$. There exist many different types of matrix decomposition such as eigenvalue decomposition, Singular Value Decomposition (SVD), Nonnegative Matrix Factorization (NMF), PLU Decomposition, QR Decomposition, Cholesky Decomposition, and Dictionary Learning (DL). I use these for my proposed IRMD (see Section 4.1.2).

SVD decomposes the matrix as $\boldsymbol{X} = \widetilde{\boldsymbol{U}}\boldsymbol{\Lambda}\widetilde{\boldsymbol{V}}^\top$ where columns of $\widetilde{\boldsymbol{U}} \in \mathbb{R}^{d\times k}$ are eigenvectors ($\widetilde{\boldsymbol{u}}$) of $\boldsymbol{X}^\top\boldsymbol{X}$, columns of $\widetilde{\boldsymbol{V}} \in \mathbb{R}^{n\times k}$ are eigenvectors ($\widetilde{\boldsymbol{v}}$) of $\boldsymbol{X}\boldsymbol{X}^\top$, $\boldsymbol{\Lambda} \in \mathbb{R}^{k\times k}$ is a diagonal matrix, containing singular values. NMF targets decomposition of a matrix with nonnegative entries $\boldsymbol{X} \in \mathbb{R}^{d\times n}_{\geq 0}$ into $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^\top$ where $\boldsymbol{U} \in \mathbb{R}^{d\times k}_{\geq 0}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times k}_{\geq 0}$ are also nonnegative. PLU decomposition is a method for solving linear systems of equations based on Gaussian elimination using elementary matrices. It decomposes matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ as $\boldsymbol{X} = \boldsymbol{P}\widetilde{\boldsymbol{L}}\widetilde{\boldsymbol{U}}$ where $\boldsymbol{P} \in \mathbb{R}^{d\times d}$ is the permutation matrix. The matrices $\widetilde{\boldsymbol{L}} \in \mathbb{R}^{d\times k}$ and $\widetilde{\boldsymbol{U}} \in \mathbb{R}^{k\times n}$ are lower and upper triangular matrices, respectively. The QR decomposition factorizes the matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ as $\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{R}$ where $\boldsymbol{R} \in \mathbb{R}^{k\times n}$ is upper triangular and $\boldsymbol{Q} \in \mathbb{R}^{d\times k}$ is an orthogonal matrix whose columns, as basis vectors, span the same space as the columns of $\boldsymbol{X}$. The DL tries to decompose matrix into $\mathbb{R}^{d\times n} \ni \boldsymbol{X} = \boldsymbol{D}\boldsymbol{R}$ where $\boldsymbol{D} \in \mathbb{R}^{d\times k}$ is the dictionary whose columns are basis vectors also called atoms, and $\mathbb{R}^{k\times n} \ni \boldsymbol{R} = [\boldsymbol{r}_1, \dots, \boldsymbol{r}_n]$ is the representation (components).

2.2.7 Polyhedron Curvature

I require the concepts of polyhedron curvature and angular defect in my proposed CAD algorithm (see Section 4.2.1). In the following, I introduce these concepts.

A *polytope* is a geometrical object in $\mathbb{R}^d$ whose faces are planar. The special cases of polytope in $\mathbb{R}^2$ and $\mathbb{R}^3$ are called *polygon* and *polyhedron*, respectively. Some examples for polyhedron are cube, tetrahedron, octahedron, icosahedron, and dodecahedron with four, eight, and twenty triangular faces, and twelve flat faces, respectively. Consider a polygon where τ_j and μ_j are the interior and exterior angles at the j-th vertex; we have $\tau_j + \mu_j = \pi$. A similar analysis holds in $\mathbb{R}^3$ for Fig. 2.1-a. In this figure, a vertex of a polyhedron and its opposite cone are shown where the opposite cone is defined to have perpendicular faces to the faces of the polyhedron at the vertex. The intersection of a unit sphere centered at the vertex and the opposite cone is shown in the figure. This intersection is a geodesic on the unit sphere. According to Thomas Harriot's theorem [97], if this geodesic on the unit sphere is a triangle, its area is $\mu_1+\mu_2+\mu_3-\pi = 2\pi-(\tau_1+\tau_2+\tau_3)$. The generalization of this theorem from a geodesic triangular polygon (3-gon) to an k-gon is $\mu_1 + \dots + \mu_k - k\pi + 2\pi = 2\pi - \sum_{a=1}^{k} \tau_a$ [97], where the polyhedron has k faces meeting

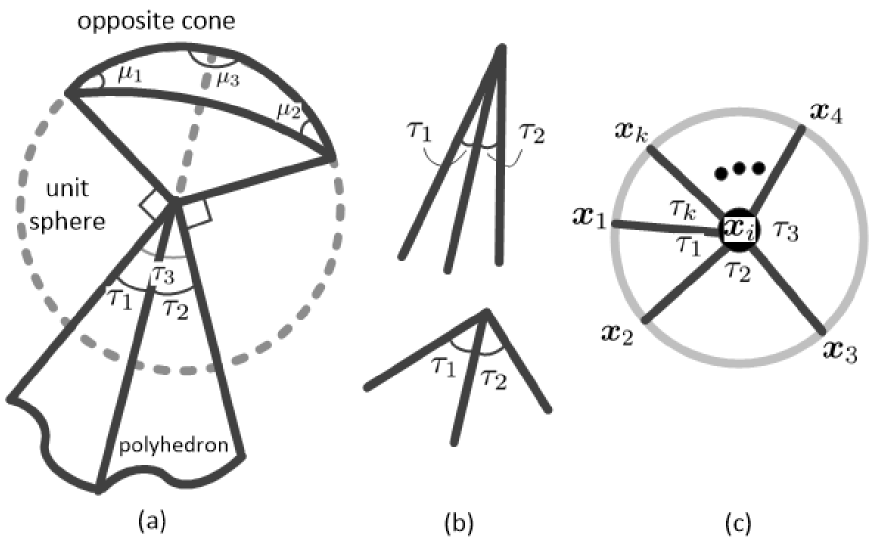

Figure 2.1: Polyhedron curvature: (a) polyhedron vertex, unit sphere, and the opposite cone, (b) large and small curvature, (c) a point and its neighbors normalized on a unit hyper-sphere around it.

at the vertex.

Rene Descartes defined *angular defect* at a vertex $\boldsymbol{x}$ of a polyhedron as [17]: $\mathcal{D}(\boldsymbol{x}) := 2\pi - \sum_{a=1}^{k} \tau_a$. The total defect of a polyhedron is defined as the summation of the defects over the vertices. It can be shown that the total defect of a polyhedron with v vertices, e edges, and f faces is: $\mathcal{D} := \sum_{i=1}^{v} \mathcal{D}(\boldsymbol{x}_i) = 2\pi(v - e + f)$. The term $v - e + f$ is Euler-Poincaré characteristic of the polyhedron; therefore, the total defect of a polyhedron is equal to its Euler-Poincaré characteristic. According to Fig. 2.1-b, the smaller τ angles result in sharper corner of the polyhedron. Therefore, I can consider the angular defect as the *curvature* of the vertex.

2.2.8 Ensemble Methods

Isolation forest and Mondrian forest are two ensemble methods for batch anomaly detection and online classification, respectively. I require these two methods in my proposed iMondrian forest algorithm (see Section 4.3.1). In the following, I introduce these methods.

2.2.8.1 Isolation Forest

An iForest [90] is an ensemble of isolation trees. An *isolation tree* is an extremely randomized tree where the tree is a proper binary tree and its splitting dimension q and splitting value p are randomly selected at every node. The tree grows until every leaf includes exactly one data point, i.e., $|\mathcal{X}| = 1$ in the leaf node. Let $h(\boldsymbol{x})$ denote the path length for a data point $\boldsymbol{x}$ in the tree where the path length is defined as the number of edges $\boldsymbol{x}$ traverses from the root to the leaf it belongs to. The average height of an isolation tree is $\log(n)$. As the structure of the isolation tree is equivalent to the binary search tree, the estimation of average path length in isolation trees is: $c(n) := 2\,h(n-1) - \big(2\,(n-1)/n\big)$, where $h(i)$ is the i-th harmonic number, defined as $h(i) := \ln(i) + 0.5772156649$, where the added constant is the Euler's constant. The anomaly score of a point $\boldsymbol{x}$ is:

$$s(\boldsymbol{x}) := 2^{-\mathbb{E}(l(\boldsymbol{x}))/c(n)}, \tag{2.28}$$

where $\mathbb{E}(l(\boldsymbol{x}))$ is the expected path length for the data point $\boldsymbol{x}$ among the trees of the forest:

$$\mathbb{E}(l(\boldsymbol{x})) := \frac{1}{|\mathcal{F}|} \sum_{t=1}^{|\mathcal{F}|} l_t(\boldsymbol{x}), \tag{2.29}$$

where $l_t(\boldsymbol{x})$ is the path length of $\boldsymbol{x}$ in the t-th tree and $|\mathcal{F}|$ is the population of trees in the forest. The intuition of anomaly score in iForests is that the anomalies tend to be isolated sooner, i.e., shallower in the tree.

2.2.8.2 Mondrian Forest

A *Mondrian forest* [83] is an ensemble of Mondrian trees which are based on the Mondrian process. *Mondrian processes* [111] are families of random hierarchical binary partitions and probability distributions over tree data structures. While Mondrian processes are infinite structures, Mondrian trees are restrictions of Mondrian processes on a finite set of points. Every node r in the Mondrian tree has a *split time* τ_r which increases with the depth of the node. The split time is zero at the root and infinite at the leaves of the tree.

Let $\hat{\mathcal{B}}_r := (\hat{\ell}_{r1}, \hat{u}_{r1}] \times \cdots \times (\hat{\ell}_{rd}, \hat{u}_{rd}]$ for the r-th node, where $\hat{\ell}_{rj}$ and $\hat{u}_{rj}$ are the lower and upper bounds of hyper-rectangular block $\hat{\mathcal{B}}_r$ along dimension j. The Mondrian tree considers the smallest block containing the data points in a node; therefore, it defines $\mathcal{B}_r := (\ell_{r1}, u_{r1}] \times \cdots \times (\ell_{rd}, u_{rd}]$ where ℓ_{rj} and u_{rj} are the lower and upper bounds of the smallest hyper-rectangular block $\mathcal{B}_r$ along dimension j. For a node indexed by r, let $\boldsymbol{\ell}_{\mathcal{X}_b} = [\ell_{r1}, \dots, \ell_{rd}]^\top$ and $\boldsymbol{u}_{\mathcal{X}_b} = [u_{r1}, \dots, u_{rd}]^\top$; thus, $\boldsymbol{\ell}_{\mathcal{X}_b} := \min(\{\boldsymbol{x}_i^{(b)} \mid \forall i\})$ and $\boldsymbol{u}_{\mathcal{X}_b} := \max(\{\boldsymbol{x}_i^{(b)} \mid \forall i\})$ where $\mathcal{X}_b = \{\boldsymbol{x}_i^{(b)}\} = \{\boldsymbol{x}_i \mid \boldsymbol{x}_i \in \mathcal{B}_r\}$. For the r-th node, the split time of a node is determined as $\tau_{\text{parent}(j)} + e$ where e is a random variable from an exponential distribution with a rate which is a function of $\boldsymbol{\ell}_{\mathcal{X}_b}$ and $\boldsymbol{u}_{\mathcal{X}_b}$. Depending on whether the split time of the node is smaller or greater than the split time of its parent, it is put before or after the parent node in the tree. Mondrian trees can be updated with new data making them suitable for online streaming domains.

2.3 Open Problems in Data Reduction & Contributions of Thesis

In this section, I list several open problems in different categories of dimensionality and numerosity reduction and I explain which of my proposed methods have tackled those problems. In this thesis, I build a taxonomy for data reduction (as seen in Fig. 1.2) and try to fill the gaps and open problems in every category of this taxonomy. Moreover, I apply the proposed methods to application domains. I look for insights into understanding features, dimensions, and instances in data. Since the cutting edge of deep learning usually does not do this analysis, I try to tie theoretical knowledge about manifolds into the modern machine and deep learning. Note that Chapters 3 and 4 are my contributions and Chapter 5 is my contributions in different applications.

2.3.1 Open Problems & My Contributions in Dimensionality Reduction

There are several open problems in manifold learning (dimensionality reduction) which I address in this proposal.

2.3.1.1 Open Problems in Spectral Dimensionality Reduction

- A problem with the supervised subspace learning methods is that they see the pairs of classes with the same eye; although, the distances and confusion of classes are different from each other. A weighting procedure can be useful for addressing this issue; the proposed WFDA does this.

- There exist several spectral dimensionality reduction methods which are based on the generalized eigenvalue problem [42]. One can propose a novel spectral dimensionality reduction method which uses the optimization of generalized eigenvalue problem. I propose RDA which uses this general form of optimization and, therefore, generalizes PCA, SPCA, and FDA.

- A problem is that most of the manifold learning methods, such as PCA, and LLE, use MSE or ℓ_2 norm in their formulation. However, in image fidelity assessment, it is shown that MSE is not promising enough [134]. SSIM [135, 133] has shown its merit in image quality assessment. The SSIM considers luminance, contrast, and non-structural distortions all together but gives more attention to structural distortions which are more noticed by human vision. I tackle this problem by proposing image structure manifold which captures the intrinsic features of an image in terms of structural similarity and distortions and can discriminate the various types of image distortions. Three methods, which are SSIM kernel, ISCA, and LLISE, are proposed for learning this manifold.

2.3.1.2 Open Problems in Probabilistic Dimensionality Reduction

- An open problem of the dimensionality reduction methods is that these methods either do not specify the distribution of the embedded data points in the embedded space and merely focus on preserving the local or global distances, such as LLE [110], Isomap [126], MVU [136], and Sammon mapping [113], or they only restrict the distribution in the embedded space to be a specific distribution, such as SNE [69] and t-SNE [94]. There is a lack of a method in the literature which gives the user a freedom to choose the distribution in the embedding space. In this proposal, I propose QQE which tackles this problem. The proposed QQE is capable of both distribution transformation and manifold embedding where the desired distribution can be chosen by the user.

2.3.1.3 Open Problems in Neural Network-based Dimensionality Reduction

- After the tremendous progress of neural networks trained by backpropagation [112], it is a good time to move on to newer trainign algorithms for neural networks to have more insights in neural nets; Geoffrey Hinton has said in one of his recent seminars. My proposed backpropagation is a novel training algorithm for neural networks with a projection based perspective.

- Triplet [118] and contrastive [65] losses for training the Siamese triplet networks have conceptual similarities with FDA because in both Siamese network and FDA, the inter-class and intra-class variances are tried to be maximized and minimized, respectively. For this similarity, one can propose a fusion of these methods for the sake of deep metric learning. The proposed Fisher losses, FDT and FDC losses, are the result of this fusion.

- There is a degree of freedom in triplet mining for training Siamese networks. It is how the triplets are sampled. It is shown in [139] that sampling of the triplets also matters in learning deep embeddings. With Siamese triplet networks, drawing more informative and stable triplets from the pool of samples will lead to qualitatively more salient embeddings. The sampling based triplet mining methods in the literature sample the triplets from the existing embedded data instances [139] so it does not use the stochastic information of the embedding space. By proposing BUT and BUNCA, I draw the positive and negative samples for every anchor instance in a dynamic manner stochastically.

2.3.2 Open Problems & My Contributions in Numerosity Reduction

There are several open problems in numerosity reduction and anomaly detection which I address in this thesis.

2.3.2.1 Open Problems in Algorithms Based on Variance & Geometry

- As surveyed in [28], most of the numerosity reduction methods are proposed merely for classification. However, regression and clustering are also important in pattern recognition. Some of the proposed methods, such as DROP [137] and ENN [138], cannot be used for regression and clustering while some other methods, such as SOS [76] and SE [89], can be slightly modified to be useful for these tasks although their papers only tackle classification. The proposed PSA, IRMD, and CAD are task agnostic.

- Most of, but not all of, the numerosity reduction methods do not rank the data and just retain a subset of data by removing the rest of data. The *scoring* or *ranking* of data instances can have important information. For example, with the opposite perspective to prototype selection, it can be used for anomaly or outlier detection.

Also, it is common in eigenvalue based feature extraction methods, such as PCA, to order the projection directions. This *ordering* can be applied in prototype selection using ranking the instances. An example of existing methods which rank instances is SOS. The proposed PSA, IRMD, and CAD are capable of ranking the instances.

- There is a gap in numerosity reduction and anomaly detection for methods based on geometry. The proposed CAD and iCAD and their kernel variants are based on polyhedron curvature and give a geometrical insight to the problem. Because of this insight, the concepts of anomaly landscape and anomaly path are also proposed which can be used for image denoising.

2.3.2.2 Open Problems in Algorithms Based on Ensemble Learning

- On one hand, isolation forest is an existing method for batch anomaly detection. Isolation forest is not capable of handling streaming data. On the other hand, Mondrian forest is an existing method for online random forest. One can fuse these two methods for having online anomaly detection. The proposed iMondrian forest is a novel hybrid of isolation forest and Mondrian forest. iMondrian forest takes the idea of isolation, using the depth of a node in a tree, and implements it in the Mondrian forest structure. The result is a new data structure which can accept streaming data in an online manner while being used for anomaly detection.

2.3.3 Open Problems & My Contributions in Applications of Data Reduction

There are some open problems in applications of data reduction, which I tackle in this thesis.

2.3.3.1 Open Problems in Medical Image Analysis

- Embedding the histopathology patches requires to embed the similar patterns close to each other and put the different patterns far from one another. This results in increasing the inter-class variance (variances of different patches) and decreasing the intra-class variance (variances of similar patches) in the embedded data. This reminds us of the intuition of FDA (see Section 2.2.1). Hence, one can embed the histopathology data using the proposed Fisher losses.

- Training the triplet Siamese networks requires triplets of anchor, positive, and negative patches. Most of the triplet mining methods, use extreme distances (nearest/farthest neighbors) within the batch. There is gap of methods which use the triplets with extreme distances in the whole dataset, in an offline manner. As the patterns of histopathology data are similar and dissimilar in different tissue types, this offline mining can be effective in histopathology triplet mining. I propose offline triplet mining to tackle this issue.

- In online triplet mining, the triplets are usually sampled from the existing embedded batch instances. However, one can consider the distribution of embedded data stochastically and draw triplet samples from the distribution. The proposed BUT and BUNCA algorithms are used for online triplet sampling for histopathology data.

2.3.3.2 Open Problems in Image Processing & Computer Vision

- Eigenfaces [130] and Fisherfaces [6] have been proposed in the literature for subspace learning for the sake of face recognition. However, facial subspace learning based on generalized eigenvalue problem can be generalized using the proposed RDA in this thesis. My proposed Roweisfaces generalize the methods of eigenfaces and Fisherfaces for face recognition.

- Fisherposes [56] is a 3D action recognition method which makes use of FDA for subspace learning for embedding the body poses. The proposed RDA can be used to generalize this method for embedding the body poses for the sake of action recognition. The proposed Roweisposes generalizes eigenposes, supervised eigenposes, and Fisherposes.

- There exist different image denoising methods; however, a possible approach for a geometrical approach exists for image denoising. The proposed anomaly path, in the CAD method, is used for image denoising using anomaly paths. This method has a geometrical approach where the noisy images can be seen as a vertex with high curvature in a polyhedron.

2.4 Summary of the Chapter

This chapter reviewed the related literature for spectral, probabilistic, and neural network-based dimensionality reduction as well as the prototype selection and anomaly detection approaches for numerosity reduction. I explained the technical background of linear subspace learning based on generalized eigenvalue problem, SSIM, quantile-quantile plot, Siamese network, triplet loss, contrastive loss, Bayesian updating theorem, matrix decomposition, polyhedron curvature, isolation forest, and Mondrian forest. Finally, the open problems and my contributions were explained in dimensionality reduction, numerosity reduction, and application of data reduction. Please note that I have mentioned all required background in this chapter so that chapters 3 and 4 purely include all my contributions. Moreover, chapter 5 contains my proposed methods for applications of data reduction.

Chapter 3

Proposed Algorithms for Dimensionality Reduction

Dimensionality reduction tries to find a new set of features by finding a discriminative subspace or embedding space. In this chapter, I propose several algorithms for dimensionality reduction. Note that the algorithms, which are proposed in this thesis, are developed based on the mathematics and techniques developed in previous work, introduced in Chapter 2. This chapter divides dimensionality reduction into three categories which are spectral (Section 3.1), probabilistic (Section 3.2), and neural network-based (Section 3.3) dimensionality reduction methods. In spectral category, it proposes WFDA (Section 3.1.1), RDA (Section 3.1.2), and image quality aware embedding (Section 3.1.3) where the last contains three proposed methods, i.e., SSIM kernel (Section 3.1.3.2), ISCA (Section 3.1.3.3), and LLISE (Section 3.1.3.4). This chapter also proposes QQE (Section 3.2.1) in the probabilistic dimensionality reduction group. It divides the neural network-based methods into shallow and deep nets where backprojection (Section 3.3.1) is proposed for the former and Fisher losses (FDT and FDC losses) (Section 3.3.2.1), as well as BUT and BUNCA (Section 3.3.2.2), are proposed in the latter. Note that some applications of these methods, in the fields of medical image analysis, image processing, and computer vision, will be proposed in Chapter 5.

3.1 Spectral Dimensionality Reduction

In the following, I propose new algorithms for spectral dimensionality reduction. My proposed methods are WFDA, RDA, and image quality aware embedding (including SSIM kernel, ISCA, and LLISE).

3.1.1 Weighted Fisher Discriminant Analysis

In this section, I propose WFDA, including CW-FDA and AW-FDA, and Weighted Kernel FDA (W-KFDA), including Cosine Weighted Kernel FDA (CW-KFDA), AW-KFDA.

3.1.1.1 Motivation and Formulation

FDA, explained in Section 2.2.1 treats all pairs of classes in the same way (see Eq. (2.3)); however, some classes might be much further from one another compared to other classes. In other words, the distances of classes are different. Treating closer classes need more attention because classifiers may more easily confuse them whereas classes far from each other are generally easier to separate. Hence, a weighting procedure might be more appropriate. The optimization in WFDA is:

$$\begin{aligned} \underset{\boldsymbol{U}}{\text{maximize}} \quad & \textbf{tr}(\boldsymbol{U}^\top \widehat{\boldsymbol{S}}_B\, \boldsymbol{U}), \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{S}_W\, \boldsymbol{U} = \boldsymbol{I}, \end{aligned} \tag{3.1}$$

where the $\boldsymbol{S}_W \in \mathbb{R}^{d\times d}$ and $\boldsymbol{S}_B \in \mathbb{R}^{d\times d}$ are the intra-class (within) and weighted inter-class (between) scatters, respectively [43]:

$$\boldsymbol{S}_W := \sum_{r=1}^{c}\sum_{i=1}^{n_r} n_r(\boldsymbol{x}_i^{(r)} - \boldsymbol{\mu}^{(r)})(\boldsymbol{x}_i^{(r)} - \boldsymbol{\mu}^{(r)})^\top = \sum_{r=1}^{c} n_r\, \breve{\boldsymbol{X}}_r\, \breve{\boldsymbol{X}}_r^\top, \tag{3.2}$$

$$\widehat{\boldsymbol{S}}_B := \sum_{r=1}^{c}\sum_{\ell=1}^{c} \alpha_{r\ell}\, n_r\, n_\ell(\boldsymbol{\mu}^{(r)} - \boldsymbol{\mu}^{(\ell)})(\boldsymbol{\mu}^{(r)} - \boldsymbol{\mu}^{(\ell)})^\top = \sum_{r=1}^{c} n_r\, \boldsymbol{M}_r\, \boldsymbol{A}_r\, \boldsymbol{N}\, \boldsymbol{M}_r^\top, \tag{3.3}$$

where $\mathbb{R}^{d\times n_r} \ni \breve{\boldsymbol{X}}_r := [\boldsymbol{x}_1^{(r)} - \boldsymbol{\mu}^{(r)}, \dots, \boldsymbol{x}_{n_r}^{(r)} - \boldsymbol{\mu}^{(r)}]$, $\mathbb{R}^{d\times c} \ni \boldsymbol{M}_r := [\boldsymbol{\mu}^{(r)} - \boldsymbol{\mu}^{(1)}, \dots, \boldsymbol{\mu}^{(r)} - \boldsymbol{\mu}^{(c)}]$, and $\mathbb{R}^{c\times c} \ni \boldsymbol{N} := \textbf{diag}([n_1, \dots, n_c]^\top)$. The mean of the r-th class is $\mathbb{R}^d \ni \boldsymbol{\mu}^{(r)} := (1/n_r)\sum_{i=1}^{n_r} \boldsymbol{x}_i^{(r)}$. Also, $\mathbb{R} \ni \alpha_{r\ell} \geq 0$ is the weight for the pair of the r-th and ℓ-th classes, $\mathbb{R}^{c\times c} \ni \boldsymbol{A}_r := \textbf{diag}([\alpha_{r1}, \dots, \alpha_{rc}])$. In FDA, we have $\alpha_{r\ell} = 1,\ \forall r, \ell \in \{1, \dots, c\}$. However, it is better for the weights to be decreasing with the distances of classes to concentrate more on the nearby classes. I denote the distances of the r-th and ℓ-th classes by $d_{r\ell} := ||\boldsymbol{\mu}^{(r)} - \boldsymbol{\mu}^{(\ell)}||_2$. The solution of Eq. (3.1) is the generalized eigenvalue problem $(\widehat{\boldsymbol{S}}_B, \boldsymbol{S}_W)$ [42].

3.1.1.2 Cosine Weighted Fisher Discriminant Analysis

Literature has shown that cosine similarity works very well with FDA. Moreover, according to the opposition-based learning [128], capturing similarity and dissimilarity of data points can improve the performance of learning. A promising operator for capturing similarity and dissimilarity (opposition) is cosine. Hence, I propose CW-FDA, as a manually weighted method, with cosine to be the weight defined as:

$$\alpha_{r\ell} := 0.5 \times \big[1 + \cos\big(\measuredangle(\boldsymbol{\mu}^{(r)}, \boldsymbol{\mu}^{(\ell)})\big)\big] = 0.5 \times \big[1 + \frac{\boldsymbol{\mu}^{(r)\top}\boldsymbol{\mu}^{(\ell)}}{||\boldsymbol{\mu}^{(r)}||_2||\boldsymbol{\mu}^{(\ell)}||_2}\big], \tag{3.4}$$

to have $\alpha_{r\ell} \in [0,1]$. Note that as I do not care about $\alpha_{r,r}$, because inter-class scatter for $r = \ell$ is zero, I set $\alpha_{rr} = 0$.

3.1.1.3 Automatically Weighted Fisher Discriminant Analysis

In AW-FDA, at the same time where we want to maximize the Fisher criterion, the optimal weights are found. Hence, there are $c+1$ matrix optimization variables which are $\boldsymbol{V}$ and $\boldsymbol{A}_k \in \mathbb{R}^{c\times c}, \forall k \in \{1,\dots,c\}$. Moreover, to use the betting on sparsity principle [26], I can make the weight matrix sparse, so I use "ℓ_0" norm for the weights to be sparse. The optimization problem is as follows

$$\begin{aligned} \underset{\boldsymbol{U},\boldsymbol{A}_r}{\text{maximize}} \quad & \textbf{tr}(\boldsymbol{U}^\top \widehat{\boldsymbol{S}}_B \boldsymbol{U}), \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U} = \boldsymbol{I}, \\ & ||\boldsymbol{A}_r||_0 \leq k, \quad \forall r \in \{1,\dots,c\}. \end{aligned} \tag{3.5}$$

I use alternating optimization to solve this problem:

$$\boldsymbol{U}^{(\tau+1)} := \arg\max_{\boldsymbol{U}} \Big(\textbf{tr}(\boldsymbol{U}^\top \widehat{\boldsymbol{S}}_B^{(\tau)} \boldsymbol{U}) \,\big|\, \boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U} = \boldsymbol{I}\Big), \tag{3.6}$$

$$\boldsymbol{A}_r^{(\tau+1)} := \arg\min_{\boldsymbol{A}_r} \Big(-\,\textbf{tr}(\boldsymbol{U}^{(\tau+1)\top} \widehat{\boldsymbol{S}}_B \boldsymbol{U}^{(\tau+1)}) \,\big|\, ||\boldsymbol{A}_r||_0 \leq k\Big), \forall r, \tag{3.7}$$

where τ denotes the iteration.

Since I use an iterative solution for the optimization, it is better to normalize the weights in the weighted inter-class scatter; otherwise, the weights gradually explode to maximize the objective function. I use $\breve{\boldsymbol{A}}_r := \boldsymbol{A}_r/||\boldsymbol{A}_r||_F^2$ for weights.

As discussed before, the solution to Eq. (3.6) is the generalized eigenvalue problem $(\widehat{\boldsymbol{S}}_B^{(\tau)}, \boldsymbol{S}_W)$. I use a step of gradient descent to solve Eq. (3.7) followed by satisfying the "ℓ_0" norm constraint. The gradient is calculated using chain rule:

$$\mathbb{R}^{c\times c} \ni \frac{\partial f}{\partial \boldsymbol{A}_r} = \mathbf{vec}^{-1}_{c\times c}\Big[(\frac{\partial \breve{\boldsymbol{A}}_r}{\partial \boldsymbol{A}_r})^\top (\frac{\partial \widehat{\boldsymbol{S}}_B}{\partial \breve{\boldsymbol{A}}_r})^\top \mathbf{vec}(\frac{\partial f}{\partial \widehat{\boldsymbol{S}}_B})\Big]. \tag{3.8}$$

For the derivation of the gradient, please see [55]. After the gradient descent step, to satisfy the condition $||\boldsymbol{A}_r||_0 \leq k$, the solution is projected onto the set of this condition. Because $-f$ should be maximized, this projection is to set the $(c-k)$ smallest diagonal entries of $\boldsymbol{A}_r$ to zero [71]. After solving the optimization in Eq. (3.5), the p leading columns of $\boldsymbol{U}$ are the OW-FDA projection directions that span the subspace.

3.1.1.4 Weighted Kernel Fisher Discriminant Analysis

The problem of treating the pairs of classes similarly also exists in kernel FDA. Here, for the reason explained in Chapter 1, I also propose the kernel version of WFDA. I define the optimization for W-KFDA as:

$$\begin{aligned} \underset{\boldsymbol{Y}}{\text{maximize}} \quad & \mathbf{tr}(\boldsymbol{Y}^\top \widehat{\boldsymbol{\Delta}}_B \boldsymbol{Y}), \\ \text{subject to} \quad & \boldsymbol{Y}^\top \boldsymbol{\Delta}_W \boldsymbol{Y} = \boldsymbol{I}, \end{aligned} \tag{3.9}$$

where the intra-class and weighted inter-class scatters in the feature space are, respectively, defined as (see [43] for derivation):

$$\boldsymbol{\Delta}_W := \sum_{r=1}^{c} n_r \boldsymbol{K}_r \boldsymbol{H}_r \boldsymbol{K}_r^\top, \tag{3.10}$$

$$\widehat{\boldsymbol{\Delta}}_B := \sum_{r=1}^{c}\sum_{\ell=1}^{c} \alpha_{r\ell}\, n_r\, n_\ell (\boldsymbol{\xi}^{(r)} - \boldsymbol{\xi}^{(\ell)})(\boldsymbol{\xi}^{(r)} - \boldsymbol{\xi}^{(\ell)})^\top = \sum_{r=1}^{c} n_r\, \boldsymbol{\Xi}_r\, \boldsymbol{A}_r\, \boldsymbol{N}\, \boldsymbol{\Xi}_r^\top, \tag{3.11}$$

where $\mathbb{R}^{n_r\times n_r} \ni \boldsymbol{H}_r := \boldsymbol{I} - (1/n_r)\mathbf{1}\mathbf{1}^\top$ is the centering matrix, the (i,j)-th entry of $\boldsymbol{K}_r \in \mathbb{R}^{n\times n_r}$ is $\boldsymbol{K}_r(i,j) := k(\boldsymbol{x}_i, \boldsymbol{x}_j^{(r)})$, the i-th entry of $\boldsymbol{\xi}^{(r)} \in \mathbb{R}^n$ is $\boldsymbol{\xi}^{(r)}(i) := (1/n_r)\sum_{j=1}^{n_r} k(\boldsymbol{x}_i, \boldsymbol{x}_j^{(r)})$, and $\mathbb{R}^{n\times c} \ni \boldsymbol{\Xi}_r := [\boldsymbol{\xi}^{(r)} - \boldsymbol{\xi}^{(1)}, \dots, \boldsymbol{\xi}^{(r)} - \boldsymbol{\xi}^{(c)}]$. The solution to Eq. (3.9) is the generalized eigenvalue problem $(\widehat{\boldsymbol{\Delta}}_B, \boldsymbol{\Delta}_W)$ and the p leading columns of $\boldsymbol{Y}$ span the subspace.

3.1.1.5 Manually Weighted Kernel FDA

All the existing weighting methods, such as APAC [91], the POW method [92], CDM [146], and the kNN method [147], can be used for weight in W-KFDA.

The CW-FDA can be used in the feature space to have CW-KFDA. For this, I propose two versions of CW-KFDA: (I) In the first version, I use Eq. (3.4) or $\boldsymbol{A}_r := \textbf{diag}(\alpha_{r\ell}, \forall \ell)$ in the Eq. (3.10). (II) In the second version, I notice that cosine is based on inner product so the normalized kernel matrix between the means of classes can be used instead to use the similarity/dissimilarity in the feature space rather than in the input space. Let $\mathbb{R}^{d\times c} \ni \boldsymbol{M} := [\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_c]$. Let $\widehat{\boldsymbol{K}}_{i,j} := \boldsymbol{K}_{i,j}/\sqrt{\boldsymbol{K}_{i,i}\boldsymbol{K}_{j,j}}$ be the normalized kernel matrix [1] where $\boldsymbol{K}_{i,j}$ denotes the (i,j)-th element of the kernel matrix $\mathbb{R}^{c\times c} \ni \boldsymbol{K}(\boldsymbol{M}, \boldsymbol{M}) = \boldsymbol{\Phi}(\boldsymbol{M})^\top \boldsymbol{\Phi}(\boldsymbol{M})$. The weights are $[0,1] \ni \alpha_{r\ell} := \widehat{\boldsymbol{K}}_{r,\ell}$ or $\boldsymbol{A}_r := \textbf{diag}(\widehat{\boldsymbol{K}}_{r,\ell}, \forall \ell)$. We set $\alpha_{r,r} = 0$.

3.1.1.6 Automatically Weighted Kernel Fisher Discriminant Analysis

Similar to AW-FDA, the optimization in AW-KFDA is:

$$\begin{aligned} \underset{\boldsymbol{Y}, \boldsymbol{A}_r}{\text{maximize}} \quad & \textbf{tr}(\boldsymbol{Y}^\top \widehat{\boldsymbol{\Delta}}_B \boldsymbol{Y}), \\ \text{subject to} \quad & \boldsymbol{Y}^\top \boldsymbol{\Delta}_W \boldsymbol{Y} = \boldsymbol{I}, \\ & ||\boldsymbol{A}_r||_0 \leq k, \quad \forall r \in \{1, \dots, c\}, \end{aligned} \tag{3.12}$$

where $\widehat{\boldsymbol{\Delta}}_B := \sum_{r=1}^{c} n_r \, \boldsymbol{\Xi}_r \, \breve{\boldsymbol{A}}_r \, \boldsymbol{N} \, \boldsymbol{\Xi}_r^\top$. This optimization is solved similar to how Eq. (3.5) was solved where we have $\boldsymbol{Y} \in \mathbb{R}^{n\times d}$ rather than $\boldsymbol{U} \in \mathbb{R}^{d\times d}$. Here, the solution to Eq. (3.6) is the generalized eigenvalue problem $(\widehat{\boldsymbol{\Delta}}_B^{(\tau)}, \boldsymbol{\Delta}_W)$. Let $f(\boldsymbol{Y}, \boldsymbol{A}_k) := -\textbf{tr}(\boldsymbol{Y}^\top \widehat{\boldsymbol{\Delta}}_B \boldsymbol{Y})$. The Eq. (3.7) is solved similarly as in AW-FDA. For more details of gradient, please see [55]. After solving the optimization, the p leading columns of $\boldsymbol{Y}$ span the AW-KFDA subspace. According to the representation theory [2], any solution must lie in the span of all the training vectors, hence, $\boldsymbol{\Phi}(\boldsymbol{U}) = \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{Y}$. The projection of some data $\boldsymbol{X}_t \in \mathbb{R}^{d\times n_t}$ is $\mathbb{R}^{p\times n_t} \ni \widetilde{\boldsymbol{X}}_t = \boldsymbol{\Phi}(\boldsymbol{U})^\top \boldsymbol{\Phi}(\boldsymbol{X}_t) = \boldsymbol{Y}^\top \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{X}_t) = \boldsymbol{Y}^\top \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_t)$.

3.1.2 Roweis Discriminant Analysis

As was explained in Section 2.2.1, many of the linear spectral dimensionality reduction methods are based on generalized eigenvalue problem. This gave me a hint to propose a generalized subspace learning method based on the generalized eigenvalue problem [42].

This generalization results in a family of infinite number of subspace learning algorithms. I name this method RDA after Prof. Sam T. Roweis (rest in peace).

3.1.2.1 Methodology

RDA aims at maximizing $\textbf{tr}(\boldsymbol{U}^\top \boldsymbol{R}_1 \boldsymbol{U})$ interpreted as the scatter of projection, while requiring the manipulated projection directions to be orthonormal. Therefore, the optimization of RDA is formulated as:

$$\underset{\boldsymbol{U}}{\text{maximize}} \quad \textbf{tr}(\boldsymbol{U}^\top \boldsymbol{R}_1 \boldsymbol{U}), \qquad \text{subject to} \quad \boldsymbol{U}^\top \boldsymbol{R}_2 \boldsymbol{U} = \boldsymbol{I}, \tag{3.13}$$

where $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are the first and second *Roweis matrices* which I define as:

$$\mathbb{R}^{d\times d} \ni \boldsymbol{R}_1 := \boldsymbol{X}\boldsymbol{H}\boldsymbol{P}\boldsymbol{H}\boldsymbol{X}^\top, \tag{3.14}$$
$$\mathbb{R}^{d\times d} \ni \boldsymbol{R}_2 := r_2\, \boldsymbol{S}_W + (1-r_2)\, \boldsymbol{I}, \tag{3.15}$$

respectively, where:

$$\mathbb{R}^{n\times n} \ni \boldsymbol{P} := r_1\, \boldsymbol{K}_y + (1-r_1)\, \boldsymbol{I}. \tag{3.16}$$

The $r_1 \in [0,1]$ and $r_2 \in [0,1]$ are the first and second *Roweis factors*. The solution to Eq. (3.13) is the generalized eigenvalue problem $(\boldsymbol{R}_1, \boldsymbol{R}_2)$ [42]. I define the *Roweis criterion* which is maximized in RDA as $\textbf{tr}(\boldsymbol{U}^\top \boldsymbol{R}_1 \boldsymbol{U})/\textbf{tr}(\boldsymbol{U}^\top \boldsymbol{R}_2 \boldsymbol{U})$. This criterion is a generalized Rayleigh-Ritz quotient.

3.1.2.2 The Special Cases of RDA and the Roweis Map

Consider the Eqs. (3.14), (3.15), and (3.16). Consider the extreme cases of r_1 and r_2 and compare the RDA optimization with the optimization of PCA, FDA, and SPCA instroduced in Chapter 2, i.e., compare Eq. (3.13) with Eqs. (2.1), (2.5), and (2.6), and notice Eq. (2.2). By these comparisons, we see that:

$$r_1 = 0,\, r_2 = 0 \implies \text{RDA} \equiv \text{PCA}, \tag{3.17}$$
$$r_1 = 0,\, r_2 = 1 \implies \text{RDA} \equiv \text{FDA}, \tag{3.18}$$
$$r_1 = 1,\, r_2 = 0 \implies \text{RDA} \equiv \text{SPCA}. \tag{3.19}$$

Hence, PCA, FDA, and SPCA are all special cases of RDA. In fact, RDA is a family of infinite number of algorithms for subspace learning with different values of r_1 and r_2. I

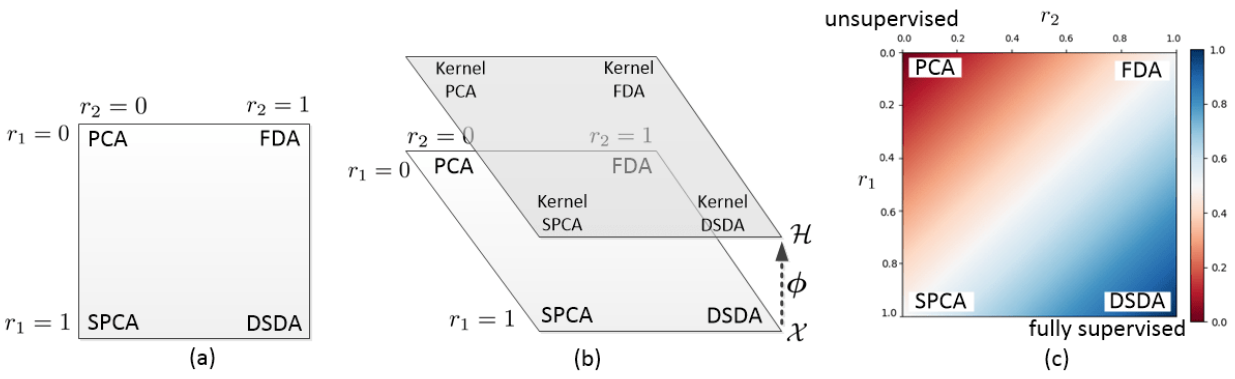

Figure 3.1: Roweis discriminant analysis: (a) The Roweis map including infinite number of subspace learning methods and its special cases, (b) the map in its input and feature spaces where the map in the feature space is pulled from the map in the input space, and (c) the supervision level shown on the Roweis map.

define a map, named *Roweis map*, which includes the infinite number of special cases of RDA where three of its corners are PCA, FDA, and SPCA. The rows and columns of the Roweis map are the values of r_1 and r_2, respectively. Figure 3.1-a shows this map. As this figure shows, every point in this map corresponds to a new subspace learning method with some specific supervision level.

The case $r_1 = r_2 = 1$ is not yet proposed in the literature, where Eq. (3.13) is:

$$\underset{\boldsymbol{U}}{\text{maximize}} \quad \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top \boldsymbol{U}), \qquad \text{subject to} \quad \boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U} = \boldsymbol{I}, \tag{3.20}$$

whose solution is the generalized eigenvalue problem $(\boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top, \boldsymbol{S}_W)$ [42]. This optimization uses the labels twice, once in the kernel over the labels and once in the within scatter; hence, I name it Double Supervised Discriminant Analysis (DSDA).

3.1.2.3 Dimensionality of the RDA Subspace

One can solve the generalized eigenvalue problem $(\boldsymbol{R}_1, \boldsymbol{R}_2)$ as $\boldsymbol{U} = \mathbf{eig}(\boldsymbol{R}_2^{-1}\boldsymbol{R}_1)$ [42], where $\mathbf{eig}(.)$ stacks the eigenvectors column-wise. We have $\mathbf{rank}(\boldsymbol{R}_2^{-1}\boldsymbol{R}_1) \leq \min\big(\mathbf{rank}(\boldsymbol{R}_2^{-1}), \mathbf{rank}(\boldsymbol{R}_1)\big) \leq \min(d, n-1)$ (see [51] for the ranks of the Roweis matrices). Therefore, $\min(d, n-1)$ is an upper bound on the dimensionality of the RDA subspace.

3.1.2.4 Kernel RDA

For the reason explained in Chapter 1, I also propose the kernel version of RDA. According to representation theory [2], any pulled solution (direction) $\boldsymbol{\phi}(\boldsymbol{u}) \in \mathcal{H}$ must lie in the span of all the training vectors pulled to $\mathcal{H}$, i.e., $\boldsymbol{\Phi}(\boldsymbol{X}) = [\boldsymbol{\phi}(\boldsymbol{x}_1), \dots, \boldsymbol{\phi}(\boldsymbol{x}_n)] \in \mathbb{R}^{t \times n}$. Hence $\mathbb{R}^t \ni \boldsymbol{\phi}(\boldsymbol{u}) = \sum_{i=1}^{n} \theta_i\, \boldsymbol{\phi}(\boldsymbol{x}_i) = \boldsymbol{\Phi}(\boldsymbol{X})\, \boldsymbol{\theta}$, where $\mathbb{R}^n \ni \boldsymbol{\theta} = [\theta_1, \dots, \theta_n]^\top$ is the unknown vector of coefficients, and $\boldsymbol{\phi}(\boldsymbol{u}) \in \mathbb{R}^t$ is the pulled RDA direction to the feature space. The pulled directions can be put together in $\mathbb{R}^{t \times p} \ni \boldsymbol{\Phi}(\boldsymbol{U}) = [\boldsymbol{\phi}(\boldsymbol{u}_1), \dots, \boldsymbol{\phi}(\boldsymbol{u}_p)]$ to have $\mathbb{R}^{t \times p} \ni \boldsymbol{\Phi}(\boldsymbol{U}) = \boldsymbol{\Phi}(\boldsymbol{X})\, \boldsymbol{\Theta}$, where $\mathbb{R}^{n \times p} \ni \boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_p]$.

In order to have RDA in the feature space, I first kernelize the objective function of the Eq. (3.13):

$$\mathbf{tr}\big(\boldsymbol{\Phi}(\boldsymbol{U})^\top \boldsymbol{\Phi}(\boldsymbol{R}_1)\, \boldsymbol{\Phi}(\boldsymbol{U})\big) \overset{(3.14)}{=} \mathbf{tr}\big(\boldsymbol{\Phi}(\boldsymbol{U})^\top \boldsymbol{\Phi}(\boldsymbol{X}) \boldsymbol{H}\boldsymbol{P}\boldsymbol{H} \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{U})\big) = \mathbf{tr}\big[\boldsymbol{\Theta}^\top \\ \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{X}) \boldsymbol{H}\boldsymbol{P}\boldsymbol{H} \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{X})\, \boldsymbol{\Theta}\big] = \mathbf{tr}\big(\boldsymbol{\Theta}^\top \boldsymbol{K}_x\, \boldsymbol{H}\boldsymbol{P}\boldsymbol{H} \boldsymbol{K}_x\, \boldsymbol{\Theta}\big) = \mathbf{tr}\big(\boldsymbol{\Theta}^\top \boldsymbol{M}\, \boldsymbol{\Theta}\big),$$

where $\mathbb{R}^{n \times n} \ni \boldsymbol{K}_x := \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{X})$ and:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{M} := \boldsymbol{K}_x\, \boldsymbol{H}\boldsymbol{P}\boldsymbol{H} \boldsymbol{K}_x. \tag{3.21}$$

In order to kernelize the constraint in the Eq. (3.13), it is easier to first consider a one-dimensional subspace and then extend it to multi-dimensional subspace. It can be shown (see [48, Appendix A]) that $\boldsymbol{\phi}(\boldsymbol{u})^\top \boldsymbol{\Phi}(\boldsymbol{S}_W)\, \boldsymbol{\phi}(\boldsymbol{u}) = \boldsymbol{\theta}^\top \big(\sum_{j=1}^{c} \boldsymbol{K}_j \boldsymbol{H}_j \boldsymbol{K}_j^\top\big) \boldsymbol{\theta} = \boldsymbol{\theta}^\top \boldsymbol{N}\, \boldsymbol{\theta}$, where c is the number of classes, n_j is the sample size of the j-th class, $\mathbb{R}^{n_j \times n_j} \ni \boldsymbol{H}_j := \boldsymbol{I} - (1/n_j)\boldsymbol{1}\boldsymbol{1}^\top$, $\mathbb{R}^{n \times n_j} \ni \boldsymbol{K}_j := \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{X}_j)$, and:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{N} := \sum_{j=1}^{c} \boldsymbol{K}_j \boldsymbol{H}_j \boldsymbol{K}_j^\top. \tag{3.22}$$

If the subspace is one-dimensional, the constraint in the Eq. (3.13) is kernelized as $\boldsymbol{\phi}(\boldsymbol{u})^\top \boldsymbol{\Phi}(\boldsymbol{R}_2)\, \boldsymbol{\phi}(\boldsymbol{u}) \overset{(3.15)}{=} r_2\, \boldsymbol{\phi}(\boldsymbol{u})^\top \boldsymbol{\Phi}(\boldsymbol{S}_W)\, \boldsymbol{\phi}(\boldsymbol{u}) + (1 - r_2)\, \boldsymbol{\phi}(\boldsymbol{u})^\top \boldsymbol{\phi}(\boldsymbol{u}) = r_2\, \boldsymbol{\theta}^\top \boldsymbol{N} \boldsymbol{\theta} + (1 - r_2)\, \boldsymbol{\theta}^\top \boldsymbol{K}_x\, \boldsymbol{\theta} = \boldsymbol{\theta}^\top \big(r_2\, \boldsymbol{N} + (1 - r_2)\, \boldsymbol{K}_x\big)\, \boldsymbol{\theta} = \boldsymbol{\theta}^\top \boldsymbol{L}\, \boldsymbol{\theta}$, where:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{L} := r_2\, \boldsymbol{N} + (1 - r_2)\, \boldsymbol{K}_x. \tag{3.23}$$

Similarly, I can extend to multi-dimensional subspace: $\mathbf{tr}\big(\boldsymbol{\phi}(\boldsymbol{U})^\top \boldsymbol{\Phi}(\boldsymbol{R}_2)\, \boldsymbol{\phi}(\boldsymbol{U})\big) = \mathbf{tr}(\boldsymbol{\Theta}^\top \boldsymbol{L}\, \boldsymbol{\Theta})$. Hence, the Roweis criterion is $\mathbf{tr}(\boldsymbol{\Theta}^\top \boldsymbol{M}\, \boldsymbol{\Theta}) / \mathbf{tr}(\boldsymbol{\Theta}^\top \boldsymbol{L}\, \boldsymbol{\Theta})$ in the feature space. The optimization in kernel RDA is:

$$\underset{\boldsymbol{\Theta}}{\text{maximize}} \quad \mathbf{tr}(\boldsymbol{\Theta}^\top \boldsymbol{M}\, \boldsymbol{\Theta}), \quad \text{subject to} \quad \boldsymbol{\Theta}^\top \boldsymbol{L}\, \boldsymbol{\Theta} = \boldsymbol{I}, \tag{3.24}$$

whose solution is the generalized eigenvalue problem $(\boldsymbol{M}, \boldsymbol{L})$. In kernel RDA, the directions are n-dimensional while in RDA, we had d-dimensional directions.

In kernel RDA, the projection and reconstruction of the training and out-of-sample data are $\widetilde{\boldsymbol{X}} = \boldsymbol{\Phi}(\boldsymbol{U})^\top \boldsymbol{\Phi}(\boldsymbol{X}) = \boldsymbol{\Theta}^\top \boldsymbol{K}_x$, $\widehat{\boldsymbol{X}} = \boldsymbol{\Phi}(\boldsymbol{U})\boldsymbol{\Phi}(\boldsymbol{U})^\top \boldsymbol{\Phi}(\boldsymbol{X}) = \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Theta}\boldsymbol{\Theta}^\top \boldsymbol{K}_x$, $\widetilde{\boldsymbol{X}}_t = \boldsymbol{\Theta}^\top \boldsymbol{K}_t$, and $\widehat{\boldsymbol{X}}_t = \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{\Theta}\boldsymbol{\Theta}^\top \boldsymbol{K}_t$, where $\boldsymbol{\Phi}(\boldsymbol{X})$ existing in the reconstructions are not necessarily available so we do not have reconstruction in kernel RDA.

3.1.2.5 Dimensionality of the Kernel RDA Subspace

One can solve Eq. (3.24) as $\boldsymbol{\Theta} = \mathbf{eig}(\boldsymbol{L}^{-1}\boldsymbol{M})$. We have $\mathbf{rank}(\boldsymbol{L}^{-1}\boldsymbol{M}) \leq \min\big(\mathbf{rank}(\boldsymbol{L}^{-1}), \mathbf{rank}(\boldsymbol{M})\big) \leq \min(n, c) - 1$ (see [51] for the ranks of the Roweis matrices in the feature space). Therefore, the dimensionality of the kernel RDA subspace is $p \leq \min(n, c) - 1$, restricted by rank of $\boldsymbol{L}$.

3.1.2.6 Special Cases of Kernel RDA

The Roweis map can have two layers, one for the input space and another for the feature space. The top layer is the bottom layer pulled to the feature space (see Fig. 3.1-b). Therefore, the four corners of Roweis map on the feature space can be considered as kernel PCA [116], kernel FDA [99], kernel SPCA [4], and kernel DSDA. The whole map includes the kernel methods of an infinite number of subspace learning algorithms.

Recall that PCA and SPCA (with $r_2 = 0$) have two types of kernelization which are using kernel trick (or dual of each method) and representation theory. This explains why there exist two types of kernel SPCA [4]. Kernel PCA using the kernel trick is already proposed [116] while the kernel PCA using representation theory is novel and proposed here.

3.1.3 Image Quality Aware Embedding

In this section, I propose subspace and manifold learning methods for image quality aware embedding. Three methods are proposed, named including SSIM kernel, ISCA, and LLISE.

3.1.3.1 Image Structure Subspace/Manifold: Image Fidelity Assessment and Manifold Learning?

It has been shown that MSE is not a promising measure for image quality, fidelity, or similarity [134]. The distortions of an image or similarities of two images can be divided into two main categories, i.e., structural and non-structural distortions [135]. The structural distortions, such as JPEG blocking distortion, Gaussian noise, and blurring, are the ones which are easily noticeable by Human Visual System (HVS), whereas the non-structural distortions, such as luminance enhancement and contrast change, do not have large impact on the visual quality of image.

SSIM [135, 133] has been shown to be an effective measure for image quality assessment. It encounters luminance and contrast change as non-structural distortions and other distortions as structural ones. Due to its performance, it has recently been noticed and used in optimization problems for tasks such as image denoising, image restoration, contrast enhancement, image quantization, compression, etc, noticing that the distance based on SSIM is quasi-convex under certain conditions [12].

So far, the fields of manifold learning and machine learning have largely used MSE and Euclidean distance in order to develop algorithms for subspace learning. PCA is an example based on Euclidean distance or ℓ_2 norm. However, MSE is not as promising as SSIM for image structure measurement [134, 133] making these algorithms not effective enough in terms of capturing the structural features of image. In this thesis, I introduce the new concept of *image structure subspace/manifold* which is a subspace/manifold capturing the intrinsic features of an image in terms of structural similarity and distortions, and can discriminate the various types of image distortions. This subspace can also be useful for parameter estimation for (or selection between) different denoising methods. The image structure subspace/manifold opens a new research field for a combination of image processing and manifold learning investigations. I propose SSIM kernel, ISCA, and LLISE for learning this subspace/manifold. These new methods use SSIM or SSIM distance, introduced in Section 2.2.2, in place of ℓ_2 norm or squared ℓ_2 norm.

3.1.3.2 SSIM Kernel

One can map the n data points $\{\boldsymbol{x}_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, to a higher-dimensional feature space hoping to have the data fall close to a simpler-to-analyze manifold in the feature space. Suppose $\boldsymbol{\phi} : \boldsymbol{x} \rightarrow \mathcal{H}$ is a function which maps the data $\boldsymbol{x}$ to the feature space. In other words, $\boldsymbol{x} \mapsto \boldsymbol{\phi}(\boldsymbol{x})$. Let t denote the dimensionality of the feature space, i.e., $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^t$. We usually have $t \gg d$. If $\boldsymbol{x}$ belongs to the set $\mathcal{X}$, i.e., $\boldsymbol{x} \in \mathcal{X}$, the kernel of two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ and is defined as $k(\boldsymbol{x}_1, \boldsymbol{x}_2) := \boldsymbol{\phi}(\boldsymbol{x}_1)^\top \boldsymbol{\phi}(\boldsymbol{x}_2)$. The kernel matrix for two datasets $\boldsymbol{X}_1 = [\boldsymbol{x}_{1,1}, \dots, \boldsymbol{x}_{1,n_1}] \in \mathbb{R}^{d \times n_1}$ and $\boldsymbol{X}_2 = [\boldsymbol{x}_{2,1}, \dots, \boldsymbol{x}_{2,n_2}] \in \mathbb{R}^{d \times n_2}$ is:

$$\mathbb{R}^{n_1 \times n_2} \ni \boldsymbol{K}(\boldsymbol{X}_1, \boldsymbol{X}_2) := \boldsymbol{\Phi}(\boldsymbol{X}_1)^\top \boldsymbol{\Phi}(\boldsymbol{X}_2), \tag{3.25}$$

where $\boldsymbol{\Phi}(\boldsymbol{X}_1) := [\boldsymbol{\phi}(\boldsymbol{x}_{1,1}), \dots, \boldsymbol{\phi}(\boldsymbol{x}_{1,n_1})] \in \mathbb{R}^{t \times n_1}$ and $\boldsymbol{\Phi}(\boldsymbol{X}_2)$ is similarly defined. The kernel between a matrix $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ and a vector $\boldsymbol{x} \in \mathbb{R}^d$ is $\mathbb{R}^n \ni \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}) := \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\phi}(\boldsymbol{x})$. I denote the kernel over dataset $\boldsymbol{X}$ by $\boldsymbol{K}_x := \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{n \times n}$.

The kernel can be written as [16, 38]:

$$\boldsymbol{K}_x = -(1/2)\, \boldsymbol{H}\boldsymbol{D}\boldsymbol{H}, \tag{3.26}$$

where $\mathbb{R}^{n \times n} \ni \boldsymbol{H} = \boldsymbol{I} - (1/n)\boldsymbol{1}\boldsymbol{1}^\top$ is the centering matrix, $\boldsymbol{I}$ and $\boldsymbol{1}$ are the identity matrix and $[1, \dots, 1]^\top$, respectively, and $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ is the distance matrix whose (i, j)-th element is a distance measure between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. For example, if the distance measure for $\boldsymbol{D}$ is $||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2$, we will have $\boldsymbol{K}_x = \boldsymbol{H}\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{H}$. The elements of $\boldsymbol{D}$ can be measured by a valid distance metric [66]. Two examples of distance metrics are Euclidean distance (resulting in metric MDS [16, 38] or PCA) and geodesic distance (resulting in Isomap [126]).

A valid distance metric based on SSIM is Eq. (2.11). Calculating this metric for every patch, I will have a distance vector $\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{R}^d$ between the two images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. I want to have a scalar distance between two images so we use this theorem: The ℓ_2 norm of a vector of metrics is also a metric (see [12] for proof). Therefore, I define the distance between two images $\boldsymbol{x}_1 \in \mathbb{R}^d$ and $\boldsymbol{x}_2 \in \mathbb{R}^d$ as:

$$\mathbb{R} \ni d(\boldsymbol{x}_1, \boldsymbol{x}_2) := ||\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2)||_2 = \left[\sum_{i=1}^{d} \Big(d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)\Big)^2\right]^{(1/2)}, \tag{3.27}$$

where $d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$ is the distance of Eq. (2.11) for the i-th patch. Note that Eq. (3.27) is equivalent to the Frobenius norm of the distance map between the two images if we have not reshaped the map to a vector. Calculating Eq. (3.27) between every two images of the dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ gives the symmetric distance matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$. Finally, according to Eq. (3.26), I define the *SSIM kernel*:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{S}_x = \boldsymbol{S}(\boldsymbol{X}, \boldsymbol{X}) := -(1/2)\, \boldsymbol{H}\boldsymbol{D}\boldsymbol{H}, \tag{3.28}$$

where $\boldsymbol{D}$ is calculated using Eq. (3.27). Similarly, as in Eq. (3.25), I can have the SSIM kernel between two different datasets: $\boldsymbol{S}(\boldsymbol{X}_1, \boldsymbol{X}_2) = -(1/2)\, \boldsymbol{H}_1\boldsymbol{D}\boldsymbol{H}_2$ where $\boldsymbol{D} \in \mathbb{R}^{n_1 \times n_2}$

is similarly calculated between the images, one from $\boldsymbol{X}_1$ and one from $\boldsymbol{X}_2$. Note that $\boldsymbol{H}_1 \in \mathbb{R}^{n_1 \times n_1}$ and $\boldsymbol{H}_2 \in \mathbb{R}^{n_2 \times n_2}$. The SSIM kernel measures the similarity of data points which are images. Notice that the kernel here should not be confused with the filter kernel in signal processing.

The SSIM kernel can be used in the spectral dimensionality reduction algorithms in order to learn the *image structure subspace* which captures the intrinsic structural features of an image and discriminates the different types of image distortions. For example, the SSIM kernel can be used in MDS [16], kernel PCA [35], and LE [7].

3.1.3.3 Image Structural Component Analysis

Orthonormal Bases for One Image: My goal is to find a subspace spanned by p directions for some desired p. Consider an image block $\breve{\boldsymbol{x}} \in \mathbb{R}^q$ which is centered (its mean is removed). I want to project it onto a p-dimensional subspace and then reconstruct it back, where $p \leq q$. Assume $\mathbb{R}^{q \times p} \ni \boldsymbol{U} := [\boldsymbol{u}_1, \dots, \boldsymbol{u}_p]$ is a matrix whose columns are the projection directions spanning the subspace. The projection and reconstruction of $\breve{\boldsymbol{x}}$ are $\boldsymbol{U}^\top \breve{\boldsymbol{x}}$ and $\boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}}$, respectively. I want to minimize the reconstruction error with orthonormal bases of the subspace; therefore (similar to PCA optimization [35] but with SSIM distance):

$$\begin{aligned} \underset{\boldsymbol{U} \in \mathbb{R}^{q \times p}}{\text{minimize}} \quad & ||\breve{\boldsymbol{x}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}}||_S^2, \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}. \end{aligned} \tag{3.29}$$

This subspace is illustrated in Fig. 3.2. According to Eq. (2.10) and noticing the orthonormality of projection directions, $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$, I have:

$$\mathbb{R} \ni f(\boldsymbol{U}) := ||\breve{\boldsymbol{x}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}}||_S^2 = \frac{\breve{\boldsymbol{x}}^\top (\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top)\, \breve{\boldsymbol{x}}}{\breve{\boldsymbol{x}}^\top (\boldsymbol{I} + \boldsymbol{U}\boldsymbol{U}^\top)\, \breve{\boldsymbol{x}} + c}. \tag{3.30}$$

See [47] for derivation. The gradient of the $f(\boldsymbol{U})$ is [47]:

$$\mathbb{R}^{q \times p} \ni \boldsymbol{G}(\boldsymbol{U}) := \frac{\partial f(\boldsymbol{U})}{\partial \boldsymbol{U}} = \frac{-2\,(1 + f(\boldsymbol{U}))}{||\breve{\boldsymbol{x}}||_2^2 + ||\boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}}||_2^2 + c}\, \breve{\boldsymbol{x}}\breve{\boldsymbol{x}}^\top \boldsymbol{U}. \tag{3.31}$$

I partition a d-dimensional image into $b = \lceil d/q \rceil$ non-overlapping blocks each of which is a reshaped vector $\breve{\boldsymbol{x}} \in \mathbb{R}^q$. The parameter q is an upper bound on the desired dimensionality of the subspace of block ($p \leq q$). This parameter should not be a very large number due to the spatial variety of image statistics, yet also not very small so as to be able to capture the image structure. Also note that p is an upper bound on the rank of $\boldsymbol{U}\boldsymbol{U}^\top \in \mathbb{R}^{q \times q}$.

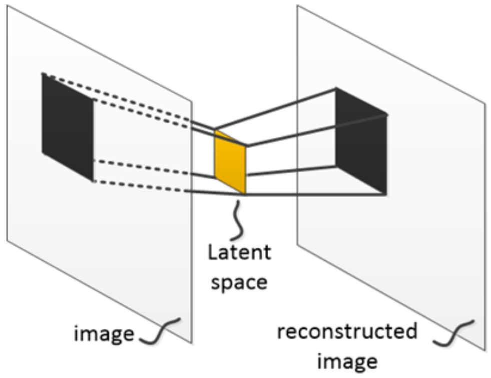

Figure 3.2: A schematic diagram of the core parts of the ISCA algorithm. Every block of image is reconstructed using an ISCA subspace.

there are b instances of p-dimensional subspaces, one for each of the blocks. Considering all the b blocks in an image, the problem in Eq. (3.29) becomes:

$$\begin{aligned} \underset{\boldsymbol{U}_i \in \mathbb{R}^{q \times p}}{\text{minimize}} \quad & \sum_{i=1}^{b} ||\breve{\boldsymbol{x}}_i - \boldsymbol{U}_i \boldsymbol{U}_i^\top \breve{\boldsymbol{x}}_i||_S^2, \\ \text{subject to} \quad & \boldsymbol{U}_i^\top \boldsymbol{U}_i = \boldsymbol{I}, \quad \forall i \in \{1, \dots, b\}, \end{aligned} \tag{3.32}$$

where $\boldsymbol{x}_i \in \mathbb{R}^q$ and $\boldsymbol{U}_i \in \mathbb{R}^{q \times p}$ are the i-th block and the bases of its subspace, respectively.

For solving this optimization, I use the Alternating Direction Method of Multipliers (ADMM) [9, 104]. The updates are as (see [47] for details and derivations):

$$\begin{aligned} \boldsymbol{U}_i^{(k+1)} &:= \boldsymbol{U}_i^{(k)} - \eta\, \boldsymbol{G}(\boldsymbol{U}_i^{(k)}) - \eta\, \rho\, (\boldsymbol{U}_i^{(k)} - \boldsymbol{V}_i^{(k)} + \boldsymbol{J}_i^{(k)}), \\ \boldsymbol{V}_i^{(k+1)} &:= \boldsymbol{Q}_i\ \mathbf{diag}\Big(\mathbf{prox}_{\rho,h}\big(\sigma(\boldsymbol{U}_i^{(k+1)} + \boldsymbol{J}_i^{(k)})\big)\Big)\ \boldsymbol{\Omega}_i^\top, \\ \boldsymbol{J}^{(k+1)} &:= \boldsymbol{J}^{(k)} + \boldsymbol{U}^{(k+1)} - \boldsymbol{V}^{(k+1)}, \end{aligned} \tag{3.33}$$

where columns of $\boldsymbol{Q}_i \in \mathbb{R}^{q \times p}$ and $\boldsymbol{\Omega}_i \in \mathbb{R}^{p \times p}$ are the left and right singular vectors of $(\boldsymbol{U}_i^{(k+1)} + \boldsymbol{J}_i^{(k)})$ and $\eta > 0$ is the learning rate. Iteratively solving Eq. (3.33) until convergence gives us the $\boldsymbol{U}_i$ for for the image blocks indexed by i. The p columns of $\boldsymbol{U}_i$ are the bases for the *ISCA subspace* of the i-th block. Unlike in PCA, the ISCA bases do not have an order of importance but as in PCA, they are orthogonal capturing different features of image structure. The i-th projected block is $\boldsymbol{U}_i^\top \breve{\boldsymbol{x}}_i \in \mathbb{R}^p$ where its dimensions are *image structural components*. Note that $\breve{\boldsymbol{x}}_i$, whether it is a block in a training image or an out-of-sample image, is centered. It is noteworthy that if we consider only one block in the images, the subscript i is dropped from Eq. (3.33).

Orthonormal Bases for a Set of Images: So far, if there is a set of n images, one can find the subspace bases $\boldsymbol{U}_i$ for the i-th block in each of them using Eq. (3.33). Now, I want to find the subspace bases $\boldsymbol{U}_i$ for the i-th block in *all training images of the dataset.* In other words, I want to find the subspace for the best reconstruction of the i-th block in all training images. For this goal, I look at the optimization problem in Eq. (3.32) as an undercomplete auto-encoder neural network [62] with one hidden layer where the input layer, hidden layer, and output layer have q, p, and q neurons, respectively. The $\boldsymbol{U}_i^\top \breve{\boldsymbol{x}}$ and $\boldsymbol{U}_i \boldsymbol{U}_i^\top \breve{\boldsymbol{x}}$ fill the role of applying the first and second weight matrices to the input, respectively. The weights are $\boldsymbol{U}_i \in \mathbb{R}^{q \times p}$. Therefore, there will be b auto-encoders, each with one hidden layer.

For training the auto-encoder, I introduce the blocks in an image as the input to this network and update the weights $\boldsymbol{U}_i, \forall i$ based on Eq. (3.33). Note that I do this update of weights with only 'one' iteration of ADMM. Then, I move to the blocks in the next image and update the weights $\boldsymbol{U}_i, \forall i$ again by an iteration of Eq. (3.33). I do this for all images one by one until an epoch is completed where an epoch is defined as introducing the block in all training images of dataset to the network. After termination of an epoch, I start another epoch to tune the weights $\boldsymbol{U}_i, \forall i$ again. After training the network, I have one p-dimensional subspace for every block in all training images where the columns of the weight matrix $\boldsymbol{U}_i$ span the subspace. Note that because of ADMM, the auto-encoders are trained simultaneously and in parallel. Again, the p columns of $\boldsymbol{U}_i$ are the bases for the *ISCA subspace* of the i-th block.

Kernel Image Structural Component Analysis: For the reason explained in Chapter 1, I also propose the kernel version of ISCA. One can map the block $\breve{\boldsymbol{x}} \in \mathbb{R}^q$ to higher-dimensional feature space hoping to have the data fall close to a simpler-to-analyze manifold in the feature space. The kernel matrix for the i-th block among the n images is $\mathbb{R}^{n \times n} \ni \boldsymbol{K}_i := \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)$ where $\boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i) := [\boldsymbol{\phi}(\breve{\boldsymbol{x}}_{1,i}), \dots, \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{n,i})] \in \mathbb{R}^{t \times n}$. After calculating the kernel matrix, I normalize it [1] as $\boldsymbol{K}_i(a,b) := \boldsymbol{K}_i(a,b)/\sqrt{\boldsymbol{K}_i(a,a)\boldsymbol{K}_i(b,b)}$ where $\boldsymbol{K}_i(a,b)$ denotes the (a,b)-th element of the kernel matrix. Afterwards, the kernel is double-centered as $\boldsymbol{K}_i := \boldsymbol{H}\boldsymbol{K}_i\boldsymbol{H}$ where $\mathbb{R}^{n \times n} \ni \boldsymbol{H} := \boldsymbol{I} - (1/n)\boldsymbol{1}\boldsymbol{1}^\top$. The reason for double-centering is that Eq. (2.10) requires $\boldsymbol{\phi}(\breve{\boldsymbol{x}}_i)$ and thus the $\boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)$ to be centered (see Eq. (3.34)). Therefore, in kernel ISCA, I center the kernel rather than centering $\breve{\boldsymbol{x}}$.

According to representation theory [2], we have $\mathbb{R}^{t \times p} \ni \boldsymbol{\Phi}(\boldsymbol{U}_i) = \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)\, \boldsymbol{\Theta}_i$ where every column of $\boldsymbol{\Theta}_i := [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_p] \in \mathbb{R}^{n \times p}$ is the vector of coefficients. As I did for ISCA, first I consider learning the b subspaces for 'one' image, here. Considering $\boldsymbol{\Phi}(\boldsymbol{U}_i) = \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)\, \boldsymbol{\Theta}_i$

for the i-th block in the image, the objective function of Eq. (3.32) in feature space is $\sum_{i=1}^{b} ||\boldsymbol{\phi}(\breve{\boldsymbol{x}}_i) - \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)\,\boldsymbol{\Theta}_i\,\boldsymbol{\Theta}_i^\top \boldsymbol{k}_i||_S$ where $\mathbb{R}^n \ni \boldsymbol{k}_i := \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_i)$. Note that $\boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)$ includes mapping of the i-th block in all the n images while $\boldsymbol{\phi}(\breve{\boldsymbol{x}}_i)$ is mapping of the i-th block in the image we are considering. The constraint of Eq. (3.32) in the feature space is $\boldsymbol{\Phi}(\boldsymbol{U}_i)^\top \boldsymbol{\Phi}(\boldsymbol{U}_i) = \boldsymbol{\Theta}_i^\top \boldsymbol{K}_i \boldsymbol{\Theta}_i = \boldsymbol{I}$. Therefore, Eq. (3.32) in the feature space is:

$$\begin{aligned} \underset{\boldsymbol{\Theta}_i \in \mathbb{R}^{n\times p}}{\text{minimize}} \quad & \sum_{i=1}^{b} ||\boldsymbol{\phi}(\breve{\boldsymbol{x}}_i) - \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)\,\boldsymbol{\Theta}_i\,\boldsymbol{\Theta}_i^\top \boldsymbol{k}_i||_S^2, \\ \text{subject to} \quad & \boldsymbol{\Theta}_i^\top \boldsymbol{K}_i\,\boldsymbol{\Theta}_i = \boldsymbol{I}, \quad \forall i \in \{1, \dots, b\}. \end{aligned} \tag{3.34}$$

Noticing the constraint $\boldsymbol{\Theta}_i^\top \boldsymbol{K}_i\,\boldsymbol{\Theta}_i = \boldsymbol{I}$ and using Eq. (2.10), I have:

$$\mathbb{R} \ni f(\boldsymbol{\Theta}_i) := ||\boldsymbol{\phi}(\breve{\boldsymbol{x}}_i) - \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)\,\boldsymbol{\Theta}_i\,\boldsymbol{\Theta}_i^\top \boldsymbol{k}_i||_S^2 = \frac{k_i - \boldsymbol{k}_i^\top \boldsymbol{\Theta}_i \boldsymbol{\Theta}_i^\top \boldsymbol{k}_i}{k_i + \boldsymbol{k}_i^\top \boldsymbol{\Theta}_i \boldsymbol{\Theta}_i^\top \boldsymbol{k}_i + c}, \tag{3.35}$$

where $\mathbb{R} \ni k_i := \boldsymbol{\phi}(\breve{\boldsymbol{x}}_i)^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_i)$. See [47] for derivation. The gradient of the $f(\boldsymbol{\Theta}_i)$ is [47]:

$$\mathbb{R}^{n\times p} \ni \boldsymbol{G}(\boldsymbol{\Theta}_i) := \frac{\partial f(\boldsymbol{\Theta}_i)}{\partial\,\boldsymbol{\Theta}_i} = \frac{-2\,(1 + f(\boldsymbol{\Theta}_i))}{k_i + \boldsymbol{k}_i^\top \boldsymbol{\Theta}_i \boldsymbol{\Theta}_i^\top \boldsymbol{k}_i + c}\,\boldsymbol{k}_i \boldsymbol{k}_i^\top \boldsymbol{\Theta}_i. \tag{3.36}$$

I can simplify the constraint $\boldsymbol{\Theta}_i^\top \boldsymbol{K}_i\,\boldsymbol{\Theta}_i = \boldsymbol{I}$. As the kernel $\boldsymbol{K}_i$ is positive semi-definite, I can decompose it as:

$$\mathbb{R}^{n\times n} \ni \boldsymbol{K}_i \overset{\text{SVD}}{=} \boldsymbol{\Psi}\boldsymbol{\Upsilon}\boldsymbol{\Psi}^\top = \boldsymbol{\Psi}\boldsymbol{\Upsilon}^{(1/2)}\boldsymbol{\Upsilon}^{(1/2)}\boldsymbol{\Psi}^\top = \boldsymbol{\Delta}^\top\boldsymbol{\Delta},$$

where $\mathbb{R}^{n\times n} \ni \boldsymbol{\Delta} := \boldsymbol{\Upsilon}^{(1/2)}\boldsymbol{\Psi}^\top$. Therefore, the constraint can be written as: $\boldsymbol{\Theta}_i^\top \boldsymbol{K}_i \boldsymbol{\Theta}_i = \boldsymbol{\Theta}_i^\top \boldsymbol{\Delta}^\top \boldsymbol{\Delta} \boldsymbol{\Theta}_i = (\boldsymbol{\Delta}\boldsymbol{\Theta}_i)^\top(\boldsymbol{\Delta}\boldsymbol{\Theta}_i) = \boldsymbol{I}$.

For solving this optimization, I use the ADMM [9, 104]. The updates are as (see [47] for details and derivations):

$$\begin{aligned} \boldsymbol{\Theta}_i^{(k+1)} &:= \boldsymbol{\Theta}_i^{(k)} - \eta\,\boldsymbol{G}(\boldsymbol{\Theta}_i^{(k)}) - \eta\,\rho\,\boldsymbol{\Delta}^\top(\boldsymbol{\Delta}\boldsymbol{\Theta}_i^{(k)} - \boldsymbol{W}_i^{(k)} + \boldsymbol{J}_i^{(k)}), \\ \boldsymbol{W}_i^{(k+1)} &:= \boldsymbol{Q}_i\,\textbf{diag}\Big(\textbf{prox}_{\rho,h}\big(\sigma(\boldsymbol{\Delta}\boldsymbol{\Theta}_i^{(k+1)} + \boldsymbol{J}_i^{(k)})\big)\Big)\,\boldsymbol{\Omega}_i^\top, \\ \boldsymbol{J}^{(k+1)} &:= \boldsymbol{J}^{(k)} + \boldsymbol{\Delta}\boldsymbol{\Theta}^{(k+1)} - \boldsymbol{W}^{(k+1)}, \end{aligned} \tag{3.37}$$

where columns of $\boldsymbol{Q}_i \in \mathbb{R}^{q\times p}$ and $\boldsymbol{\Omega}_i \in \mathbb{R}^{p\times p}$ are the left and right singular vectors of $(\boldsymbol{\Delta}\boldsymbol{\Theta}_i^{(k+1)} + \boldsymbol{J}_i^{(k)})$. Iteratively solving Eq. (3.37) until convergence gives us the $\boldsymbol{\Theta}_i$ for for

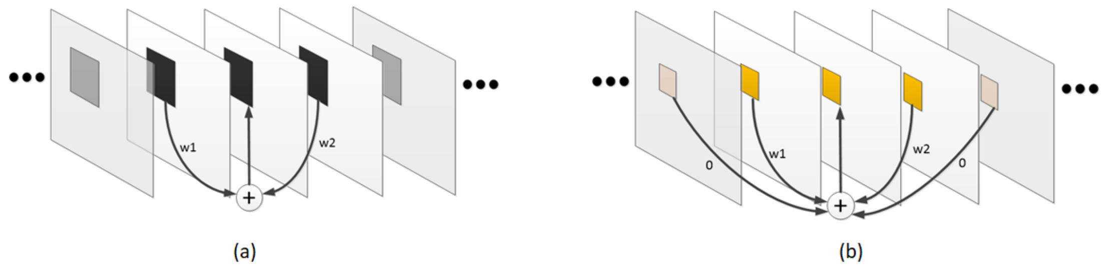

Figure 3.3: A schematic diagram of the core parts of the LLISE algorithm. (a) Every block of image is reconstructed linearly using the corresponding blocks of its nearest neighbors. (b) Using the same obtained weights, the low-dimensional embedding block is obtained.

the image blocks indexed by i. The p columns of $\boldsymbol{\Theta}_i$ are the bases for the *kernel ISCA subspace* of the i-th block. The i-th projected block is $\boldsymbol{\Theta}_i^\top \boldsymbol{k}_i \in \mathbb{R}^p$ and its dimensions are the *kernel image structural components*. Note that $\boldsymbol{k}_i$, whether it is the kernel over a block in a training image or an out-of-sample image, is normalized and centered. Again, with the auto-encoder approach, one can solve these equations in successive epochs in order to find the b subspaces for all the n training images.

3.1.3.4 Locally Linear Image Structural Embedding

In LLISE, I find a p-dimensional image structure manifold for every block. I denote the i-th block in the j-th image by $\breve{\boldsymbol{x}}_{j,i} \in \mathbb{R}^q$. In LLISE, I first center every image block by removing its mean.

k-Nearest Neighbors: For every block $\breve{\boldsymbol{x}}_i$ ($i \in \{1, \dots, b\}$), amongst the n images, a kNN graph is formed using pairwise Euclidean distances between that i-th block in the n images. Therefore, every block in every image has k neighbors. Let ${}_r\breve{\boldsymbol{x}}_{j,i} \in \mathbb{R}^q$ denote the r-th neighbor of $\breve{\boldsymbol{x}}_{j,i}$ and let the matrix $\mathbb{R}^{q\times k} \ni \breve{\boldsymbol{X}}_{j,i} := [{}_1\breve{\boldsymbol{x}}_{j,i}, \dots, {}_k\breve{\boldsymbol{x}}_{j,i}]$ include the neighbors of $\breve{\boldsymbol{x}}_{j,i}$.

Linear Reconstruction by the Neighbors: For every block $\breve{\boldsymbol{x}}_i$, I want the j-th image to be linearly reconstructed by its k neighbors:

$$\begin{aligned}
\underset{\widetilde{\boldsymbol{W}}_i}{\text{minimize}} \quad & \sum_{i=1}^{b} \varepsilon(\widetilde{\boldsymbol{W}}_i) := \sum_{i=1}^{b}\sum_{j=1}^{n} \Big|\Big| \breve{\boldsymbol{x}}_{j,i} - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}\, {}_r\breve{\boldsymbol{x}}_{j,i} \Big|\Big|_S^2, \\
\text{subject to} \quad & \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^2 = 1, \quad \forall i \in \{1, \dots, b\}, \ \ \forall j \in \{1, \dots, n\},
\end{aligned} \tag{3.38}$$

where $\mathbb{R}^{n\times k} \ni \widetilde{\boldsymbol{W}}_i := [\widetilde{\boldsymbol{w}}_{1,i}, \dots, \widetilde{\boldsymbol{w}}_{n,i}]^\top$ includes the weights for the i-th block in the images and $\mathbb{R}^k \ni \widetilde{\boldsymbol{w}}_{j,i} := [{}_1\widetilde{w}_{j,i}, \dots, {}_k\widetilde{w}_{j,i}]^\top$ includes the weights of linear reconstruction of the i-th block in the j-th image using its k neighbors. The second constraint ensures $\widetilde{\boldsymbol{w}}_{j,i}^\top \widetilde{\boldsymbol{w}}_{j,i} = ||\widetilde{\boldsymbol{w}}_{j,i}||_2^2 = 1$. Note that one may formulate the problem with the constraint $\sum_{r=1}^k {}_r\widetilde{w}_{j,i} = 1$ as in LLE; however, with that constraint, the weights start to explode gradually after some optimization iterations. This problem does not happen in LLE because LLE is solved in closed form and not iteratively. This linear reconstruction is illustrated in Fig. 3.3-a.

Take $f(\widetilde{\boldsymbol{w}}_{j,i}) := \big|\big|\breve{\boldsymbol{x}}_{j,i} - \sum_{r=1}^k {}_r\widetilde{w}_{j,i}\, {}_r\breve{\boldsymbol{x}}_{j,i}\big|\big|_S^2 = f(\widetilde{\boldsymbol{w}}_{j,i}) = ||\breve{\boldsymbol{x}}_{j,i} - \breve{\boldsymbol{X}}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i}||_S^2$. According to Eq. (2.10), the $f(\widetilde{\boldsymbol{w}}_{j,i})$ is simplified to (see [46] for derivation):

$$\mathbb{R} \ni f(\widetilde{\boldsymbol{w}}_{j,i}) = \frac{\breve{\boldsymbol{x}}_{j,i}^\top \breve{\boldsymbol{x}}_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} - 2\, \widetilde{\boldsymbol{w}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}^\top \breve{\boldsymbol{x}}_{j,i}}{\breve{\boldsymbol{x}}_{j,i}^\top \breve{\boldsymbol{x}}_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} + c}. \tag{3.39}$$

The gradient of $f(\widetilde{\boldsymbol{w}}_{j,i})$ with respect to $\widetilde{\boldsymbol{w}}_{j,i}$ is (see [46] for derivation):

$$\mathbb{R}^k \ni \nabla f(\widetilde{\boldsymbol{w}}_{j,i}) = \frac{2\, \breve{\boldsymbol{X}}_{j,i}^\top \Big(\big(1 - f(\widetilde{\boldsymbol{w}}_{j,i})\big) \breve{\boldsymbol{X}}_{j,i} \widetilde{\boldsymbol{w}}_{j,i} - \breve{\boldsymbol{x}}_{j,i} \Big)}{\breve{\boldsymbol{x}}_{j,i}^\top \breve{\boldsymbol{x}}_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} + c}. \tag{3.40}$$

For solving this optimization, I use the ADMM [9, 104]. The updates are as (see [46] for details and derivations):

$$\begin{aligned}
\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)} &:= \widetilde{\boldsymbol{w}}_{j,i}^{(\nu)} - \eta\, \nabla f(\widetilde{\boldsymbol{w}}_{j,i}^{(\nu)}) - \eta\, \rho\, (\widetilde{\boldsymbol{w}}_{j,i}^{(\nu)} - \widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu)} + \boldsymbol{j}_{j,i}^{(\nu)}), \\
\widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu+1)} &:= (\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)} + \boldsymbol{j}_{j,i}^{(\nu)}) / ||\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)} + \boldsymbol{j}_{j,i}^{(\nu)}||_2, \\
\boldsymbol{j}_{j,i}^{(\nu+1)} &:= \boldsymbol{j}_{j,i}^{(\nu)} + \widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)} - \widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu+1)},
\end{aligned} \tag{3.41}$$

where $\eta > 0$ is the learning rate. Iteratively solving Eq. (3.41) until convergence gives the $\widetilde{\boldsymbol{w}}_{j,i}$ for for the i-th block in the j-th image.

Linear Embedding: I find the embedding of the i-th block in every image using the obtained weights of reconstruction:

$$\begin{aligned}
&\underset{\boldsymbol{Y}_i}{\text{minimize}} && \sum_{i=1}^b \sum_{j=1}^n \big|\big|\boldsymbol{y}_{j,i} - \sum_{r=1}^n {}_r w_{j,i}\, \boldsymbol{y}_{r,i}\big|\big|_S^2, \\
&\text{subject to} && \frac{1}{n} \sum_{j=1}^n \boldsymbol{y}_{j,i} \boldsymbol{y}_{j,i}^\top = \boldsymbol{I}, \quad \sum_{j=1}^n \boldsymbol{y}_{j,i} = \boldsymbol{0}, \quad \forall i \in \{1, \dots, b\},
\end{aligned} \tag{3.42}$$

where $\boldsymbol{I}$ is the identity matrix, the rows of $\mathbb{R}^{n\times p} \ni \boldsymbol{Y}_i := [\boldsymbol{y}_{1,i}, \dots, \boldsymbol{y}_{n,i}]^\top$ are the embedded i-th block in the images, $\boldsymbol{y}_{r,i} \in \mathbb{R}^p$ is the i-th embedded block in the r-th image, and ${}_r w_{j,i}$ is the weight obtained from the linear reconstruction if $\boldsymbol{x}_{r,i}$ is a neighbor of $\boldsymbol{x}_{j,i}$ and zero otherwise. The second constraint ensures the zero mean of embedded blocks. The first and second constraints together satisfy having unit covariance for the embedded image blocks. This linear embedding is illustrated in Fig. 3.3-b.

Suppose $\mathbb{R}^n \ni \boldsymbol{w}_{j,i} := [{}_1 w_{j,i}, \dots, {}_n w_{j,i}]^\top$ and let $\mathbb{R}^n \ni \boldsymbol{1}_j := [0, \dots, 1, \dots, 0]^\top$ be the vector whose j-th element is one and other elements are zero. The Eq. (3.42) can be restated as:

$$\begin{aligned} &\underset{\boldsymbol{Y}_i}{\text{minimize}} && \sum_{i=1}^{b} \sum_{j=1}^{n} ||\boldsymbol{Y}_i^\top \boldsymbol{1}_j - \boldsymbol{Y}_i^\top \boldsymbol{w}_{j,i}||_S^2, \\ &\text{subject to} && \frac{1}{n} \boldsymbol{Y}_i^\top \boldsymbol{Y}_i = \boldsymbol{I}, \quad \boldsymbol{Y}_i^\top \boldsymbol{1} = \boldsymbol{0}, \quad \forall i \in \{1, \dots, b\}. \end{aligned} \tag{3.43}$$

Let $\theta_j(\boldsymbol{Y}_i) := ||\boldsymbol{Y}_i^\top \boldsymbol{1}_j - \boldsymbol{Y}_i^\top \boldsymbol{w}_{j,i}||_S$. According to Eq. (2.10), it is simplified to (see [46]):

$$\mathbb{R} \ni \theta_j(\boldsymbol{Y}_i) = \frac{\textbf{tr}(\boldsymbol{Y}_i^\top \boldsymbol{M}_{j,i} \boldsymbol{Y}_i)}{\textbf{tr}(\boldsymbol{Y}_i^\top \boldsymbol{\Psi}_{j,i} \boldsymbol{Y}_i) + c}, \tag{3.44}$$

where $\textbf{tr}(.)$ is the trace of matrix, $\mathbb{R}^{n\times n} \ni \boldsymbol{M}_{j,i} := \boldsymbol{1}_j \boldsymbol{1}_j^\top + \boldsymbol{w}_{j,i} \boldsymbol{w}_{j,i}^\top - 2\, \boldsymbol{1}_j \boldsymbol{w}_{j,i}^\top$, and $\mathbb{R}^{n\times n} \ni \boldsymbol{\Psi}_{j,i} := \boldsymbol{1}_j \boldsymbol{1}_j^\top + \boldsymbol{w}_{j,i} \boldsymbol{w}_{j,i}^\top = \boldsymbol{M}_{j,i} + 2\, \boldsymbol{1}_j \boldsymbol{w}_{j,i}^\top$. The gradient of $\theta_j(\boldsymbol{Y}_i)$ with respect to $\boldsymbol{Y}_i$ is (see [46]):

$$\mathbb{R}^{n\times p} \ni \nabla \theta_j(\boldsymbol{Y}_i) = \frac{2}{\textbf{tr}(\boldsymbol{Y}_i^\top \boldsymbol{\Psi}_{j,i} \boldsymbol{Y}_i) + c} \Big(\boldsymbol{M}_{j,i} - \theta_j(\boldsymbol{Y}_i)\, \boldsymbol{\Psi}_{j,i} \Big) \boldsymbol{Y}_i. \tag{3.45}$$

For solving this optimization, I use the ADMM [9, 104]. The updates are as (see [46] for details and derivations):

$$\begin{aligned} \boldsymbol{Y}_i^{(\nu+1)} &:= \boldsymbol{Y}_i^{(\nu)} - \eta \sum_{j=1}^{n} \big(\nabla \theta_j(\boldsymbol{Y}_i) \big) - \eta\, \rho\, (\boldsymbol{Y}_i - \boldsymbol{V}_i^{(\nu)} + \boldsymbol{J}_i^{(\nu)}), \\ \boldsymbol{V}_i^{(\nu+1)} &:= \Pi(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)}), \\ \boldsymbol{J}^{(\nu+1)} &:= \boldsymbol{J}^{(\nu)} + \boldsymbol{Y}^{(\nu+1)} - \boldsymbol{V}^{(\nu+1)}, \end{aligned} \tag{3.46}$$

where $\Pi(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)})$ first removes the row mean of $(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)})$ and then sets the singular values of $(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)})$ to n (see [46] for more details). Iteratively solving Eq. (3.46) until convergence gives $\boldsymbol{Y}_i$ for the image blocks indexed by i. The rows of $\boldsymbol{Y}_i$

are the p-dimensional embedded image blocks in the *LLISE manifold.* Unlike LLE, the first column of $\boldsymbol{Y}_i$ is not ignored in LLISE because it is not based on ℓ_2 norm and thus eigenvalue problem.

Embedding The Out-of-sample Data: There exist two methods in the literature for extension of LLE to out-of-sample embedding. The first method is based on the concept of eigenfunctions [8] and the second method uses linear reconstruction of the out-of-sample data [114]. The first method cannot be used for LLISE because it does not result in closed-form eigenvalue problem as in LLE. I use the second approach.

Suppose there are n_t out-of-sample images and $\breve{\boldsymbol{x}}_{j,i}^{(t)}$ denotes the i-th block in the j-th out-of-sample image. For the i-th block in every out-of-sample image, I first find the k-NN among the i-th block in training images. Let ${}_r\breve{\boldsymbol{x}}_{j,i}^{(t)}$ and $\mathbb{R}^{q\times k} \ni \breve{\boldsymbol{X}}_{j,i}^{(t)} := [{}_1\breve{\boldsymbol{x}}_{j,i}^{(t)}, \dots, {}_k\breve{\boldsymbol{x}}_{j,i}^{(t)}]$ denote the r-th training neighbor of $\breve{\boldsymbol{x}}_{j,i}^{(t)}$ and the matrix including the training neighbors of $\breve{\boldsymbol{x}}_{j,i}^{(t)}$, respectively. I want to reconstruct every out-of-sample image block by its training neighbors:

$$\begin{aligned}
\underset{\widetilde{\boldsymbol{W}}_i^{(t)}}{\text{minimize}} \quad & \sum_{i=1}^{b} \varepsilon(\widetilde{\boldsymbol{W}}_i^{(t)}) := \sum_{i=1}^{b}\sum_{j=1}^{n_t} \Big|\Big| \breve{\boldsymbol{x}}_{j,i}^{(t)} - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^{(t)}\, {}_r\breve{\boldsymbol{x}}_{j,i}^{(t)} \Big|\Big|_S^2, \\
\text{subject to} \quad & \sum_{r=1}^{k} ({}_r\widetilde{w}_{j,i}^{(t)})^2 = 1, \quad \forall i \in \{1,\dots,b\}, \quad \forall j \in \{1,\dots,n_t\},
\end{aligned} \tag{3.47}$$

where $\mathbb{R}^{n_t\times k} \ni \widetilde{\boldsymbol{W}}_i^{(t)} := [\widetilde{\boldsymbol{w}}_{1,i}^{(t)}, \dots, \widetilde{\boldsymbol{w}}_{n_t,i}^{(t)}]^\top$ includes the weights, and $\mathbb{R}^k \ni \widetilde{\boldsymbol{w}}_{j,i}^{(t)} := [{}_1\widetilde{w}_{j,i}^{(t)}, \dots, {}_k\widetilde{w}_{j,i}^{(t)}]^\top$ includes the weights of linear reconstruction of the i-th block in the j-th out-of-sample image using the i-th block in its k training neighbors. Eq. (3.47) is similar to Eq. (3.38) and is solved similarly. The embedding $\boldsymbol{y}_{j,i}^{(t)}$ of the i-th block in the j-th out-of-sample image, i.e., $\boldsymbol{x}_{j,i}^{(t)}$, is obtained by the linear reconstruction of the embedding of the i-th block in its k training neighbors:

$$\mathbb{R}^p \ni \boldsymbol{y}_{j,i}^{(t)} = \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^{(t)}\, {}_r\boldsymbol{y}_{j,i}^{(t)}, \tag{3.48}$$

where ${}_r\boldsymbol{y}_{j,i}^{(t)} \in \mathbb{R}^p$ is the embedding of ${}_r\breve{\boldsymbol{x}}_{j,i}^{(t)}$ which was found by the linear embedding of the training data, $\boldsymbol{Y}_i$.

Kernel Locally Linear Image Structural Embedding: For the reason explained in Chapter 1, I also propose the kernel version of LLISE. I normalize the kernel as $\boldsymbol{K}(a, b) :=$

$\boldsymbol{K}(a,b)/\sqrt{\boldsymbol{K}(a,a)\boldsymbol{K}(b,b)}$ where $\boldsymbol{K}(a,b)$ denotes the (a,b)-th element of the kernel matrix [1]. Then, the kernel is double-centered as $\boldsymbol{K} := \boldsymbol{HKH}$. The reason for double-centering is that Eq. (2.10) requires $\boldsymbol{\phi}(\breve{\boldsymbol{x}}_i)$ and thus the $\boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)$ to be centered. Therefore, in kernel LLISE, I center the kernel rather than centering $\breve{\boldsymbol{x}}_i$. Kernel LLISE maps data to the feature space and performs the steps of kNN and linear reconstruction in the feature space. For kNN, the Euclidean distance in the feature space is used as [115]:

$$||\boldsymbol{\phi}(\breve{\boldsymbol{x}}_{a,i}) - \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{b,i})||_2 = \sqrt{k(\breve{\boldsymbol{x}}_{a,i}, \breve{\boldsymbol{x}}_{a,i}) - 2k(\breve{\boldsymbol{x}}_{a,i}, \breve{\boldsymbol{x}}_{b,i}) + k(\breve{\boldsymbol{x}}_{b,i}, \breve{\boldsymbol{x}}_{b,i})}. \tag{3.49}$$

For every block i amongst the images, we construct the k-NN graph.

For finding the reconstruction weights $\mathbb{R}^k \ni \widetilde{\boldsymbol{w}}_{j,i} = [{}_1\widetilde{w}_{j,i}, \dots, {}_k\widetilde{w}_{j,i}]^\top$, the Eq. (3.38) is used in the feature space:

$$\begin{aligned} \underset{\widetilde{\boldsymbol{W}}_i}{\text{minimize}} \quad & \varepsilon(\widetilde{\boldsymbol{W}}_i) := \sum_{i=1}^{b}\sum_{j=1}^{n} \Big|\Big|\boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}) - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}\, \boldsymbol{\phi}({}_r\breve{\boldsymbol{x}}_{j,i})\Big|\Big|_S^2, \\ \text{subject to} \quad & \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^2 = 1, \quad \forall i \in \{1,\dots,b\}, \quad \forall j \in \{1,\dots,n\}. \end{aligned} \tag{3.50}$$

Let $f^\phi(\widetilde{\boldsymbol{w}}_{j,i}) := \big|\big|\boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}) - \sum_{r=1}^{k} {}_r\widetilde{w}_{ij}\, \boldsymbol{\phi}({}_r\breve{\boldsymbol{x}}_{j,i})\big|\big|_S^2$. According to Eq. (2.10), I have [46]:

$$\mathbb{R} \ni f^\phi(\widetilde{\boldsymbol{w}}_{j,i}) = \frac{k_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \boldsymbol{K}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} - 2\, \widetilde{\boldsymbol{w}}_{j,i}^\top \boldsymbol{k}_{j,i}}{k_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \boldsymbol{K}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} + c}, \tag{3.51}$$

where $\mathbb{R} \ni k_{j,i} := \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i})^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i})$, $\mathbb{R}^k \ni \boldsymbol{k}_{j,i} := \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i})^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i})$, and $\mathbb{R}^{k \times k} \ni \boldsymbol{K}_{j,i} := \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i})^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i})$. The gradient of $f^\phi(\widetilde{\boldsymbol{w}}_{j,i})$ with respect to $\widetilde{\boldsymbol{w}}_{j,i}$ is [46]:

$$\mathbb{R}^k \ni \nabla f^\phi(\widetilde{\boldsymbol{w}}_{j,i}) = \frac{2\Big(\big(1 - f^\phi(\widetilde{\boldsymbol{w}}_{j,i})\big)\boldsymbol{K}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} - \boldsymbol{k}_{j,i}\Big)}{k_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \boldsymbol{K}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} + c}. \tag{3.52}$$

I can use Eq. (3.41) for solving Eq. (3.50) with some slight changes. The linear embedding in kernel LLISE is the same as the linear embedding in LLISE. The rows of obtained $\boldsymbol{Y}_i$ are the i-th embedded block of the images in *kernel LLISE manifold*.

For embedding every out-of-sample image, I solve this optimization problem:

$$\begin{aligned} \underset{\widetilde{\boldsymbol{W}}_i^{(t)}}{\text{minimize}} \quad & \sum_{i=1}^{b} \varepsilon(\widetilde{\boldsymbol{W}}_i^{(t)}) := \sum_{i=1}^{b}\sum_{j=1}^{n_t} \Big|\Big|\, \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}^{(t)}) - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^{(t)}\, \boldsymbol{\phi}({}_r\breve{\boldsymbol{x}}_{j,i}^{(t)})\Big|\Big|_S^2, \\ \text{subject to} \quad & \sum_{r=1}^{k} ({}_r\widetilde{w}_{j,i}^{(t)})^2 = 1, \quad \forall i \in \{1,\dots,b\}, \quad \forall j \in \{1,\dots,n_t\}, \end{aligned} \tag{3.53}$$

which is similar to Eq. (3.50) and is solved similarly.

3.2 Probabilistic Dimensionality Reduction

In the following, I propose new algorithms for probabilistic dimensionality reduction. My proposed method is QQE which deals with CDF of data.

3.2.1 Quantile-Quantile Embedding

Using the quantile-quantile plot, introduced in Section 2.2.3, we propose QQE for giving the choice of embedding distribution to user. Here, I provide my definition for distribution transformation:

Definition 1 (distribution transformation) *For a sample $\{\boldsymbol{x}_i^0\}_{i=1}^n$ of size n in $\mathbb{R}^d$ space, the mapping $\boldsymbol{x}_i^0 \mapsto \boldsymbol{x}_i, \forall i \in \{1, \dots, n\}$ is a distribution transformation where the distribution of $\{\boldsymbol{x}_i\}_{i=1}^n$ is the known desired distribution and the local distances of nearby points in $\{\boldsymbol{x}_i^0\}_{i=1}^n$ are preserved in $\{\boldsymbol{x}_i\}_{i=1}^n$ as much as possible.*

Distribution transformation can be performed in two approaches. In the first approach, (i) the distribution of data is transformed to the "exact" reference distribution, (ii) while in the second approach, only the "shape" of the reference distribution is considered to transform to. Transformation to *exact* reference distribution means that all the moments of data distribution, including mean and variance, will match those of the reference distribution. However, transformation to the *shape* of reference distribution refers to an scale and mean invariant transformation. In this case, the mean of data does not change much and just the shape of data becomes similar to the reference shape. The scale or variance of data does not change significantly but changes in a way to become roughly a multiplication of the scale or variance of reference distribution. In the following, I detail the two approaches, respectively. Then, I introduce manifold embedding using QQE. Finally, I explain the unsupervised and supervised approaches for QQE.

3.2.1.1 Distribution Transformation to Exact Reference Distribution

When the d qq-plots are obtained by the fuzzy qq-plot, I can use them to embed the data for distribution transformation. Consider the transformation of an initial sample

$\{\boldsymbol{x}_i^0\}_{i=1}^n$ to $\{\boldsymbol{x}_i\}_{i=1}^n$. I want the distribution of sample $\{\boldsymbol{x}_i\}_{i=1}^n$ to become the same as the distribution of the reference sample $\{\boldsymbol{y}_{\sigma(i)}\}_{i=1}^n$ or the reference distribution. Therefore, the qq-plot of every dimension should be a line with slope one and intercept zero [103]. Let $Q_l(\boldsymbol{u}_i) \in \mathbb{R}$ denote the l-th dimension of $\mathbb{R}^d \ni \boldsymbol{Q}(\boldsymbol{u}_i) = [Q_1(\boldsymbol{u}_i), \dots, Q_d(\boldsymbol{u}_i)]^\top$ which is used for the i-th data point in the l-th qq-plot. Consider $Q_l(\boldsymbol{u}_i)$ for the matched data and the reference sample, denoted by $Q_{X,l}(\boldsymbol{u}_i)$ and $Q_{Y,l}(\boldsymbol{u}_i)$, respectively. In order to have the line in the qq-plot, I should minimize $\sum_{i=1}^n \sum_{l=1}^d \big(Q_{X,l}(\boldsymbol{u}_i) - Q_{Y,l}(\boldsymbol{u}_i)\big)^2$. According to Eqs. (2.20) and (2.21), this cost function is equivalent to $\sum_{i=1}^n \sum_{l=1}^d (x_{i,l} - y_{\sigma(i),l})^2$ where $x_{i,l}$ and $y_{\sigma(i),l}$ denote the l-th dimension of $\boldsymbol{x}_i = [x_{i,1}, \dots, x_{i,d}]^\top$ and $\boldsymbol{y}_{\sigma(i)} = [y_{\sigma(i),1}, \dots, y_{\sigma(i),d}]^\top$, respectively. In vector form, the cost function is restated as:

$$\mathcal{L}_1 := \frac{1}{2} \sum_{i=1}^n \|\boldsymbol{x}_i - \boldsymbol{y}_{\sigma(i)}\|_2^2. \tag{3.54}$$

On the other hand, according to our definition of distribution transformation, I should also preserve the local distances of the nearby data points as far as possible to embed the data locally [114]. For preserving the local distances, I minimize the differences of local distances between the data and transformed data. Using the k-nearest neighbors (k-NN) graph for the set $\{\boldsymbol{x}_i\}_{i=1}^n$. Let $\mathcal{N}_i$ denote the set containing the indices of the k neighbors of $\boldsymbol{x}_i$. The cost to be minimized is:

$$\mathcal{L}_2 := \frac{1}{a} \sum_{i=1}^n \sum_{j \in \mathcal{N}_i} w_{ij} \big(d_x(i,j) - d_x^0(i,j)\big)^2, \tag{3.55}$$

where $d_x(i,j) := \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$, $d_x^0(i,j) := \|\boldsymbol{x}_i^0 - \boldsymbol{x}_j^0\|_2$, and $a := \sum_{i=1}^n \sum_{j \in \mathcal{N}_i} d_x^0(i,j)$ is the normalization factor. The weight $w_{ij} := 1/d_x^0(i,j)$ gives more value to closer points as expected. Note that if $k = n-1$, the Eq. (3.55) is the cost function used in Sammon mapping [113]. I use this cost as a regularization term in our optimization. Therefore, the optimization is:

$$\underset{\boldsymbol{X}}{\text{minimize}} \quad \mathcal{L} := \frac{1}{2} \sum_{i=1}^n \Big(\|\boldsymbol{x}_i - \boldsymbol{y}_{\sigma(i)}\|_2^2 + \frac{\lambda}{a} \sum_{j \in \mathcal{N}_i} w_{ij} \big(d_x(i,j) - d_x^0(i,j)\big)^2 \Big), \tag{3.56}$$

where $\lambda > 0$ is the regularization parameter. The gradient of the cost function with respect to $x_{i,l}$ is (see [52] for derivation):

$$\frac{\partial \mathcal{L}}{\partial x_{i,l}} = (x_{i,l} - \boldsymbol{y}_{\sigma(i)}) + \frac{\lambda}{a} \sum_{j \in \mathcal{N}_i} \frac{d_x(i,j) - d_x^0(i,j)}{d_x(i,j)\, d_x^0(i,j)} (x_{i,l} - x_{j,l}). \tag{3.57}$$

The second derivative of the cost function with respect to $x_{i,l}$ is (see [52] for derivation):

$$\frac{\partial^2 \mathcal{L}}{\partial x_{i,l}^2} = 1 + \frac{\lambda}{a} \sum_{j \in \mathcal{N}_i} \Big(\frac{d_x(i,j) - d_x^0(i,j)}{d_x(i,j)\, d_x^0(i,j)} - \frac{(x_{i,l} - x_{j,l})^2}{\big(d_x(i,j)\big)^3} \Big). \tag{3.58}$$

We use the quasi-Newton's method for solving this optimization problem inspired by [113]. If we consider the vectors component-wise, the diagonal quasi-Newton's method updates the solution as:

$$x_{i,l}^{(\nu+1)} := x_{i,l}^{(\nu)} - \eta \left| \frac{\partial^2 \mathcal{L}}{\partial x_{i,l}^2} \right|^{-1} \frac{\partial \mathcal{L}}{\partial x_{i,l}}, \tag{3.59}$$

$\forall i \in \{1, \dots, n\}, \forall l \in \{1, \dots, d\}$, where ν is the index of iteration, $\eta > 0$ is the learning rate, and $|.|$ denotes the absolute value guaranteeing that we move toward the minimum and not maximum in the Newton's method.

3.2.1.2 Distribution Transformation to the Shape of Reference Distribution

One can ignore the location and scale of the reference distribution and merely change the distribution of the observed sample to look like the "shape" of the reference distribution regardless of its location and scale. Recall that if the qq-plot is a line, the shapes of the distributions are the same where the intercept and slope of the line correspond to the location and scale [103]. Therefore, in my optimization, rather than trying to make the qq-plot a line with slope one and intercept zero, I try to make it the closest line possible. This line can be found by fitting a line as a least squares problem, i.e., a linear regression problem. For the qq-plot of every dimension, I fit a line to the qq-plot. If I define $\mathbb{R}^n \ni \breve{\boldsymbol{Q}}_{Y,l} := [Q_{Y,l}(\boldsymbol{u}_1), \dots, Q_{Y,l}(\boldsymbol{u}_n)]^\top$, let $\mathbb{R}^{n \times 2} \ni \boldsymbol{\Gamma}_l := [\boldsymbol{1}_{n\times 1}, \breve{\boldsymbol{Q}}_{Y,l}]$. Fitting a line to the qq-plot of the l-th dimension is the following least squares problem:

$$\underset{\boldsymbol{\beta}_l}{\text{minimize}} \quad \frac{1}{2} \big\| \boldsymbol{Q}_X(\boldsymbol{u}_i) - \boldsymbol{\Gamma}_l\, \boldsymbol{\beta}_l \big\|_2^2 \overset{(2.20)}{=} \frac{1}{2} \big\| \boldsymbol{x}_l - \boldsymbol{\Gamma}_l\, \boldsymbol{\beta}_l \big\|_2^2, \tag{3.60}$$

whose solution is [26]:

$$\mathbb{R}^2 \ni \boldsymbol{\beta}_l = (\boldsymbol{\Gamma}_l^\top \boldsymbol{\Gamma}_l)^{-1} \boldsymbol{\Gamma}_l^\top \boldsymbol{x}_l, \tag{3.61}$$

where $\mathbb{R}^n \ni \boldsymbol{x}_l := [x_{1,l}, \dots, x_{n,l}]^\top$. The n points on the line fitted to the qq-plot of the l-th dimension are:

$$\mathbb{R}^n \ni \boldsymbol{\mu}_l := \boldsymbol{\Gamma}_l\, \boldsymbol{\beta}_l = [\mu_{\sigma(1),l}, \dots, \mu_{\sigma(n),l}]^\top, \tag{3.62}$$

which are used instead of $\boldsymbol{Q}_Y(\boldsymbol{u}_i), \forall i$ in our optimization. Defining $\mathbb{R}^d \ni \breve{\boldsymbol{\mu}}(\boldsymbol{y}_{\sigma(i)}) := [\mu_{\sigma(i),1}, \dots, \mu_{\sigma(i),d}]^\top$, the optimization problem is:

$$\underset{\boldsymbol{Y}}{\text{minimize}} \quad \mathcal{L} := \frac{1}{2} \sum_{i=1}^{n} \Big(\|\boldsymbol{x}_i - \breve{\boldsymbol{\mu}}(\boldsymbol{y}_{\sigma(i)})\|_2^2 + \frac{\lambda}{a} \sum_{j \in \mathcal{N}_i} w_{ij} \big(d_x(i,j) - d_x^0(i,j)\big)^2 \Big). \tag{3.63}$$

Similar to Eq. (3.57), the gradient is:

$$\frac{\partial \mathcal{L}}{\partial x_{i,l}} = (x_{i,l} - \mu_{\sigma(i),l}) + \frac{\lambda}{a} \sum_{j \in \mathcal{N}_i} \frac{d_x(i,j) - d_x^{(0)}(i,j)}{d_x(i,j)\, d_x^{(0)}(i,j)} (x_{i,l} - x_{j,l}), \tag{3.64}$$

and the second derivative is the same as Eq. (3.58). We again solve using diagonal quasi-Newton's method.

3.2.1.3 Manifold Embedding

QQE can be used for manifold embedding in a lower dimensional embedding space where the embedding distribution can be determined by the user. As an initialization, the high dimensional data are embedded in a lower dimensional embedding space using a dimensionality reduction method. Thereafter, the low dimensional embedding data are transformed to a desired distribution using QQE.

Any dimensionality reduction method can be utilized for the initialization of data in the low dimensional subspace. Some examples are PCA [35] (or metric MDS [16]), FDA [43], Isomap [126], LLE [110], t-SNE [131], and deep features like triplet Siamese features [118] and ResNet features [67]. After the initialization, a reference sample is drawn from the reference distribution or is taken from the user. The dimensionality of the reference sample is equal to the dimensionality of the low dimensional embedding space. I transform the distribution of the low dimensional data to the reference distribution using QQE. Again, the distribution transformation can be either to the exact or shape of the desired distribution.

3.2.1.4 Unsupervised and Supervised Embedding

QQE, for both tasks of distribution transformation and manifold embedding, can be used in either supervised or unsupervised manners. In an unsupervised manner, the distribution of all the data points is transformed to the desired distribution; however, in the supervised manner, the data points of each class are transformed to have the desired distribution. Hence, in the supervised case, the user can even choose different distributions for the different classes. Note that in both unsupervised and supervised cases, the distribution transformation can be either to the exact or shape of reference distribution. For the supervised case in the distribution transformation task, the distribution of every class is transformed; in the manifold learning task, the distribution of low dimensional data of every class is transformed no matter whether the dimensionality reduction method for initialization is unsupervised or supervised.

3.3 Neural Network-Based Dimensionality Reduction

In neural network-based dimensionality reduction, the network can be either shallow or deep. For the shallow network, I propose the backprojection algorithm for training network using a projection-based perspective. For deep networks, I propose Fisher losses for Siamese nets and BUT/BUNCA methods for triplet mining.

3.3.1 Shallow Network: Backprojection

In this section, I propose backprojection for neural network-based dimensionality reduction by shallow networks.

3.3.1.1 Projection and Backprojection in Network

In a neural network, every layer without its activation function acts as a linear projection. Without the nonlinear activation functions, a network/autoencoder is reduced to a linear projection/principal component analysis [35]. If $\boldsymbol{U}$ denotes the projection matrix (i.e., the weight matrix of a layer), $\boldsymbol{U}^\top \boldsymbol{x}$ projects $\boldsymbol{x}$ onto the column space of $\boldsymbol{U}$. The reverse operation of projection is called reconstruction or backprojection and is formulated as $\boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}$ which shows the projected data in the input space dimensionality (note that it is $\boldsymbol{U}\boldsymbol{f}^{-1}(\boldsymbol{f}(\boldsymbol{U}^\top \boldsymbol{x}))$ if we have a nonlinear function $\boldsymbol{f}(.)$ after the linear projection). At the initialization, a layer acts as a random projection which is a promising feature extractor. Fine tuning the weights using labels makes the features more useful for discrimination of classes.

3.3.1.2 Definitions

Consider a training set $\mathcal{X} := \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$ and their one-hot encoded labels $\mathcal{Y} := \{\boldsymbol{y}_i \in \mathbb{R}^p\}_{i=1}^n$ where n, d, and p are the sample size, dimensionality of data, and dimensionality of labels, respectively. I denote the dimensionality or the number of neurons in layer m by d_m. By convention, we have $d_0 := d$ and $d_{n_\ell} = p$ where n_ℓ is the number of layers and p is the dimensionality of the output layer. Let the data after the activation function of the m-th layer be denoted by $\boldsymbol{x}^{(m)} \in \mathbb{R}^{d_m}$. Let the projected data in the m-th layer be $\mathbb{R}^{d_m} \ni \boldsymbol{z}^{(m)} := \boldsymbol{U}_m^\top \boldsymbol{x}^{(m-1)}$ where $\boldsymbol{U}_m \in \mathbb{R}^{d_{m-1} \times d_m}$ is the weight matrix of the m-th layer. Note that $\boldsymbol{x}^{(m)} = \boldsymbol{f}_m(\boldsymbol{z}^{(m)})$ where $\boldsymbol{f}_m(.)$ is the activation function in the m-th layer. By convention, $\boldsymbol{x}^{(0)} := \boldsymbol{x}$. The data are projected and passed through the activation functions layer by layer; hence, $\boldsymbol{x}^{(m)}$ is calculated as:

$$\mathbb{R}^{d_m} \ni \boldsymbol{x}^{(m)} := \boldsymbol{f}_m(\boldsymbol{U}_m^\top \boldsymbol{f}_{m-1}(\boldsymbol{U}_{m-1}^\top \cdots \boldsymbol{f}_1(\boldsymbol{U}_1^\top \boldsymbol{x}))) = \boldsymbol{f}_m(\boldsymbol{U}_m^\top \boldsymbol{x}^{(m-1)}). \tag{3.65}$$

In a mini-batch gradient descent set-up, let $\{\boldsymbol{x}_i\}_{i=1}^b$ be a batch of size b. For a batch, I denote the outputs of activation functions at the m-th layer by $\mathbb{R}^{d_m \times b} \ni \boldsymbol{X}^{(m)} := [\boldsymbol{x}_1^{(m)}, \dots, \boldsymbol{x}_b^{(m)}]$.

Now, consider the one-hot encoded labels of batch, denoted by $\boldsymbol{y} \in \mathbb{R}^p$. I take the inverse activation function of the labels and then reconstruct or *backproject* them to the previous layer to obtain $\boldsymbol{y}^{(n_\ell - 1)}$. I do similarly until the layer m. Let $\boldsymbol{y}^{(m)} \in \mathbb{R}^{d_m}$ denotes the backprojected data at the m-th layer, calculated as:

$$\boldsymbol{y}^{(m)} := \boldsymbol{U}_{m+1} \boldsymbol{f}_{m+1}^{-1}(\boldsymbol{U}_{m+2} \boldsymbol{f}_{m+2}^{-1}(\cdots \boldsymbol{U}_{n_\ell} \boldsymbol{f}_{n_\ell}^{-1}(\boldsymbol{y}))) = \boldsymbol{U}_{m+1} \boldsymbol{f}_{m+1}^{-1}(\boldsymbol{y}^{(m+1)}). \tag{3.66}$$

By convention, $\boldsymbol{y}^{(n_\ell)} := \boldsymbol{y}$. The backprojected batch at the m-th layer is $\mathbb{R}^{d_m \times b} \ni \boldsymbol{Y}^{(m)} := [\boldsymbol{y}_1^{(m)}, \dots, \boldsymbol{y}_b^{(m)}]$. I use $\boldsymbol{X} \in \mathbb{R}^{d \times b}$ and $\boldsymbol{Y} \in \mathbb{R}^{p \times b}$ to denote the column-wise batch matrix and its one-hot encoded labels.

3.3.1.3 Optimization

In the backprojection algorithm, I optimize the layers' weights one by one. Consider the m-th layer whose loss I denote by $\mathcal{L}_m$:

$$\underset{\boldsymbol{U}_m}{\text{minimize}} \quad \mathcal{L}_m := \sum_{i=1}^b \ell(\boldsymbol{x}_i^{(m)} - \boldsymbol{y}_i^{(m)}) = \sum_{i=1}^b \ell\big(\boldsymbol{f}_m(\boldsymbol{U}_m^\top \boldsymbol{x}_i^{(m-1)}) - \boldsymbol{y}_i^{(m)}\big), \tag{3.67}$$

where $\ell(.)$ is a loss function such as the squared ℓ_2 norm (or MSE), cross-entropy, etc. The loss $\mathcal{L}_m$ tries to make the projected data $\boldsymbol{x}_i^{(m)}$ as similar as possible to the backprojected

```
1  Procedure:  UpdateLayerWeights($\mathcal{U}$, $\boldsymbol{X}$, $\boldsymbol{Y}$, $m$)
2  Input:  weights: $\mathcal{U} := \{\boldsymbol{U}_r\}_{r=1}^{n_\ell}$, batch data: $\boldsymbol{X} \in \mathbb{R}^{d\times b}$, batch labels: $\boldsymbol{Y} \in \mathbb{R}^{p\times b}$,
     layer: $m \in [1, n_\ell]$
3  $\boldsymbol{X}^{(0)} := \boldsymbol{X}$
4  for layer $r$ from $1$ to $(m-1)$ do
5      $\boldsymbol{Z}^{(r)} := \boldsymbol{U}_r^\top \boldsymbol{X}^{(r-1)}$
6      $\boldsymbol{X}^{(r)} := \boldsymbol{f}_r(\boldsymbol{Z}^{(r)})$
7  $\boldsymbol{Y}^{(n_\ell)} := \boldsymbol{Y}$
8  for layer $r$ from $(n_\ell - 1)$ to $m$ do
9      $\boldsymbol{Y}^{(r+1)} := \Pi(\boldsymbol{Y}^{(r+1)})$
10     $\boldsymbol{Y}^{(r)} := \boldsymbol{U}_{r+1}\, \boldsymbol{f}_{r+1}^{-1}(\boldsymbol{Y}^{(r+1)})$
11 $\boldsymbol{U}_m := \boldsymbol{U}_m - \eta\,(\partial \mathcal{L}_m / \partial \boldsymbol{U}_m)$
12 Return $\boldsymbol{U}_m$
```

Algorithm 1: Updating the weights of a layer in backprojection

data $\boldsymbol{y}_i^{(m)}$ by tuning the weights $\boldsymbol{U}_m$. This is because the output of the network is supposed to be equal to the labels, i.e., $\boldsymbol{x}^{(n_\ell)} \approx \boldsymbol{y}$. In order to tune the weights for Eq. (3.67), I use a step of gradient descent. Using chain rule, the gradient is:

$$\mathbb{R}^{d_{m-1}\times d_m} \ni \frac{\partial \mathcal{L}_m}{\partial \boldsymbol{U}_m} = \sum_{i=1}^{b} \textbf{vec}^{-1}_{d_{m-1}\times d_m}\Big[\big(\frac{\partial \boldsymbol{z}_i^{(m)}}{\partial \boldsymbol{U}_m}\big)^\top \big(\frac{\partial \boldsymbol{f}_m(\boldsymbol{z}_i^{(m)})}{\partial \boldsymbol{z}_i^{(m)}}\big)^\top \frac{\partial \ell(\boldsymbol{f}_m(\boldsymbol{z}_i^{(m)}))}{\partial \boldsymbol{f}_m(\boldsymbol{z}_i^{(m)})}\Big], \tag{3.68}$$

whose terms are detailed in [50].

The procedure for updating weights in the m-the layer is shown in Algorithm 1. Until the layer m, data is projected and passed through activation functions layer by layer. Also, the label is backprojected and passed through inverse activation functions until the layer m. A step of gradient descent is used to update the layer's weights where $\eta > 0$ is the learning rate. Note that the backprojected label at a layer may not be in the feasible domain of its inverse activation function. Hence, at every layer, I should project the backprojected label onto the feasible domain [106]. I denote projection onto the feasible set by $\Pi(.)$.

3.3.1.4 Different Procedures

So far, I explained how to update the weights of a layer. Here, I detail updating the entire network layers. In terms of the order of updating layers, I can have three different procedures for a backprojection algorithm. I can update layers from the first to the last layer (i.e., forward procedure), from the last layer to the first layer (i.e., backward procedure), and going back and forth (i.e., forward-backward procedure). One alternative approach is to make updating of layers dependent only on the weights tuned by previous mini-batch. In that approach, the training of layers can be parallelized within mini-batch.

3.3.1.5 Kernel Backprojection Algorithm

For the reason explained in Chapter 1, I also propose the kernel version of backprojection. Suppose $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathcal{H}$ is the pulling function to the feature space. Let t denote the dimensionality of the feature space, i.e., $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^t$. Let the matrix-form of $\mathcal{X}$ and $\mathcal{Y}$ be denoted by $\mathbb{R}^{d\times n} \ni \breve{\boldsymbol{X}} := [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]$ and $\mathbb{R}^{p\times n} \ni \breve{\boldsymbol{Y}} := [\boldsymbol{y}_1, \dots, \boldsymbol{y}_n]$. The kernel matrix for the training data $\breve{\boldsymbol{X}}$ is defined as $\mathbb{R}^{n\times n} \ni \breve{\boldsymbol{K}} := \boldsymbol{\Phi}(\breve{\boldsymbol{X}})^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}})$ where $\mathbb{R}^{t\times n} \ni \boldsymbol{\Phi}(\breve{\boldsymbol{X}}) := [\boldsymbol{\phi}(\boldsymbol{x}_1), \dots, \boldsymbol{\phi}(\boldsymbol{x}_n)]$. I normalize the kernel matrix [1] as $\breve{\boldsymbol{K}}(i,j) := \breve{\boldsymbol{K}}(i,j)/\big[\breve{\boldsymbol{K}}(i,i)\breve{\boldsymbol{K}}(j,j)\big]^{1/2}$ where $\breve{\boldsymbol{K}}(i,j)$ denotes the (i,j)-th element of the kernel matrix. According to representation theory [2], the projection matrix $\boldsymbol{U}_1 \in \mathbb{R}^{d\times d_1}$ can be expressed as a linear combination of the projected training data. Hence, I have $\mathbb{R}^{t\times d_1} \ni \boldsymbol{\Phi}(\boldsymbol{U}_1) = \boldsymbol{\Phi}(\breve{\boldsymbol{X}})\,\boldsymbol{\Theta}$ where every column of $\boldsymbol{\Theta} := [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{d_1}] \in \mathbb{R}^{n\times d_1}$ is the vector of coefficients. The projection of the pulled data is $\mathbb{R}^{d_1\times n} \ni \boldsymbol{\Phi}(\boldsymbol{U}_1)^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}}) = \boldsymbol{\Theta}^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}})^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}}) = \boldsymbol{\Theta}^\top \breve{\boldsymbol{K}}$.

In the kernel backprojection algorithm, in the first network layer, I project the pulled data from the feature space with dimensionality t to another feature space with dimensionality d_1. The projections of the next layers are the same as in backprojection. In other words, *kernel backprojection applies backprojection in the feature space rather than the input space.* In a mini-batch set-up, I use the columns of the normalized kernel corresponding to the batch samples, denoted by $\{\boldsymbol{k}_i \in \mathbb{R}^n\}_{i=1}^b$. Therefore, the projection of the i-th data point in the batch is $\mathbb{R}^{d_1} \ni \boldsymbol{\Theta}^\top \boldsymbol{k}_i$. In kernel backprojection, the dimensionality of the input is n and the kernel vector $\boldsymbol{k}_i$ is fed to the network as input. By replacing the $\boldsymbol{x}_i$ by $\boldsymbol{k}_i$, Algorithm 1 is applicable for kernel backprojection. In the test phase, I normalize the kernel over the matrix $[\breve{\boldsymbol{X}}, \boldsymbol{x}_t]$ where $\boldsymbol{x}_t \in \mathbb{R}^d$ is the test data point. Then, I take the portion of normalized kernel which corresponds to the kernel over the training versus test data, denoted by $\mathbb{R}^n \ni \boldsymbol{k}_t := \boldsymbol{\Phi}(\breve{\boldsymbol{X}})^\top \boldsymbol{\Phi}(\boldsymbol{x}_t)$. The projection at the first layer is then $\mathbb{R}^{d_1} \ni \boldsymbol{\Theta}^\top \boldsymbol{k}_t$.

3.3.2 Deep Network

In this section, I propose algorithms for neural network-based dimensionality reduction by deep networks. The proposed methods are Fisher losses, including FDT and FDC, and triplet mining using Bayesian updating, including BUT and BUNCA.

3.3.2.1 Fisher Loss for Training Siamese Network

The triplet and contrastive loss functions, introduced in Section 2.2.4, are used for training Siamese networks. These loss functions increase and decrease the inter-class and intra-class variances of embedded data, respectively. The same goal is tackled by FDA (see Section 2.2.1). This motivated me to propose Fisher loss functions for Siamese networks.

Network Structure for Our Proposed Losses: Consider any arbitrary neural network as the backbone. This network can be either a multi-layer perception or a convolutional network. Let q be the number of its output neurons, i.e., the dimensionality of its embedding space. I add a fully connected layer after the q-neurons layer to a new embedding space (output layer) with $p \leq q$ neurons. Denote the weights of this layer by $\boldsymbol{U} \in \mathbb{R}^{q \times p}$. I name the first q-dimensional embedding space as the *latent space* and the second p-dimensional embedding space as the *feature space*. My proposed loss functions are network-agnostic as they can be used for any network structure and topology of the backbone. The overall network structure for the usage of the proposed loss functions is depicted in Fig. 3.4.

Consider a triplet $\{\boldsymbol{x}_a, \boldsymbol{x}_n, \boldsymbol{x}_d \in \mathbb{R}^d\}$ or a pair $\{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d\}$. I feed the triplet or pair to the network. I denote the latent embedding of data by $\{\boldsymbol{o}_a, \boldsymbol{o}_n, \boldsymbol{o}_d \in \mathbb{R}^q\}$ and $\{\boldsymbol{o}_1, \boldsymbol{o}_2 \in \mathbb{R}^q\}$ while the feature embedding of data is denoted by $\{\mathbf{f}(\boldsymbol{x}_a), \mathbf{f}(\boldsymbol{x}_n), \mathbf{f}(\boldsymbol{x}_d) \in \mathbb{R}^p\}$ and $\{\mathbf{f}(\boldsymbol{x}_1), \mathbf{f}(\boldsymbol{x}_2) \in \mathbb{R}^p\}$. The last layer of network is projecting the latent embedding to the feature space where the activation function of the last layer is linear because of unsupervised feature extraction. Hence, the last layer acts as a linear projection $\mathbf{f}(\boldsymbol{x}) = \boldsymbol{U}^\top \boldsymbol{o}$. During the training, the latent space is adjusted to extract some features; however, the last-layer projection fine-tunes the latent features in order to have better discriminative features.

Fisher Discriminant Triplet Loss: As in neural networks, the loss function is usually minimized, I minimize the negative of Fisher criterion:

$$\underset{\boldsymbol{U}}{\text{minimize}} \quad -J = \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U}) - \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_B \boldsymbol{U}). \tag{3.69}$$

This problem is ill-defined because by increasing the total scatter of embedded data, the inter-class scatter also gets larger and this objective function gets decreased. Therefore,

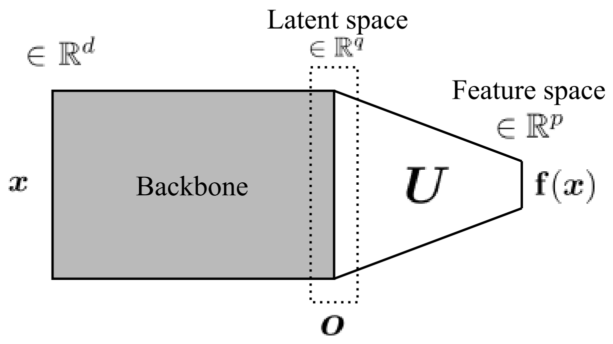

Figure 3.4: The network structure for the proposed Fisher loss functions.

the embedding space scales up and explodes gradually to increase the term $\mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_B \boldsymbol{U})$. In order to control this issue, I penalize the total scatter of the embedded data, denoted by $\boldsymbol{S}_T \in \mathbb{R}^{d \times d}$:

$$\min_{\boldsymbol{U}} \; \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U}) - \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_B \boldsymbol{U}) + \epsilon\, \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_T \boldsymbol{U}), \tag{3.70}$$

where $\epsilon \in (0,1)$ is the regularization parameter. The total scatter can be considered as the summation of the inter- and intra-class scatters (see Eq. 2.4). Hence:

$$\begin{aligned}&\mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U}) - \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_B \boldsymbol{U}) + \epsilon\, \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_T \boldsymbol{U}) = \mathbf{tr}\big(\boldsymbol{U}^\top (\boldsymbol{S}_W - \boldsymbol{S}_B + \epsilon\, \boldsymbol{S}_T) \boldsymbol{U}\big) \\ &\overset{(2.4)}{=} \mathbf{tr}\big(\boldsymbol{U}^\top ((\epsilon+1)\, \boldsymbol{S}_W + (\epsilon-1)\, \boldsymbol{S}_B) \boldsymbol{U}\big) \overset{(a)}{=} (2-\lambda)\, \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U}) - \lambda\, \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_B \boldsymbol{U}),\end{aligned} \tag{3.71}$$

where (a) is because $(0,1) \ni \lambda := 1 - \epsilon$. It is recommended for ϵ and λ to be close to one and zero, respectively because the total scatter should be controlled not to explode. For example, a good value can be $\lambda = 0.1$.

I want the inter-class scatter term to get larger than the intra-class scatter term by a margin $\alpha > 0$. Hence, the FDT loss, to be minimized, is defined as:

$$\ell_{\text{FDT}} = \Big[(2-\lambda)\, \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_W \boldsymbol{U}) - \lambda\, \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}_B \boldsymbol{U}) + \alpha\Big]_+, \tag{3.72}$$

where we defer the mathematical definition of intra- and inter-class scatter matrices in our loss functions to the following sections.

Fisher Discriminant Contrastive Loss: Rather than the triplets of data, one can consider the pairs of samples. For this goal, I propose the FDC loss function defined as:

$$\ell_{\text{FDC}} = (2-\lambda)\, \mathbf{tr}(\boldsymbol{U}^\top \widetilde{\boldsymbol{S}}_W \boldsymbol{U}) + \big[-\lambda\, \mathbf{tr}(\boldsymbol{U}^\top \widetilde{\boldsymbol{S}}_B \boldsymbol{U}) + \alpha\big]_+, \tag{3.73}$$

where the intra- and inter-class scatter matrices, which will be defined in the following sections, consider the anchor-neighbor and anchor-distant pairs.

Scatter Matrices in FDT: Let the output embedding of the backbone, i.e. the second-to-last layer of total structure, be denoted by $\boldsymbol{o} \in \mathbb{R}^q$. We call this embedding the *latent embedding*. Consider the latent embeddings of anchor, neighbor, and distant, denoted by $\boldsymbol{o}_a$, $\boldsymbol{o}_n$, and $\boldsymbol{o}_d$, respectively. If having a mini-batch of b triplets, I can define $\mathbb{R}^{q\times b} \ni \boldsymbol{O}_W := [\boldsymbol{o}_a^1 - \boldsymbol{o}_n^1, \dots, \boldsymbol{o}_a^b - \boldsymbol{o}_n^b]$ and $\mathbb{R}^{q\times b} \ni \boldsymbol{O}_B := [\boldsymbol{o}_a^1 - \boldsymbol{o}_d^b, \dots, \boldsymbol{o}_a^b - \boldsymbol{o}_d^b]$ where $\boldsymbol{o}^i$ is the i-th sample in the mini-batch. The intra- and inter-class scatter matrices are, respectively, defined as:

$$\mathbb{R}^{q\times q} \ni \boldsymbol{S}_W := \sum_{i=1}^{b} (\boldsymbol{o}_a^i - \boldsymbol{o}_n^i)(\boldsymbol{o}_a^i - \boldsymbol{o}_n^i)^\top = \boldsymbol{O}_W \boldsymbol{O}_W^\top, \tag{3.74}$$

$$\mathbb{R}^{q\times q} \ni \boldsymbol{S}_B := \sum_{i=1}^{b} (\boldsymbol{o}_a^i - \boldsymbol{o}_d^i)(\boldsymbol{o}_a^i - \boldsymbol{o}_d^i)^\top = \boldsymbol{O}_B \boldsymbol{O}_B^\top. \tag{3.75}$$

The ranks of the intra- and inter-class scatters are $\min(q, b-1)$. As the subspace of FDA can be interpreted as the eigenspace of $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$, the rank of the subspace would be $\min(q, b-1) = b-1$ because we usually have $b < q$. In order to improve the rank of the embedding subspace, I slightly strengthen the main diagonal of the scatter matrices [99]:

$$\boldsymbol{S}_W := \boldsymbol{O}_W \boldsymbol{O}_W^\top + \mu_W \boldsymbol{I}, \tag{3.76}$$

$$\boldsymbol{S}_B := \boldsymbol{O}_B \boldsymbol{O}_B^\top + \mu_B \boldsymbol{I}, \tag{3.77}$$

where $\mu_W, \mu_B > 0$ are small positive numbers, e.g., 10^{-4}. Hence, the embedding subspace becomes full rank with $q \geq p$.

Scatter Matrices in FDC: As in the regular contrastive loss, I consider the pairs of anchor-neighbor and anchor-distant for the FDC loss. Let y be zero and one when the pair $\{\boldsymbol{x}_1^i, \boldsymbol{x}_2^i\}$ is an anchor-neighbor or anchor-distant pair, respectively. The latent embedding of this pair is denoted by $\{\boldsymbol{o}_1^i, \boldsymbol{o}_2^i\}$. The intra- and inter-class scatter matrices in the FDC loss are, respectively, defined as:

$$\widetilde{\boldsymbol{S}}_W := \sum_{i=1}^{b} (1-y)(\boldsymbol{o}_1^i - \boldsymbol{o}_2^i)(\boldsymbol{o}_1^i - \boldsymbol{o}_2^i)^\top + \mu_W \boldsymbol{I} = \widetilde{\boldsymbol{O}}_W \widetilde{\boldsymbol{O}}_W^\top + \mu_W \boldsymbol{I}, \tag{3.78}$$

$$\widetilde{\boldsymbol{S}}_B := \sum_{i=1}^{b} y(\boldsymbol{o}_1^i - \boldsymbol{o}_2^i)(\boldsymbol{o}_1^i - \boldsymbol{o}_2^i)^\top + \mu_B \boldsymbol{I} = \widetilde{\boldsymbol{O}}_B \widetilde{\boldsymbol{O}}_B^\top + \mu_B \boldsymbol{I}, \tag{3.79}$$

where $\widetilde{\boldsymbol{O}}_W := [\{\boldsymbol{o}_1^i - \boldsymbol{o}_2^i \,|\, y = 0\}]$ and $\widetilde{\boldsymbol{O}}_B := [\{\boldsymbol{o}_1^i - \boldsymbol{o}_2^i \,|\, y = 1\}]$.

Note that in both FDT and FDC loss functions, there exist the weight matrix $\boldsymbol{U}$ and the intra- and inter-class scatter matrices. By back-propagation, both the last layer and the previous layers are trained because $\boldsymbol{U}$ in loss affects the last layer, and the scatter matrices in loss impact all the layers.

3.3.2.2 Triplet Mining Using Bayesian Updating

For training a triplet Siamese network, triplets, containing anchor, positive, and negative, should be sampled. Here, I aim to draw the positive and negative samples for every anchor instance in a dynamic manner. The main idea is to sample the positive and negative instances of triplets for every anchor in a mini-batch of data from some distributions rather than from the embedded data points themselves. This gives the triplet network more opportunity to explore the embedding space for increasing and decreasing the inter- and intra-class variances because the triplet information is not restricted to only the embedded data but is instead stochastic.

Preliminaries and Notations: Consider a q-dimensional training dataset $\{\boldsymbol{z}_i\}_{i=1}^n$ where $\boldsymbol{z}_i \in \mathbb{R}^q$. The class labels of instances are $\{y_i\}_{i=1}^n$. Suppose I have c number of classes in the dataset. I use the mini-batch (of size b) stochastic gradient descent for training the network. Let n^j denote the training sample size per class in a mini-batch. I show the i-th training instance of the j-th class in a mini-batch by $\boldsymbol{z}_i'^j$. Let $\boldsymbol{x}_i'^j \in \mathbb{R}^d$ denote the embedding of $\boldsymbol{z}_i'^j$ by the triplet network where the dimensionality of embedding space is d.

The data for each class are accumulated by receiving new mini-batches of data. Let n_0^j denote the sample size of accumulated data for the j-th class so far. The sample size per j-th class in a mini-batch is denoted by n'^j. In this work, we have $n'^1 = \cdots = n'^c = n' = \lceil b/c \rceil$ and $n_0^1 = \cdots = n_0^c = n_0$ because we take the same sample size per class in the mini-batch. This n' is the sample size of new incoming data per class in every mini-batch. The accumulated data for the j-th class so far are denoted by $\boldsymbol{x}^{0,j}$. Also, $\boldsymbol{\mu}^j$ and $\boldsymbol{\Sigma}^j$ are the mean and covariance of the distribution of the j-th class, respectively.

Sampling Algorithm: I assume a multivariate normal distribution for the embedded data of every class. This assumption makes sense according to the central limit theorem and the fact that the normal distribution is the most common continuous distribution. In the first batch, where there is not already any embedding of training data, I use Maximum Likelihood Estimation (MLE) to estimate the distribution parameters. The mean and

```
Procedure: TrainTripletNetwork($\{\boldsymbol{z}_i\}_{i=1}^n$, $\{y_i\}_{i=1}^n$)
Input: training data: $\{\boldsymbol{z}_i\}_{i=1}^n$, training labels: $\{y_i\}_{i=1}^n$
for all required epochs do
    for all batches in epoch do
        $\{\boldsymbol{x}_i\}_{i=1}^b \leftarrow$ Feed $\{\boldsymbol{z}_i\}_{i=1}^b$ to the triplet network
        for class $j$ from 1 to $c$ do
            if it is first mini-batch then
                $\boldsymbol{\mu}^{0,j} := (1/n') \sum_{i=1}^{n'} \boldsymbol{x}_i'^j$
                $\boldsymbol{\Sigma}^{0,j} := (1/n') \sum_{i=1}^{n'} (\boldsymbol{x}_i'^j - \boldsymbol{\mu}^{0,j})(\boldsymbol{x}_i'^j - \boldsymbol{\mu}^{0,j})^\top$
            else
                $\boldsymbol{\mu}'^j := (1/n') \sum_{i=1}^{n'} \boldsymbol{x}_i'^j$
                $\boldsymbol{\mu}^{0,j} := (n'\boldsymbol{\mu}'^j + n_0\boldsymbol{\mu}^{0,j})/(n' + n_0)$
                if $n' + n_0 > d + 1$ then
                    $\boldsymbol{\Upsilon} := n'\boldsymbol{\Sigma}'^j + n_0\boldsymbol{\Sigma}^{0,j} + \frac{n'n_0}{n'+n_0}(\boldsymbol{\mu}^{0,j} - \boldsymbol{\mu}'^j)(\boldsymbol{\mu}^{0,j} - \boldsymbol{\mu}'^j)^\top$
                    $\boldsymbol{\Sigma}^{0,j} := \boldsymbol{\Upsilon}^{-1}/(n' + n_0 - d - 1)$
                else
                    $\boldsymbol{\Sigma}^{0,j} := (1/n') \sum_{i=1}^{n'} (\boldsymbol{x}_i'^j - \boldsymbol{\mu}'^j)(\boldsymbol{x}_i'^j - \boldsymbol{\mu}'^j)^\top$
        for instance $i$ from 1 to $b$ do
            anchor $\leftarrow \boldsymbol{x}_i$
            for class $j$ from 1 to $c$ do
                if $j = y_i$ then
                    Sample $(c-1)$ positive instances $\sim \mathcal{N}(\boldsymbol{\mu}^{0,j}, \boldsymbol{\Sigma}^{0,j})$
                else
                    Sample a negative instance $\sim \mathcal{N}(\boldsymbol{\mu}^{0,j}, \boldsymbol{\Sigma}^{0,j})$
        Minimize the triplet/NCA loss with the $(b \times (c-1))$ triplets.
```

Algorithm 2: Dynamic Triplet Sampling with Bayesian Updating

covariance of the embedded data of every class are estimated by the sample mean and covariance matrix, respectively.

In later batches after the first batch, we do have some existing data per class, denoted by $n_0^j, \forall j$. According to Bayesian updating, the mean and covariance of distribution of every class can be updated (see [119]). I update the mean and covariance matrix of the distribution of every class by the expectation of marginal distributions for the mean and covariance. According to the expectations of these two distributions which can be found in [119], the updates of mean and covariance of the j-th class can be performed by Eqs. (2.25) and (2.26), respectively.

The proposed dynamic triplet sampling is summarized in Algorithm 2. The mean and covariance of every class are estimated by MLE at the initial batch. In the following batches, Bayesian updating is exploited for updating the mean and covariance of classes. After the means and covariances are updated, I sample the triplets. For every instance of a batch, considered as an "anchor", a negative instance is sampled from each different class resulting in $(c - 1)$ negatives per anchor. Accordingly, $(c - 1)$ positive instances are also sampled from the same class of anchor. Overall, $(b \times (c - 1))$ triplets are sampled in every mini-batch while the distributions of classes are being updated dynamically.

Optimization of the Loss Functions: The proposed dynamic triplet sampling can be used for either triplet or NCA loss functions, introduced in Section 2.2.4. I name these methods by BUT and BUNCA, respectively.

3.4 Summary of the Chapter

This chapter was on dimensionality reduction and proposed several new algorithms in different categories of dimensionality reduction, i.e., spectral, probabilistic, and neural network-based algorithms. In spectral dimensionality reduction category, I proposed WFDA, RDA, and image quality aware embedding (including SSIM kernel, ISCA, and LLISE). QQE was proposed as a probabilistic dimensionality reduction method. The neural network-based methods were divided into shallow and deep networks where backprojection was proposed for the former and Fisher losses (FDT and FDC losses) and BUT/BUNCA triplet mining approaches were proposed for the latter.

Among the categories of dimensionality reduction, spectral methods are fast but cannot usually handle large volumes of data. The probabilistic methods are more robust to outliers usually. The neural network-based methods have the advantage of being able to handle larger volumes of data. The proposed dimensionality reduction methods can be used for feature extraction from data. This feature extraction can be used to find an embedding space for better representation of data or separation of classes.

Chapter 4

Proposed Algorithms for Numerosity Reduction

Numerosity reduction reduces the number or cardinality of data instances by either ranking them or discarding the non-informative ones. In this chapter, I propose several algorithms for numerosity reduction. Note that the algorithms, which are proposed in this thesis, are developed based on the mathematics and techniques developed in previous work, introduced in Chapter 2. This chapter divides numerosity reduction into three categories which are variance-based (Section 4.1), geometry-based (Section 4.2), and isolation-based (Section 4.3) methods. In variance-based category, it proposes PSA (Section 4.1.1) and IRMD (Section 4.1.2). This chapter proposes CAD (Section 4.2.1) in the geometry-based numerosity reduction group. In isolation-based category, it proposes iMondrian forest (Section 4.3.1). Note that some applications of these methods, in the field of image processing and denoising, will be proposed in Chapter 5.

4.1 Algorithms Based on Variance

In the following, I propose new algorithms for variance-based numerosity reduction. My proposed algorithms are PSA and IRMD.

4.1.1 Principal Sample Analysis

Assume there exist C classes indexed by $j = \{1, \dots, C\}$. The N training samples are denoted by X, and the N^j training samples from the j^{th} class are denoted by X^j. The PSA algorithm consists of four stages: preprocessing, finding sets of major samples, ranking major samples, and ranking minor samples (see Fig. 4.1). This algorithm is mostly based on intra-class and inter-class variances of data and in this manner, it is similar to the approach of FDA. In the PSA algorithm, I denote dimensionality of data by D.

4.1.1.1 Preprocessing

PSA applies regression on the samples of every class. However, this requires at least $(D-1)$ samples in every class to regress data on a $(D-1)$-dimensional space of samples of the class. Therefore, the number of samples of every class must be at least $(D-1)$. If this condition does not exist for a class, either the number of samples of the class should be increased or the dimensionality of data should be decreased.

4.1.1.2 Finding Sets of Major Samples

Regression Score: A good representative set of samples should contain samples which can predict all the data points in the class with minimum error. Inspired by RANdom SAmple Consensus (RANSAC), N_M^j samples of the j^{th} class, denoted by X_M^j, are randomly selected from the samples X^j for several iterations. As in RANSAC, a regression method such as linear regression [26] is applied to the selected samples for several iterations, but with a difference. In our problem, there does not exist any label as required by regression. To overcome this challenge, regression is performed $(D-1)$ times where each one of the dimensions is considered once as the label for regression and the rest of dimensions form the observations for regression. In every iteration of RANSAC, the regression is performed on all N^j samples of the class and also on the N_M^j major samples:

$$\text{All samples:} \quad \beta_a^j = \Big((\mathcal{X}^j[-d])^\top \mathcal{X}^j[-d]\Big)^{-1} (\mathcal{X}^j[-d])^\top \mathcal{X}^j[d], \tag{4.1}$$

$$\text{Major samples:} \quad \beta_M^j = \Big((\mathcal{X}_M^j[-d])^\top \mathcal{X}_M^j[-d]\Big)^{-1} (\mathcal{X}_M^j[-d])^\top \mathcal{X}_M^j[d]. \tag{4.2}$$

If the selected samples form a good representative of the all the data for class j, the two vectors β_a^j and β_M^j are closely parallel and have cosine close to 1. Therefore, the regression

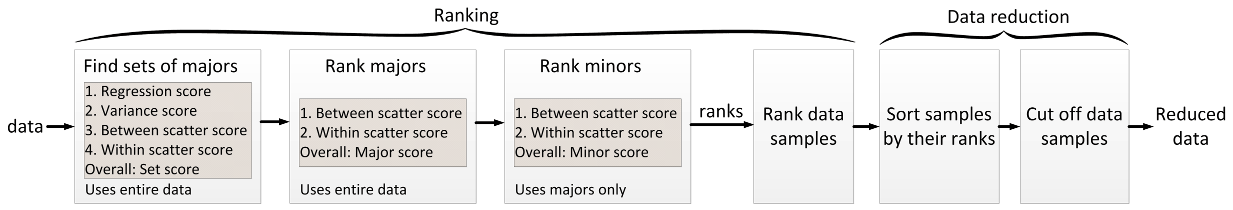

Figure 4.1: Overall structure for principal sample analysis

score $s_M^{j,R}$ of major samples is:

$$s_M^{j,R} = \cos(\beta_M^j, \beta_a^j) = \frac{(\beta_M^j)^\top \beta_a^j}{||\beta_M^j||_2 ||\beta_a^j||_2}. \tag{4.3}$$

Variance Score: The major samples are supposed not to be close to each other because very nearby samples share similar information and therefore are redundant. The scatter of major samples of class j is:

$$S_M^{j,\nu} = \sum_{i=1}^{N_M^j} (x_{M,i}^j - \overline{x}_M^j)(x_{M,i}^j - \overline{x}_M^j)^\top, \tag{4.4}$$

where $x_{M,i}^j$ is the i^{th} sample in the set X_M^j, and $\overline{x}_M^j = (1/N_M^j) \sum_{i=1}^{N_M^j} x_{M,i}^j$ is the mean of samples in set X_M^j. As the eigenspace of the scatter matrix carries information about the variance of the data [26], the variance score is:

$$s_M^{j,\nu} = \mathbf{tr}(S_M^{j,\nu}) \propto \mathrm{Var}(x_M^j), \tag{4.5}$$

where $\mathbf{tr}(.)$ is the trace of matrix.

Between Scatter Score: The major samples are supposed to be farther from the other classes for the sake of more discrimination. Therefore, the between scatter of major samples of the j^{th} class are:

$$S_M^{j,B} = \sum_{c=1, c\neq j}^{C} \sum_{k=1}^{N^c} w_k^c (\overline{x}_M^j - x_k^c)(\overline{x}_M^j - x_k^c)^\top, \tag{4.6}$$

where C is the number of classes, x_k^c is the k^{th} sample of class c, $\overline{x}_M^j$ is the mean of major samples in class j, and w_k^c is the weight associated with x_k^c, calculated as:

$$w_k^c = \frac{1}{2}\big(1 + \cos(x_k^c, \overline{x}^c)\big) = \frac{1}{2}\big(1 + \frac{x_k^{c\top} \overline{x}^c}{||x_k^c||_2 ||\overline{x}^c||_2}\big), \tag{4.7}$$

where $\overline{x}^c$ is the mean of class c. This weighting, which is in the range $[0, 1]$, gives more weight to the samples close (parallel) to the mean of its own class. The between scatter score is:

$$s_M^{j,B} = \mathbf{tr}(S_M^{j,B}). \tag{4.8}$$

Within Scatter Score: Although it is better for the major samples to be sparse, as explained in variance score, they should also be close to each other to *represent the core of the class*. The within scatter of the major samples of the j^{th} class is:

$$S_M^{j,W} = \sum_{i=1}^{N_M^j} \sum_{k=1,k\neq i}^{N_M^j} w_{M,k}^j (x_{M,i}^j - x_{M,k}^j)(x_{M,i}^j - x_{M,k}^j)^\top, \tag{4.9}$$

where weight $w_{M,k}^j$ is the same as equation (4.7) if substituting x_k^c and $\overline{x}^c$ with $x_{M,k}^j$ and $\overline{x}_M^j$ respectively. The within scatter score is:

$$s_M^{j,W} = \mathbf{tr}(S_M^{j,W}). \tag{4.10}$$

Ranking Sets: The score of a set of major samples (set score) is finally calculated as,

$$s_M^j = s_M^{j,R} \times s_M^{j,\nu} \times s_M^{j,B} \times (1/s_M^{j,W}), \tag{4.11}$$

because a better set should have larger regression score, larger variance score, larger between scatter score, and smaller within scatter score. The algorithm is performed for every class j. In every iteration, the set score of selected samples is found and finally the set of samples having the best set score is returned.

4.1.1.3 Ranking Major Samples

Between Scatter Score: The major samples are supposed to be far from the other classes. The between scatter of a major sample $x_{M,i}^j$ in class j from the samples of other classes is:

$$S_{M,i}^{j,B} = \sum_{c=1,c\neq j}^{C} \sum_{k=1}^{N^c} w_k^c (x_{M,i}^j - x_k^c)(x_{M,i}^j - x_k^c)^\top, \tag{4.12}$$

where weight w_k^c is the same as equation (4.7). The between scatter score of a sample in the set is then found as,

$$s_{M,i}^{j,B} = \mathbf{tr}(S_{M,i}^{j,B}). \tag{4.13}$$

Within Scatter Score: It is better for the major samples to be close to the samples of their own class. The within scatter of a major sample $x^j_{M,i}$ in class j from the samples of its own class is:

$$S^{j,W}_{M,i} = \sum_{k=1}^{N^j} w^j_k (x^j_{M,i} - x^j_k)(x^j_{M,i} - x^j_k)^\top, \tag{4.14}$$

where weight w^j_k is the same as equation (4.7) if substituting x^c_k and $\overline{x}^c$ with x^j_k and $\overline{x}^j$ respectively. The within scatter score of a sample in the set is:

$$s^{j,W}_{M,i} = \mathbf{tr}(S^{j,W}_{M,i}). \tag{4.15}$$

Ranking: The score of a major sample is:

$$s^j_{M,i} = s^{j,B}_{M,i} \times (1/s^{j,W}_{M,i}), \tag{4.16}$$

because the better sample in the major set is farther from the samples of other classes and is closer to samples of its own class. This score is found for every sample in the best major set of the class and the major samples are ranked by these scores.

4.1.1.4 Ranking Minor Samples

Between Scatter Score: It is better for the minor samples to be far from the major samples of other classes. Here, the major samples of other classes are assumed to be proper and purer representatives of their classes; thus, the found major samples are used rather than whole samples of other classes. The between scatter score of a minor sample $x^j_{m,i}$ in class j from the major samples of other classes is:

$$S^{j,B}_{m,i} = \sum_{c=1,c\neq j}^{C} \sum_{k=1}^{N^c_M} w^c_{M,k} (x^j_{m,i} - \mathbf{x}^c_{M,k})(x^j_{m,i} - \mathbf{x}^c_{M,k})^\top, \tag{4.17}$$

where $\mathbf{x}^c_{M,k}$ is the indexed sample in the sorted major samples according their score (i.e., $\mathbf{x}^c_{M,1}$ has the best rank in set of majors). The weight $w^c_{M,k}$ is:

$$w^c_{M,k} = \frac{N^c_M - k + 1}{N^c_M (N^c_M + 1)/2}, \tag{4.18}$$

which gives larger weight to better ranked major samples because they are more important to their class. The between scatter score of a minor sample is:

$$s^{j,B}_{m,i} = \mathbf{tr}(S^{j,B}_{m,i}). \tag{4.19}$$

Within Scatter Score: The minor samples should also be close to the major samples of their own class. The within scatter of a minor sample from the major samples of its class is:

$$S_{m,i}^{j,W} = \sum_{k=1}^{N_M^j} w_{M,k}^j (x_{m,i}^j - \mathbf{x}_{M,k}^j)(x_{m,i}^j - \mathbf{x}_{M,k}^j)^\top, \tag{4.20}$$

where weight w_k^j is the same as equation (4.18) if substituting N_M^c with N_M^j. The within scatter score of a minor sample is:

$$s_{m,i}^{j,W} = \mathbf{tr}(S_{m,i}^{j,W}). \tag{4.21}$$

Ranking: The score of a minor sample is:

$$s_{m,i}^j = s_{m,i}^{j,B} \times (1/s_{m,i}^{j,W}), \tag{4.22}$$

because a better minor sample is farther from the major samples of other classes and is closer to major samples of its own class. This score is found for every minor sample of class and the minor samples are ranked by the scores. Finally, in every class, the ranks of minor samples are concatenated after the ranks of major samples to have the ranks of all samples in the class.

4.1.1.5 Principal Sample Analysis for Regression and Clustering Tasks

One can extend PSA to be useful for regression and clustering tasks as well as classification. Applying slight changes to PSA makes it useful for regression and clustering tasks. In regression and clustering, there exists merely one set or class of data; hence, the between scatter scores in PSA should be set to 1 because we do not have several classes to compute their between scatters. Moreover, calculation of scatters in PSA can be simplified to iterate over only one existing class which is the whole dataset. For regression case, the regression score can also be simplified because the labels of regression are now available enabling us to omit the loop over the $(D-1)$ dimensions in PSA. Finally, PSA can be used for regression and clustering after these changes.

4.1.2 Instance Ranking by Matrix Decomposition

In this section, I propose Instance Ranking by Matrix Decomposition (IRMD). The general idea of IRMD is that the more important data points fall close to the more informative directions of data. These informative directions can be found with the help of matrix decomposition. In the following, I explain the details of this proposed method.

4.1.2.1 Instance Ranking Using Matrix Decomposition

As was explained in Section 2.2.6, the matrix decomposition factorizes a matrix into the product of two matrices $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^\top$. The matrices $\boldsymbol{U} \in \mathbb{R}^{d\times k}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times k}$ can be interpreted as kernel or similarity over columns (instances) and rows (features) of $\boldsymbol{X}$, respectively. From another perspective, $\boldsymbol{U}$ and $\boldsymbol{V}$ can be considered as bases and coefficients of instances, respectively. Comparing the p-th columns in $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^\top$, I have $\boldsymbol{x}_p = \sum_{j=1}^{k} \boldsymbol{V}(p,j)\,\boldsymbol{u}_j \in \text{span}\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_k\}$; therefore, columns of $\boldsymbol{U}$ are the bases for column space or range of $\boldsymbol{X}$, i.e., columns of $\boldsymbol{U}$ span the space of instances. In SVD, for example, the columns of $\boldsymbol{U}$ are principal components of $\boldsymbol{X}$ showing its maximum variations. The idea is that the basis vectors capture the most informative directions in the data in terms of some type of information such as variation. Hence, it is expected that the more important instances are more similar to these basis vectors. Therefore, the instances can be ranked based on their similarity to the basis vectors. There are several basis vectors so I fuse their information into one similarity score as explained in the following.

Measuring Similarity with Basis Vectors: Suppose that $\boldsymbol{u}$ is a basis vector and $\boldsymbol{x}$ is an instance. The cosine is used for the similarity metric because it can be written in simple closed-form matrix operations. If the two vectors $\boldsymbol{x}$ and $\boldsymbol{u}$ are already normalized and have unit length, the cosine is reduced to the inner product $\cos(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{x}^\top\boldsymbol{u}$. Thus, I normalize the instances (columns of $\boldsymbol{X}$), i.e., $\widetilde{\boldsymbol{X}}(:,j) = \boldsymbol{X}(:,j)/||\boldsymbol{X}(:,j)||_2$, $\forall j \in \{1,\dots,n\}$, where $\widetilde{\boldsymbol{X}}$ is the normalized dataset. I assume that basis vectors $\boldsymbol{U}$ are already orthonormal; otherwise, they should be normalized as well which is the case with the basis of NMF, DL, and PLU decomposition. Having k basis vectors as columns of $\boldsymbol{U} \in \mathbb{R}^{d\times k}$ and n instances as columns of $\boldsymbol{X} \in \mathbb{R}^{d\times n}$, the cosine of basis vectors and instances are $\mathbb{R}^{n\times k} \ni \cos(\boldsymbol{X},\boldsymbol{U}) = \widetilde{\boldsymbol{X}}^\top\boldsymbol{U}$.

I would like these scores to be positive in order to be ready for fusion, so I use $|\widetilde{\boldsymbol{X}}^\top\boldsymbol{U}|_\varepsilon \in \mathbb{R}^{n\times k}$ in range $[\varepsilon, 1]$ where the safe absolute value $|\boldsymbol{A}|_\varepsilon := \max(|\boldsymbol{A}|, \varepsilon)$ prevents the elements of matrix $\boldsymbol{A}$ from being zero and ε is a small positive number (e.g., 0.001). The intuition of absolute value is that in measuring similarity with a basis vector, I should care only about the direction of the basis vector and not its sign of direction. The intuition of safe absolute value is that I want to fuse the k scores of every instance by multiplication so having a small score regarding one of the basis should not spoil all scores.

In order to fuse the scores of similarities of every instance with different basis vectors, I use weighted product of scores, $|s_1|_\varepsilon^{w_1} \times \dots \times |s_k|_\varepsilon^{w_k}$. This can be written in logarithmic form, $-(w_1 \log|s_1|_\varepsilon + \dots + w_k \log|s_k|_\varepsilon)$, which gives a closed-form matrix where the negative signs cancel with those obtained from logarithms of scores in range $[\varepsilon, 1]$. Finally, the overall score of every instance with respect to the k basis vectors can be obtained as the entries of:

$$\mathbb{R}^n \ni \boldsymbol{s} = -\log\big(|\widetilde{\boldsymbol{X}}^\top \boldsymbol{U}|_\varepsilon\big)\boldsymbol{w}, \tag{4.23}$$

where $\mathbb{R}^k \ni \boldsymbol{w} = [w_1, \dots, w_k]$ contains the weights regarding the k basis vectors. In case the eigenvalues are obtained (such as SVD, SPCA, and FDA) or are not obtained (such as NMF, DL, PLU and QR decompositions) from decomposition or subspace learning, I use $w_i = \frac{1/\lambda_i}{\sum_j 1/\lambda_j}$ and $w_i = \frac{2i}{k(k+1)}$, $\forall i = 1, \dots, k$, respectively. When having eigenvalues, the basis vectors and eigenvalues are sorted in descending order. Note that smaller weight gives more importance as the range of $|s_i|_\varepsilon$ is $[\varepsilon, 1]$. Moreover, as the weights should be positive, if there are any negative eigenvalue, I shift all values to become positive $\lambda_i \leftarrow \lambda_i - 2\lambda_k \mathbb{I}(\lambda_k < 0)$ where $\mathbb{I}(.)$ is the indicator function (1 if its condition is satisfied and 0 otherwise).

Unsupervised Cases: In unsupervised learning, we are given a dataset X without labels. The matrix $\boldsymbol{U}$ can be obtained from its decomposition, i.e., in SVD: $\boldsymbol{X} = (\widetilde{\boldsymbol{U}})(\boldsymbol{\Lambda}\widetilde{\boldsymbol{V}}^\top) = \boldsymbol{U}\boldsymbol{V}^\top$, in NMF: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^\top$, in PLU: $\boldsymbol{X} = (\boldsymbol{P}\widetilde{\boldsymbol{L}})(\widetilde{\boldsymbol{U}}) = \boldsymbol{U}\boldsymbol{V}^\top$, in QR: $\boldsymbol{X} = (\boldsymbol{Q})(\boldsymbol{R}) = \boldsymbol{U}\boldsymbol{V}^\top$, and in DL: $\boldsymbol{X} = (\boldsymbol{D})(\boldsymbol{R}) = \boldsymbol{U}\boldsymbol{V}^\top$ (see Section 2.2.6). To be more clear, for example for SVD, I take left singular matrix $\widetilde{\boldsymbol{U}}$ to be $\boldsymbol{U}$ and $\boldsymbol{\Lambda}\widetilde{\boldsymbol{V}}^\top$ to be $\boldsymbol{V}^\top$. The scores of instances are calculated using Eq. (4.23).

Regression Cases: In regression, there exist some independent variables $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ (I call them observations) and some dependant variables, or labels, $\boldsymbol{Y} \in \mathbb{R}^{\ell\times n}$. The goal of regression is to predict the $\boldsymbol{Y}$ from given $\boldsymbol{X}$. Instance ranking may help regression in terms of finding the most important instances for this prediction. Considering merely $\boldsymbol{X}$ takes into account the distribution and variation of data regardless of the labels. On the other hand, considering only $\boldsymbol{Y}$ ignores the effect of $\boldsymbol{X}$ and concentrates on the output labels to be predicted and the relation of instances in terms of labels. These two scenarios have their own merits so I fuse them. The explained methodology for unsupervised cases can be applied once to $\boldsymbol{X}$ and once on $\boldsymbol{Y}$. Let the scores obtained from processing $\boldsymbol{X}$ and $\boldsymbol{Y}$ be $\boldsymbol{s}_X$ and $\boldsymbol{s}_Y$, respectively. The fusion of these scores can be done by multiplying these two scores which are both positive: $\mathbb{R}^n \ni \boldsymbol{s} = \boldsymbol{s}_X \odot \boldsymbol{s}_Y$, where $\odot$ denotes Hadamard product.

Classification Cases: In classification, the dataset $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ and the corresponding possible labels $\boldsymbol{Y} \in \mathbb{R}^{\ell\times n}$ exist, while every column of $\boldsymbol{Y}$ encodes one of the $|\mathcal{C}|$ classes. Every important instance should be a satisfactory representative of its own class. Let the instances of class c be denoted by $\boldsymbol{X}_c$. If $\boldsymbol{U}_{X_c}$ denotes the basis matrix obtained from decomposition of $\boldsymbol{X}_c$ and $\widetilde{\boldsymbol{X}}_c$ is the normalized $\boldsymbol{X}_c$, the scores of instances in class c are obtained as $\mathbb{R}^{n_c} \ni \boldsymbol{s}_{X_c} = -\log\big(|\widetilde{\boldsymbol{X}}_c^\top \boldsymbol{U}_{X_c}|_\varepsilon\big)\boldsymbol{w}$.

On the other hand, the instances should be important in terms of discrimination of classes. Here, as in the case of regression, I cannot find the scores for solely the labels $\boldsymbol{Y}$. The reason is that the labels of a class are all similar and the rank of $\boldsymbol{Y}$ will be only $|\mathcal{C}|$ with so many repetitive columns if it is one-hot encoded. Therefore, I put the labels $\boldsymbol{Y}$ alongside $\boldsymbol{X}$ to yield a new matrix. However, in order to bias instances of every class to fall closer to each other in the space, it is best to choose $\boldsymbol{Y}$ to be encoded by one-hot encoding resulting in $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{C}|\times n}$. Finally, concatenating $\boldsymbol{X}$ and $\boldsymbol{E}$ results in $\boldsymbol{D} := [\boldsymbol{X}^\top, \boldsymbol{E}^\top]^\top \in \mathbb{R}^{(d+|\mathcal{C}|)\times n}$. The same approach is used for finding the scores of instances based on $\boldsymbol{D}$: $\mathbb{R}^n \ni \boldsymbol{s}_D = -\log\big(|\widetilde{\boldsymbol{D}}^\top \boldsymbol{U}_D|_\varepsilon\big)\boldsymbol{w}$, where $\widetilde{\boldsymbol{D}}$ and $\boldsymbol{U}_D$ are the normalized $\boldsymbol{D}$ and basis vectors from decomposition of $\boldsymbol{D}$, respectively. Finally, the two scores $\boldsymbol{s}_X$ and $\boldsymbol{s}_D$ are fused similarly as before: $\mathbb{R}^n \ni \boldsymbol{s} = \boldsymbol{s}_X \odot \boldsymbol{s}_D$, and the reduced dataset ($\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Y}}$) is obtained.

4.1.2.2 Instance Ranking Using Subspace Learning

The goal of subspace learning is to project data form the original d-dimensional space to a lower dimensional subspace with dimensionality k. The projection is formulated as $\boldsymbol{U}^\top\boldsymbol{X}$ where $\boldsymbol{U} \in \mathbb{R}^{d\times k}$ is the projection matrix. Interestingly, the projection matrix can be considered as the basis matrix in the matrix decomposition. The reason for this claim is that assuming $\mathbb{R}^{k\times n} \ni \boldsymbol{V}^\top := \boldsymbol{U}^\top\boldsymbol{X}$, the reconstruction of matrix $\boldsymbol{X}$ can be written as $\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^\top$, which is the matrix decomposition of $\boldsymbol{X}$. Therefore, the columns of the projection matrix $\boldsymbol{U}$ can be interpreted as the basis vectors with which the instances can be compared. Hence, considering the projection matrix as $\boldsymbol{U}$, we can have scores for all the unsupervised, regression, and classification cases.

4.2 Algorithm Based on Geometry

I propose Curvature Anomaly Detection (CAD) and Kernel Curvature Anomaly Detection (K-CAD) for anomaly detection and iCAD and Kernel Inverse Curvature Anomaly Detection (K-iCAD) for numerosity reduction. These methods have a geometry-based approach.

The main idea of these methods is as follows. Every data point is considered to be the vertex of a hypothetical polyhedron (see Fig. 2.1-a). For every point, we find its kNN. The k neighbors of the point (vertex) form the k faces of a polyhedron meeting at that vertex.

Then, the more curvature that point (vertex) has, the more anomalous (or less important in terms of numerosity reduction) it is because it is far away (different) from its neighbors. The concept of polyhedron curvature was introduced in Section 2.2.7. Please note that this view to anomaly detection and numerosity reduction is completely novel.

4.2.1 Curvature Anomaly Detection

Because of the idea explained above, *anomaly score* s_A is proportional to the curvature. Since, according to the equation of angular effect (see Section 2.2.7), the curvature is proportional to minus the summation of angles, one can consider the anomaly score to be inversely proportional to the summation of angles. Without loss of generality, I assume the angles to be in range $[0, \pi]$ (otherwise, I take the smaller angle). The less the angles between two edges of the polyhedron, the more their cosine. As the anomaly score is inversely proportional to the angles, I can use cosine for the anomaly score: $s_A(\boldsymbol{x}_i) \propto 1/\tau_a \propto \cos(\tau_a)$. I define the anomaly score to be the summation of cosine of the angles of the polyhedron faces meeting at that point: $s_A(\boldsymbol{x}_i) := \sum_{a=1}^{k} \cos(\tau_a) = \sum_{a=1}^{k} (\breve{\boldsymbol{x}}_a^\top \breve{\boldsymbol{x}}_{a+1})/(||\breve{\boldsymbol{x}}_a||_2 ||\breve{\boldsymbol{x}}_{a+1}||_2)$ where $\breve{\boldsymbol{x}}_a := \boldsymbol{x}_a - \boldsymbol{x}_i$ is the a-th edge of the polyhedron passing through the vertex $\boldsymbol{x}_i$, $\boldsymbol{x}_a$ is the a-th neighbor of $\boldsymbol{x}_i$, and $\breve{\boldsymbol{x}}_{a+1}$ denotes the next edge sharing the same polyhedron face with $\breve{\boldsymbol{x}}_a$ where $\breve{\boldsymbol{x}}_{k+1} = \breve{\boldsymbol{x}}_1$.

Note that finding the pairs of edges which belong to the same face is difficult and time-consuming so I relax this calculation to the summation of the cosine of angles between all pairs of edges meeting at the vertex $\boldsymbol{x}_i$:

$$s_A(\boldsymbol{x}_i) := \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \frac{\breve{\boldsymbol{x}}_a^\top \breve{\boldsymbol{x}}_b}{||\breve{\boldsymbol{x}}_a||_2 ||\breve{\boldsymbol{x}}_b||_2}, \tag{4.24}$$

where $\breve{\boldsymbol{x}}_a := \boldsymbol{x}_a - \boldsymbol{x}_i$, $\breve{\boldsymbol{x}}_b := \boldsymbol{x}_b - \boldsymbol{x}_i$, and $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ denote the a-th and b-th neighbor of $\boldsymbol{x}_i$. In Eq. (4.24), I have omitted the redundant angles because of symmetry of inner product. Note that the Eq. (4.24) implies that I normalize the k neighbors of $\boldsymbol{x}_i$ to fall on the unit hyper-sphere centered at $\boldsymbol{x}_i$ and then compute their cosine similarities (see Fig. 2.1-c).

The mentioned relaxation is valid for the following reason. Take two edges meeting at the vertex $\boldsymbol{x}_i$. If the two edges belong to the same polyhedron face, the relaxation is exact. Consider the case where the two edges do not belong to the same face. These two edges are connected with a set of polyhedron faces. If we tweak one of the two edges to increase/decrease the angle between them, the angle of that edge with its neighbor edge on the same face also increases/decreases. Therefore, the changes in the additional angles of relaxation are consistent with the changes of the angles between the edges sharing the same faces.

After scoring the data points, one can sort the points and find a suitable threshold visually using a scree plot of the scores. However, in order to find anomalies automatically, I apply K-means clustering, with two clusters, to the scores. The cluster with the larger mean is the cluster of anomalies because the higher the score, the more anomalous the point. For finding anomalies for out-of-sample data, I find kNN for the out-of-sample point where the neighbors are from the training points. Then, I calculate the anomaly score using Eq. (4.24). The K-means cluster whose mean is closer to the calculated score determines whether the point is normal or anomaly.

4.2.2 Kernel Curvature Anomaly Detection Algorithm

For the reason explained in Chapter 1, I also propose the kernel version of CAD, named K-CAD) to work on data in the feature space. In K-CAD, the two stages of finding kNN and calculating the anomaly score are performed in the feature space. The kernel over two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is the inner product of their pulled data, i.e., $\mathbb{R} \ni k(\boldsymbol{x}_1, \boldsymbol{x}_2) := \boldsymbol{\phi}(\boldsymbol{x}_1)^\top \boldsymbol{\phi}(\boldsymbol{x}_2)$. The Euclidean distance in the feature space is [115]: $||\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{\phi}(\boldsymbol{x}_j)||_2 = \sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2k(\boldsymbol{x}_i, \boldsymbol{x}_j) + k(\boldsymbol{x}_j, \boldsymbol{x}_j)}$. Using this distance, I find the k-NN of the dataset in the feature space.

After finding k-NN in the feature space, I calculate the score in the feature space. I pull the vectors $\breve{\boldsymbol{x}}_a$ and $\breve{\boldsymbol{x}}_b$ to the feature space so $\breve{\boldsymbol{x}}_a^\top \breve{\boldsymbol{x}}_b$ is changed to $k(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_b) = \boldsymbol{\phi}(\breve{\boldsymbol{x}}_a)^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_b)$. Let $\boldsymbol{K}_i \in \mathbb{R}^{k \times k}$ denote the kernel of neighbors of $\boldsymbol{x}_i$ whose (a, b)-th element is $k(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_b)$. The vectors in Eq. (4.24) are normalized. In the feature space, this is equivalent to normalizing the kernel $k(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_b) := k(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_b) / \sqrt{k(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_a)\, k(\breve{\boldsymbol{x}}_b, \breve{\boldsymbol{x}}_b)}$ [1]. If $\boldsymbol{K}'_i$ denotes the normalized kernel $\boldsymbol{K}_i$, the anomaly score in the feature space is:

$$s_A(\boldsymbol{x}_i) := \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \boldsymbol{K}'_i(a, b), \tag{4.25}$$

where $\boldsymbol{K}'_i(a, b)$ is the (a, b)-th element of the kernel. The K-means clustering and out-of-sample anomaly detection are similarly performed as in CAD.

My observations in experiments showed that the anomaly score in K-CAD is ranked inversely for some kernels such as RBF, Laplacian, and polynomial (different degrees) in various datasets. In other words, for example, in K-CAD with linear (i.e., CAD), cosine, and sigmoid kernels, the more anomalous points have greater score but in K-CAD with RBF, Laplacian, and polynomial kernels, the smaller score is assigned to the more anomalous points. I conjecture that the reason lies in the characteristics of the kernels. I defer more investigations for the reason as a future work. In conclusion, for the mentioned kernels, one should either multiply the scores by -1 or take the K-means cluster with smaller mean as the anomaly cluster.

4.2.3 Inverse Curvature Anomaly Detection Algorithm

If the anomaly detection uses scores, one can see instance ranking and numerosity reduction in the opposite perspective of anomaly detection. Therefore, the ranking scores can be considered as the anomaly scores multiplied by -1: $s_R(\boldsymbol{x}_i) := -1 \times s_A(\boldsymbol{x}_i) = -\sum_{a=1}^{k-1} \sum_{b=a+1}^{k} (\breve{\boldsymbol{x}}_a^\top \breve{\boldsymbol{x}}_b)/(||\breve{\boldsymbol{x}}_a||_2 ||\breve{\boldsymbol{x}}_b||_2)$. I sort the ranking scores in descending order. The data point with larger ranking score is more important. As the order of ranking scores is inverse of the order of anomaly scores, I name this method iCAD.

Prototype selection can be performed in two approaches: (I) the data points are sorted and a portion of the points having the best ranks is retained, or (II) a portion of data points is retained as prototypes and the rest of points are discarded. Some examples of the fist approach is IRMD, PSA, SOS, and SE. DROP3 and ENN are examples for the second approach. The iCAD can be used for both approaches. The first approach is ranking the points with the ranking score. For the second approach, I apply K-means clustering, with two clusters, to the ranking scores and take the points of the cluster with larger mean.

4.2.4 Kernel Inverse Curvature Anomaly Detection Algorithm

For the reason explained in Chapter 1, I also propose the kernel version of iCAD. One can perform iCAD in the feature space to have *Kernel iCAD (K-iCAD)*. The ranking score is again the anomaly score multiplied by -1 to reverse the ranks of scores: $s_R(\boldsymbol{x}_i) := -1 \times s_A(\boldsymbol{x}_i) = -\sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \boldsymbol{K}_i'(a, b)$. Again, there are two approaches where the points are ranked or K-means is applied on the scores. Note that for what was mentioned before, I do not multiply by -1 for some kernels including RBF, Laplacian, and polynomial. Note that iCAD and K-iCAD are task agnostic and can be used for data reduction in classification, regression, and clustering. For classification, I apply the method for every class while in regression and clustering, the method is applied on the entire data.

4.3 Algorithm Based on Isolation

As was explained in Section 2.2.8, isolation forest and Mondrian forest are proposed for batch anomaly detection and online classification, respectively. Here, I combine them in order to propose iMondrian forest for batch and online anomaly detection.

4.3.1 iMondrian Forest: Batch Processing

Training: The iMondrian forest is an ensemble of iMondrian trees. Algorithm 3 shows this ensemble where $\mathcal{X} := \{\boldsymbol{x}_i\}_{i=1}^n$ is the batch of data and $|\mathcal{F}|$ is the number of trees in the forest. Inspired by [90], the data in a batch can be subsampled with subsampling size $\psi = 256$ for growing the tree. If subsampling is used, $\mathcal{X}$ denotes the sample of data and $n = \psi$. The iMondrian tree is grown recursively as detailed in Algorithm 4. As with Mondrian trees, bounds of hyper-rectangular blocks are defined $\mathcal{B}_r := (\ell_{r1}, u_{r1}] \times \cdots \times (\ell_{rd}, u_{rd}]$ along each of d dimensions for the r-th node. Let $\mathcal{X}_b := \{\boldsymbol{x}_i^{(b)}\} := \{\boldsymbol{x}_i \mid \boldsymbol{x}_i \in \mathcal{B}_r\}$ be the subset of data which exist in the smallest block enclosing the node. For a node, the lower and upper bounds of $\mathcal{B}_r$ along the features are denoted by $\boldsymbol{\ell}_{\mathcal{X}_b}$ and $\boldsymbol{u}_{\mathcal{X}_b}$, respectively.

In order to split a block, I sample a random variable e from an exponential distribution with the rate $\lambda = \sum_{j=1}^d (\boldsymbol{u}_{\mathcal{X}_b}(j) - \boldsymbol{\ell}_{\mathcal{X}_b}(j))$ which is the linear dimension of $\mathcal{B}_r$. I set the split time of a node to the split time of its parent plus e. I sample the dimension of the split, q, from a discrete distribution proportional to $(\boldsymbol{u}_{\mathcal{X}_b}(j) - \boldsymbol{\ell}_{\mathcal{X}_b}(j))$. I sample the value of the split, p, from a continuous uniform distribution $U(\boldsymbol{\ell}_{\mathcal{X}_b}(q), \boldsymbol{u}_{\mathcal{X}_b}(q))$. The tree is grown until every node contains a single data point, i.e., $|\mathcal{X}| = 1$.

```
1 Procedure: BatchTraining($\mathcal{X}$, $|\mathcal{F}|$)
2 Input: $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, $|\mathcal{F}|$: number of trees
3 for tree $t$ from 1 to $|\mathcal{F}|$ do
4     $\mathcal{F} \leftarrow \mathcal{F} \cup$ iMondrianTree(root, $\mathcal{X}$, 0)
5 Return Forest $\mathcal{F}$
```

Algorithm 3: Batch training in iMondrian forest

Evaluation: After growing the iMondrian trees in the forest, I calculate the path length of every tree for a data point $\boldsymbol{x}$ as in Algorithm 5. The path length for the t-th tree, $l_t(\boldsymbol{x})$, is the number of edges traversed by the point from the root to the node containing point $\boldsymbol{x}$. I calculate the expected path length in the iMondrian forest using Eq. (2.29) and the

```
1  Procedure: iMondrianTree($r$, $\mathcal{X}$, $\tau_{\text{parent}}$)
2  Input: $r$: node pointer, $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, $\tau_{\text{parent}}$: split time of the parent node
3  $\mathcal{X}_b = \{\boldsymbol{x}_i^{(b)}\} = \{\boldsymbol{x}_i \mid \boldsymbol{x}_i \in \mathcal{B}_r\}$
4  $\boldsymbol{\ell}_{\mathcal{X}_b} \leftarrow \min(\{\boldsymbol{x}_i^{(b)} \mid \forall i\})$
5  $\boldsymbol{u}_{\mathcal{X}_b} \leftarrow \max(\{\boldsymbol{x}_i^{(b)} \mid \forall i\})$
6  if $|\mathcal{X}_b| > 1$ then
7      $e \sim \text{Exp}\big(\lambda = \sum_{j=1}^{d}(\boldsymbol{u}_{\mathcal{X}_b}(j) - \boldsymbol{\ell}_{\mathcal{X}_b}(j))\big)$
8      $\tau \leftarrow \tau_{\text{parent}} + e$
9      $q \leftarrow$ sample from $\{1, \dots, d\}$ with distribution $\propto (\boldsymbol{u}_{\mathcal{X}_b}(j) - \boldsymbol{\ell}_{\mathcal{X}_b}(j))$ for the $j$-th
         dimension
10     $p \sim U(\boldsymbol{\ell}_{\mathcal{X}_b}(q), \boldsymbol{u}_{\mathcal{X}_b}(q))$
11     $\mathcal{X}_{\text{left}} \leftarrow \{\boldsymbol{x} \in \mathcal{X}_b \mid \boldsymbol{x}(q) < p\}$
12     $\mathcal{X}_{\text{right}} \leftarrow \{\boldsymbol{x} \in \mathcal{X}_b \mid \boldsymbol{x}(q) \geq p\}$
13     Left $\leftarrow$ iMondrianTree(leftChild($r$), $\mathcal{X}_{\text{left}}, \tau$)
14     Right $\leftarrow$ iMondrianTree(rightChild($r$), $\mathcal{X}_{\text{right}}, \tau$)
15     Return internalNode{leftChild: Left, rightChild: Right, splitDim: q,
         splitVal: p, time: $\tau$, dim$_{\text{min}}$: $\boldsymbol{\ell}_{\mathcal{X}_b}$, dim$_{\text{max}}$: $\boldsymbol{u}_{\mathcal{X}_b}$, population: $|\mathcal{X}_b|$}
16 else
17     Return leafNode{time: $\infty$, dim$_{\text{min}}$: $\boldsymbol{\ell}_{\mathcal{X}_b}$, dim$_{\text{max}}$: $\boldsymbol{u}_{\mathcal{X}_b}$, population: 1}
```

Algorithm 4: Constructing iMondrian tree

```
1  Procedure: PathLength($\boldsymbol{x}$, $t$, $l$)
2  Input: $\boldsymbol{x}$: data point, $t$: iMondrian tree, $l$: current path length (initialized to
     0)
3  $q \leftarrow t$.splitDim
4  $p \leftarrow t$.splitVal
5  if $\boldsymbol{x}(q) < p$ then
6      Return PathLength($\boldsymbol{x}$, $t$.leftChild, $l + 1$)
7  else
8      Return PathLength($\boldsymbol{x}$, $t$.rightChild, $l + 1$)
```

Algorithm 5: Calculation of path length

anomaly score for point $\boldsymbol{x}$ using Eq. (2.28). For determining whether a point in the dataset is normal or an anomaly, one can either use the threshold $s = 0.5$ as in [90] or K-means

```
1 Procedure: ExtendIMondrianForest($\mathcal{X}^{(n)}$, $\mathcal{F}$)
2 Input: $\mathcal{X}^{(n)} = \{\boldsymbol{x}_i^{(n)}\}_{i=1}^{m}$: new data, $\mathcal{F}$: Forest
3 for $\boldsymbol{x}_i^{(n)} \in \mathcal{X}^{(n)}$ do
4     for tree $t \in \mathcal{F}$ do
5         ExtendIMondrianTree($t$.root, $\boldsymbol{x}_i^{(n)}$, 0)
```

Algorithm 6: Extension of iMondrian forest

clustering. In the threshold approach, the point is determined as anomaly if $s(\boldsymbol{x}) > 0.5$. In the K-means approach, I assign the scores of training data into two clusters and take the points in the cluster with greater mean as the anomaly points. The theoretical reason for threshold 0.5 is that the expected path length for the data point (Eq. (2.29)) is the estimation of the average path length (see $c(n)$ in Section 2.2.8.1) when $s = 0.5$ (see p. 415 in [90], same holds for iMondrian). The empirical reason is that the results of $s = 0.5$ and K-means are almost the same (as I will show in the experiments in Chapter 6).

In batch processing, for an out-of-sample data point, I feed the data point to the trees of iMondrian forest and calculate the score using Eq. (2.28). Then, I can use the threshold $s = 0.5$ again or assign the point to the cluster whose mean is closer to the score of the point. My experiments showed that both the threshold and clustering approaches have almost equally good performance for batch processing.

4.3.2 iMondrian Forest: Online Processing

Training: A major advantage of iMondrian forests is their ability to be updated online for new data. Let $\mathcal{X}^{(n)} := \{\boldsymbol{x}_i^{(n)}\}_{i=1}^{m}$ denote the m new data points. I process data points one-by-one to extend each tree in the forest (see Algorithm 6). Algorithm 7 describes how I extend each iMondrian tree for $\boldsymbol{x}^n$. The tree is extended recursively starting from the root. The lower and upper errors of deviation of a point from the smallest block contained by the node r are calculated as $\mathbb{R}^d \ni \boldsymbol{e}_\ell := \max(r.\text{dim}_{\min} - \boldsymbol{x}^{(n)}, \boldsymbol{0})$ and $\mathbb{R}^d \ni \boldsymbol{e}_u := \max(\boldsymbol{x}^{(n)} - r.\text{dim}_{\max}, \boldsymbol{0})$, respectively, where $\text{dim}_{\min}$ and $\text{dim}_{\max}$ are the upper and lower bounds of the block along different dimensions. I sample a random variable e from an exponential distribution with the rate $\lambda = \sum_{j=1}^{d}(\boldsymbol{e}_\ell(j) + \boldsymbol{e}_u(j))$.

In the case where the split time of the node r is greater than the split time of its parent plus e, a new node is created above the node r. Note that it started from the root and is moving downwards so the new node is added before the current node for which a

```
Procedure: ExtendIMondrianTree($r, \boldsymbol{x}^{(n)}, \tau_{\text{parent}}$)
Input: $r$: node pointer, new data point: $\boldsymbol{x}^{(n)}$, $\tau_{\text{parent}}$: split time of the parent
  node
$\boldsymbol{e}_\ell \leftarrow \max(r.\text{dim}_{\text{min}} - \boldsymbol{x}^{(n)}, \boldsymbol{0})$
$\boldsymbol{e}_u \leftarrow \max(\boldsymbol{x}^{(n)} - r.\text{dim}_{\text{max}}, \boldsymbol{0})$
$e \sim \text{Exp}\big(\lambda = \sum_{j=1}^{d}(\boldsymbol{e}_\ell(j) + \boldsymbol{e}_u(j))\big)$
if $\tau_{parent} + e < r.\tau$ then
    $q \leftarrow$ sample from $\{1, \ldots, d\}$ with distribution $\propto (\boldsymbol{e}_\ell(j) + \boldsymbol{e}_u(j))$ for the $j$-th
      dimension
    if $\boldsymbol{x}^{(n)}(q) > r.dim_{max}(q)$ then
        $p \sim U\big(r.\text{dim}_{\text{max}}(q), \boldsymbol{x}^{(n)}(q)\big)$
    else if $\boldsymbol{x}^{(n)}(q) < r.dim_{min}(q)$ then
        $p \sim U\big(\boldsymbol{x}^{(n)}(q), r.\text{dim}_{\text{min}}(q)\big)$
    newNode $\leftarrow$ internalNode{splitDim: $q$, splitVal: $p$, time: $\tau_{\text{parent}} + e$, dim$_{\text{min}}$:
      min($r$.dim$_{\text{min}}$, $\boldsymbol{x}^{(n)}$), dim$_{\text{max}}$: max($r$.dim$_{\text{max}}$, $\boldsymbol{x}^{(n)}$), population: $r$.population
      + 1}
    newNode.parent $\leftarrow$ $r$.parent
    if $\boldsymbol{x}^{(n)}(q) > p$ then
        newNode.leftChild $\leftarrow r$
        newNode.rightChild $\leftarrow$ iMondrianTree (rightSibling($r$), $\boldsymbol{x}^{(n)}$,
          newNode.time)
    else
        newNode.leftChild $\leftarrow$ iMondrianTree (leftSibling($r$), $\boldsymbol{x}^{(n)}$, newNode.time)
        newNode.rightChild $\leftarrow r$
else
    $r$.dim$_{\text{min}}$ $\leftarrow$ min($r$.dim$_{\text{min}}$, $\boldsymbol{x}^{(n)}$)
    $r$.dim$_{\text{max}}$ $\leftarrow$ max($r$.dim$_{\text{max}}$, $\boldsymbol{x}^{(n)}$)
    if $\boldsymbol{x}(r.splitDim) \leq r.splitVal$ then
        ExtendIMondrianTree($r$.leftChild, $\boldsymbol{x}^{(n)}$, $r.\tau$)
    else
        ExtendIMondrianTree($r$.rightChild, $\boldsymbol{x}^{(n)}$, $r.\tau$)
```

Algorithm 7: Extension of iMondrian tree

condition holds. In this case, I randomly pick a split dimension q from the distribution proportional to $(\boldsymbol{e}_\ell(j) + \boldsymbol{e}_u(j))$. If the value on dimension q of the data point is *greater* than the upper bound of the current block, then the split value p is sampled from the uniform distribution $U\big(r.\text{dim}_{\max}(q), \boldsymbol{x}^{(n)}(q)\big)$. If the value is *lower*, then p is sampled from $U\big(\boldsymbol{x}^{(n)}(q), r.\text{dim}_{\min}(q)\big)$. Depending on the split value and the feature of data point, I create an iMondrian tree as the left or right sibling of the node r.

In the case where the split time of the node r is less than the split time of its parent plus e, I simply descend down the tree and call the extending function recursively for the left or right of the node r depending on the split dimension and split values of the children.

Evaluation: After the extension of the trees of iMondrian forest, I can process data points through the forest to calculate their anomaly scores using Eq. (2.28). This can be done for all the new points and any other out-of-sample points. Whenever the trees have been updated, I should also ideally process previous batches of data through the forest again to recalculate their anomaly scores. This is expected since more data will lead to an improved model and a better structure for detection of false negative or positive points. However, for performance reasons, in practice this recalculation of scores could be done for just a window of the latest points. For online processing, our experiments showed that the threshold $s = 0.5$ is not necessarily the best threshold and K-means clustering works more better. Hence, I use K-means to cluster all the training points into two clusters and set the cluster with greater mean as anomalous. The out-of-sample points are assigned to the cluster whose mean is closer to their score.

4.4 Summary of the Chapter

This chapter was on numerosity reduction and proposed several new algorithms in different categories of numerosity reduction, i.e., variance-based, geometry-based, and isolation-based algorithms. In variance-based numerosity reduction category, I proposed PSA and IRMD. CAD was proposed as a geometry-based numerosity reduction method. The algorithm proposed in the isolation-based methods category was iMondrian forest.

In the variance-based numerosity reduction methods, the variances of data between the similar and dissimilar points as well as the most informative directions of data are considered. The geometry-based methods have an interesting visual perspective to data. The isolation-based methods have an isolation point of view where the anomalies or less important points fall away from other points and can be isolated. The proposed numerosity reduction methods can be used for ranking data instances by the importance or information. They may also be useful for outlier and anomaly detection.

Chapter 5

Proposed Algorithms for Applications of Data Reduction

To asses my proposed methods, this chapter proposes some applications for data reduction and reports their experimental results. Dimensionality reduction can be used in different applications for extracting informative features from raw data. Moreover, numerosity reduction and prototype selection can be used for selecting the most informative data instances in various applications. Two main categories of applications, i.e., medical image analysis (Section 5.1), image processing, and computer vision (Section 5.2), are introduced. In medical image analysis, I focus on histopathology data where Fisher loss (Section 5.1.3), triplet mining based on extreme distances (Section 5.1.4), and BUT / BUNCA (Section 5.1.5) for histopathology are proposed. In image processing and computer vision, I propose Roweisfaces for face recognition (Section 5.2.1), Roweisposes for 3D action recognition (Section 5.2.2), and image denoising by anomaly path (Section 5.2.3).

5.1 Medical Image Analysis

In this section, I propose algorithms for application of dimensionality reduction on medical image analysis. The histopathology projects which I worked on were funded by the available grant and data we had. I worked on histopathology project because it is a search problem on images for ranking. Moreover, histopathology has very large data as gigapixel images; hence, compact and meaningful embeddings are required to learn patterns of data and improve search and classification. Here, first, I introduce the utilized histopathology datasets.

Then, I explain the proposed using Siamese network for histopathology embedding, Fisher losses (FDT and FDC) for histopathology embedding, triplet mining for histopathology embedding, and dynamic triplet mining (using BUT/BUNCA) for histopathology embedding.

5.1.1 Datasets

For the experiments, I used different challenging histopathology datasets. In the following, I introduce the characteristics of these datasets.

CRC1 dataset: One of the histopathology datasets is the Colorectal Cancer 1 (CRC1) dataset [79]. This dataset is available at this link: click here. There exist 5000 histological images of 150×150 pixels in this dataset. It contains tissue patches from eight tissue types of colorectal cancer tissue slides. The tissue types are background (empty), adipose tissue, mucosal glands, debris, immune cells (lymphoma), complex stroma, simple stroma, and tumor epithelium. Some sample patches of CRC1 tissue types can be seen in Fig. 5.3.

CRC2 dataset: One of the histopathology datasets is the Colorectal Cancer 2 (CRC2) dataset [78]. This dataset is available at this link: click here. There exist 100,000 histological images of 224×224 pixels in this dataset. It includes nine classes of tissues, namely adipose, background, debris, lymphocytes, mucus, smooth muscle, normal colon mucosa (normal), cancer-associated stroma, and colorectal adenocarcinoma epithelium (tumor).

TCGA dataset: Another histopathology dataset is The Cancer Genome Atlas (TCGA) dataset [15]. This dataset is available at this link: click here. TCGA Whole Slide Images (WSIs) come from 25 different organs for 32 different cancer subtypes. We use the three most common sites, which are prostate, gastrointestinal, and lung [15, 77]. These organs have a total of 9 cancer subtypes, i.e., Prostate adenocarcinoma (PRAD), Testicular germ cell tumors (TGCT), Oesophageal carcinoma (ESCA), Stomach adenocarcinoma (STAD), Colonic adenocarcinoma (COAD), Rectal adenocarcinoma (READ), Lung adenocarcinoma (LUAD), Lung squamous cell carcinoma (LUSC), and Mesothelioma (MESO).

5.1.2 Siamese Network for Histopathology Embedding

I use the triplet loss and Siamese network, introduced in Chapter 2, for medical image analysis. In this section, I report the experimental result and analysis on medical images.

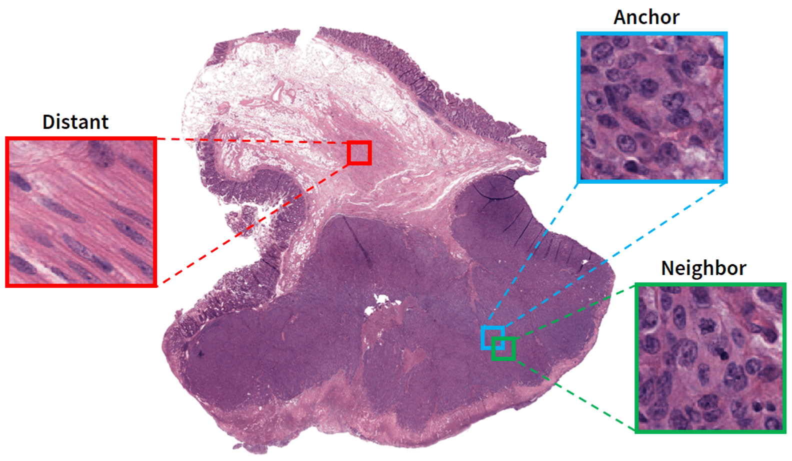

Figure 5.1: An example of the type 1 triplet generation from a sample WSI (from COAD subtype) from TCGA dataset.

5.1.2.1 Triplet Generation from TCGA Data

The TCGA dataset includes WSIs and not labeled patches. Hence, for generating triplets, one cannot simply take the patches from tissue types. The triplet is composed of anchor, positive (neighbor), and negative (distant) patches, in which anchor and neighbor are defined as similar and anchor and distant as dissimilar pairs. Inspired by [72], I utilized spatial correlation as one of the approaches to define the similarity among patches extracted from WSIs. In other words, I assumed that similar patterns usually emerge in an adjacent neighborhood, while the dissimilar layouts often appear in the spatially remote neighborhood. More specifically, a neighbor patch was selected within a certain range of the anchor's patch center of the same WSI. On the other hand, I used several alternatives for choosing the distant patch. The distant sample was chosen from (1) the same WSI as long as it was spatially remote, (2) another WSI associated with the same cancer subtype, (3) another WSI associated with other subtypes of the same organ, or (4) another WSI associated with another organ. An example of a type 1 triplet generation is depicted in Fig. 5.1.

5.1.2.2 Embeddings

First, I split the CRC1 dataset into 60% and 40% portions. I trained a triplet Siamese network with the ResNet-18 [67] as backbone. I trained the triplet network with the triplets from the train set of CRC1. The embedding of the test set of CRC1 data with this network is shown in Fig. 5.2-a. I applied UMAP [98] to visualize the 128-dimensional representations in 2D.

The triplets extracted from the CRC training set were sampled in a supervised manner as the labels of tissues were used. However, as I described in Section 5.1.2.1, the triplets of TCGA data were sampled using the spatial and tissue type information in an unsupervised manner. As a result, I trained extra two models on triplets extracted from TCGA. The first one was trained on all three anatomical sites while the second model was only trained on the gastrointestinal data from TCGA as the CRC1 data is also related to this anatomical site (organ). Similarly, the CRC1 test embeddings encoded by these models are shown in Figs. 5.2-b and 5.2-c, respectively. As these figures show, the CRC1 tissues have been well separated.

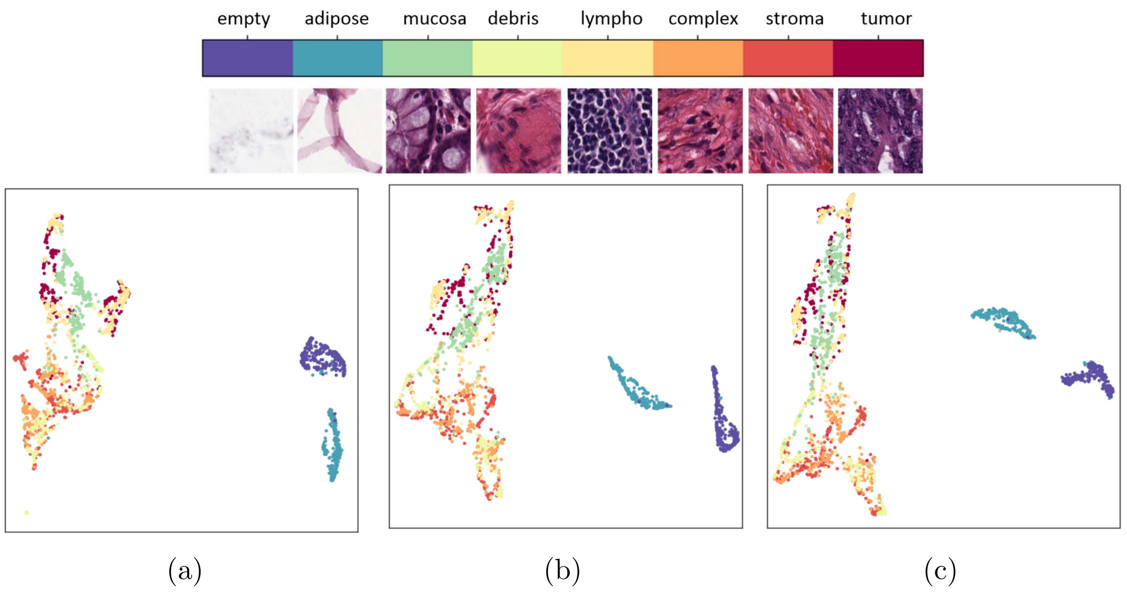

Figure 5.2: CRC1 test embeddings by (a) trained network with CRC1 training triplets, (b) trained network with TCGA training triplets (three organs), (c) trained network with TCGA training triplets (gastrointestinal organ).

5.1.3 Fisher Loss for Histopathology Embedding

I use the Fisher losses, introduced in Chapter 3, for medical image analysis. In this section, I report the experimental result and analysis on medical images.

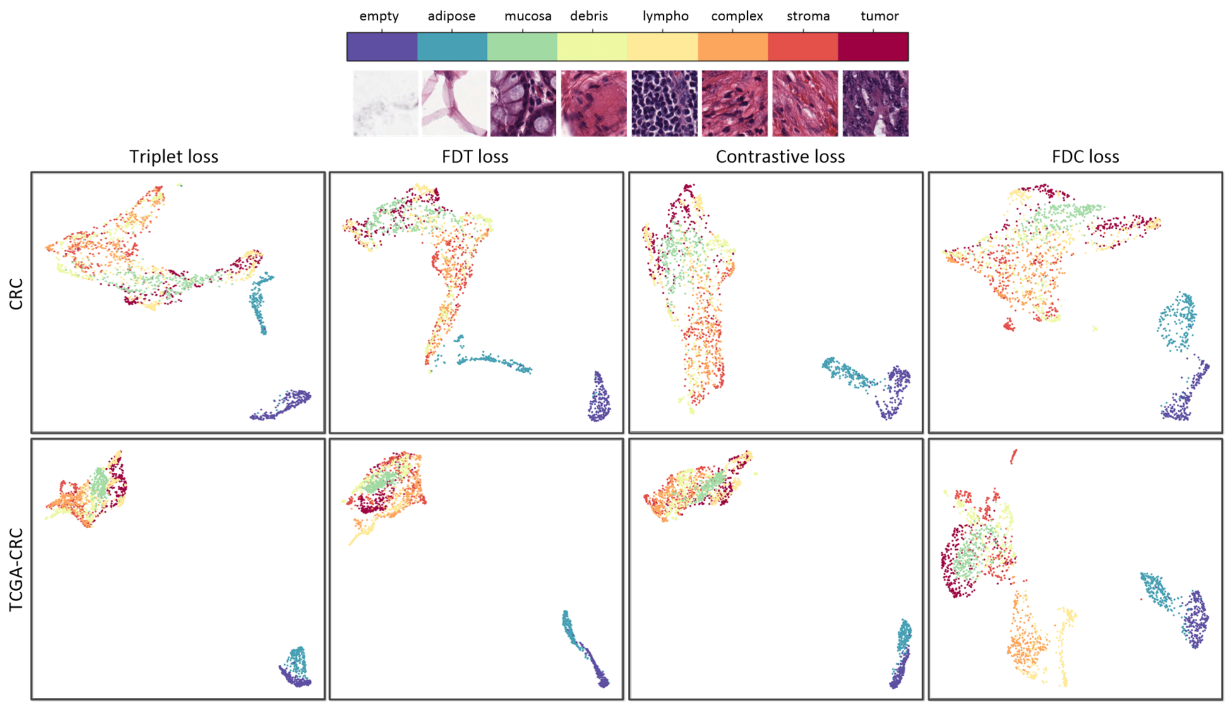

Figure 5.3: Application of Fisher Loss for histopathology data: Embedding of the CRC1 test data for different loss functions (top row: CRC1, bottom row: TCGA).

5.1.3.1 Processing on Datasets

I split the CRC1 data into train/test sets with 60%–40% portions. Using the training set, I extracted 22,528 triplets by considering the tissue types as the classes. For the TCGA data, by sampling patches from slides, I extracted 22,528 triplets to test the proposed losses with a large triplet sample size. The anchor and neighbor patches were selected from one WSI, but I used four ways of extraction of the distant patch, i.e., from the same WSI but far from the anchor, from another WSI of the same cancer subtype as an anchor, from another cancer subtype but the same anatomic site as anchor, and from another anatomic site.

5.1.3.2 Visualization of Embedding of Histopathology Data

I used $\lambda = 0.1$ in the FDT and FDC losses. I also used UMAP [98] for visualizing the 128-dimensional embedded data. For embedding the histopathology data, I performed two

Table 5.1: Experiments for Fisher losses on the CRC1 and TCGA datasets: Accuracy of 1-NN search for different loss functions.

	CRC1	TCGA-CRC
triplet	95.75%	95.50%
FDT ($\lambda = 0.01$)	96.45%	97.60%
FDT ($\lambda = 0.1$)	96.05%	96.40%
FDT ($\lambda = 0.8$)	95.35%	95.95%
contrastive	95.55%	96.55%
FDC ($\lambda = 0.01$)	94.25%	96.55%
FDC ($\lambda = 0.1$)	96.40%	98.10%
FDC ($\lambda = 0.8$)	97.00%	97.05%

different experiments. In the first experiment, I trained and tested the Siamese network using the CRC1 data. The second experiment was to train the Siamese network using TCGA data and test it using the CRC1 test set. The latter, which we denote by TCGA-CRC, is more difficult because it tests generalization of the feature space, which is trained by different data from the test data, although with a similar texture. Figure 5.3 shows the embeddings of the CRC1 test sets in the feature spaces trained by CRC1 and TCGA data. The embeddings by all losses, including FDT and FDC, are acceptable, noticing that the histopathology data are hard to discriminate even by a human (see the sample patches in Fig. 5.3). As expected, the empty and adipose data, which are similar, are embedded closely. Comparing the TCGA-CRC embeddings of contrastive and FDC losses shows FDC has discriminated classes slightly better. Overall, the good embedding in TCGA-CRC shows that the proposed losses can train a generalizing feature space, which is very important in histopathology analysis because of the lack of labeled data [73].

5.1.3.3 Numerical Comparison of Embeddings

The accuracy rates of the 1-NN search for the embedding test set of histopathology data by different loss functions are reported in Table 5.1. As the results show, in most cases, the FDT and FDC losses have outperformed the triplet and contrastive losses, respectively.

5.1.4 Triplet Mining for Histopathology Embedding

I use the triplet loss and triplet mining, introduced in Chapter 2, for medical image analysis. In this section, in addition to the existing offline triplet mining methods, I propose online mining and report the experimental result and analysis on medical images.

5.1.4.1 Online Mining by Extreme Distances

In addition to the triplet mining methods, reviewed in Chapter 2, I propose four additional online methods based on extreme distances. In the mini-batch, I consider every instance once as an anchor and take its nearest/farthest same-class instance as the easiest/hardest positive and its nearest/farthest other-class instance as the hardest/easiest negative instance. Hence, four different cases, i.e., Easiest Positive Easiest Negative (EPEN), Easiest Positive Hardest Negative (EPHN), Hardest Positive Easiest Negative (HPEN), and Hardest Positive Hardest Negative (HPHN), exist. Considering the extreme values, especially the farthest, was inspired by the opposition-based learning [128]. HPHN is equivalent to BH, already explained. I can also have a mixture of these four cases (i.e., *assorted* case) where for every anchor in the mini-batch, one of the cases is randomly considered. The proposed online mining loss functions are as follows:

$$\mathcal{L}_{\text{EPEN}} := \sum_{i=1}^{c} \sum_{a=1}^{w} \Big[m + \min_{p \in \{1,\dots,w\} \setminus \{a\}} \mathcal{D}(y_a^i, y_p^i) - \max_{\substack{j \in \{1,\dots,c\} \setminus \{i\} \\ n \in \{1,\dots,w\}}} \mathcal{D}(y_a^i, y_n^j) \Big]_+, \tag{5.1}$$

$$\mathcal{L}_{\text{EPHN}} := \sum_{i=1}^{c} \sum_{a=1}^{w} \Big[m + \min_{p \in \{1,\dots,w\} \setminus \{a\}} \mathcal{D}(y_a^i, y_p^i) - \min_{\substack{j \in \{1,\dots,c\} \setminus \{i\} \\ n \in \{1,\dots,w\}}} \mathcal{D}(y_a^i, y_n^j) \Big]_+, \tag{5.2}$$

$$\mathcal{L}_{\text{HPEN}} := \sum_{i=1}^{c} \sum_{a=1}^{w} \Big[m + \max_{p \in \{1,\dots,w\} \setminus \{a\}} \mathcal{D}(y_a^i, y_p^i) - \max_{\substack{j \in \{1,\dots,c\} \setminus \{i\} \\ n \in \{1,\dots,w\}}} \mathcal{D}(y_a^i, y_n^j) \Big]_+, \tag{5.3}$$

$$\mathcal{L}_{\text{Assorted}} := \sum_{i=1}^{c} \sum_{a=1}^{w} \Big[m + \min_{p \in \{1,\dots,w\} \setminus \{a\}} / \max \mathcal{D}(y_a^i, y_p^i) - \min_{\substack{j \in \{1,\dots,c\} \setminus \{i\} \\ n \in \{1,\dots,w\}}} / \max \mathcal{D}(y_a^i, y_n^j) \Big]_+, \tag{5.4}$$

where m is the margin, $[\cdot]_+$ is the standard Hinge loss, and $\min / \max$ denotes random selection between the minimum and maximum operators.

5.1.4.2 Offline Triplet Mining by Extreme Distances

In the offline triplet mining approach, the processing of data is not performed during the training of the triplet network but beforehand. The extreme distances are calculated only once on the whole training dataset and not repeatedly in the mini-batches during the training. The histopathology patterns in the input space cannot be distinguished, especially for the visually similar tissues [73]. Hence, I work on the extreme distances in the feature space trained using the class labels. The block diagram of the proposed offline triplet mining is depicted in Fig. 5.4. In the following, I explain the steps of mining.

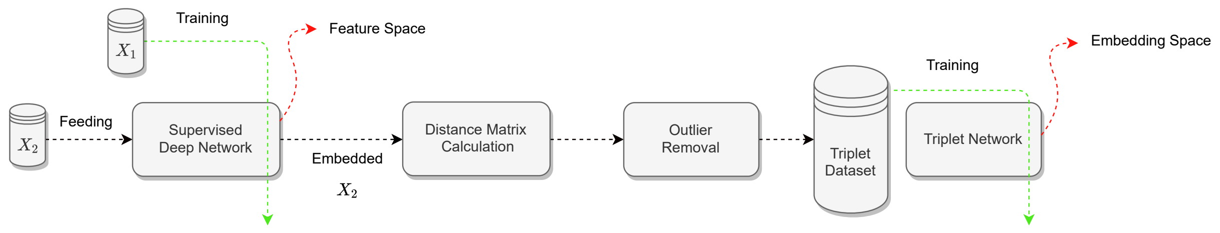

Figure 5.4: Block diagram for the offline triplet mining approach.

Training Supervised Feature Space: I first train a feature space in a supervised manner. For example, a deep network with a cross-entropy loss function can be used for training this space where the embedding of the one-to-last layer is extracted. I want the feature space to use the labels for better discrimination of classes by increasing their inter-class distances. Hence, I use a set of training data, call it X_1, for training the supervised network.

Distance Matrix in the Feature Space: After training the supervised network, I embed another set of the training data, denoted by X_2 (where $X_1 \cup X_2 = X$ and $X_1 \cap X_2 = \varnothing$), in the feature space. I compute a distance matrix on the embedded data in the feature space. Therefore, using a distance matrix, I can find cases with extreme distances. I consider every $x \in X_2$ as an anchor in a triplet where its nearest or farthest neighbors from the same and other classes are considered as its positive and negative instances, respectively. Again, there are four different cases with extreme distances, i.e., EPEN, EPHN, HPEN, and HPHN, in addition to the *assorted* case where one of the extreme cases is randomly selected for a triplet. There might exist some outliers in data whose embeddings fall much apart from others. In that case, merely one single outlier may become the hardest negative for all anchors. To prevent this problem, for every data instance in X_1 embedded in the feature space, I standardize the distances from other instances using the Z-score normalization. I consider the instances having distances above the 99-th percentile (i.e., normalized distances above the threshold 2.3263) as outliers and ignore them.

Training the Triplet Network: After preparing the triplets in any of the extreme cases, a triplet network [118] is trained using the triplets for learning an embedding space for better discrimination of dissimilar instances while holding the similar instances close enough. I call the spaces learned by the supervised and triplet networks as the feature space and embedding space, respectively (see Fig. 5.4).

Table 5.2: Results of offline triplet mining on the training and test data

	Train					Test				
	R@1	R@4	R@8	R@16	Acc.	R@1	R@4	R@8	R@16	Acc.
EPEN	92.60	97.66	98.85	99.48	95.87	89.86	96.78	98.20	99.11	94.58
EPHN	**94.82**	**98.46**	**99.27**	**99.70**	**97.10**	**94.50**	**98.41**	**99.25**	**99.67**	**97.21**
HPEN	93.22	96.93	97.71	98.35	96.16	87.11	97.01	98.83	99.59	94.10
HPHN	81.62	89.73	93.15	95.78	91.19	42.71	71.07	86.13	95.32	71.25
assorted	86.40	93.65	95.93	97.66	92.53	88.56	97.31	98.95	99.52	94.60

Table 5.3: Results of online triplet mining on the training and test data

	Train					Test				
	R@1	R@4	R@8	R@16	Acc.	R@1	R@4	R@8	R@16	Acc.
BA [19]	95.13	98.45	99.20	99.60	97.73	82.42	93.94	96.93	98.58	90.85
BSH [118]	95.83	98.77	**99.42**	**99.65**	98.00	84.70	94.78	97.34	98.75	91.74
HPHN [68]	91.52	97.14	98.60	99.34	96.09	**86.65**	95.80	97.81	99.04	93.20
NCA [61]	**96.45**	**98.92**	99.40	99.69	**98.40**	78.93	92.58	96.39	98.47	89.65
PNCA [100]	93.59	98.06	99.04	99.53	97.08	80.45	93.02	96.34	98.42	88.72
EP [141]	84.30	94.30	96.94	98.38	92.78	74.00	90.35	95.00	97.93	85.88
EP-D	86.11	95.90	97.86	99.00	93.30	77.23	92.14	96.18	98.49	87.95
DWS [139]	84.43	94.78	97.27	98.59	92.25	83.74	94.36	96.72	98.33	92.20
EPEN	87.44	95.89	97.84	98.90	94.03	85.48	95.40	97.65	98.92	92.57
EPHN	95.44	98.68	99.22	99.57	97.90	85.34	94.80	97.49	98.81	91.77
HPEN	89.53	96.67	98.21	99.15	95.04	85.38	95.30	97.55	98.82	92.56
assorted	93.73	97.98	99.02	99.57	97.12	86.57	**96.18**	**98.25**	**99.30**	**93.44**

5.1.4.3 Comparison of Embeddings

Using the Recall@k (R@k) measure, I compared the offline and online triplet mining based on extreme distances with the baseline methods, introduced in Chapter 2, in Tables 5.2 and 5.3, respectively. The baseline methods which I compared with are BA [19], BSH [118], BH [68], EP [141], DWS [139], NCA [61], and PNCA [100]. I also compared with EP-D where we use Euclidean distance rather than inner product in EP. As these tables show, the offline mining can generate a better statistical representation of the population by working on the whole dataset.

5.1.5 Dynamic Triplet Sampling for Histopathology Embedding

I use dynamic triplet sampling using BUT and BUNCA, introduced in Chapter 3, for medical image analysis. In this section, I report the experimental result and analysis on

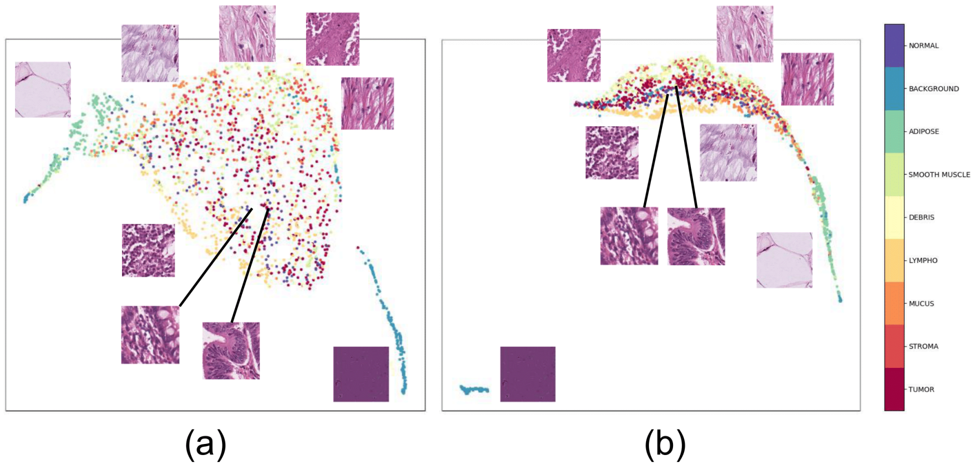

Figure 5.5: 2D visualization of test embeddings: (a) CRC2 using BUT and (b) CRC2 using BUNCA.

medical images.

5.1.5.1 Experimental Setup

For the experiments, I used the CRC2 dataset [78]. I split the training data into 70% and 30% portions for training and validation sets. I used ResNet-18 network [67] as the backbone of *triplet* network. Using the validation set, early stopping was employed, and the maximum number of epochs was set to 50. The batch size was 45, where every batch contains five instances per class (i.e., $n' = 5$). The learning rate was set to 10^{-5}, and the dimensionality of the embedding space was 128.

5.1.5.2 Visualization of Embedding Spaces

The 2D visualization of spaces was performed using UMAP [98] applied to the embedded data. Figure 5.5 illustrates the embedding of test sets of the CRC2 data using the BUT and BUNCA sampling methods. The obtained embedding spaces for the histopathology data are meaningful. The histopathology patches with similar patterns have been embedded close to each other as expected. In embedding using the BUT approach (see Fig. 5.5-a), the patches are embedded from smoothest to roughest patterns in a circular manner. These patches, with smoothest to roughest [79] patterns, are adipose (with thin stripes of fat), mucus, smooth muscle, debris, stroma, tumor, normal, and lymphocyte (with a rough

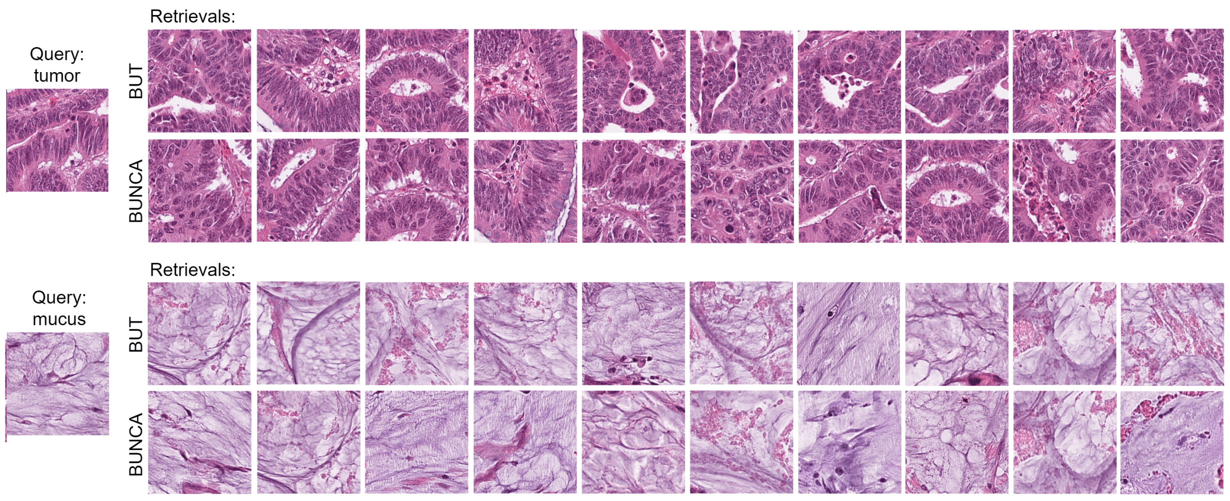

Figure 5.6: Histopathology image retrieval in the embedded spaces learned using the BUT and BUNCA approaches. The retrievals are sorted from left to right.

pattern). Moreover, the background patch with no pattern is separated from the tissues, as expected. In embedding using the BUNCA approach (see Fig. 5.5-b), the patches with a considerable amount of roughness are embedded closely. For example, adipose, mucus, stroma, and smooth muscle, which are smoother, fall close to each other while tumor, normal, lymphocyte, and debris, with diverse patterns, are embedded close to each other. Again, the background patches are embedded far from the tissue types. The meaningfulness of the learned embedded spaces shows the effectiveness of the proposed BUT and BUNCA approaches.

5.1.5.3 Retrieval of Histopathology Patches

Query retrieval can be very useful for histopathology data in hospitals where similar patches are extracted from the database to rely on already diagnosed cases. The type of disease or tissue can be found out by a majority vote amongst the retrievals [77]. Figure 5.6 shows retrievals for two different tissue types, which are tumor and mucus. The former has more complex patterns, in contrast to the latter one. As the figure shows, the retrievals are very similar to the pattern of query patch.

Table 5.4: Comparison of BUT and BUNCA approaches with the baselines on the CRC2 dataset.

	R@1	R@4	R@8	R@16
BA [19]	38.54	66.76	80.64	89.97
BSH [118]	30.85	60.39	77.73	90.33
BH [68]	79.09	92.60	96.00	97.95
EP [141]	69.94	87.88	93.20	96.38
DWS [139]	76.06	91.31	95.34	97.58
NCA [61]	77.87	92.25	95.92	98.01
PNCA [100]	78.85	92.24	95.80	97.78
BUT	79.14	92.32	95.60	97.65
BUNCA	78.67	92.28	95.64	97.71

5.1.5.4 Comparison with Baseline Methods

The results for the CRC2 histopathology data are reported in Table 5.4 and are compared to the baseline approaches, which we compared with, are BA [19], BSH [118], BH [68], EP [141], DWS [139], NCA [61], and PNCA [100]. On this data, the performance of BUNCA is closer to BUT. In most cases, BUT has the best performance against all the baseline approaches. On this dataset, BUNCA performs better than BA, BSH, EP, DWS, NCA, and is comparable with PNCA. Overall, these two tables demonstrate the effectiveness of the proposed mining approaches for triplet training.

5.2 Image Processing and Computer Vision

In this section, I propose algorithms for application of dimensionality reduction in image processing and computer vision. The proposed methods are Roweisfaces, Roweisposes, and image denoising by anomaly path inspired by CAD.

5.2.1 Face Recognition: Roweisfaces

I use the RDA subspace learning method, introduced in Chapter 3, for facial image recognition and embedding. In this section, I report the experimental result and analysis on facial images.

5.2.1.1 Rowiesfaces and Special Cases

I used the ORL face dataset [127] (see Section 6.1.1.1 for explanation of this dataset). The data were standardized to have zero mean and unit variance. I divided the dataset into two classes of images having and not having eye glasses. I trained nine special cases, $r_1, r_2 \in \{0, 0.5, 1\}$, of RDA and kernel RDA using the facial dataset. I name the facial eigenvectors, or ghost faces, in RDA as *Roweisfaces*. The existing special cases of Roweisfaces in the literature are eigenfaces ($r_1 = r_2 = 0$) [130], Fisherfaces ($r_1 = 0, r_2 = 1$) [6], and supervised eigenfaces ($r_1 = 1, r_2 = 0$) [4, 35]. For $r_1 = r_2 = 1$ in Roweisfaces, I use the name *double supervised eigenfaces*. I name facial embedding using kernel RDA as *kernel Roweisfaces* whose existing special cases are kernel eigenfaces ($r_1 = r_2 = 0$) [142], kernel Fisherfaces ($r_1 = 0, r_2 = 1$) [143], kernel supervised eigenfaces ($r_1 = 1, r_2 = 0$) [4, 35]. For $r_1 = r_2 = 1$ in kernel Roweisfaces, I use the name *kernel double supervised eigenfaces*.

The eigenvectors in kernel PCA are n-dimensional and not d-dimensional so one can show the ghost faces only in Roweisfaces and not kernel Roweisfaces. The trained Roweisfaces are shown in Fig. 5.7 for the special cases. The PCA case has captured different features such as eyes, hair, lips, nose, and face border. However, the more I consider the labels by increasing r_1 and r_2, the more features related to eyes and cheeks are extracted because of the more discrimination of having or not having glasses. Increasing r_1 tends to extract more features like Haar wavelet features which are useful for face feature detection (see Viola-Jones face detector [132]). Increasing r_2, however, fades out the irrelevant features leaving merely the eyes which are important. The double supervised eigenfaces have a mixture of Haar features and fading out unimportant features.

5.2.1.2 Projections in Rowiesfaces

The top two dimensions of projection of the facial images into the RDA and kernel RDA (with RBF kernel for $\boldsymbol{K}_x$) subspaces are shown in Fig. 5.8. As expected, kernel RDA separates the classes better that RDA. Also, the larger the supervision level, the better the separation. For the same reasons explained for Fig. 6.3, the two classes are collapsed into one-dimensional lines for $r_2 = 1$ in kernel RDA.

5.2.1.3 Reconstructions in Rowiesfaces

As explained before, data cannot be reconstructed in kernel RDA but it can be done in RDA. Some reconstructed images for the facial dataset in Roweisfaces are shown in Fig.

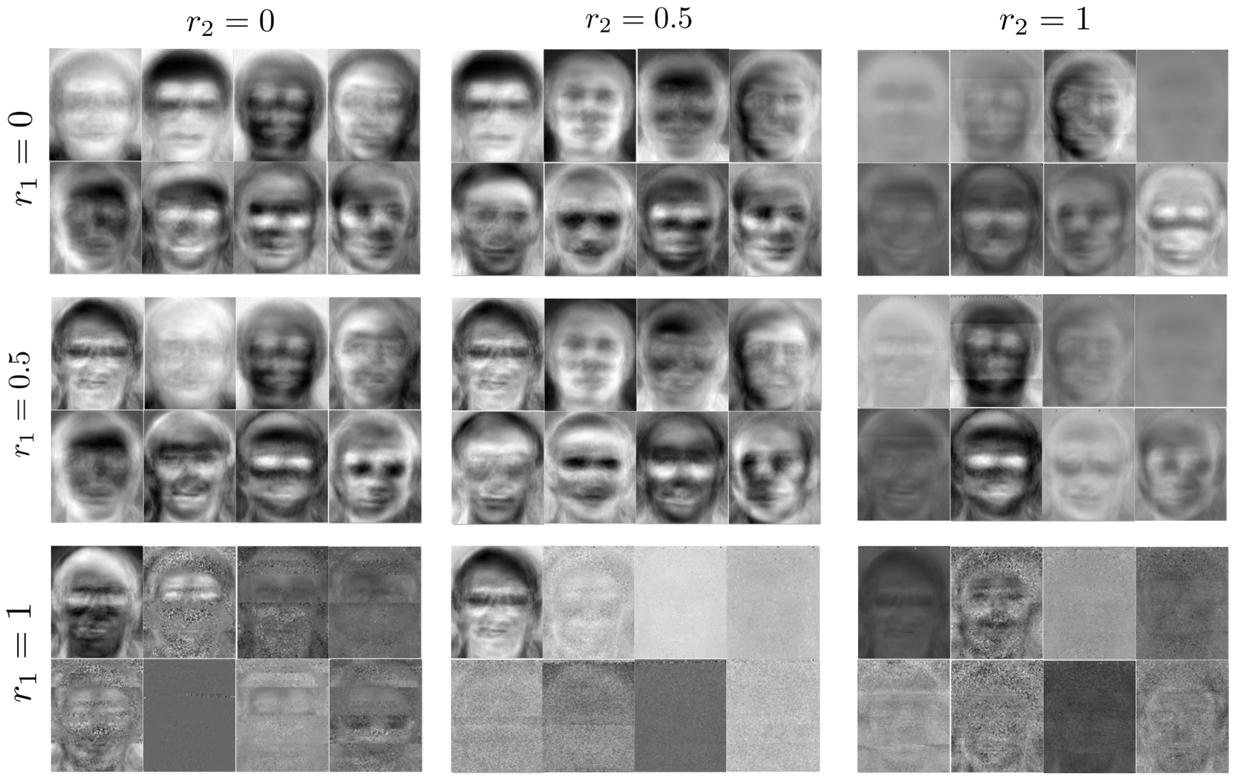

Figure 5.7: The Roweisfaces: the eight leading eigenvectors for the special cases in the Roweis map where the first, fifth, and eighth eigenvectors of every case are at the top-left, bottom-left, and bottom-right, respectively.

5.9. The quality of reconstruction falls down as r_2 is increased. I explain the reason in the following. The reconstruction error for $\boldsymbol{XA}$ is $||\boldsymbol{XA} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{XA}||_F^2$ where $\boldsymbol{A}$ is a symmetric matrix. Minimizing this error where the bases are orthonormal is:

$$\begin{aligned} \underset{\boldsymbol{U}}{\text{minimize}} \quad & ||\boldsymbol{XA} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{XA}||_F^2, \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}, \end{aligned} \tag{5.5}$$

whose Lagrangian is simplified to $\mathcal{L} = \mathbf{tr}(\boldsymbol{A}^2\boldsymbol{X}^\top\boldsymbol{X} - \boldsymbol{X}\boldsymbol{A}^2\boldsymbol{X}^\top\boldsymbol{U}\boldsymbol{U}^\top)$ noticing the constraint. Setting the derivative of Lagrangian to zero results in $\boldsymbol{X}\boldsymbol{A}^2\boldsymbol{X}^\top\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda}$ which is the eigenvalue problem for $\boldsymbol{X}\boldsymbol{A}^{\boldsymbol{2}}\boldsymbol{X}^\top$. Comparing this to the solution of Eq. (3.13) and noticing Eq. (3.14) shows we can have $\boldsymbol{A}^2 = \boldsymbol{HPH}$, $r_2 = 0$, and $\boldsymbol{R}_2 = \boldsymbol{I}$ for minimization of reconstruction error. Hence, the best setting for reconstruction is to have $r_2 = 0$. In addition, if $r_1 = 0$, the objective in Eq. (5.5) becomes the error between $\breve{\boldsymbol{X}}$ and $\widehat{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{U}^\top\breve{\boldsymbol{X}}$

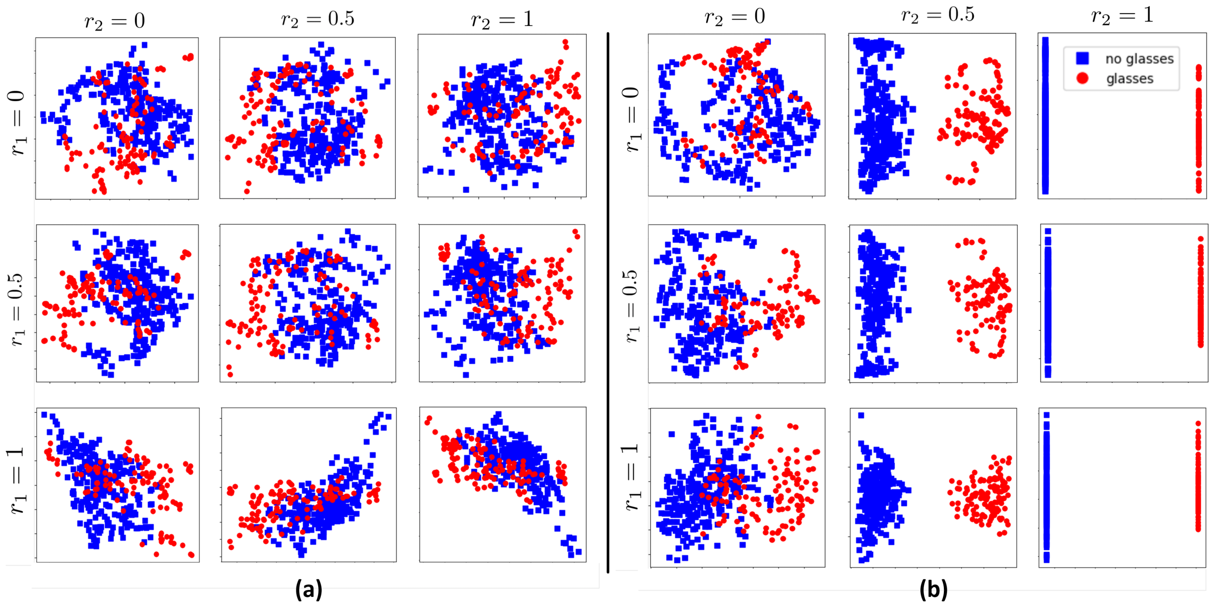

Figure 5.8: The first two dimensions of the projected data in (a) Roweisfaces and (b) kernel Roweisfaces.

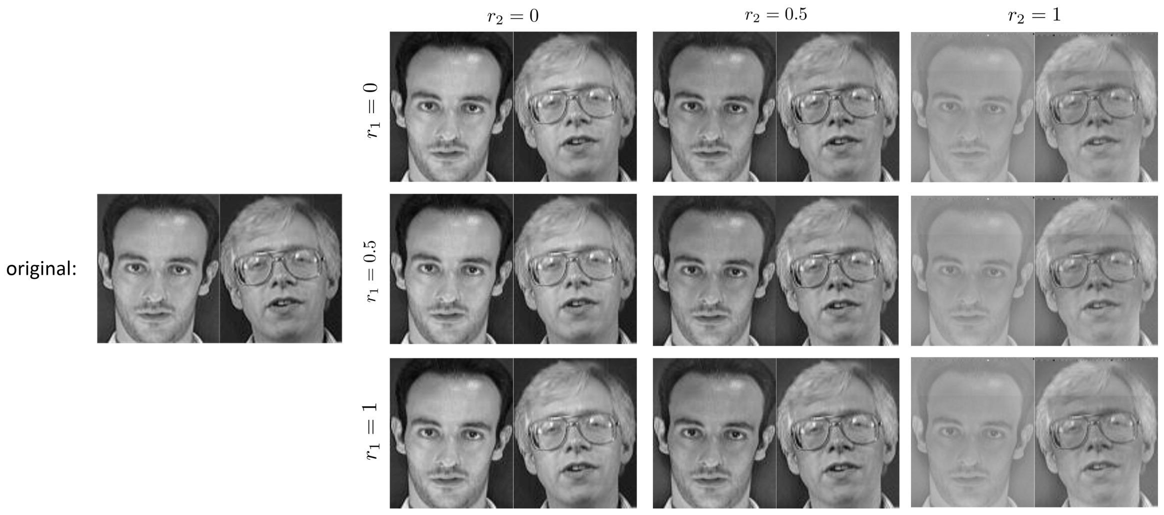

Figure 5.9: The reconstructed images after projection into the RDA subspaces.

which is the reconstruction error of centered data. This explains why PCA (with $r_2 = 0$) is the best linear method for reconstruction.

5.2.2 Action Recognition: Roweisposes

I use the RDA subspace learning method, introduced in Chapter 3, for 3D action recognition. In this section, I report the experimental result and analysis on action recognition datasets.

5.2.2.1 Review of Fisherposes

The Fisherposes method [56] is an action recognition approach which uses 3D skeletal data as input and constructs a Fisher subspace, for discrimination of body poses, using FDA. Instead of using the raw data, it applies some pre-processing on the 3D data. These pre-processing steps include skeleton alignment by translating the hip joint to the origin and aligning the shoulders to cancel the orientation of body. Moreover, the scales of skeletons are removed and some informative joints are selected amongst all the available joints.

After the pre-processing step, different body poses are selected out of the dataset where every action can be decomposed into a sequence of some of these poses. The body poses are considered as classes and the instances of a body pose are used as the data of that class. The information of joints in a body pose are concatenated to form a vector. Using these data vectors, the FDA subspace is trained to discriminate the body poses of an action recognition dataset.

Using Euclidean distance of the projected frame onto the FDA subspace from the projection of training data, the pose of a frame is recognized. Windowing is also applied to eliminate the frames which do not belong to any of the poses well enough. The distance of projection onto the subspace is used as a criterion for windowing. Finally a Hidden Markov Model (HMM) [44] is used to learn the sequences of recognized poses as different actions. For some datasets in which some actions contain similar poses without consideration of movement of body, histogram of trajectories is also used to discriminate those actions.

5.2.2.2 Roweisposes

I propose *Roweisposes* for action recognition. This method is based on basic subspace learning approaches which make use of generalized eigenvalue problem [42]. This method, which uses RDA [51], generalizes the Fisherposes method [56]. Some of the special cases of Roweisposes are based on PCA, SPCA, and DSDA and I name them *eigenposes*, *supervised eigenposes*, and *double supervised eigenposes*, respectively. The Roweisposes method includes infinite number of subspace learning methods for embedding the body poses useful for action recognition.

5.2.2.3 Datasets

For validating the effectiveness of the proposed Roweisposes, I used three publicly available datasets following the paper [56]. In the following, I introduce the characeristics of these datasets.

TST Dataset: The first dataset is the TST fall detection [29] which includes two categories of normal and fall actions. The normal actions are sitting, grasping, walking, and lying down, and the fall actions are falling front, back, side, and falling backward which ends up sitting. The number of subjects performing the actions are 11.

UTKinect Dataset: The UTKinect dataset [140] contains 10 actions which are walking, sitting down, standing up, picking up, carrying, throwing, pushing, pulling, waving, and clapping hands. This dataset has 10 subjects performing these actions.

UCFKinect Dataset: The UCFKinect dataset [22] includes 16 actions which are balancing, climbing ladder, ducking, hopping, kicking, leaping, punching, running, stepping back, stepping front, stepping left, stepping right, twisting left, twisting right, and vaulting. The number of subjects in this dataset is 16.

5.2.2.4 Performance of Roweisposes

The average of accuracies over the cross validation folds are reported in Table 5.5. The first part of table reports the related work and the state-of-the-art performances on the TST, UTKinect, and UCFKinect datasets. The second part of table contains the performances of the four extreme cases of Roweisposes which are eigenposes (with $r_1 = 0, r_2 = 0$), Fisherposes (with $r_1 = 0, r_2 = 1$), supervised eigenposes (with $r_1 = 1, r_2 = 0$), and double supervised eigenposes (with $r_1 = 1, r_2 = 1$). The third part of table reports the performance of some of the middle special cases of Roweisposes in the Roweis map. These cases are $(r_1 = 0, r_2 = 0.5)$, $(r_1 = 1, r_2 = 0.5)$, $(r_1 = 0.5, r_2 = 0)$, $(r_1 = 0.5, r_2 = 1)$, and $(r_1 = 0.5, r_2 = 0.5)$.

Regarding comparison of the special cases of Roweisposes against each other, it is seen that, except some exceptions, higher supervision level mostly results in better performance. In other words, usually, Fisherposes and supervised eigenposes have superior or comparable performance than eigenposes. This is especially true for the UTKinect dataset while in the other datasets, the performances are comparable. This is expected because more level of supervision make more use of the labels and thus improves the recognition by learning a more discriminative subspace for poses. The performance of double supervised eigenposes is not necessarily much better than the other cases. This fact has been shown to be also

Table 5.5: Comparison of the special cases of Roweisposes with other action recognition methods. The rates are average accuracy.

	TST	UTKinect	UCFKinect
Roweisposes ($r_1 = 0, r_2 = 0$)	81.44%	38.50%	87.19%
Roweisposes ($r_1 = 0, r_2 = 1$) [56]	76.14%	82.50%	79.22%
Roweisposes ($r_1 = 1, r_2 = 0$)	82.20%	70.50%	86.80%
Roweisposes ($r_1 = 1, r_2 = 1$)	76.52%	79.00%	71.72%
Roweisposes ($r_1 = 0, r_2 = 0.5$)	79.17%	83.50%	80.02%
Roweisposes ($r_1 = 1, r_2 = 0.5$)	80.68%	82.50%	86.25%
Roweisposes ($r_1 = 0.5, r_2 = 0$)	79.92%	41.00%	88.36%
Roweisposes ($r_1 = 0.5, r_2 = 1$)	80.30%	82.50%	69.45%
Roweisposes ($r_1 = 0.5, r_2 = 0.5$)	81.82%	80.50%	86.25%

true for facial image and the MNIST dataset in the paper [51]. That paper shows that only in some regression problems, which is not the case study of this paper, DSDA outperforms other cases.

The performance of the cases $r_1 = 0, r_2 = 1$ is slightly different than what is reported for Fisherposes in paper [56]. This has two small reasons. The first reason is that FDA as a special case of RDA uses the optimization problem (2.5) but the Fisherposes method in paper [56] uses optimization with the between scatter. The second reason is that this paper uses Euclidean distance for simplicity of the method and elimination of hyperparameters. The paper [56] uses a regularized Mahalanobis distance instead. I defer trying the Roweisposes method with Mahalanobis distance to the future work. The middle special cases of Roweisposes also show that the performance of this method is almost stable in most cases of Roweisposes in the Roweis map.

5.2.3 Image Denoising by Anomaly Path

I use the CAD anomaly detection method, introduced in Chapter 4, for image and video-frame denoising. In this section, I propose anomaly path using CAD and report the experimental result and analysis on some video images/frames.

5.2.3.1 Anomaly Landscape and Anomaly Paths

I define *anomaly landscape* to be the landscape in the input space whose value at every point $\boldsymbol{x}_i$ in the space is the anomaly score computed by Eq. (4.24) or (4.25). The point $\boldsymbol{x}_i$ in the space can be either the training or out-of-sample point but the kNN is obtained from the training data. We can have two types of anomaly landscape where all the training data points or merely the non-anomaly training points are used for kNN. In the latter type, the training phase of CAD or K-CAD are performed before calculating the anomaly landscape for the whole input space.

I also define the *anomaly path* as the path that an anomalous point has traversed from its not-known-yet normal version to become anomalous. Conversely, it is the path that an anomalous point should traverse to become normal. In other words, *an anomaly path can be used to make a normal sample anomalous or vice-versa.* At every point on the path, I calculate the kNN again because the neighbors may change slightly during the path. For anomaly path, I use the second type of anomaly landscape where the path is like going up/down the mountains in this landscape. For finding the anomaly path for every anomaly point, I use gradient descent where the gradient of the Eq. (4.24) is used:

$$\frac{\partial s_A(\boldsymbol{x}_i)}{\partial \boldsymbol{x}_i} = \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \Bigg[\frac{1}{||\breve{\boldsymbol{x}}_a||_2 ||\breve{\boldsymbol{x}}_b||_2} \Big[- (\breve{\boldsymbol{x}}_a + \breve{\boldsymbol{x}}_b) + \breve{\boldsymbol{x}}_a^\top \breve{\boldsymbol{x}}_b \big(\frac{\breve{\boldsymbol{x}}_a}{||\breve{\boldsymbol{x}}_a||_2^2} + \frac{\breve{\boldsymbol{x}}_b}{||\breve{\boldsymbol{x}}_b||_2^2} \big) \Big] \Bigg]. \qquad (5.6)$$

See [49] for derivation. The anomaly path can be computed in CAD and not K-CAD because the gradient in K-CAD cannot be computed analytically. The anomaly path can have many applications one of which is image denoising as explained in our experiments.

5.2.3.2 Visualization of Anomaly Landscape

For several synthetic datasets, I show the anomaly landscape and anomaly paths for CAD in Fig. 6.28. The K-CAD does not have anomaly paths as mentioned before. The landscapes in this figure are of the second type and the paths are shown by red traces which simulate climbing down the mountains in the landscape.

5.2.3.3 Image Denoising

One of the applications for anomaly path is image denoising where several similar reference images exist; for example, in video where neighbor frames exist for a frame. For experiment, I used the first 100 frames of Frey face dataset. I selected one of the frames and applied different types of noises, i.e., Gaussian noise, Gaussian blurring, salt & pepper impulse noise, and JPEG blocking to it all with the same mean squared errors (MSE $= 625$). To make the experiment more difficult, I removed the non-distorted frame from dataset. Figure 5.10 shows the iterations of denoising for different noise types where $k = 3$ is used. The

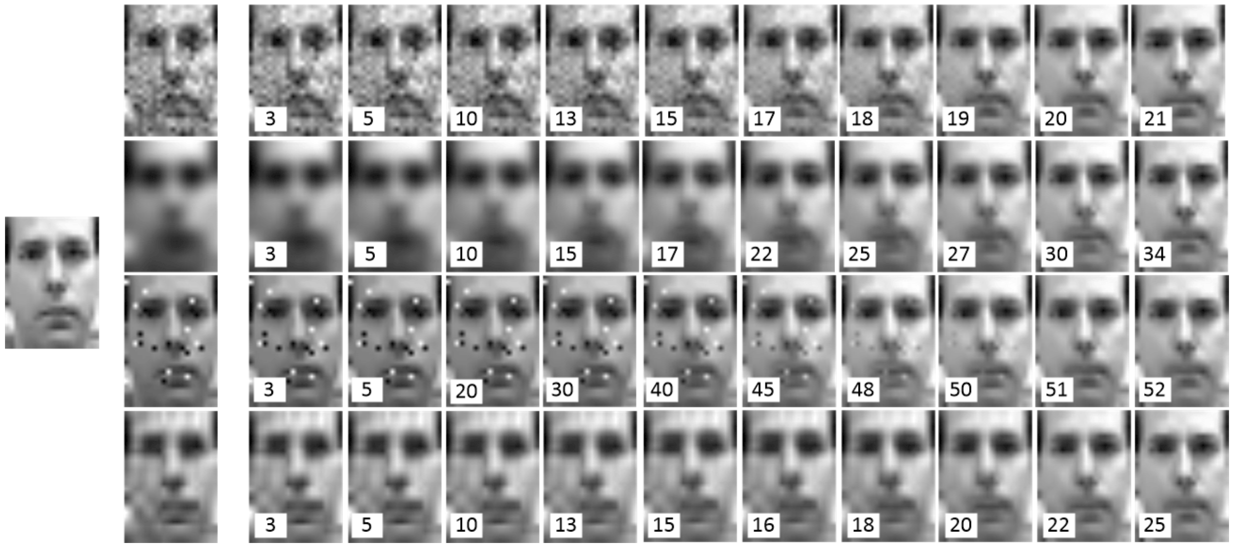

Figure 5.10: Image denoising using anomaly paths: the most left image is the original image and the first to fourth rows are for Gaussian noise, Gaussian blurring, salt & pepper impulse noise, and JPEG blocking. The numbers are the iteration indices.

distorted images were considered as anomalous instances. The neighbor frames were found for the distorted image and the anomaly path was used to gradually convert the distorted image to a normal image using its neighbors. As this figure shows, every distorted image traverses the anomaly path and gradually becomes more and more normal until it converges to the normal version of image. These experiments show that the proposed anomaly path can be used for image denoising especially in video frames where the neighbor images exist for every frame. Hence, one can use the proposed image denoising method for noise filtering in videos. I showed by this experiment that the proposed video image denoising can remove various distortions and noises such as Gaussian noise, Gaussian blurring, salt & pepper impulse noise, and JPEG blocking.

5.3 Summary of the Chapter

In this chapter, I proposed some applications for data reduction and reported their simulation results. I proposed different applications of data reduction including medical image analysis, image processing, and computer vision. In medical image analysis, I focused on histopathology data where Fisher loss, triplet mining based on extreme distances, and

BUT and BUNCA triplet mining approaches were proposed for histopathology. In image processing and computer vision, I proposed Roweisfaces for face recognition, Roweisposes for 3D action recognition, and image denoising by anomaly path.

Note that the proposed data reduction methods are very useful for gigapixel images such as histopathology because they have huge dimensionality due to the number of pixels and huge numerosity as they need to be divided into many patches. Moreover, note that ghost faces are very important especially in facial recognition and computer vision. I proposed Roweisfaces as ghost faces in RDA for facial recognition. The proposed anomaly path can also be used for converting anomalous data instance to normal or vice-versa. It can also be used for video image denoising which is a very important topic in image and video processing.

Chapter 6

Experiments and Analysis

This chapter reports the experimental results on the proposed algorithms in data reduction, excluding the results on applications already reported in Chapter 5. It reports the simulation results on WFDA (Section 6.1.1.1), RDA (Section 6.1.1.2), SSIM kernel (Section 6.1.1.3), ISCA (Section 6.1.1.4), LLISE (Section 6.1.1.5), QQE (Section 6.1.2.1), back-projection (Section 6.1.3.1), Fisher losses (Section 6.1.3.2), and BUT/BUNCA (Section 6.1.3.3) in dimensionality reduction. In numerosity reduction, it also reports the results on PSA (Section 6.2.1.1), IRMD (Section 6.2.1.2), CAD (Section 6.2.2.1), and iMondrian (Section 6.2.3).

6.1 Experiments for Dimensionality Reduction

In this section, I report the experiments for the proposed algorithms in spectral, probabilistic, and neural network-based dimensionality reduction.

6.1.1 Experiments for Spectral Dimensionality Reduction

In this section, I report the experiments for the proposed algorithms in spectral dimensionality reduction including WFDA, RDA, and image quality aware embedding (SSIM kernel, ISCA, and LLISE).

Table 6.1: Experiments for WFDA: Accuracy of 1NN classification for different obtained subspaces. In each cell of input or feature spaces, the first and second rows correspond to the classification accuracy of training and test data, respectively.

	FDA	APAC	POW	CDM	kNN ($k = 1$)	kNN ($k = 3$)	kNN ($k = c-1$)	CW-FDA version 1	CW-FDA version 2	AW-FDA ($k = 1$)	AW-FDA ($k = 3$)	AW-FDA ($k = c-1$)
Input	97.01%	97.01%	97.01%	74.62%	97.76%	97.76%	97.01%	97.01%	–	97.76%	97.01%	96.26%
space	92.42%	93.93%	96.96%	45.45%	96.96%	98.48%	92.42%	92.42%	–	87.87%	93.93%	93.93%
Feature	97.01%	97.01%	97.01%	91.79%	95.52%	97.76%	97.01%	97.01%	97.01%	100%	100%	100%
space	83.33%	86.36%	89.39%	77.27%	80.30%	83.33%	83.33%	84.84%	87.87%	100%	100%	100%

6.1.1.1 Experiments for Weighted FDA

Dataset: For experiments, I used the public ORL face recognition dataset [127]. This dataset includes 40 classes, each having ten different poses of the facial picture of a subject, resulting in 400 total images. For computational reasons, I selected the first 20 classes and resampled the images to 44×36 pixels. Please note that massive datasets are not feasible for the KFDA/FDA because of having generalized eigenvalue problem. The data were split into training and test sets with 66%/33% portions and were standardized to have mean zero and variance one.

Evaluation of the Embedding Subspaces: For the evaluation of the embedding subspaces, I used the 1-Nearest Neighbor (1NN) classifier because it is useful to evaluate the subspace by the closeness of the projected data samples. The training and out-of-sample (test) accuracy of classifications are reported in Table 6.1. In the input space, kNN with $k = 1, 3$ have the best results but in $k = c - 1$, AW-FDA outperforms it in generalization (test) result. The performances of CW-FDA and AW-FDA with $k = 1, 3$ are promising, although not the best. For instance, AW-FDA with $k = 1$ outperforms weighted FDA with APAC, POW, and CDM methods in the training embedding, while it has the same performance as kNN. In most cases, AW-FDA with all k values has better performance than FDA, which shows the effectiveness of the obtained weights compared to the equal weights in FDA. Also, the sparse k in AW-FDA outperforming FDA (with dense weights equal to one) validates the betting on sparsity.

Comparison of Fisherfaces: In the feature space, where we used the radial basis kernel, AW-KFDA has the best performance with entirely accurate recognition. Both versions of CW-KFDA outperform regular KFDA and KFDA with CDM, and kNN (with $k = 1, c - 1$) weighting. They also have better generalization than APAC, kNN with all k values. Overall, the results show the effectiveness of the proposed weights in the input and feature spaces. Moreover, the existing weighting methods, which were for the

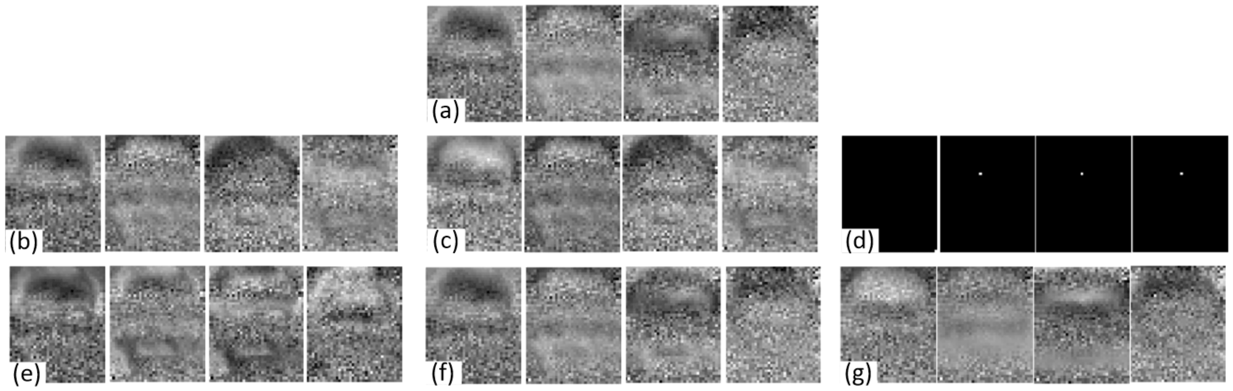

Figure 6.1: Experiments for WFDA: the leading Fisherfaces in (a) FDA, (b) APAC, (c) POW, (d) CDM, (e) kNN, (f) CW-FDA, and (g) AW-FDA.

input space, have outstanding performance when used in our proposed weighted KFDA (in feature space). This shows the validness of the proposed weighted KFDA even for the existing weighting methods.

Figure 6.1 depicts the four leading eigenvectors obtained from the different methods, including the FDA itself. These ghost faces, or so-called Fisherfaces [6], capture the critical discriminating facial features to discriminant the classes in subspace. Note that Fisherfaces cannot be shown in kernel FDA as its projection directions are n dimensional. CDM has captured some pixels as features because its all weights have become zero for its possible flaw which is perfect classification (see Fig. 6.2). The Fisherfaces, in most of the methods including CW-FDA, capture information of facial organs such as hair, forehead, eyes, chin, and mouth. The features of AW-FDA are more akin to the Haar wavelet features, which are useful for facial feature detection [132].

Comparison of the Weights: I show the obtained weights in different methods in Fig. 6.2. The weights of APAC and POW are too small, while the range of weights in the other methods is more reasonable. The weights of CDM have become all zero because the samples were purely classified (recall the flaw of CDM). The weights of kNN method are only zero and one, which is a flaw of this method because, amongst the neighbors, some classes are closer. This issue does not exist in AW-FDA with different k values. Moreover, although not all the obtained weights are visually interpretable, some non-zero weights in AW-FDA or AW-KFDA, with e.g. $k = 1$, show the meaningfulness of the obtained weights noticing the similarity of some facial classes. For example, the non-zero pairs $(2, 20)$, $(4, 14)$, $(13, 6)$, $(19, 20)$, $(17, 6)$ in AW-FDA and the pairs $(2, 20)$, $(4, 14)$, $(19, 20)$, $(17, 14)$ in AW-KFDA make sense visually because of having eyeglasses so their classes are

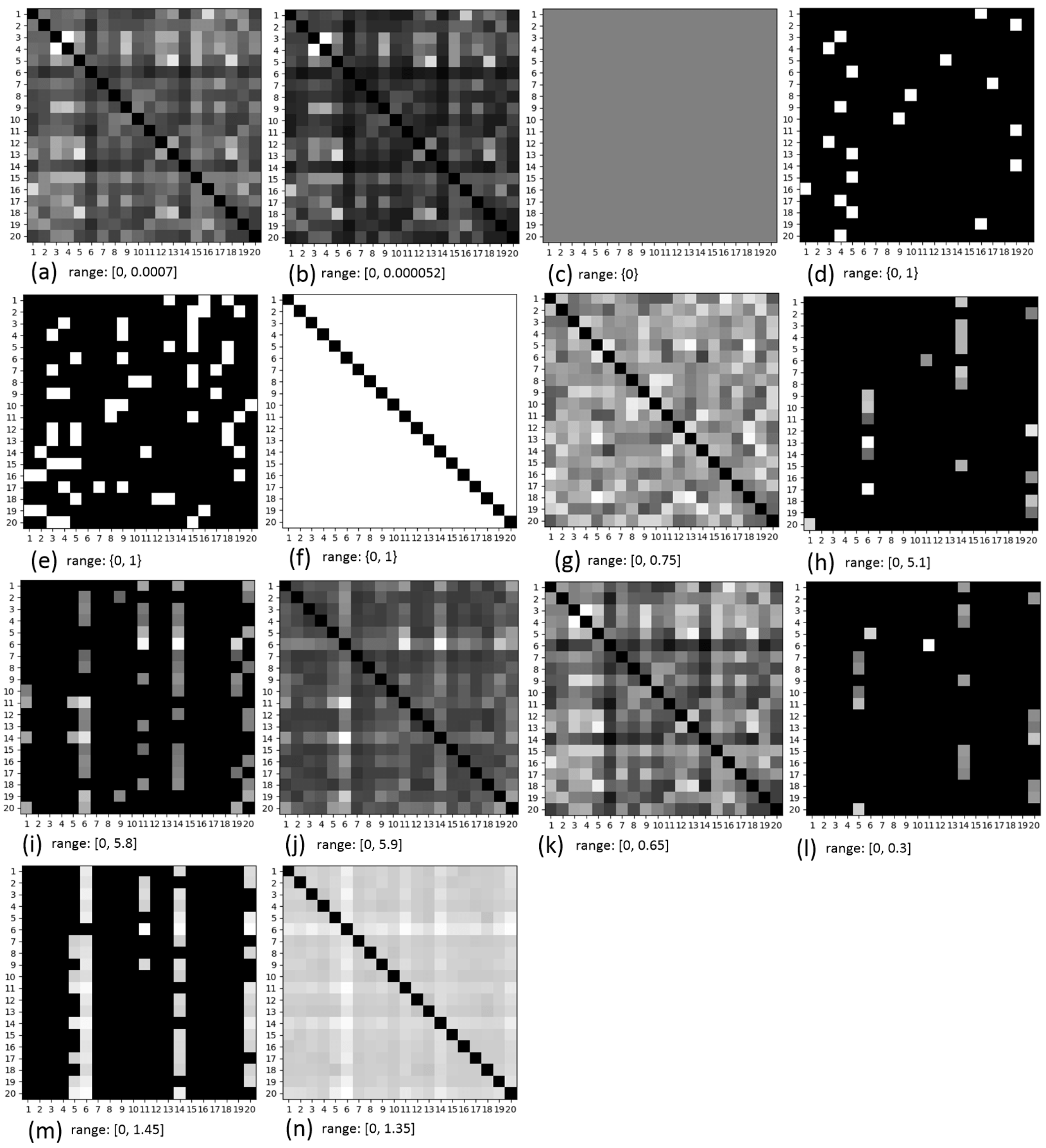

Figure 6.2: Experiments for WFDA: the weights in (a) APAC, (b) POW, (c) CDM, (d) kNN with $k = 1$, (e) kNN with $k = 3$, (f) kNN with $k = c-1$, (g) CW-FDA, (h) AW-FDA with $k = 1$, (i) AW-FDA with $k = 3$, (j) AW-FDA with $k = c - 1$, (k) CW-KFDA, (l) AW-KFDA with $k = 1$, (m) AW-KFDA with $k = 3$, (n) AW-KFDA with $k = c - 1$. The rows and columns index the classes.

close to one another.

6.1.1.2 Experiments for RDA

Visualization for Synthetic Nonlinear Datasets: One of the applications of subspace learning is data visualization. For the first experiment, I created two synthetic datasets which are highly nonlinear. The two datasets are binary XOR and concentric rings having two dimensions and two classes. The sample size in every dataset is 400 where 70% and 30% of the data are used for training and out-of-sample (test), respectively. The datasets are shown in Fig. 6.3.

As these datasets are highly nonlinear, RDA does not separate the classes as well as kernel RDA. Therefore, the first two dimensions of the embedding in nine special cases of kernel RDA (with RBF kernel for $\boldsymbol{K}_x$) are illustrated in Fig. 6.3. Note that I used the Kronecker delta kernel for $\boldsymbol{K}_y$ inspired by [4]. As this figure shows, kernel PCA does not perform as well as kernel SPCA, kernel FDA, and kernel DSDA. Overall, the larger the Roweis factors get, the better the two classes are separated which is expected because the supervision level is increased. By sweeping $r_2 = 0 \rightarrow 1$, the two classes are almost collapsed into two one dimensional lines because the rank of kernel RDA is restricted by $c-1 = 1$ when $r_2 = 1$ but this restriction does not exist for $r_2 = 0$. Another interpretation is because of taking the within scatter into account when r_2 is closer to one so the classes are collapsed. Moreover, these figures show that RDA is capable of handling out-of-sample data well enough.

RDA for Classification and Regression: For evaluating RDA on classification and regression, I evaluated nine special cases of RDA on the MNIST [84] and some regression benchmarks [4], respectively. For the sake of brevity, I do not report the results in this thesis. The reader can refer to [48] and [51] to see the results. The results showed that RDA and kernel RDA mostly perform better in classification for larger supervision levels, as expected. In regression, the results showed that DSDA works better than the other special cases because of having larger supervision level.

6.1.1.3 Experiments for SSIM Kernel

Dataset: I made a dataset out of the standard *Lena* image. Six different types of distortions were applied on the original *Lena* image (see Fig. 6.4), each of which had 20 images in the dataset with different MSE values. Therefore, the size of training set is 121 including the original image. The six used distortions are stretching contrast, Gaussian

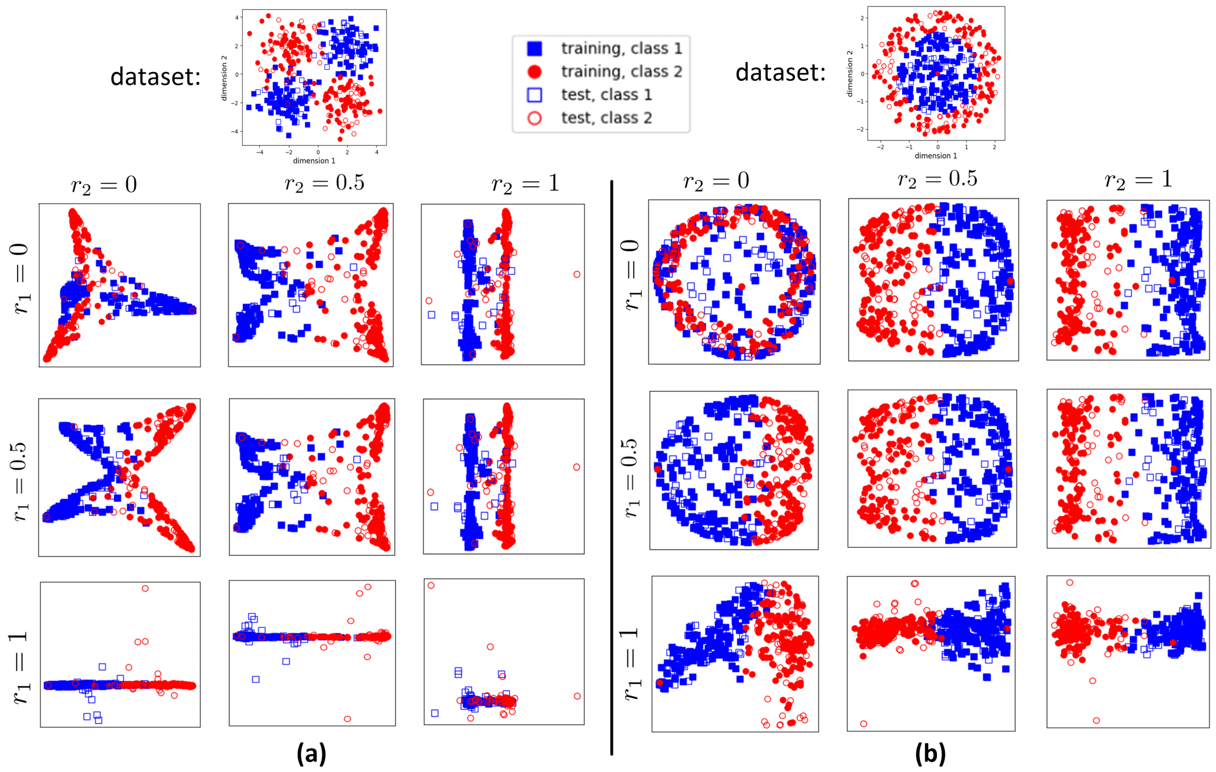

Figure 6.3: Visualization using RDA: the first two dimensions of the projected data in kernel RDA for (a) binary XOR dataset and (b) concentric rings dataset.

noise, enhancing luminance, Gaussian blurring, salt & pepper impulse noise, and JPEG distortion. For every type of distortion, 20 different levels of MSE, i.e., from MSE = 45 to MSE = 900 with step 45, were generated in order to have images on the equal-MSE or *iso-error* hypersphere [133]. This dataset is also used for the ISCA and LLISE experiments.

Comparison of Kernels: I experimented with the basic subspace learning methods using different kernels including my proposed SSIM kernel. The baseline kernels are Radial Basis Function (RBF) $\exp(-\gamma\,||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2^2)$, linear kernel $\boldsymbol{x}_1^\top \boldsymbol{x}_2$, polynomial $(\gamma\,\boldsymbol{x}_1^\top \boldsymbol{x}_2 + 1)^3$, sigmoid $\tanh(\gamma\,\boldsymbol{x}_1^\top \boldsymbol{x}_2 + 1)$, cosine $\boldsymbol{x}_1^\top \boldsymbol{x}_2 / (||\boldsymbol{x}_1||_2 ||\boldsymbol{x}_2||_2)$, and geodesic kernel (Eq. (3.26) with geodesic distance matrix), where $\gamma := 1/d$. Figures 6.5 and 6.6 show the subspaces obtained from experiments using different kernels.

As can be seen in Fig. 6.5, the image structure subspace propagates the different types of distortions out from the non-distorted original image while the distortions mostly exist on separate trajectories. The more distorted an image is, the further from the original image it

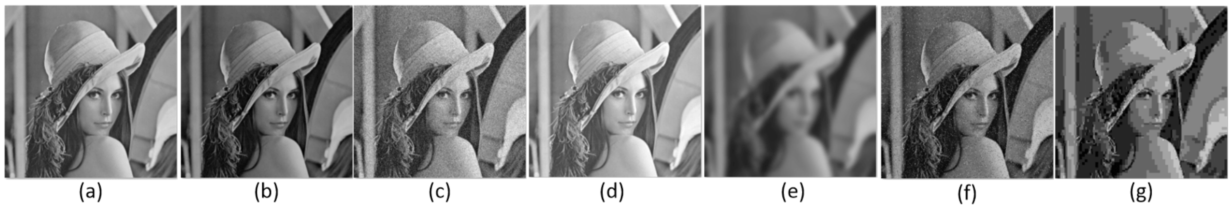

Figure 6.4: Examples from the training dataset: (a) original image, (b) contrast stretched, (c) Gaussian noise, (d) luminance enhanced, (e) Gaussian blurring, (f) salt & pepper impulse noise, and (g) JPEG distortion.

is projected or embedded. The subspace obtained using the SSIM kernel discriminates the different distortions much more properly compared to other kernels. In kernel PCA using SSIM, the first dimension puts the luminance enhancement and contrast stretching (non-structural distortions) apart from the structural distortions. Moreover, the third dimension separates the Gaussian blurring as a structural distortion. The fourth dimension shows that the distortions have tilted around this direction in the image distortion subspace. In linear and polynomial kernels, Gaussian noise is not separated from the original image. Also, contrast stretching and impulse noise are treated similarly in a linear kernel. In a polynomial kernel, contrast stretching, impulse noise, and JPEG distortion are not properly separated. The fourth dimension in both linear and polynomial kernels are not promising discriminators. Moreover, in SSIM kernel, the distances of embedded images are almost equal which is expected because the steps of MSE levels of images were equal. This is while most of the other kernels could not preserve the equal distances of images in the embedding space.

According to Fig. 6.5, the results of the SSIM kernel and RBF kernel look very similar, but not identical, especially in the first dimensions. We provide the reason here: Let $d := ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2$ and $r = d^2$. The Taylor series expansion of RBF kernel is:

$$\exp(-\gamma r) \approx 1 - \gamma r + \frac{\gamma^2}{2} r^2 - \frac{\gamma^3}{6} r^3 + \cdots \tag{6.1}$$

On the other hand, the SSIM kernel is $\boldsymbol{S}_x \propto -\boldsymbol{D}$ (Eq. (3.28)). Every element of $\boldsymbol{D}$ is based on $d(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in Eq. (3.27). We have $d(\boldsymbol{x}_1, \boldsymbol{x}_2) \propto d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$. Therefore, $\boldsymbol{S}_x \propto -d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = -\sqrt{r}$ where $r = (d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2))^2$. By Taylor series expansion, we have:

$$\boldsymbol{S}_x \propto -\sqrt{r} \approx -\frac{5}{16} - \frac{15}{16} r + \frac{5}{16} r^2 - \frac{1}{16} r^3 + \cdots \tag{6.2}$$

Comparing Eqs. (6.1) and (6.2) gives the hint for why SSIM and RBF kernels had similar

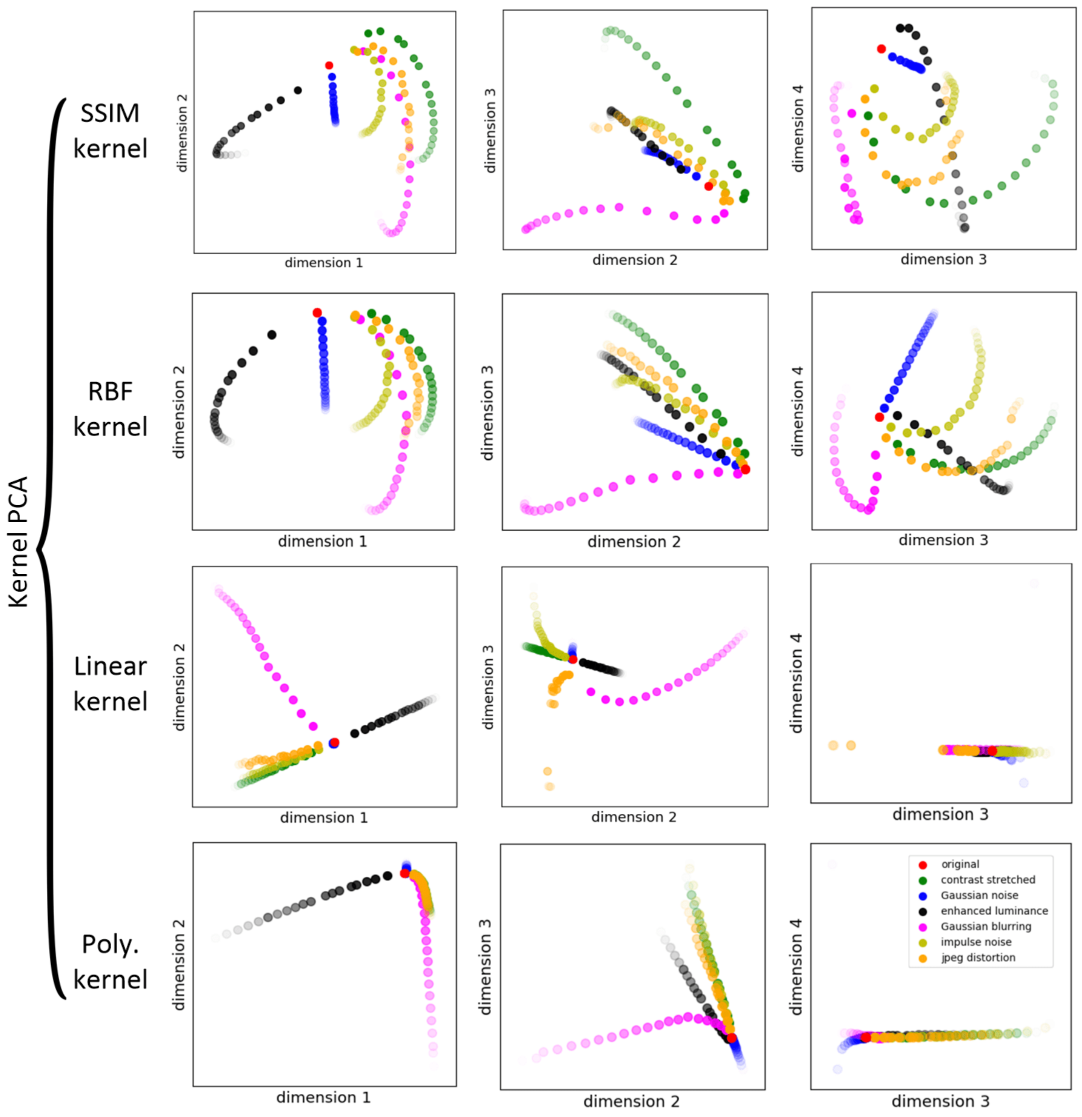

Figure 6.5: Image structure subspaces: The first to fourth rows correspond to SSIM kernel, RBF kernel, linear kernel (or dual PCA or MDS [16]), and polynomial kernel in kernel PCA, respectively. More transparent points correspond to more distorted images.

results. Note that the r in the RBF kernel is based on Euclidean distance while the r in SSIM kernel is based on SSIM distance.

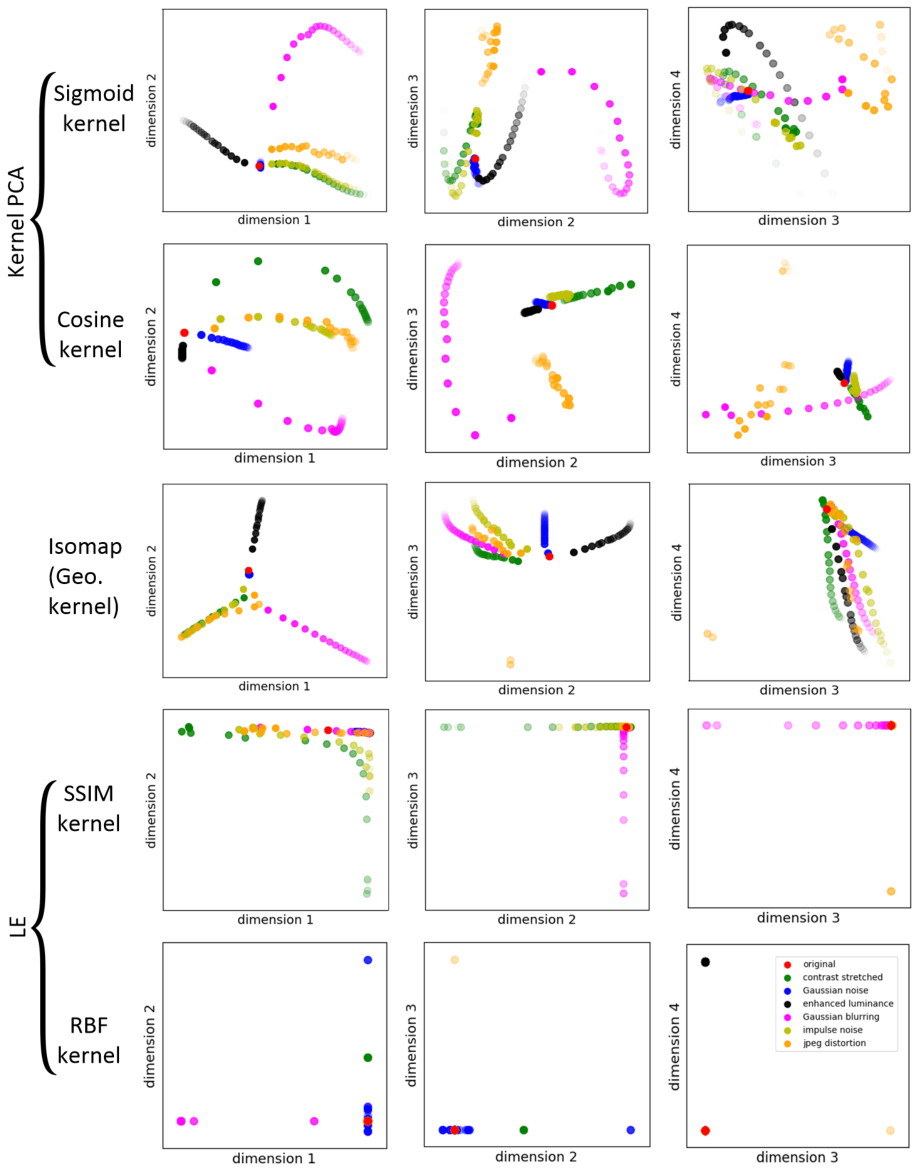

Figure 6.6: Image structure subspaces: The first and second rows correspond to sigmoid kernel and cosine kernel in kernel PCA, respectively. The third row is for geodesic kernel (Isomap) [126]. The fourth and fifth rows correspond to LE using SSIM kernel and RBF kernel [7], respectively. More transparent points correspond to more distorted images.

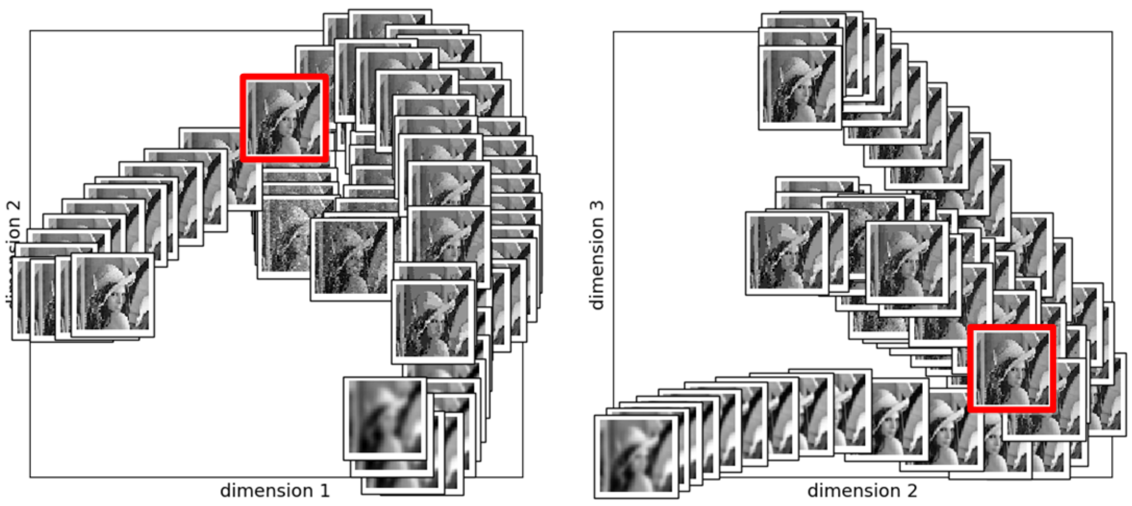

Figure 6.7: The projected images onto the subspace of kernel PCA using SSIM kernel. The image with a thick red frame is the original image.

It is also noteworthy that the proposed SSIM kernel is a universal kernel. Paper [124] has shown that according to the Stone-Weierstrass theorem, the universal kernels can be expanded in certain types of Taylor or Fourier series. The RBF kernel is an example. Eqs. (6.1) and (6.2) show that the SSIM kernel can be expanded by Taylor series similar to the RBF kernel. Hence, it is a universal kernel, so it can be used in generative models such as generative moment matching networks [88].

In Fig. 6.6, we can see that the sigmoid kernel does not properly discriminate stretching contrast and impulse noise. Moreover, the Gaussian noise is not well separated from the original image. The cosine kernel separates these two but does not preserve the almost uniform distance of different intensities of a distortion, as we had for SSIM kernel. In the first two dimensions of Isomap, on the other hand, the image with Guassian noise and the original image are not separated, and contrast stretching, impulse noise, and JPEG distortion are not discriminated. In the second, third, and fourth dimensions of Isomap, we see that very high amount of JPEG distortion is noticed while low value of JPEG distortion is not respected.

Figure 6.6 also includes the subspaces of LE using SSIM kernel and RBF kernel [7]. The LE using SSIM kernel strongly outperforms the LE using RBF kernel because its second and third dimensions discriminate contrast stretching (non-structural distortion)

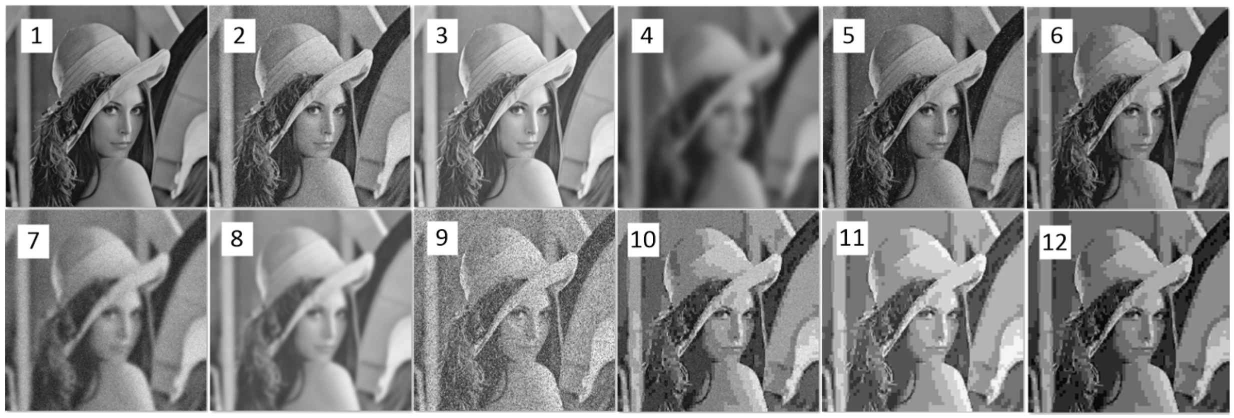

Figure 6.8: Out-of-sample images with different types of distortions having MSE = 500: (1) stretching contrast, (2) Gaussian noise, (3) luminance enhancement, (4) Gaussian blurring, (5) impulse noise, (6) JPEG distortion, (7) Gaussian blurring + Gaussian noise, (8) Gaussian blurring + luminance enhancement, (9) impulse noise + luminance enhancement, (10) JPEG distortion + Gaussian noise, (11) JPEG distortion + luminance enhancement, and (12) JPEG distortion + stretching contrast.

from Gaussian blurring (structural distortion). The first and second dimensions of LE using SSIM kernel also show much better scatter of data points in the subspace. That is while LE fails to discriminate the distortions properly. Finally, for the sake of better visualization of what the scatter plots in Figs. 6.5 and 6.6 mean, see Fig. 6.7.

Out-of-sample Projection: For out-of-sample projection using the SSIM kernel, I created 12 test images with MSE = 500 having different distortions. Figure 6.8 shows the test images where the distortions are reported in the caption of figure. Some of the test images have a combination of different distortions in order to evaluate the image structure subspace with harder out-of-sample images. This dataset is also used for the ISCA and LLISE experiments.

The projection of these out-of-sample images onto the kernel PCA subspace obtained using SSIM kernel is shown in Fig. 6.9. As expected, the images 1 to 6 which have solely one type of distortion fall close enough to the projected training samples of their distortion. Note that in dimensions 3 and 4 of projection, some points might be thought to fall onto each other but a 3D imagination shows that they are apart in the other dimension. As expected, the images having a combination of two distortions fall between the two embedded distortions. This shows that the learned embedding space is meaningful and interpretable. For example, image 7 falls between the projection of Gaussian blurring

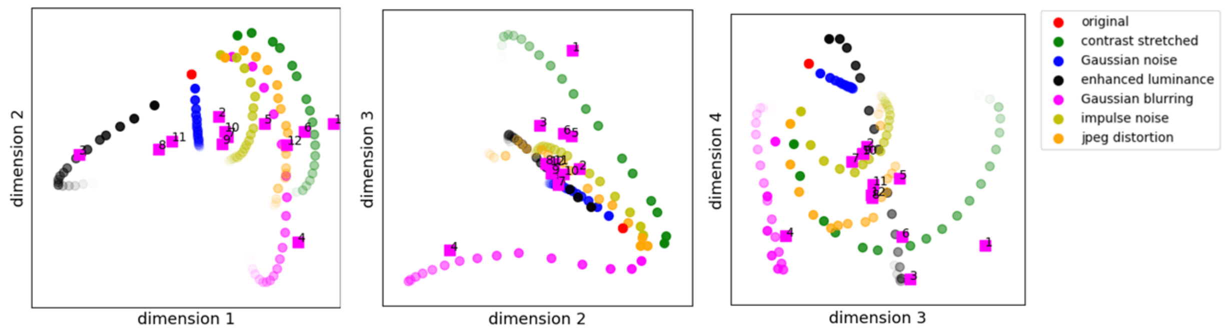

Figure 6.9: Experiments for SSIM kernel: Projection of out-of-sample images onto the image structure subspace. The pink square points are the out-of-sample images.

and Gaussian noise. Likewise, image 8 falls between projection of Gaussian blurring and luminance enhancement, image 9 falls between impulse noise and luminance enhancement, image 10 falls between JPEG distortion and Gaussian noise, image 11 falls between JPEG and luminance, and image 12 falls between JPEG and stretching contrast. Note that for example, image 12 falls closer to JPEG distortion than to contrast stretching because JPEG distortion is a structural distortion and carries more amount of image quality distortion. In conclusion, the image structure subspace also supports distortion of out-of-sample images even if they have a combination of different distortions.

Note that the proposed image structure subspace can be very useful for image processing, machine learning, and classification. It provides and opens a new field of research for further investigations. One can use metric learning in this learned subspace for image processing purposes.

6.1.1.4 Experiments for ISCA

Training: In our experiments for ISCA, the parameters used were $\rho = 1$ and $\eta = 0.1$, and for kernel ISCA, we used $\rho = 0.1$ and $\eta = 0.1$. These parameters should be set small enough to have progress in optimization without oscillating behaviour. I took $q = 64$ (8×8 blocks), $p = 4$, and $d = 512 \times 512 = 262144$. One of the dimensions of the trained $\boldsymbol{U} = \cup_{i=1}^{b} \boldsymbol{U}_i$, $\boldsymbol{V} = \cup_{i=1}^{b} \boldsymbol{V}_i$, and $\boldsymbol{J} = \cup_{i=1}^{b} \boldsymbol{J}_i$ for ISCA are shown in Fig. 6.10. The dual variable $\boldsymbol{J}$ has captured the edges because edges carry much of the structure information. As expected, $\boldsymbol{U}$ and $\boldsymbol{V}$ are close (*Lena* can be seen in them by noticing scrupulously). Note that the variables in kernel ISCA are not q-dimensional and thus cannot be displayed in image form.

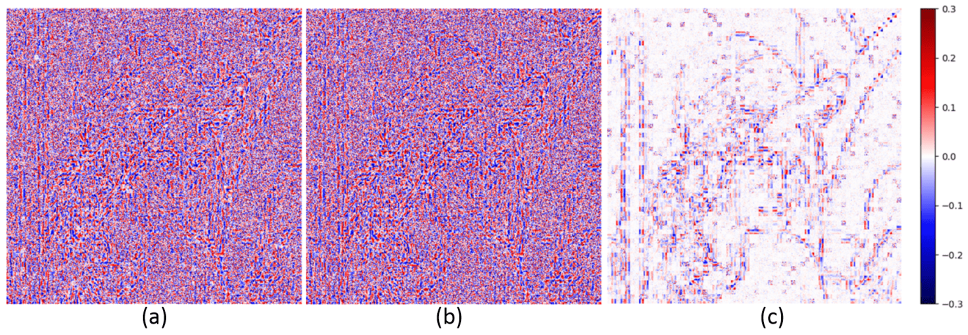
Figure 6.10: Experiments for ISCA: the first dimension of the trained (a) $\boldsymbol{U}$, (b) $\boldsymbol{V}$, and (c) $\boldsymbol{J}$ for ISCA.

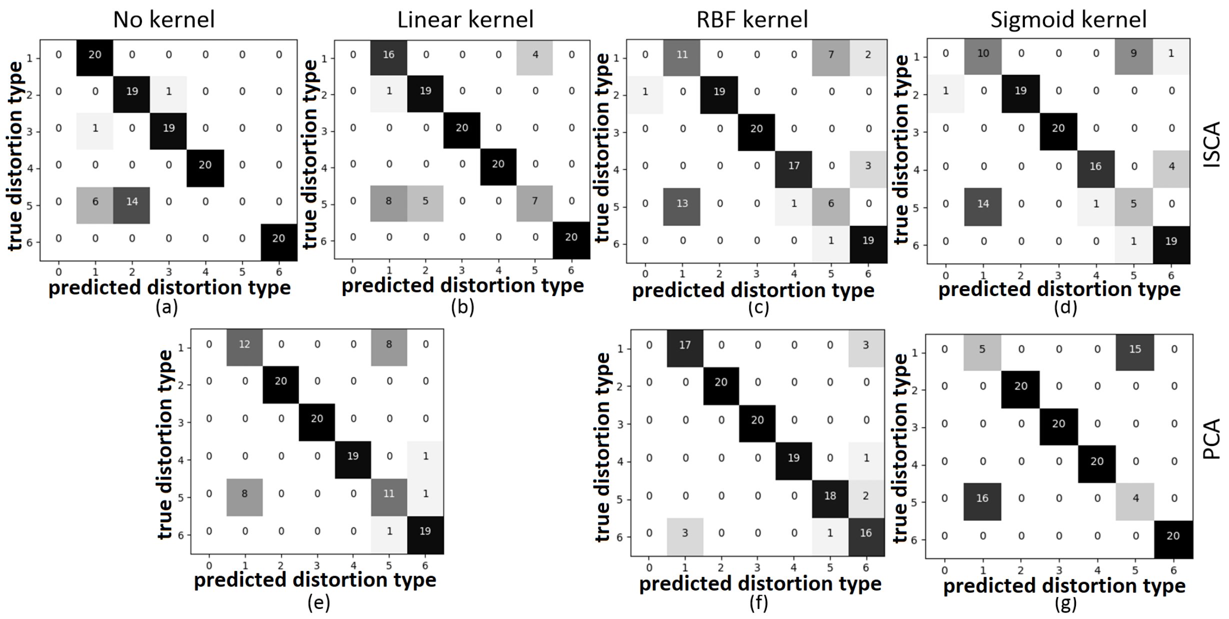

Figure 6.11: Experiments for ISCA: confusion matrices for recognition of distortion types with a 1NN classifier used in the subspace. Matrices (a) and (e) correspond to ISCA and PCA (or linear-kernel PCA), respectively. Matrices (b) to (d) are for kernel ISCA with linear, RBF, and sigmoid kernels. Matrices (f) and (g) are for kernel PCA with RBF and sigmoid kernels. The 0 label in matrices correspond to the original image and the labels 1 to 6 are the distortion types with the same order as in Fig. 6.4.

Projections and Comparisons: In order to evaluate the trained ISCA and kernel

Table 6.2: Experiments for ISCA: recognition of distortions for out-of-sample images. Letters O, C, G, L, B, I, and J correspond to original image, contrast stretch, Gaussian noise, luminance enhanced, blurring, impulse noise, and JPEG distortion, respectively.

image	1	2	3	4	5	6	7	8	9	10	11	12
distortion	C	G	L	B	I	J	B + G	B + L	I + L	J + G	J + L	J + C
ISCA	69.3% O	49.1% G	69.7% O	99.8% B	30.3% G	96.4% J	55.2% B	98.7% B	48.9% G	39.4% J	96.4% J	97.9% J
	30.2% C	27.2% I	29.6% C	0.2% J	23.8% I	3.6% B	19.9% G	1.3% J	33.3% I	32.9% B	3.6% B	2.1% B
kernel ISCA (linear)	88.1% C	59.2% G	99.8% L	95.9% B	37.4% G	80.4% J	40.8% G	93.4% B	45.6% I	38.7% G	70.2% J	74.1% J
	11.2% I	25.2% I	0.1% O	3.4% J	32.3% I	17.6% B	33.4% B	5.8% J	39.4% G	21.7% J	27.0% B	25.0% B
kernel ISCA (RBF)	72.0% C	79.1% G	99.2% L	70.6% B	39.6% I	74.3% J	44.8% G	88.1% L	48.2% L	33.1% G	82.8% L	43.1% J
	10.9% I	5.0% B	0.5% G	13.1% C	36.4% C	13.5% C	28.5% B	6.6% B	37.7% G	21.6% L	8.9% G	30.0% B
kernel ISCA (sigmoid)	80.3% C	76.3% G	99.6% L	76.2% B	38.5% I	79.3% J	47.9% G	81.7% L	52.1% L	37.9% G	80.7% L	43.9% J
	7.6% I	6.8% I	0.2% G,B	10.6% J	36.5% C	10.6% C	24.6% B	9.8% B	26.6% G	19.8% L	11.3% G	31.0% B

ISCA subspaces, I projected the training images onto these subspaces. For projecting an image, each of its blocks was projected onto the subspace of that block. After projecting all the images, I used the 1-Nearest Neighbor (1NN) classifier to recognize the distortion type of every block. The 1NN is useful to evaluate the subspace by closeness of the projected distortions. The distortion type of an image comes from a majority vote among the blocks. The linear, RBF, and sigmoid kernels were tested for kernel ISCA. The confusion matrices for distortion recognition are shown in Fig. 6.11. Mostly kernel ISCA performed better than ISCA because it works in feature space; although, ISCA performed better for some distortions like contrast stretching and blurring. Moreover, I compared with PCA and kernel PCA. PCA showed weakness in contrast stretching. RBF and sigmoid kernels in kernel PCA did not perform well for JPEG distortion and contrast stretching, respectively. As expected, ISCA and kernel ISCA have performed better than PCA and kernel PCA in JPEG distortion which is a structural distortion.

Out-of-sample Projections: For out-of-sample projection, I created 12 test images with MSE = 500 having different distortions and some having a combination of different distortions (see Fig. 6.8). I did the same 1NN classification for these images. Table 6.2 reports the top two votes of blocks for every image with the percentage of blocks voting for those distortions. ISCA did not recognize luminance enhancement well enough because, for Eq. (2.10), the block is centered while in kernel ISCA, the block is centered in feature space. Overall, both ISCA and kernel ISCA performed very compelling even in recognizing the combination of distortions. This shows that the learned subspaces are capable of handling a mixture of distortions.

Reconstruction: The images can be reconstructed after projection onto the ISCA subspace. For reconstruction, every block is reconstructed as $\boldsymbol{U}_i\boldsymbol{U}_i^\top \breve{\boldsymbol{x}}_i \in \mathbb{R}^q$ where the mean of

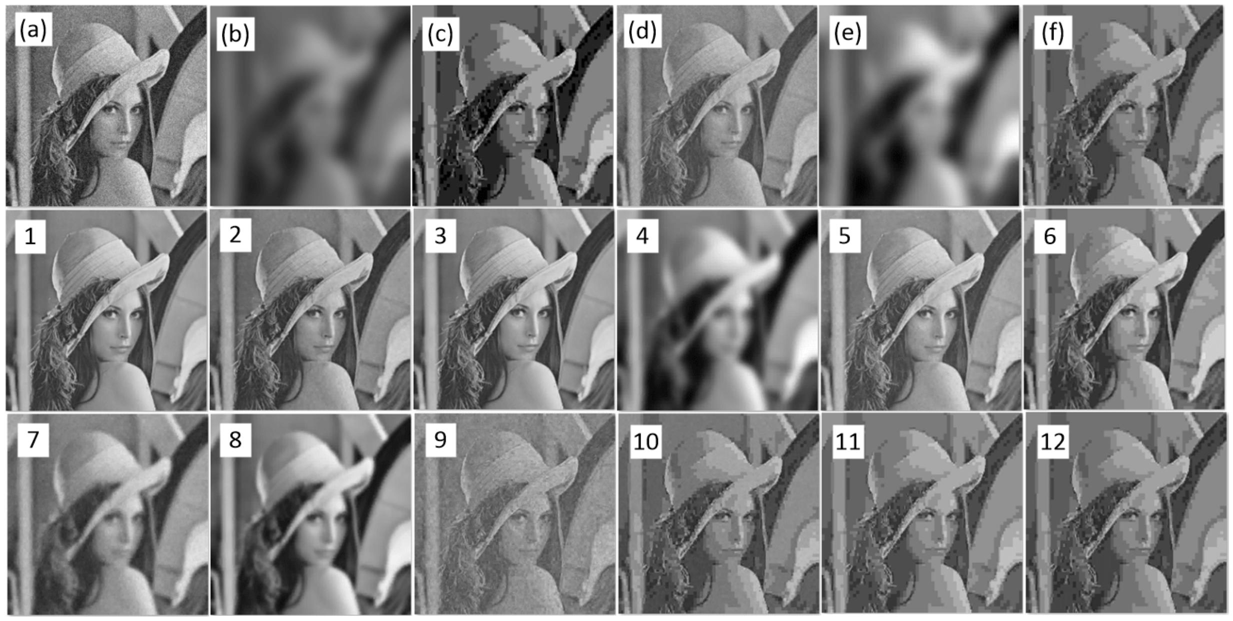

Figure 6.12: Experiments for ISCA: reconstruction of images in ISCA. Reconstruction of the training images (a), (b), and (c) are shown in (d), (e), and (f), respectively. The reconstruction of out-of-sample images shown in Fig. 6.8 are shown in the second and third rows.

block should be added to the reconstruction. Similar to kernel PCA, reconstruction cannot be done in kernel ISCA because $\boldsymbol{\Theta}_i \boldsymbol{\Theta}_i^\top \boldsymbol{k}_i \in \mathbb{R}^n \neq \mathbb{R}^q$. Figure 6.12 shows reconstruction of some of training and out-of-sample images. As expected, the reconstructed images, for both training and out-of-sample images, are very similar to the original images.

6.1.1.5 Experiments for LLISE

Embedding the Training Images: I embedded the blocks in the training images. In kNN, I used $k = 10$. For linear reconstruction, I used $\rho = \eta = 0.1$ in LLISE and $10\rho = \eta = 0.1$ in kernel LLISE. For linear embedding, I used $\rho = \eta = 0.01$. These parameters should be set small enough to have progress in optimization without oscillating behaviour. I took $q = 64$ (8×8 blocks), $p = 4$, and $d = 512 \times 512 = 262144$. In order to evaluate the obtained embedded manifold, I used the 1NN classifier to recognize the distortion type of every block. Again, the distortion type of an image comes from a majority vote among the blocks. The polynomial $(\gamma\, \breve{\boldsymbol{x}}_1^\top \breve{\boldsymbol{x}}_2 + 1)^3$, RBF $\exp(-\gamma\, ||\breve{\boldsymbol{x}}_1 - \breve{\boldsymbol{x}}_2||_2^2)$, and sigmoid

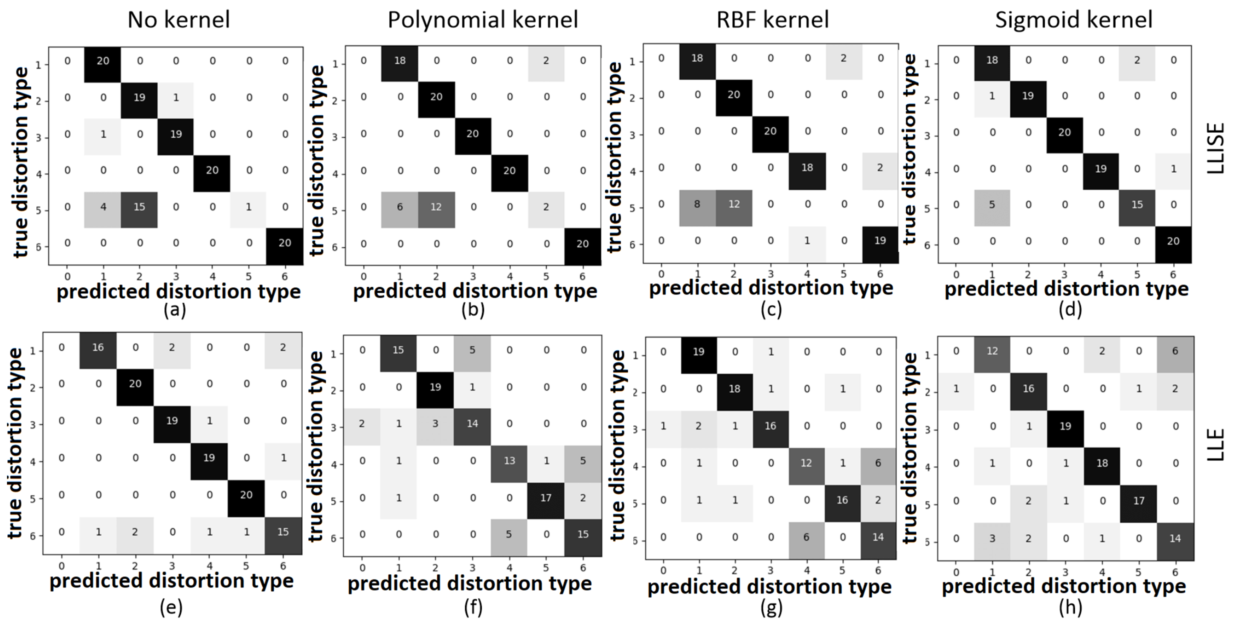

Figure 6.13: Experiments for LLISE: confusion matrices for recognition of distortion types with a 1NN classifier used in the embedded space. Matrices (a) and (e) correspond to LLISE and LLE, respectively. Matrices (b) to (d) are for kernel LLISE and (f) and (g) are for kernel LLE with polynomial, RBF, and sigmoid kernels, respectively. The 0 label corresponds to the original image and the labels 1 to 6 are the distortion types with the same order as in Fig. 6.4.

$\tanh(\gamma\, \breve{\boldsymbol{x}}_1^\top \breve{\boldsymbol{x}}_2 + 1)$ kernels were tested for kernel LLISE, where $\gamma := 1/q$. The confusion matrices for distortion recognition are shown in Fig. 6.13. Also, the LLISE and kernel LLISE are compared with LLE and kernel LLE in this figure. Except for impulse noise, LLISE and kernel LLISE had better performance compared to LLE and kernel LLE. In other distortions, especially in JPEG distortion and contrast stretch, the performances of LLE and kernel LLE were not acceptable because LLE uses ℓ_2 norm rather than SSIM distance.

Out-of-sample Embedding: For linear reconstruction, I used $\rho = \eta = 0.1$ in LLISE and $10\rho = \eta = 0.1$ in kernel LLISE. The same 1NN classification was done for the test images. Table 6.3 reports the top two votes of blocks for every image with the percentage of blocks voting for those distortions. This table also shows the recognition of distortions using LLE and kernel LLE. Note that LLE does not perform block-wise and thus it has only one recongnition label for the whole image. As expected for LLISE and kernel LLISE,

Table 6.3: Experiments for LLISE: recognition of distortions for out-of-sample images. Letters O, C, G, L, B, I, and J correspond to original image, contrast stretch, Gaussian noise, luminance enhanced, blurring, impulse noise, and JPEG distortion, respectively.

image	1	2	3	4	5	6	7	8	9	10	11	12
distortion	C	G	L	B	I	J	B + G	B + L	I + L	J + G	J + L	J + C
LLISE	42.9% C	42.2% G	35.6% L	44.5% B	31.2% G	43.2% J	46.8% G	41.3% B	55.9% G	33.5% G	39.6% J	40.7% J
	22.8% L	29.3% I	29.6% C	15.7% J	28.4% I	16.8% B	34.4% I	14.7% J	39.6% I	26.5% I	16.3% B	16.3% L
kernel LLISE (polynomial)	69.7% C	23.7% L	97.5% L	74.3% B	45.4% C	76.6% J	20.7% G	78.1% L	35.1% G	27.7% L	76.9% L	45.1% J
	14.7% I	21.3% G	0.8% C	14.5% J	19.5% I	13.9% B	17.9% B	5.7% I	23.5% L	16.6% G	7.0% B	28.6% B
kernel LLISE (RBF)	62.1% C	25.6% L	72.6% L	58.1% B	42.3% C	59.6% J	23.7% B	58.1% L	33.3% L	26.5% L	57.8% L	40.0% J
	12.0% I	16.1% B	9.5% C	16.3% J	17.0% I	16.4% B	18.5% J	11.9% B	24.7% G	19.6% J	13.0% B	24.8% B
kernel LLISE (sigmoid)	63.0% C	28.5% L	92.4% L	68.5% B	51.9% C	55.6% J	39.2% B	77.8% L	57.0% L	31.1% L	76.1% L	42.8% J
	14.5% I	24.5% C	3.5% B	15.5% J	14.0% I	19.7% B	19.5% L	10.7% B	12.6% C	24.6% B	10.8% B	29.4% B
LLE	C	L	L	B	C	J	B	C	C	L	L	B
kernel LLE (polynomial)	L	C	L	B	C	J	B	L	L	B	L	J
kernel LLE (RBF)	L	C	C	J	C	J	B	L	L	B	L	J
kernel LLE (sigmoid)	C	L	L	B	C	J	J	C	L	C	C	C

in most cases, at least one of the two top votes recognized the type of distortion(s) the out-of-sample images had. However, LLE and kernel LLE performed poorly on the out-of-sample images. This shows the effectiveness of the proposed LLISE and kernel LLISE even for a mixture of distortions.

6.1.2 Experiments for Probabilistic Dimensionality Reduction

In this section, I report the experiments for the proposed algorithms in probabilistic dimensionality reduction including QQE.

6.1.2.1 Experiments for QQE

Discussion on Impact of Hyperparameters: For all experiments of QQE, I set $\lambda = 0.1$, $\eta = 0.01$, and $k = 10$. QQE is not yet applicable on out-of-sample data so these parameters cannot be determined by validation; however, here, I briefly discuss the impact of these hyperparameters. The learning rate η should be set small enough to have progress in optimization without oscillating behaviour. I empirically found $\eta = 0.01$ to be good for different datasets. The larger number of neighbors k results in slower pacing of optimization because of Eqs. (3.57) and (3.58). Very small k, however, does not capture the local patterns of data [114]. The value $k = 10$ is fairly proper. The regularization parameter λ determines the importance of distance preserving compared to the quantile-quantile

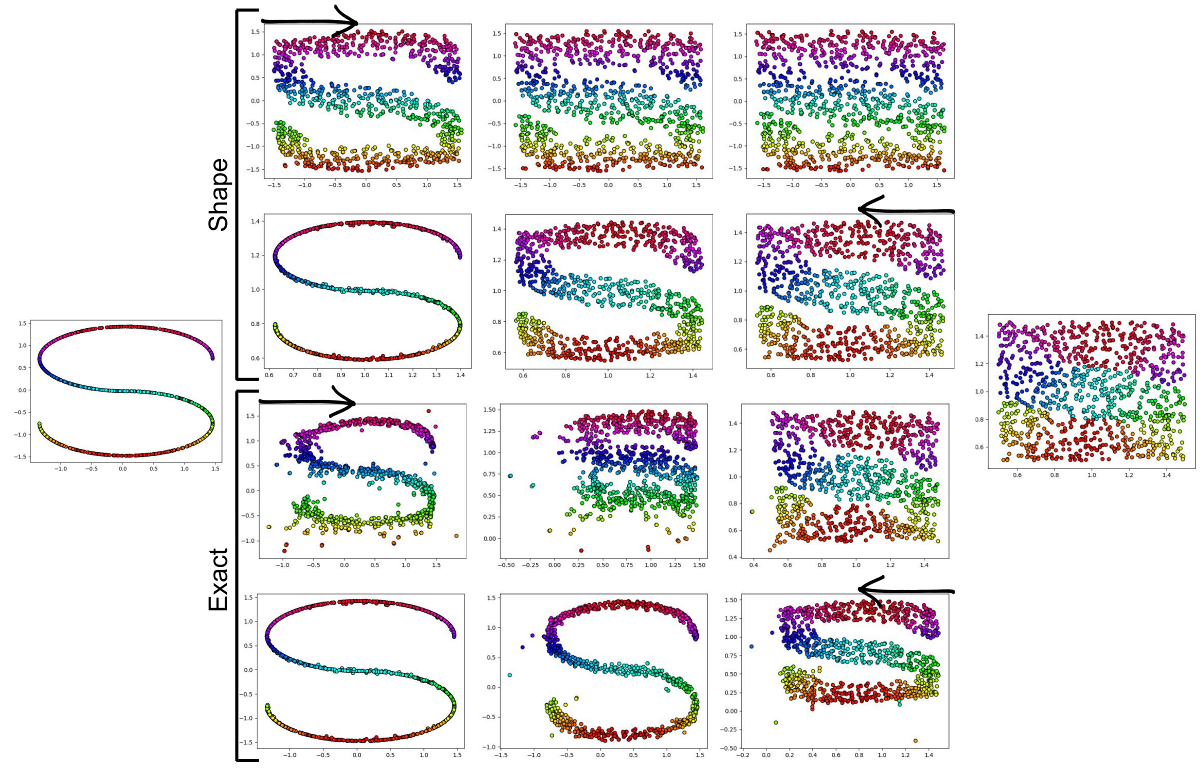

Figure 6.14: Experiments for QQE: Distribution transformation of S-shape and uniform data to each other. The first and second pair of rows correspond to transformation of shape and exact distributions, respectively. The arrows show the direction of gradual changes.

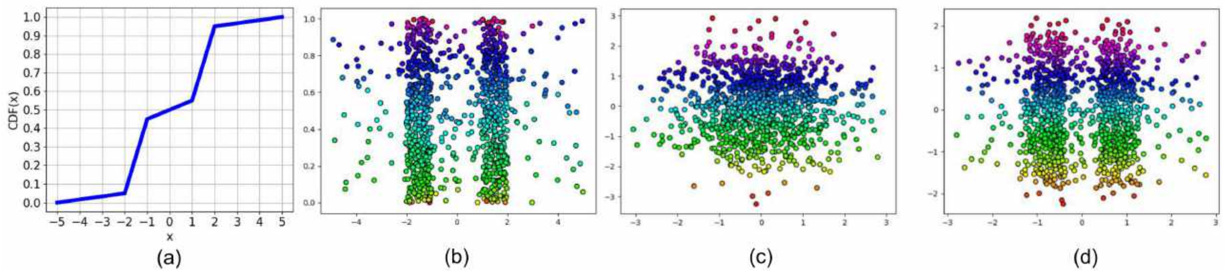

Figure 6.15: Experiments for QQE: Distribution transformation using (a) CDF of reference distribution: (b) the reference data, (c) Gaussian data, and (d) transformed data.

plot of distributions. The larger this parameter gets, the less important the distribution transformation becomes compared to preserving distances; hence, the slower the progress

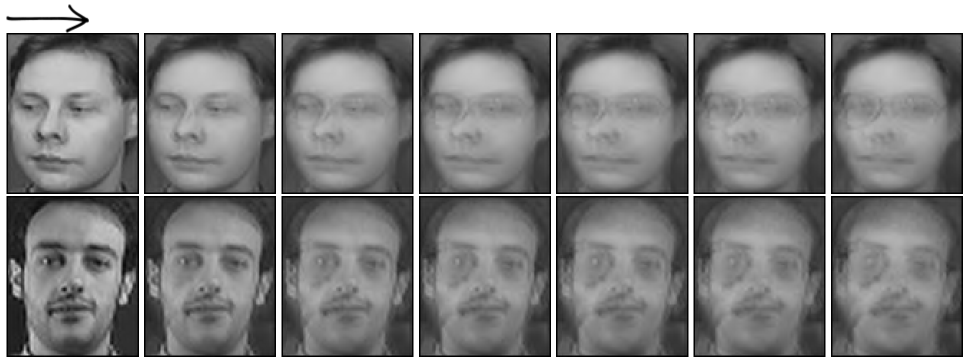

Figure 6.16: Experiments for QQE: Distribution transformation of facial images without eyeglasses to the shape of images with eyeglasses. The arrow shows the direction of gradual changes.

of optimization gets. The value $\lambda = 0.1$ was empirically found to be proper for different datasets.

Distribution Transformation for Synthetic Data: To visually show how distribution transformation works, I report the results of QQE on some synthetic datasets. In the following, I report several different possible cases for distribution transformation.

Standard Reference Distributions: A simple option for the reference distribution is a standard probability distribution. As an example, I drew a sample of size 1000 from the two dimensional uniform distribution in range $[0.5, 1.5]$ in both dimensions. This sample is depicted at the right hand side of Fig. 6.14. I also created an S-shape dataset, with mean zero and scale three, illustrated at the left hand side of Fig. 6.14. This S-shape distribution is chosen as an example of non-standard (or strangely) distributed data which is complicated to process. This strange shape illustrates the progress of QQE iterations properly. As this figure shows, in transforming the S-shape data to the shape of uniform distribution, the dataset gradually expands to fill the gaps and become similar to uniform without changing its mean and scale. In transforming to the exact uniform distribution, however, the mean and scale of data change gradually, by translation and contraction, to match the moments of the reference distribution.

Given Reference Sample: We may have a reference sample which we want to transform the distribution of data to its distribution. An example is the S-shape data shown in Fig. 6.14 where we transform the uniform data to its distribution. In shape transformation, two gaps appear first to imitate the S shape and then the stems become narrower iteratively. In exact transformation, however, the mean and scale of data also change.

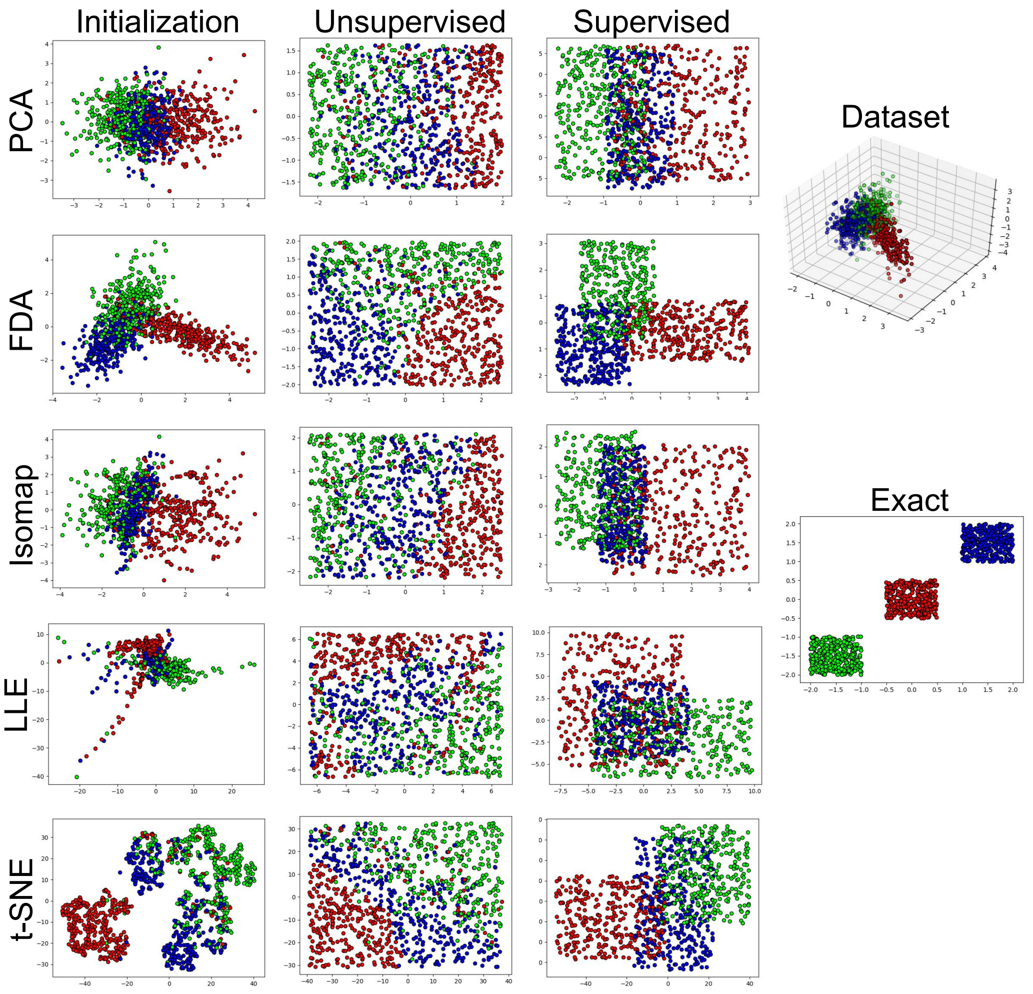

Figure 6.17: Experiments for QQE: Unsupervised and supervised exact manifold embedding of the synthetic data with different initializations. Transformation to exact reference distribution is also shown. The initialization of LLE is scaled by constant to be in range of other embeddings.

Note that exact transformation is harder than shape transformation because of change of moments; thus, some points jump at initial iterations and then converge gradually. I defer a more robust QQE to the future work.

Given Cumulative Distribution Function: Instead of a standard reference distribution or a reference sample, the user can give a desired CDF for the distribution to

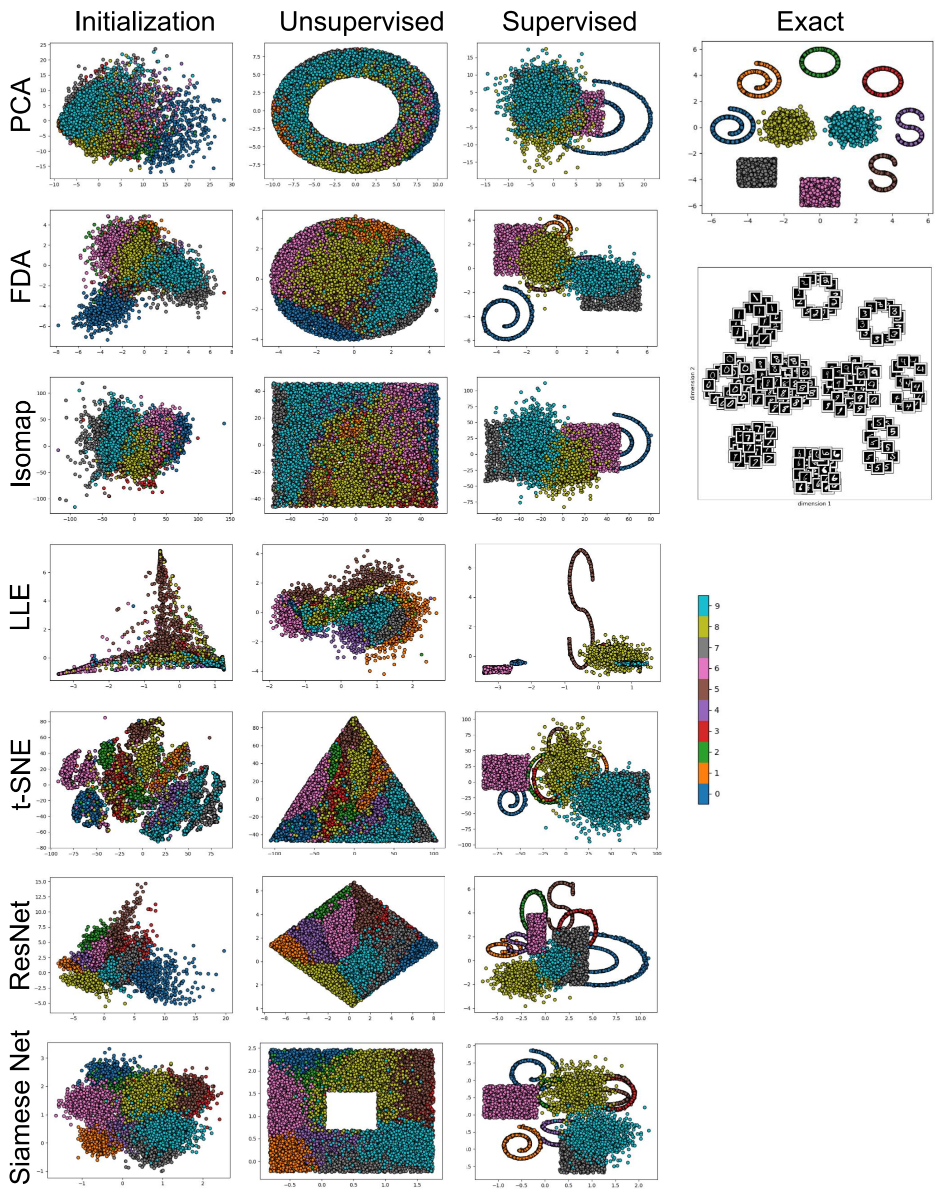

Figure 6.18: Experiments for QQE: Unsupervised and supervised exact manifold embedding of the image data with different initializations. Transformation to exact reference distribution is also shown. The initialization of LLE is scaled by constant to be in range of other embeddings.

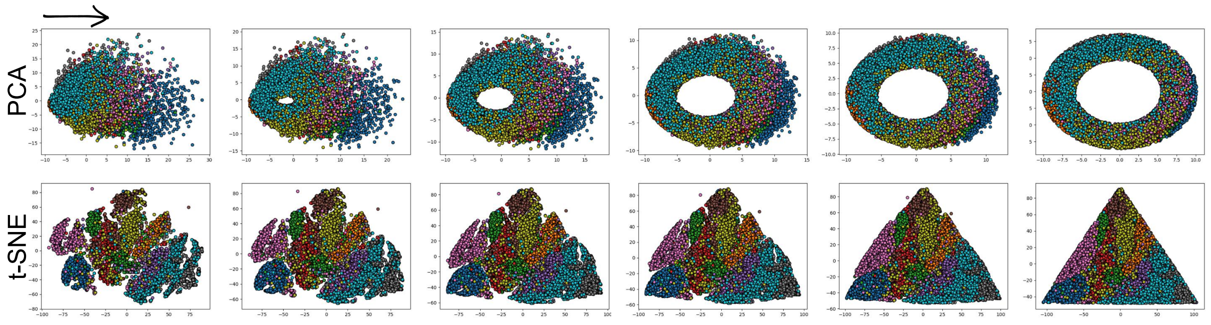

Figure 6.19: Experiments for QQE: Some iterations of unsupervised manifold embedding initialized by PCA and t-SNE. The arrow shows the direction of gradual changes.

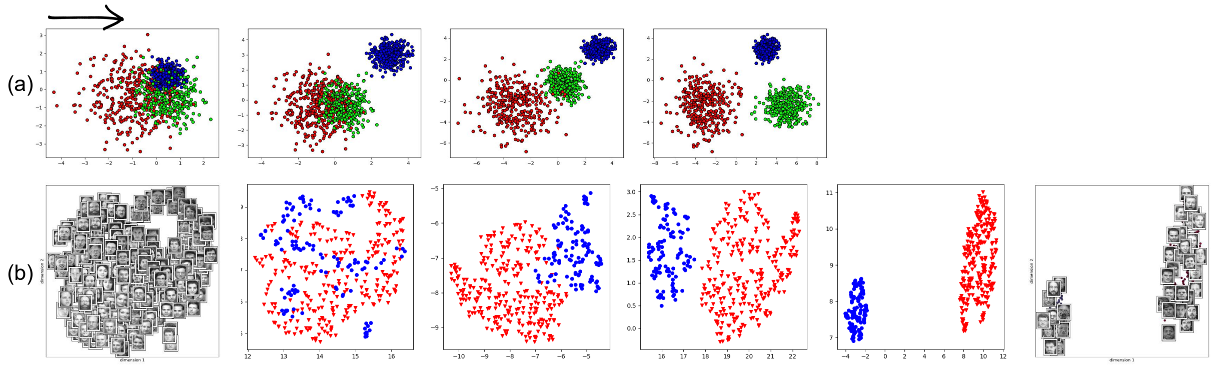

Figure 6.20: Experiments for QQE: Separation and discrimination of classes in synthetic and image data. The arrow shows the direction of gradual changes.

have. The reference sample can be sampled using the inverse CDF. The CDF can be multivariate; however, for the sake of visualization, Fig. 6.15-a shows an example multi-modal univariate CDF. I used this CDF and uniform distribution for the first and second dimension of the reference sample, respectively, shown in Fig. 6.15-b. QQE was applied on the Gaussian data shown in Fig. 6.15-c and its distribution changed to have a CDF similar to the reference CDF (see Fig. 6.15-d).

Distribution Transformation for Image Data: The distribution transformation can be used for any real data such as images. I divided the ORL facial images [127] into two sets of with and without eyeglasses. The set with eyeglasses was taken as the reference sample and we transformed the set without glasses to have the shape of reference distribution. Figure 6.16 illustrates the gradual change of two example faces from not having eyeglasses to having them. The glasses have appeared gradually in the eye regions of faces.

Manifold Embedding for Synthetic Data: To test QQE for manifold embedding, I created a three dimensional synthetic dataset having three classes shown in Fig. 6.17. Different dimensionality reduction methods, including PCA [35], FDA [43], Isomap [126], LLE [110], and t-SNE [131], were used for initialization (see Fig. 6.17). I used uniform distribution as reference and transformed the embedded data in unsupervised manner. As Fig. 6.17 shows, the embeddings of the entire dataset have changed to have the shape of uniform distribution but the order and adjacency of classes/points differ according to the initialization methods. On the other hand, the supervised QQE has made the shape of distribution of every class uniform, depicted in Fig. 6.17. Finally, supervised transformation of the embedded data to the exact reference distributions, which are uniform distributions with different means, are shown in Fig. 6.17. In exact transformation, the order of points differ depending on the initialization method but the data patterns are similar so I show only one result.

Image Manifold Embedding: QQE can be used for manifold embedding of real data such as images. For the experiments, I sampled 10000 images from the MNIST digit dataset [84] with 1000 images per digit. This sampling is because of computational reasons for the time complexity of QQE. I used different initialization methods, i.e., PCA [35], FDA [43], Isomap [126], LLE [110], t-SNE [131], ResNet-18 features [67] (with cross entropy loss after the embedding layer), and deep triplet Siamese features [118] (with ResNet-18 as the backbone network). Any embedding dimensionality can be used but here, for visualization, I took it to be two.

Figure 6.18 shows the experiments. For unsupervised QQE, I took ring stripe, filled circle, uniform (square), Gaussian mixture model, triangle, diamond, and thick square as the reference distribution for embedding initialized by PCA, FDA, Isomap, LLE, t-SNE, ResNet, and Siamese net, respectively. As shown in Fig. 6.18, the shape of embedding has changed to the desired while the local distances are preserved as much as possible. Figure 6.19 illustrates some iterations of changes in PCA and t-SNE embeddings as examples.

For supervised transformation to the shape of references distributions, I used different distributions to show that QQE can use any various references for different classes. Helix, circle, S-shape, uniform, and Gaussian were used for the digits 0/1, 2/3, 4/5, 6/7, 8/9, respectively. Figure 6.18 depicts the supervised transformation to shapes of distributions. Moreover, I set the means of reference distribution to be on a global circular pattern. This resulted in the transformation to the exact reference distributions shown in Fig. 6.18. The embedded digit images are also shown in this figure.

QQE for Separation of Classes: QQE can be used for separation and discrimination of classes; although, it does not yet support out-of-sample data. For this, reference distributions with far-away means can be chosen where transformation to the exact distribution is used. Hence, the classes move away to match the first moments of reference distributions. I experimented this for both synthetic and image data. A two dimensional synthetic dataset with three mixed classes was created as shown in Fig. 6.20. The three classes are gradually separated by QQE to match three Gaussian reference distributions with apart means.

For image data, I used the ORL face dataset [127] with two classes of faces with and without eyeglasses. The distribution transformation was performed in the input (pixel) space. The two dimensional embeddings, for visualization in Fig. 6.20, were obtained using UMAP [98]. The dataset was standardized and the reference distributions were set to be two Gaussian distributions with apart means. As the figure shows, the two classes are mixed first but gradually the two classes are completely separated by QQE.

6.1.3 Experiments for Neural Network-Based Dimensionality Reduction

In this section, I report the experiments for the proposed algorithms in neural network-based dimensionality reduction including backprojection, Fisher losses (FDT and FDC), and dynamic triplet mining (BUT and BUNCA).

6.1.3.1 Experiments for Backprojection

Datasets: For experiments, I created two synthetic datasets with 300 data points each, one for binary-class and one for three-class classification (see Figs. 6.21 and 6.22). For more difficulty, I set different variances for the classes. The data were standardized as a preprocessing. I limit myself to introduction of this new approach with small synthetic experiments. Validation on larger real-world datasets is ongoing research in our research lab.

Neural Network Settings: I implemented a neural network with three layers whose number of neurons are $\{15, 20, p\}$ where $p = 1$ and $p = 3$ for the binary and ternary classification, respectively. In different experiments, I used MSE loss for the middle layers and MSE or cross-entropy losses for the last layer. Moreover, I used Exponential Linear Unit (ELU) [14] or linear functions for activation functions of the middle layers while

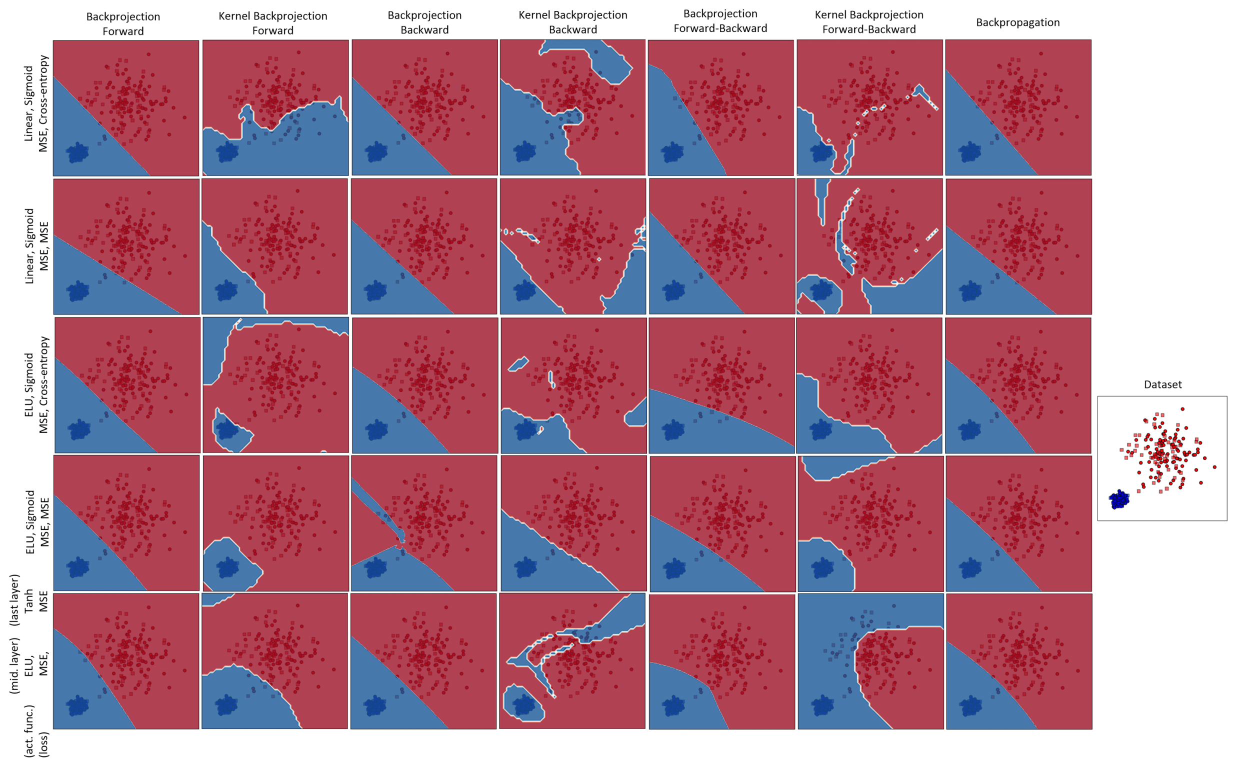

Figure 6.21: Discrimination of two classes by different training algorithms with various activation functions and loss functions. The label for each row indicates the activation functions and the loss functions for the middle then the last layers.

sigmoid or hyperbolic tangent (tanh) were used for the last layer. The derivative and inverse of these activation functions are as the following:

$$\text{ELU: } f(z) = \begin{cases} e^z - 1, z \leq 0 \\ z, z > 0 \end{cases}, f'(z) = \begin{cases} e^z, z \leq 0 \\ 1, z > 0 \end{cases}, f^{-1}(y) = \begin{cases} \ln(y+1), y \leq 0 \\ y, y > 0 \end{cases},$$

$$\text{Linear: } f(z) = z, \quad f'(z) = 1, \quad f^{-1}(y) = y,$$

$$\text{Sigmoid: } f(z) = \frac{1}{1+e^{-z}}, \quad f'(z) = f(1-f), \quad f^{-1}(y) = \ln(\frac{y}{1-y}),$$

$$\text{Tanh: } f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad f'(z) = 1 - f^2, \quad f^{-1}(y) = 0.5\ln(\frac{1+y}{1-y}).$$

Note that we bound the output values of the inverse functions $f^{-1}(.)$ for computational reasons in computer programming. Mostly, a learning rate of $\eta = 10^{-4}$ was used for

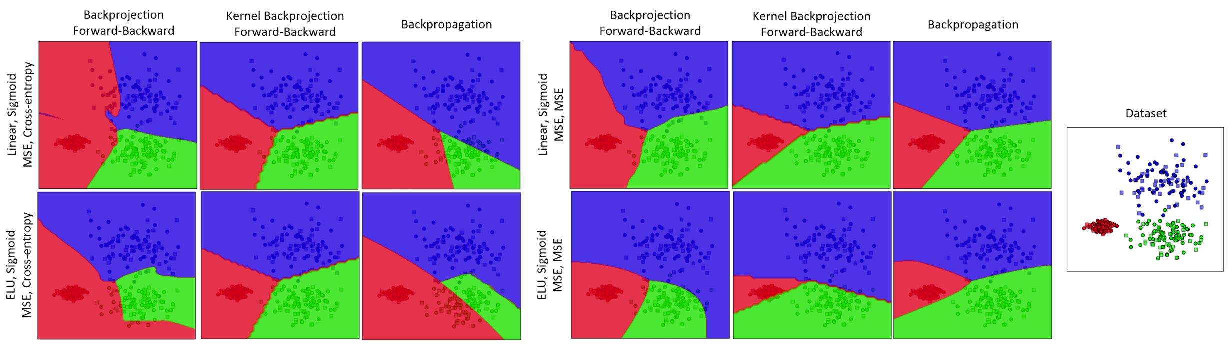

Figure 6.22: Experiments for Backprojection: Discrimination of three classes by different training algorithms with various activation functions and loss functions.

backprojection and backpropagation and $\eta = 10^{-5}$ was used for kernel backprojection. These learning rates should be set small enough to have progress in optimization without much oscillation.

Comparison of Procedures: The performance of different forward, backward, and forward-backward procedures in backprojection and kernel backprojection are illustrated in Fig. 6.21. In these experiments, the RBF kernel was used in kernel backprojection. Although the performance of these procedures are not identical but all of them are promising discrimination of classes. This shows that all three proposed procedures work well for backprojection in the input and feature spaces. In other words, the algorithm is fairly robust to the order of updating layers.

Comparison to Backpropagation: The performances of backprojection, kernel backprojection, and backpropagation are compared in the ternary classification shown in Fig. 6.22. In Fig. 6.22, the linear kernel was used. The results for binary classification are shown in Fig. 6.21. Comparison to backpropagation in these figures shows that backprojection's performance nearly matches that of backpropagation.

In the different experiments, the mean time of every epoch was often 0.08, 0.11, and 0.2 seconds for backprojection, kernel backprojection, and backpropagation, respectively, where the number of epochs were fairly similar in the experiments. This shows that backprojection is *faster* than backpropagation. This is because backpropagation updates the weights one by one while backprojection updates layer by layer.

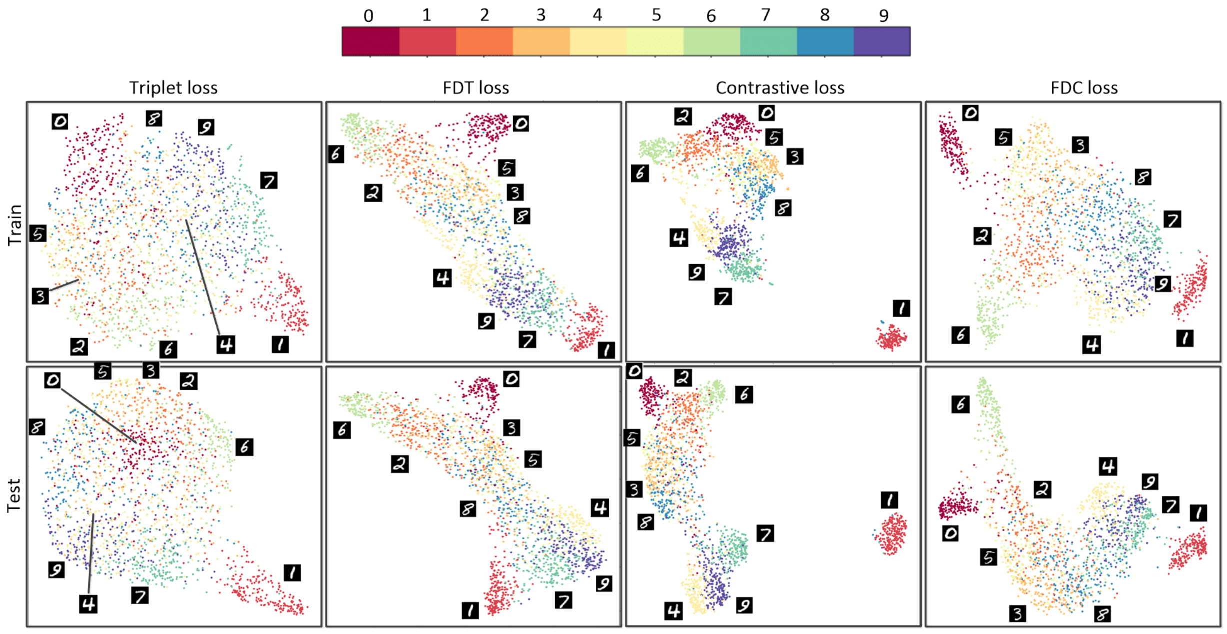

Figure 6.23: Experiments for Fisher losses: Embedding of the training and test sets of MNIST dataset in the feature spaces of different loss functions.

6.1.3.2 Experiments for Fisher Loss

Visual Comparison of Emebddings: For the experiments, I used the public MNIST dataset [84]. In the experiments, I used ResNet-18 [67] as the backbone in our Siamese network structure (see Fig. 3.4). In our experiments, I set $q = 300$, $p = 128$, $b = 32$, and $\alpha = 0.25$. The learning rate was set to 10^{-5} in all experiments.

The embeddings of the train/test sets of the MNIST dataset in the feature spaces of different loss functions are illustrated in Fig. 6.23 where $\lambda = 0.1$ was used for FDT and FDC. I used UMAP [98] for visualizing the 128-dimensional embedded data. As can be seen, both embeddings of train and test data by the FDT loss are much more discriminating than the embedding of triplet loss. On the other hand, comparing the embeddings of contrastive and FDC losses shows that their performances are both good enough as the classes are well separated. Interestingly, the similar digits usually are embedded as close classes in the feature space, and this shows the meaningfulness of the trained subspace. For example, the digit pairs (3, 8), (1, 7), and (4, 9) with the second writing format of digit four can transition into each other by slight changes, and that is why they are embedded close together.

Table 6.4: Experiments for Fisher losses on the MNIST data: Accuracy of 1-NN search for different loss functions.

	Accuracy
triplet	82.21%
FDT ($\lambda = 0.01$)	82.76%
FDT ($\lambda = 0.1$)	85.74%
FDT ($\lambda = 0.8$)	79.59%
contrastive	89.99%
FDC ($\lambda = 0.01$)	78.47%
FDC ($\lambda = 0.1$)	89.00%
FDC ($\lambda = 0.8$)	87.71%

Numerical Comparison of Embeddings: In addition to visualization, I can assess the embeddings numerically. For the evaluation of the embedded subspaces, we used the 1-Nearest Neighbor (1-NN) search because it is useful to evaluate the subspace by the closeness of the projected data samples. The accuracy rates of the 1-NN search for the embedding test data by different loss functions are reported in Table 6.4. I report the results for different values of $\lambda \in \{0.01, 0.1, 0.8\}$ in order to analyze the effect of this hyper-parameter. As the results show, in most cases, the FDT and FDC losses have outperformed the triplet and contrastive losses, respectively. Moreover, I see that $\lambda = 0.1$ is often better performing. This can be because the large value of λ (e.g., 0.8) imposes less penalty on the total scatter, which may cause the embedding space to expand gradually. The very small value of λ (e.g., 0.01), on the other hand, puts too much emphasis on the total scatter where the classes do not tend to separate well enough, so they do not increase the total scatter.

Comparison of the Latent and Feature Spaces: The last layer of network (see Fig. 3.4) behaves as a linear projection of the latent space onto the feature space. This projection fine-tunes the embeddings for better discrimination of classes. The latent embedding of the MNIST train set for both FDT and FDC loss functions can be seen in Fig. 6.24. Comparing them to the feature embeddings of the MNIST train set in Fig. 6.23 shows that the feature embedding discriminates the classes much better than the latent embedding.

6.1.3.3 Experiments for BUT & BUNCA

Experimental Setup: For the experiments, I used the MNIST dataset [84]. I split the training data into 70% and 30% portions for training and validation sets. The test set with

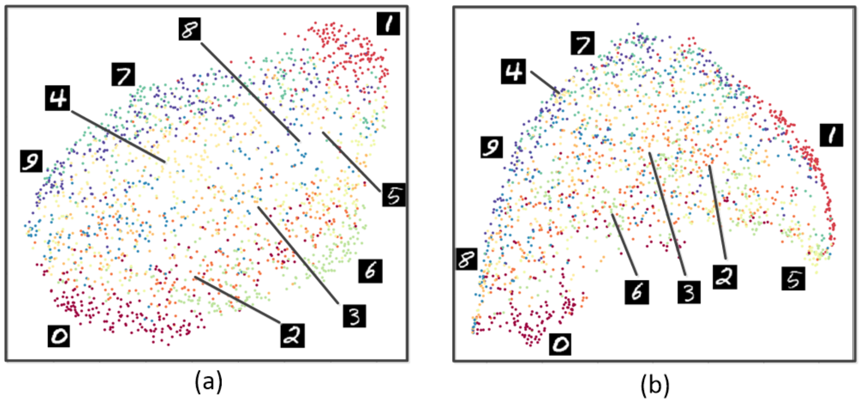

Figure 6.24: Experiments for Fisher loss: The latent embedding of MNIST training for (a) FDT and (b) FDC.

10,000 images was used for the test. The batch size was 50 where every batch contains five instances per class (i.e., $n' = 5$). The rest of setup was like those reported in Section 5.1.5.1.

Visualization of Embedding Spaces: The 2D visualization of spaces was performed using UMAP [98] applied to the embedded data. Figure 6.25 illustrates the embedding of test sets of the MNIST and CRC data using the BUT and BUNCA sampling methods. As apparent in this figure, the learned embedding spaces are interpretable. In embeddings of MNIST data, the similar digits, in the style of writing, fall close to one another. Closely embedded digits by BUT (see Fig. 6.25-a) are the digits 1 and 7, 7 and 9, 3 and 8, and 4 (second style of writing) and 9. Likewise, closely embedded digits by BUNCA (see Fig. 6.25-b) are the digits 0 and 6, 1 and 7, 7 and 9, 3 and 8, and 2 and 3 (because continuing the underneath curve of 2 results in 3).

Query Retrieval: For the evaluation of the embedding space, one can see the embedded instances as a database where nearby cases can be retrieved as matched cases for a query instance. The retrievals are extracted using the nearest neighbors in the embedding space. Because of representation learning, the retrievals are expected to be similar to the query in terms of pattern. In Fig. 6.26, I illustrate the top ten retrievals for query examples for the MNIST data. The retrievals in the embedding spaces using both BUT and BUNCA approaches are shown to visually verify the similarity matching.

In Fig. 6.26, the retrievals for a digit 4 with the second style of writing are depicted. As expected, the retrievals are very similar to the pattern of the query image. Compared to the last retrievals, the first retrievals are more similar to the query as expected. For this query

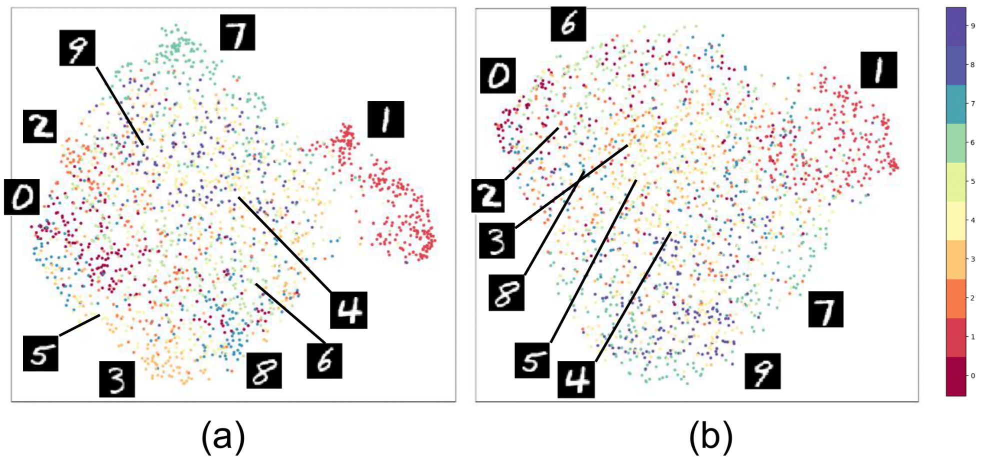

Figure 6.25: 2D visualization of test embeddings: (a) MNIST using BUT and (b) MNIST using BUNCA.

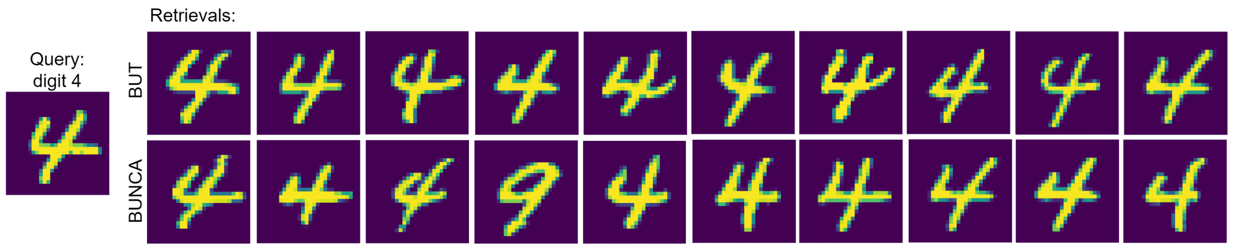

Figure 6.26: MNIST image retrieval (for digit 4) in the embedded spaces learned using the BUT and BUNCA approaches. The retrievals are sorted from left to right.

example in the BUNCA approach, one of the retrievals is wrong, but it is interpretable. The second writing style of digit "4" is very similar to digit "9" and can be morphed into it by a slight change.

Comparison with Baseline Methods: In Table 6.5, I compare the proposed BUT and BUNCA approaches with the existing triplet mining methods in the literature. This table reports the Recall@k (R@k) measure on the embedded test data, for different values of k. The baseline approaches, which I compare with, are BA [19], BSH [118], BH [68], EP [141], DWS [139], NCA [61], and PNCA [100]. Among these methods, DWS is a sampling method that samples from the existing instances in the mini-batch in contrast to my proposed approach, which samples from the distribution of data.

Table 6.5 reports the results for the MNIST dataset. The proposed BUT approach

Table 6.5: Experiments for BUT and BUNCA: Comparison of the proposed triplet mining approaches with the baselines on the MNIST dataset.

	R@1	R@4	R@8	R@16
BA [19]	79.31	93.53	96.55	98.21
BSH [118]	78.95	92.61	96.09	98.17
BH [68]	85.75	95.31	97.43	98.63
EP [141]	73.34	90.09	95.08	97.68
DWS [139]	76.44	91.35	95.72	97.68
NCA [61]	85.40	95.48	97.46	98.76
PNCA [100]	83.71	94.69	97.31	98.55
BUT	88.03	96.25	98.15	99.09
BUNCA	78.67	92.44	95.77	98.02

outperforms all other methods. Moreover, BUNCA performs better than EP and DWS, where DWS is also a sampling approach for triplet mining. This table demonstrates the effectiveness of the proposed mining approaches for triplet training.

6.2 Experiments for Numerosity Reduction

In this section, I report the experiments for the proposed algorithms in numerosity reduction based on variance, geometry, and isolation.

6.2.1 Experiments for Algorithms Based on Variance

In this section, I report the experiments for the proposed algorithms in the variance-based numerosity reduction including PSA and IRMD.

6.2.1.1 Experiments for PSA

I compare PSA with SRS, SDM, and the entire dataset (100% of samples). In SRS, all samples of a class have equal probability of selection and the sampling from instances of a class is without replacement. In all experiments of SRS, sampling is performed 20 times and the average result is considered as the result of that fold in cross validation. In SDM, the samples of a class are ranked by their distance from the mean of the class in ascending order. The selection of samples is done from the best ranked samples which are closest to

Table 6.6: Experiments for PSA: Results on MNIST dataset

	Portion	SVM	LDA	QDA	RF	LR	NB
PSA	6%	47.97%	**84.52%**	**94.83%**	**61.14%**	86.24%	**84.86%**
	50&	65.42%	84.49%	**95.01%**	**61.47%**	86.80%	**85.46%**
	90%	71.90%	**84.55&**	94.95%	**61.76%**	**86.85%**	85.32%
SRS	6%	52.70%	84.32%	94.36%	60.07%	**86.33%**	84.83%
	50%	69.91%	**84.55%**	94.98%	60.30%	**86.85%**	85.28%
	90%	72.13%	**84.55%**	**95.01%**	60.44%	**86.85%**	**85.34%**
SDM	6%	**64.46%**	77.97%	74.33%	57.27%	73.95%	68.69%
	50%	**71.08%**	81.31%	87.48%	57.43%	82.73%	78.44%
	90%	**75.21%**	83.95%	93.91%	60.20%	86.71%	84.14%
Entire Data	100%	76.22%	84.51%	95.03%	60.67%	86.78%	85.30%

the mean. Note that for PSA, SRS, and SDM, I experiment with different amounts of data reduction, retaining 6%, 20%, 60% (or 50%), 90%, and 100% of the data before carrying out classification. I set the appropriate portions according to the size of each dataset. Six different classifiers are used for verifying the effectiveness of the proposed method when using any classifier; I use Support Vector Machines (SVM), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Random Forest (RF), Logistic Regression (LR), and Gaussian Naive Bayes (NB).

The MNIST dataset [84] was used for experiments. PCA is applied on the image data as a preprocessing step. The accuracy rates of experiments on this dataset are reported in Table 6.6. As shown in this table, PSA with portion 90% outperforms using the entire data for LDA, RF, LR, and NB classifiers. It is interesting that PSA even outperforms using the entire dataset in portions 6% (using LDA and RF) and 50% (using RF, LR, and NB). In the cases LDA (6%), QDA and NB (6% and 50%), and RF (6%, 50%, and 90%), PSA strongly outperforms SRS. Moreover, in all portions using all classifiers except SVM, PSA strongly outperforms SDM in this dataset. To better compare PSA, SRS, and SDM, see Fig. 6.27 which depicts the top ranked samples of every class in MNIST dataset using these three methods. For SRS, the random samples of one of the runs are shown. Comparing PSA and SDM, PSA has captured more diverse samples because of taking both similarities and variances into account, while SDM gives top ranks to the samples close to mean and thus the top ranks are very similar. Comparing PSA and SRS, more diverse but 'better' representative samples of digits are selected by PSA. For PSA, this is specifically seen for digits 1 (different slopes), 2 and 3 (styles), 4 (two different types of writing), 5, 6, 7, and 8 (styles), and 9 (slopes).

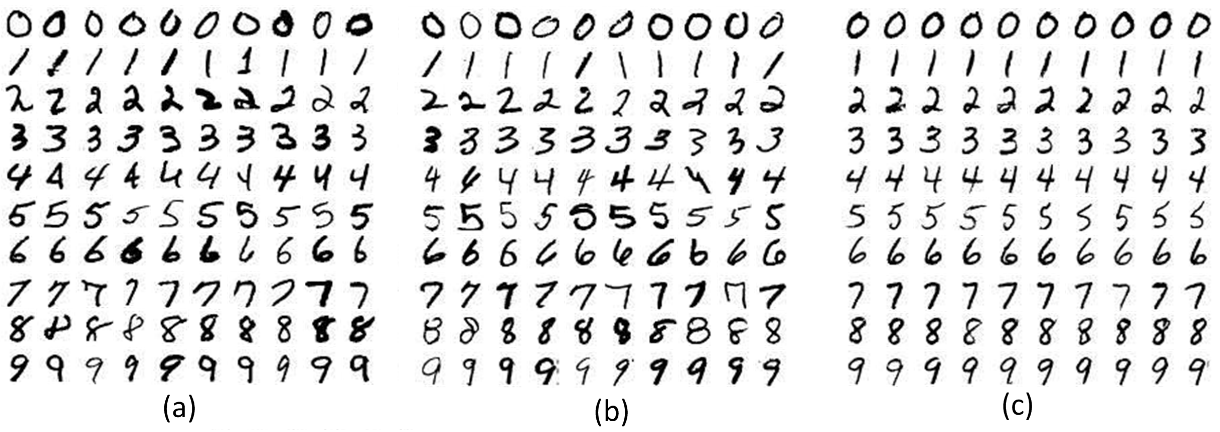

Figure 6.27: Experiments for PSA: Top ranked samples of MNIST dataset using (a) PSA, (b) SRS, and (c) SDM methods.

6.2.1.2 Experiments for IRMD

Datasets: Several datasets with various characteristics, taken from the machine learning repository, are used for experiments in this section. For classification cases, I used the Page Blocks dataset. The Facebook metrics dataset was used for regression experiments. For clustering cases, two datasets, Isolet and Iris, were utilized. These datasets are available in the UCI machine learning repository [20]. There is enough variety in the selected datasets in terms of number of instances, features, and classes. Note that in the Facebook Metrics dataset, we have $d = 7$ and $\ell = 12$. The Isolet dataset includes negative values so NMF cannot be used for it. For all experiments, datasets were shuffled and the results are the average of values in 10-fold cross validation.

Classification Cases: Table 6.7 reports the results of experiments for classification. In Tables 6.7, 6.8, and 6.9, underlined bold, bold, and underlined values show the best, the second best, and the third best values, respectively. The rates are accuracy and the reported times (in seconds) are the average times over folds for computing ranking/reduction. Our proposed methods are compared with our own implementations of SOS [76], SE [89], ascending Sorted By Distance from Mean (SDM), ENN [138], DROP3 [137], and NR scenario. Three classifiers, 1-Nearest Neighbor (1NN), Linear Discriminant Analysis (LDA), and SVM are used with three different levels of reduction each, i.e., 20%, 50%, and 70%. Note that in ENN and DROP3, the amount of reduction cannot be selected and thus in highlighted comparisons for different reduction levels, I exclude them. In comparing time of ranking, SDM usually is the best because its scoring is merely sorting distances. As

Table 6.7: Experiments for IRMD: Comparison of instance selection methods in classification. Rates are accuracy and times are in seconds. Numbers in parentheses for ENN and DROP3 show the percentage of retained data.

Page Block Dataset														
		SVD	**NMF**	**PLU**	**QR**	**DL**	**SPCA**	**FDA**	**SOS**	**SE**	**SDM**	**ENN**	**DROP3**	**No Reduction (NR)**
	Time:	5.74E−2	2.31E+1	5.96E−2	6.05E−2	3.28E+1	4.67E−1	**4.04E−2**	5.43E−1	1.51E−1	**2.95E−2**	3.90E−1	4.20E+3	×
1NN	20% data:	37.56%	73.83%	56.45%	**92.08%**	41.23%	**83.02%**	73.77%	1.93%	34.27%	33.12%	(95.61%)	(1.98%)	95.57%
	50% data:	64.59%	87.74%	71.97%	**93.51%**	67.02%	89.34%	83.55%	1.35%	**95.19%**	58.03%	95.66%	48.45%	
	70% data:	85.05%	91.81%	84.83%	**94.09%**	77.74%	92.56%	88.45%	1.20%	**95.28%**	75.05%			
LDA	20% data:	92.89%	93.23%	92.65%	90.17%	93.16%	**93.45%**	93.18%	0.36%	**93.71%**	92.34%	94.79%	80.26%	94.53%
	50% data:	93.62%	93.73%	93.53%	93.87%	93.66%	94.04%	**94.55%**	0.25%	**94.55%**	**94.24%**			
	70% data:	94.00%	94.04%	93.89%	94.06%	94.00%	94.18%	**94.51%**	0.25%	**94.48%**	94.40%			
SVM	20% data:	71.64%	65.87%	82.52%	**90.11%**	70.48%	64.18%	69.22%	2.52%	**89.07%**	71.09%	94.86%	87.33%	89.76%
	50% data:	71.09%	86.48%	83.17%	**91.85%**	82.07%	83.33%	82.21%	1.99%	**94.24%**	84.27%			
	70% data:	91.10%	90.31%	90.46%	**91.44%**	83.09%	88.08%	87.99%	1.53%	**94.37%**	89.20%			

Table 6.8: Experiments for IRMD: Comparison of instance selection methods in regression. The values are mean absolute error and times are in seconds.

Facebook dataset											
		SVD	**NMF**	**PLU**	**QR**	**DL**	**SPCA**	**SOS**	**SE**	**SDM**	**NR**
	Time:	7.62E−3	1.63E+0	6.51E−3	7.01E−3	1.24E+1	**6.41E−3**	1.40E−2	8.40E−3	**4.81E−3**	×
LR	20% data:	8.09E+3	2.94E+4	1.62E+4	**5.28E+3**	1.57E+4	**5.71E+3**	6.07E+3	1.89E+4	7.50E+3	5.81E+3
	50% data:	6.45E+3	1.15E+4	8.95E+3	**4.85E+3**	8.61E+3	**5.20E+3**	6.25E+3	6.59E+3	5.69E+3	
	70% data:	6.08E+3	8.34E+3	6.90E+3	**4.95E+3**	6.76E+3	5.80E+3	5.90E+3	5.83E+3	**5.30E+3**	
RF	20% data:	8.64E+3	2.80E+4	1.54E+4	**5.53E+3**	1.49E+4	7.08E+3	**6.56E+3**	1.09E+4	6.58E+3	6.17E+3
	50% data:	7.03E+3	1.43E+4	1.04E+4	**5.02E+3**	9.69E+3	7.38E+3	6.76E+3	**6.10E+3**	7.19E+3	
	70% data:	6.76E+3	1.05E+4	7.66E+3	**5.03E+3**	7.52E+3	6.30E+3	6.26E+3	6.32E+3	**6.03E+3**	
MLP	20% data:	1.11E+4	5.37E+4	1.84E+4	**7.11E+3**	1.73E+4	1.66E+4	**7.46E+3**	4.70E+4	2.31E+4	5.72E+3
	50% data:	6.19E+3	1.45E+4	8.80E+3	**4.75E+3**	8.26E+3	5.67E+3	6.02E+3	**5.62E+3**	6.64E+3	
	70% data:	6.19E+3	1.15E+4	7.18E+3	**5.14E+3**	6.81E+3	**5.55E+3**	5.95E+3	5.86E+3	6.00E+3	

reported in Table 6.7, I perform better than SOS, SE, ENN, and DROP3 in terms of time in many cases. Also, in many cases, I hold the best or the second best positions in accuracy. In some cases, the proposed methods even outperform NR. Not surprisingly, FDA performs very well combined with the LDA classifier. In some cases, SVD provides very good results which makes sense since it minimizes the reconstruction error. Also, the orthogonal bases of QR-decomposition seem to lead to good performance.

Regression Cases: The results of regression are reported in Table 6.8. Linear Regression (LR), Random Forest (RF), and Multi-Layer Perceptron (MLP) are the used methods for regression. The number of trees and maximum depth in RF were both 100. In MLP, the ReLu activation function, ADAM optimizer, and two hidden layers with 100 and 50 neurons were used. Regarding existing methods for numerosity reduction, SOS and SE are not proposed for regression but I compare them here. However, ENN and DROP3 are

Table 6.9: Experiments for IRMD: Comparison of instance selection methods in clustering. Rates are adjusted rand index (best is 1) and times are in seconds.

		SVD	NMF	PLU	QR	DL	SOS	SE	SDM	NR
Isolet dataset										
	Time:	5.30E−1	×	**3.14E−1**	3.25E−1	3.62E+3	1.32E+0	3.39E−1	**2.10E−1**	×
K-means	20% data:	3.32E−1	×	3.24E−1	**5.31E−1**	3.67E−1	**4.80E−1**	3.02E−1	2.11E−1	4.86E−1
	50% data:	**4.40E−1**	×	3.95E−1	**4.95E−1**	4.19E−1	**4.95E−1**	4.15E−1	4.22E−1	
	70% data:	4.53E−1	×	4.57E−1	**4.82E−1**	4.66E−1	**4.88E−1**	4.51E−1	4.67E−1	
Birch	20% data:	3.77E−1	×	3.23E−1	**5.04E−1**	3.74E−1	**4.70E−1**	3.09E−1	2.29E−1	5.13E−1
	50% data:	4.58E−1	×	4.38E−1	**5.20E−1**	4.12E−1	**4.89E−1**	4.48E−1	3.92E−1	
	70% data:	4.76E−1	×	**4.84E−1**	**5.00E−1**	4.77E−1	4.82E−1	4.57E−1	4.79E−1	
Iris Dataset										
		SVD	NMF	PLU	QR	DL	SOS	SE	SDM	NR
	Time:	**1.60E−3**	2.62E−1	**1.32E−3**	**1.60E−3**	9.23E−2	3.70E−3	3.00E−1	1.90E−3	×
K-means	20% data:	**6.84E−1**	2.41E−1	1.36E−1	1.32E−1	6.68E−2	**6.92E−1**	5.02E−1	4.89E−1	7.03E−1
	50% data:	**7.80E−1**	4.95E−1	6.21E−1	5.93E−1	7.05E−1	**7.38E−1**	4.85E−1	5.77E−1	
	70% data:	7.17E−1	7.18E−1	6.95E−1	6.53E−1	**7.32E−1**	**7.35E−1**	6.27E−1	7.12E−1	
Birch	20% data:	**6.89E−1**	3.77E−1	1.72E−1	1.07E−1	1.80E−1	**6.41E−1**	5.10E−1	2.90E−1	5.93E−1
	50% data:	**7.04E−1**	5.03E−1	6.15E−1	**6.35E−1**	6.05E−1	5.34E−1	5.11E−1	5.92E−1	
	70% data:	**6.85E−1**	6.16E−1	**6.77E−1**	6.61E−1	5.76E−1	5.79E−1	6.23E−1	6.04E−1	

only applicable to classification. The proposed methods always outperform SOS and SE in terms of both time and accuracy. QR-decomposition takes the best place probably because of orthogonality. SPCA is also performing well because it captures dependencies of labels and instances using HSIC. SVD has acceptable result because of both orthogonality and minimum reconstruction error. In all cases, the proposed methods even outperform NR, interestingly.

Clustering Cases: Table 6.9 summarizes the results for clustering. Two clustering methods, i.e., K-means and Birch are used for experiments. I apply SOS and SE beyond their proposed use of classification. Again, the proposed methods always outperform SOS and SE in ranking time and in most cases, my methods also have the best performance in comparisons. QR-decomposition is performing well in Isolet dataset. SVD shows promising results in both Isolet and Iris datasets. The possible reasons of good performances of SVD and QR were already explained. DL performs well enough in Iris dataset because it benefits from sparsity.

6.2.2 Experiments for Algorithm Based on Geometry

In this section, I report the experiments for the proposed algorithms in the geometry-based numerosity reduction including CAD, iCAD, K-CAD, and K-iCAD.

6.2.2.1 Experiments for Anomaly Detection by CAD

Synthetic Datasets: I examined CAD and iCAD on three two-dimensional synthetic datasets, i.e., two moons and two homogeneous and heterogeneous clusters. Figure 6.28 shows the results for CAD and K-CAD with RBF and polynomial (degree three) kernels. As expected, the abnormal and core points are correctly detected as anomalous and normal points, respectively. The boundary points are detected as anomaly in CAD while they are correctly recognized as normal points in K-CAD. In heterogeneous clusters data, the larger cluster is correctly detected as normal in CAD but not in K-CAD; however, if the threshold is manually changed (rather than by K-means) in K-CAD, the larger cluster will also be correctly recognized. As seen in this figure, the scores are reverse in RBF and polynomial kernels which is consistent to my explanation in Section 4.2.2.

Real Datasets: I did experiments on several real datasets of anomaly detection. The datasets, which are taken from [108], are speech, opt. digits, arrhythmia, wine, and musk with 1.65%, 3%, 15%, 7.7%, and 3.2% portions of anomalies, respectively. The sample size of these datasets are 3686, 5216, 452, 129, and 3062 and their dimensionality are 400, 64, 274, 13, and 166, respectively. I compared CAD and K-CAD with RBF and polynomial (degree 3) kernels to Isolation forest, LOF, one-class SVM (RBF kernel), and EE. I used $k = 10$ in LOF, CAD, and K-CAD. The average area under the ROC curve (AUC) and the average time for both training and test phases over 10-fold Cross Validation (CV) are reported in Table 6.10. For wine data, because of small sample size, I used 2-fold CV. The system running the methods was Intel Core i7, 3.60 GHz, with 32 GB RAM. In most cases, K-CAD has better performance than CAD; although CAD is useful and effective especially for anomaly path (see Section 5.2.3). For speech and optdigits datasets, RBF kernel has better performance than polynomial and for other datasets, polynomial kernel is better. Mostly, K-CAD is faster in both training and test phases because K-CAD uses kernel matrix and normalizing the matrix rather than element-wise cosines in CAD. In speech and optdigits datasets, the proposed method outperforms all the baseline methods in both training and test AUC rates. In arrhythmia data, K-CAD with polynomial kernel has beter results than isolation forest. For wine dataset, K-CAD with polynomial kernel is better than isolation forest, SVM, and EE. In musk data, K-CAD with both RBF and polynomial kernels is better than isolation forest and SVM.

6.2.2.2 Experiments for Prototype Selection by iCAD

I performed experiments on several real datasets, i.e., pima, image segment, Facebook metrics, and iris datasets, from the UCI machine learning repository [20]. The first two

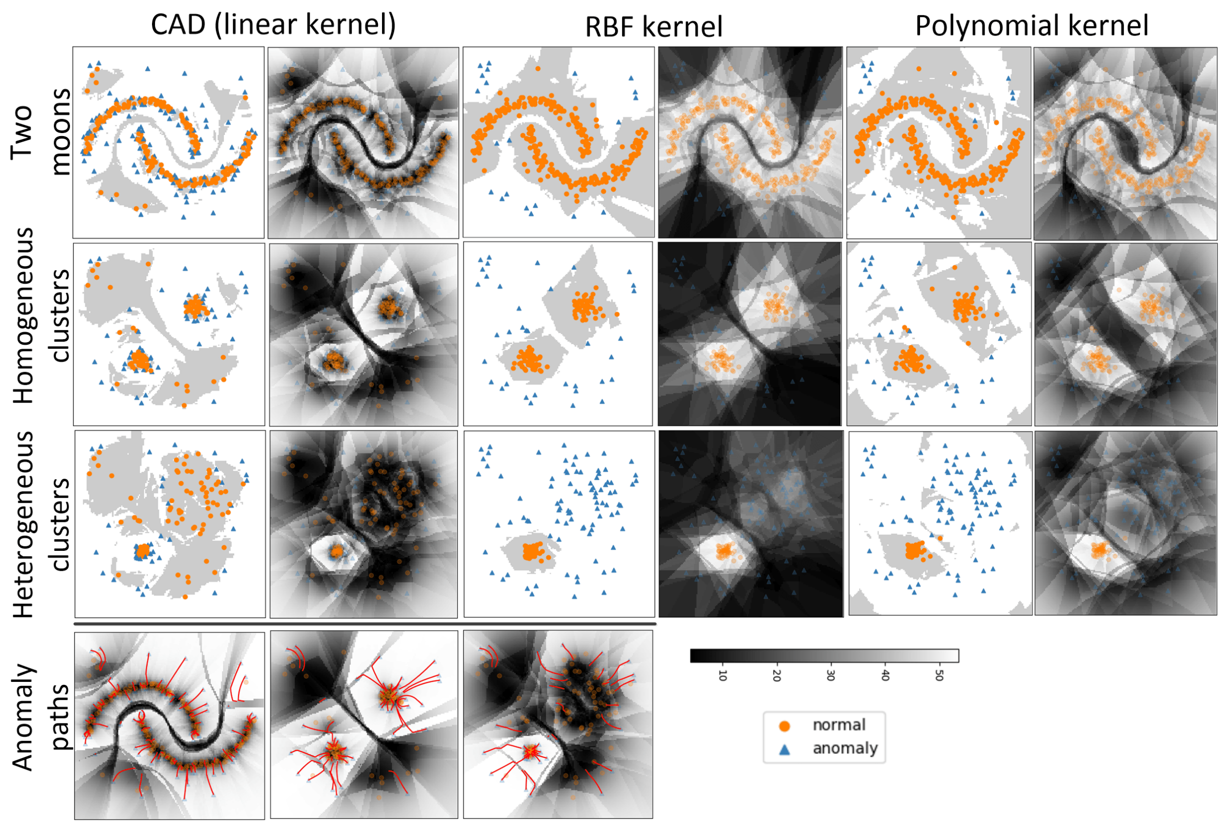

Figure 6.28: Anomaly detection, anomaly scores, anomaly landscape, and anomaly paths for synthetic datasets. In the gray and white plots, the gray and white colors show the regions determined as normal and anomaly, respectively. The gray-scale plots are the anomaly scores.

datasets were used for classification, the third for regression, and the last one for clustering. The sample size of datasets are 768, 2310, 500, and 150, and their dimensionality are 8, 19, 19, and 4, respectively. The number of classes/clusters in pima, image segment, and iris are 2, 7, and 3, respectively. I used 1-Nearest Neighbor (1-NN), Linear Discriminant Analysis (LDA), SVM, Linear Regression (LR), Random Forest (RF), Multi-Layer Perceptron (MLP) with two hidden layers, K-means and Birch clustering methods in experiments. Table 6.11 reports the average accuracy and time over 10-fold CV and comparison to IRMD (with QR decomposition), PSA, SOS, SDM, ENN, DROP3, and NR.

The iCAD and K-iCAD are reported in both rank-based and retaining-based versions of prototype selection. For pima and image segment datasets, iCAD and K-iCAD are

Table 6.10: Experiments for CAD: Comparison of anomaly detection methods. Rates are AUC percentage and times are in seconds.

			CAD	K-CAD (rbf)	K-CAD (poly)	Iso Forest	LOF	SVM	EE
Speech	**Train:**	**Time:**	14.84 ± 0.21	13.54 ± 0.21	13.87 ± 0.33	2.82 ± 0.06	6.53 ± 0.02	6.12 ± 0.02	23.35 ± 0.25
		AUC:	34.78 ± 0.15	76.15 ± 2.08	63.69 ± 1.48	48.37 ± 1.38	53.99 ± 1.75	46.63 ± 1.59	49.16 ± 1.84
	Test:	**Time:**	7.16 ± 0.03	1.38 ± 0.03	1.42 ± 0.03	0.06 ± 00.01	7.31 ± 0.10	0.24 ± 0.01	0.01 ± 0.00
		AUC:	42.07 ± 10.48	71.23 ± 13.18	56.15 ± 10.48	45.23 ± 12.17	53.53 ± 12.29	44.55 ± 12.09	47.25 ± 12.54
Opt digits	**Train:**	**Time:**	13.27 ± 0.11	26.96 ± 0.31	25.86 ± 0.48	0.81 ± 0.01	1.84 ± 0.02	3.10 ± 0.01	0.96 ± 0.04
		AUC:	32.67 ± 1.53	87.52 ± 1.76	77.79 ± 1.45	68.38 ± 4.64	60.84 ± 1.67	50.52 ± 3.81	39.04 ± 2.44
	Test:	**Time:**	11.24 ± 0.12	2.67 ± 0.01	2.59 ± 0.03	0.03 ± 0.01	2.13 ± 0.08	0.15 ± 00.00	00.01 ± 00.00
		AUC:	26.28 ± 7.10	88.22 ± 5.62	79.72 ± 4.81	68.36 ± 8.11	61.12 ± 11.65	37.49 ± 7.41	38.84 ± 4.29
Arrhythmia	**Train:**	**Time:**	4.76 ± 0.02	2.75 ± 0.06	2.53 ± 0.03	0.20 ± 0.01	0.07 ± 00.00	0.13 ± 00.00	0.85 ± 0.02
		AUC:	52.89 ± 0.96	48.87 ± 0.51	73.92 ± 1.12	62.43 ± 2.05	91.04 ± 0.66	88.56 ± 00.87	80.59 ± 0.65
	Test:	**Time:**	1.59 ± 0.01	0.30 ± 0.01	0.28 ± 00.00	0.02 ± 0.01	0.08 ± 0.00	0.01 ± 00.00	00.01 ± 00.00
		AUC:	48.02 ± 9.06	48.56 ± 5.38	71.88 ± 9.23	63.07 ± 11.55	90.57 ± 5.47	90.03 ± 5.63	80.32 ± 4.73
Wine	**Train:**	**Time:**	0.28 ± 0.00	0.03 ± 0.03	0.03 ± 0.01	0.09 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.05 ± 0.02
		AUC:	25.59 ± 4.28	27.04 ± 10.66	92.11 ± 7.06	79.56 ± 10.59	98.70 ± 1.29	68.59 ± 4.25	59.56 ± 37.15
	Test:	**Time:**	0.18 ± 00.00	0.02 ± 00.00	0.01 ± 0.00	0.02 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
		AUC:	23.65 ± 14.45	40.17 ± 13.12	86.97 ± 2.96	76.09 ± 10.11	92.58 ± 5.69	91.13 ± 3.83	57.70 ± 38.84
Musk	**Train:**	**Time:**	11.15 ± 0.20	9.67 ± 0.47	9.42 ± 0.03	0.98 ± 0.01	1.16 ± 0.03	3.16 ± 0.00	10.16 ± 0.32
		AUC:	40.69 ± 2.97	69.68 ± 2.97	93.45 ± 1.46	99.91 ± 00.00	41.93 ± 3.34	57.99 ± 7.34	99.99 ± 00.00
	Test:	**Time:**	4.89 ± 0.02	0.98 ± 0.02	0.98 ± 0.01	0.03 ± 0.00	1.24 ± 0.01	0.17 ± 0.00	0.01 ± 0.00
		AUC:	30.30 ± 10.37	50.00 ± 0.00	93.80 ± 3.77	99.95 ± 0.00	39.00 ± 10.55	5.71 ± 3.63	100 ± 0.00

both performing equally well but in other datasets, K-iCAD is mostly better. In terms of time, the proposed methods outperform PSA and DROP3. In pima, the proposed methods outperform all other baselines. In image segment, the proposed methods are better than IRMD, SE, and SDM. In facebook data, the proposed methods are mostly better than SOS, SE, and SDM, and in some cases better than PSA. In iris data, the proposed methods outperform all the baselines. In some cases, the proposed methods even outperform using the entire data. In retaining-based iCAD and K-iCAD, mostly, K-iCAD with RBF kernel retains the least, then K-iCAD with polynomial kernel, and then CAD.

6.2.3 Experiments for Algorithm Based on Isolation

In this section, I report the experiments for the proposed algorithm in the isolation-based numerosity reduction which is iMondrian.

6.2.3.1 Experiments for iMondrian – Synthetic Data

I created four two-dimensional synthetic datasets (a)-(d) as can be seen in Fig. 6.29. The datasets include a variety of distributions of random anomalous points. Dataset (a) has 255 and the rest of datasets have 100 inliers while the number of outliers are 45.

Table 6.11: Experiments for iCAD: Comparison of instance selection methods in classification. Classification, regression, and clustering rates are accuracy, mean absolute error, and adjusted rand index (best is 1), respectively, and times are in seconds. The left and right columns are for rank-based and retaining-based methods. Numbers in parentheses show the percentage of retained data.

Pima Dataset														
		iCAD	K-iCAD (rbf)	K-iCAD (poly)	IRMD	PSA	SOS	SE	SDM	iCAD	K-iCAD (rbf)	K-iCAD (poly)	ENN	DROP3	NR
	Time:	1.98E+0	4.41E−1	4.14E−1	9.52E−3	5.87E+0	2.13E−2	1.59E−2	4.51E−3	2.09E+0	4.73E−1	4.73E−1	1.87E−2	5.00E+0	×
1NN	20% data:	70.69%	67.70%	61.97%	67.28%	66.53%	67.06%	44.40%	62.87%	(40.79%)	(4.35%)	(12.76%)	(69.40%)	(13.44%)	
	50% data:	70.83%	66.01%	64.96%	66.64%	64.44%	67.04%	50.64%	66.39%	65.23%	66.93%	65.88%	71.74%	64.06%	68.48%
	70% data:	69.26%	67.56%	65.74%	67.95%	65.75%	66.91%	65.62%	67.81%						
LDA	20% data:	75.64%	67.82%	72.25%	70.81%	76.17%	73.56%	54.55%	65.09%						
	50% data:	76.42%	76.03%	76.43%	73.81%	76.81%	76.43%	71.35%	73.68%	65.62%	71.74%	66.14%	77.07%	75.12%	77.73%
	70% data:	76.94%	76.82%	76.81%	75.38%	76.82%	76.56%	76.55%	74.98%						
SVM	20% data:	63.28%	57.65%	62.50%	59.62%	57.68%	63.52%	42.04%	63.78%						
	50% data:	65.22%	62.25%	57.94%	61.57%	61.57%	62.25%	48.15%	60.01%	64.84%	63.92%	63.93%	67.18%	53.12%	64.57%
	70% data:	62.76%	61.28%	56.40%	55.97%	60.42%	62.63%	55.23%	64.98%						
Image Segment dataset															
		iCAD	K-iCAD (rbf)	K-iCAD (poly)	IRMD	PSA	SOS	SE	SDM	iCAD	K-iCAD (rbf)	K-iCAD (poly)	ENN	DROP3	NR
	Time:	6.25E+0	1.20E+0	1.04E+0	3.64E−2	3.28E+2	5.86E−2	4.85E−2	1.38E−2	6.13E+0	1.58E+0	1.44E+0	1.30E−1	3.13E+2	×
1NN	20% data:	90.34%	81.42%	83.72%	78.00%	90.25%	89.26%	83.41%	80.99%	(9.54%)	(3.97%)	(3.14%)	(95.25%)	(6.78%)	
	50% data:	93.41%	89.43%	92.85%	86.83%	94.76%	94.19%	84.71%	87.05%	47.22%	49.13%	49.43%	94.45%	60.34%	96.45%
	70% data:	94.84%	92.20%	95.75%	91.03%	96.06%	95.41%	90.69%	90.00%						
LDA	20% data:	89.26%	85.97%	90.86%	82.90%	90.47%	90.51%	85.23%	86.36%						
	50% data:	90.64%	90.04%	91.94%	86.32%	91.08%	91.16%	86.27%	87.83%	60.60%	58.70%	59.91%	67.57%	79.35%	91.55%
	70% data:	90.90%	90.99%	91.64%	88.26%	91.42%	91.34%	90.30%	89.48%						
SVM	20% data:	73.41%	77.74%	76.40%	71.42%	78.44%	78.35%	71.73%	71.86%						
	50% data:	80.90%	81.08%	80.30%	80.60%	77.18%	82.85%	70.60%	81.34%	39.43%	40.38%	38.70%	78.52%	55.10%	85.23%
	70% data:	84.71%	84.71%	83.67%	78.87%	82.20%	85.67%	86.79%	84.32%						
Facebook Dataset															
		iCAD	K-iCAD (rbf)	K-iCAD (poly)	IRMD	PSA	SOS	SE	SDM	iCAD	K-iCAD (rbf)	K-iCAD (poly)	ENN	DROP3	NR
	Time:	1.30E+0	6.33E−1	1.01E+0	7.01E−3	1.08E+0	1.40E−2	8.40E−3	4.81E−3	1.26E+0	3.18E−1	3.18E−1	×	×	×
LR	20% data:	7.57E+3	6.00E+3	7.45E+3	5.28E+3	9.98E+3	6.07E+3	1.89E+4	7.50E+3	(49.96%)	(6.95%)	(34.81%)			
	50% data:	6.11E+3	5.51E+3	5.86E+3	4.85E+3	6.30E+3	6.25E+3	6.59E+3	5.69E+3	6.10E+3	1.56E+4	6.41E+3	×	×	5.81E+3
	70% data:	5.70E+3	6.00E+3	5.72E+3	4.95E+3	5.55E+3	5.90E+3	5.83E+3	5.30E+3						
RF	20% data:	8.54E+3	7.87E+3	6.85E+3	5.53E+3	7.12E+3	6.56E+3	1.09E+4	6.58E+3						
	50% data:	6.41E+3	7.08E+3	6.28E+3	5.02E+3	6.20E+3	6.76E+3	6.10E+3	7.19E+3	6.36E+3	9.82E+3	6.46E+3	×	×	6.17E+3
	70% data:	5.92E+3	6.95E+3	6.19E+3	5.03E+3	5.86E+3	6.26E+3	6.32E+3	6.03E+3						
MLP	20% data:	1.348E+4	1.61E+4	1.02E+4	7.11E+3	2.12E+4	7.46E+3	4.70E+4	2.31E+4						
	50% data:	6.10E+3	5.44E+3	5.57E+3	4.75E+3	6.93E+3	6.02E+3	5.62E+3	6.64E+3	6.11E+3	5.06E+4	8.31E+3	×	×	5.72E+3
	70% data:	6.31E+3	6.06E+3	5.66E+3	5.14E+3	5.75E+3	5.95E+3	5.86E+3	6.00E+3						
Iris Dataset															
		iCAD	K-iCAD (rbf)	K-iCAD (poly)	IRMD	PSA	SOS	SE	SDM	iCAD	K-iCAD (rbf)	K-iCAD (poly)	ENN	DROP3	NR
	Time:	4.96E−1	5.30E−2	4.46E−2	1.60E−3	2.83E−1	3.70E−3	3.00E−1	1.90E−3	4.68E−1	6.07E−2	5.95E−2	×	×	×
K-means	20% data:	6.87E−1	1.56E−1	2.34E−1	1.32E−1	6.18E−1	6.92E−1	5.02E−1	4.89E−1	(60.59%)	(84.07%)	(84.07%)			
	50% data:	6.98E−1	7.13E−1	8.33E−1	5.93E−1	7.01E−1	7.38E−1	4.85E−1	5.77E−1	7.34E−1	8.06E−1	8.20E−1	×	×	7.03E−1
	70% data:	7.18E−1	7.83E−1	8.20E−1	6.53E−1	7.18E−1	7.35E−1	6.27E−1	7.12E−1						
Birch	20% data:	6.09E−1	0.00E+0	1.26E−1	1.07E−1	6.97E−1	6.41E−1	5.10E−1	2.90E−1						
	50% data:	6.48E−1	6.58E−1	7.15E−1	6.35E−1	6.11E−1	5.34E−1	5.11E−1	5.92E−1	7.71E−1	6.60E−1	6.85E−1	×	×	5.93E−1
	70% data:	6.67E−1	7.17E−1	7.37E−1	6.61E−1	6.48E−1	5.79E−1	6.23E−1	6.04E−1						

Batch Experiments: The results of batch processing in iMondrian forest on the synthetic datasets are illustrated in Fig. 6.29. The anomaly scores are shown for the input space of data. As expected, the scores are higher for anomalous points of space, which are far away from the core of distribution. Fig. 6.29 shows the results of both K-means clustering and thresholding (with threshold $s = 0.5$) which perform almost equally well. I also show the result of iForest (with threshold $s = 0.5$) for the sake of comparison.

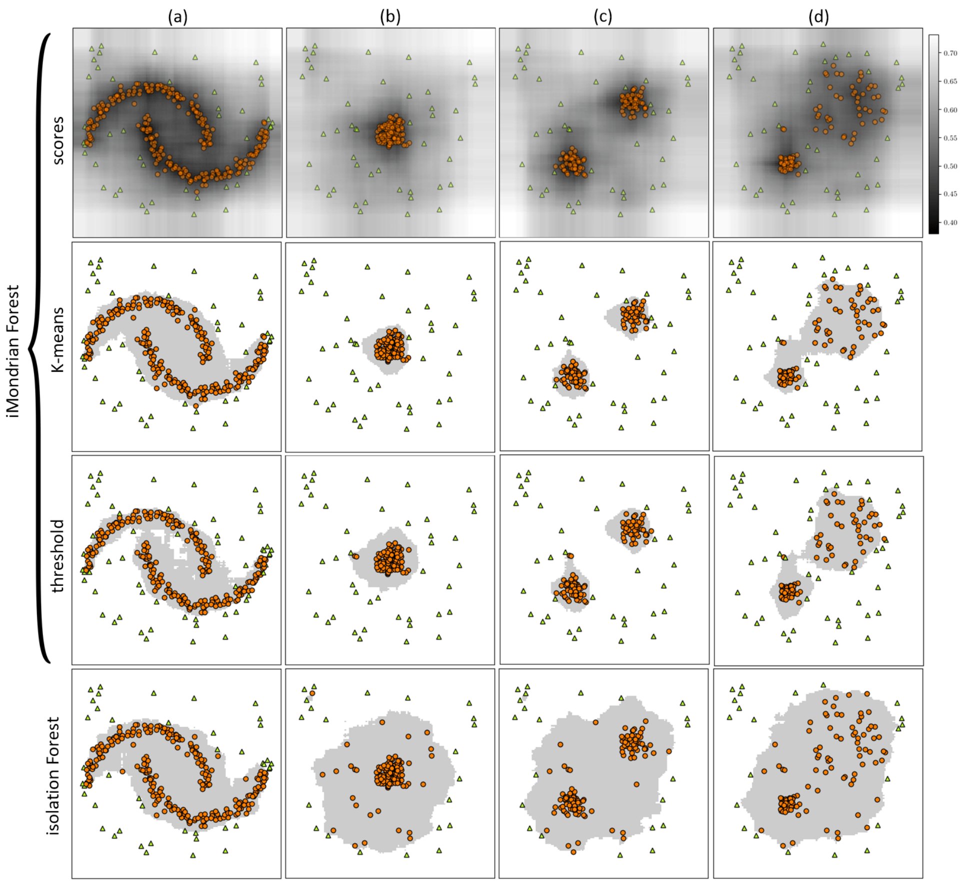

Figure 6.29: Experiments for iMondrian: Comparison of batch anomaly detection in iMondrian and iForests for the synthetic datasets (a)-(d). The orange circles and green triangles correspond to the detected normal and anomalous points, respectively, while shaded gray regions show the partition of space detected as normal.

iMondrian forest clearly performed much better than iForest due to having much fewer false negatives. It is because iMondrian trees take into account the smallest blocks containing the points within a node while iForest considers the whole block.

Online Experiments: I divided every dataset, using stratified sampling, into five subsets with equal amounts of outliers. I used these subsets to simulate streaming data by adding each subset to the existing data in a succession of five steps. Figure 6.30 shows the results of online processing using iMondrian forests on the synthetic datasets. K-means clustering was used in all of these experiments. In set (a), we see that in the second step,

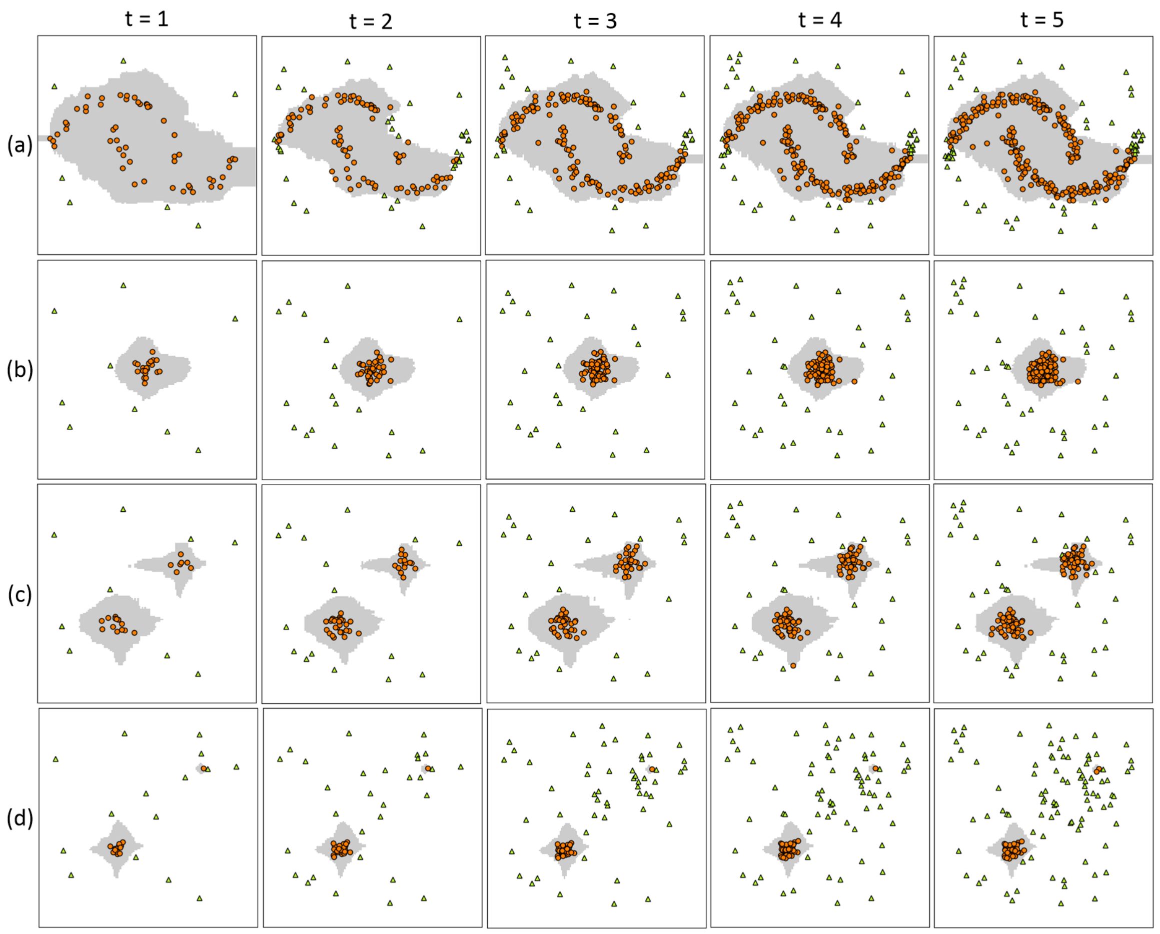

Figure 6.30: Experiments for iMondrian: Online anomaly detection in iMondrian forest for the synthetic datasets (a)-(d). The orange circles and green triangles correspond to the detected normal and anomalous points, respectively, while shaded gray regions show the partition of space detected as normal. Time steps are denoted by t in this figure.

some inliers are falsely recognized as anomalous; however, by receiving more data in the next steps, the structures of iMondrian trees have been modified correctly and those points are recognized correctly as inliers. For set (c), merely some core points of the larger blob are detected as normal. This is because in the initial steps, there happen to be far fewer points from the smaller blob so the algorithm has found that region to be anomalous; however, if that blob had become much denser in the further steps, they would be detected as normal. Overall, for different datasets, the performance of online iMondrian is acceptable.

Table 6.12: Experiments for iMondrian: Comparison of batch anomaly detection methods. Rates are AUC percentage and times are in seconds averaged over the folds. The upward arrows in AUC rates mean iMForest outperforms the other methods.

			WBC	Pima	Thyroid	Satellite	Optdigits	Ionosphere	Wine	SMTP
iMForest	Train:	Time:	2.40 ± 0.01	2.54 ± 0.04	4.96 ± 0.15	16.33 ± 0.18	5.28 ± 0.03	2.27 ± 0.02	0.48 ± 0.00	68.64 ± 1.11
		AUC:	86.35 ± 1.31	63.63 ± 1.02	95.36 ± 0.41	73.93 ± 1.19	72.90 ± 3.49	86.07 ± 0.95	99.01 ± 0.16	86.76 ± 1.47
	Test:	Time:	0.04 ± 0.00	0.07 ± 0.01	0.33 ± 0.02	1.58 ± 0.02	0.35 ± 0.00	0.02 ± 0.00	0.02 ± 0.00	7.42 ± 0.11
		AUC:	86.25 ± 5.02	63.74 ± 9.39	95.37 ± 1.62	73.67 ± 2.33	73.00 ± 7.64	83.99 ± 6.32	99.71 ± 0.28	85.12 ± 14.25
iForest	Train:	Time:	0.14 ± 0.00	0.14 ± 0.00	0.25 ± 0.01	0.46 ± 0.01	0.81 ± 0.01	0.12 ± 0.00	0.08 ± 0.00	4.41 ± 0.05
		AUC:	78.75 ± 1.62 ↑	67.49 ± 1.36	97.89 ± 0.22	70.13 ± 2.13 ↑	68.38 ± 4.64 ↑	84.74 ± 0.94 ↑	79.56 ± 10.59 ↑	90.74 ± 1.37
	Test:	Time:	0.01 ± 0.00	0.01 ± 0.00	0.02 ± 0.00	0.03 ± 0.00	0.03 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.19 ± 0.00
		AUC:	78.81 ± 6.20 ↑	68.00 ± 5.14	97.87 ± 0.84	70.13 ± 3.46 ↑	68.36 ± 8.11 ↑	84.32 ± 6.19	76.09 ± 10.11 ↑	89.44 ± 10.02
LOF	Train:	Time:	0.01 ± 0.00	0.01 ± 0.00	0.04 ± 0.00	0.65 ± 0.00	1.84 ± 0.05	0.01 ± 0.00	0.01 ± 0.00	1.05 ± 0.06
		AUC:	61.12 ± 1.56 ↑	49.91 ± 1.41 ↑	70.26 ± 2.10 ↑	52.65 ± 0.33 ↑	60.84 ± 1.67 ↑	89.59 ± 0.81	98.70 ± 1.29 ↑	53.51 ± 7.25 ↑
	Test:	Time:	0.01 ± 0.00	0.01 ± 0.00	0.05 ± 0.00	0.77 ± 0.01	2.13 ± 0.08	0.01 ± 0.00	0.01 ± 0.00	1.14 ± 0.04
		AUC:	61.94 ± 7.56 ↑	51.36 ± 7.50 ↑	66.17 ± 13.06 ↑	53.26 ± 2.32 ↑	61.12 ± 11.65 ↑	89.87 ± 7.23	92.58 ± 5.69 ↑	56.23 ± 25.90 ↑
SVM	Train:	Time:	0.03 ± 0.00	0.04 ± 0.00	0.27 ± 0.00	3.78 ± 0.00	3.10 ± 0.01	0.01 ± 0.00	0.01 ± 0.00	240.93 ± 3.81
		AUC:	49.40 ± 3.16 ↑	51.93 ± 0.02 ↑	84.36 ± 0.53 ↑	48.54 ± 0.57 ↑	50.52 ± 3.81 ↑	76.23 ± 0.86 ↑	68.59 ± 4.25 ↑	84.14 ± 1.75 ↑
	Test:	Time:	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.18 ± 0.00	0.15 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	4.89 ± 0.09
		AUC:	94.21 ± 1.35	60.01 ± 8.37 ↑	84.49 ± 4.12 ↑	64.04 ± 1.65 ↑	37.49 ± 7.41 ↑	76.61 ± 7.90 ↑	91.13 ± 3.83 ↑	83.06 ± 17.75 ↑

Table 6.13: Experiments for iMondrian: Comparison of online anomaly detection methods. Rates are AUC percentage and times are in seconds. Upward arrows mean better performance of iMForest.

	Stages	Pima	Thyroid	Satellite	Optdigits	Ionosphere	Wine	SMTP	CICIDS
iMForest	Time:	1.25	6.59	13.24	9.85	0.56	0.21	185.33	2.6E3
	AUC:	70.19	95.41	71.94	67.50	86.62	95.65	95.48	71.02
	Time:	1.48	8.40	15.26	11.88	0.65	0.22	364.03	1.0E4
	AUC:	68.07	94.27	73.73	68.00	85.07	98.91	95.01	70.95
	Time:	1.59	9.20	16.65	12.93	0.69	0.23	319.45	5.9E3
	AUC:	65.45	94.65	74.15	66.70	84.27	98.79	96.58	70.76
	Time:	1.72	10.32	18.87	14.45	0.73	0.24	349.33	7.3E3
	AUC:	64.51	94.58	74.00	66.40	83.10	98.64	93.84	70.80
	Time:	1.88	11.29	20.67	15.29	0.79	0.29	393.93	8.6E3
	AUC:	65.50	94.64	73.40	67.10	82.80	97.60	92.87	70.83
Incremental LOF	Time:	0.001	0.006	0.04	0.06	0.001	0.0009	0.71	4.1E2
	AUC:	58.81 ↑	85.61 ↑	54.23 ↑	56.87 ↑	93.06	95.65	94.90 ↑	46.58 ↑
	Time:	0.001	0.02	0.14	0.26	0.001	≈ 0	1.87	1.5E3
	AUC:	55.13 ↑	70.62 ↑	53.76 ↑	60.78 ↑	89.57	98.91	95.44	46.06 ↑
	Time:	0.001	0.04	0.29	0.58	0.003	≈ 0	3.93	3.3E3
	AUC:	53.31 ↑	72.34 ↑	52.33 ↑	62.98 ↑	88.81	91.06 ↑	58.59 ↑	45.99 ↑
	Time:	0.003	0.07	0.52	1.06	0.003	≈ 0	5.39	4.3E3
	AUC:	49.10 ↑	69.61 ↑	51.64 ↑	64.22 ↑	88.93	81.92 ↑	52.79 ↑	46.00 ↑
	Time:	0.005	0.11	0.79	1.68	0.004	≈ 0	8.02	7.9E3
	AUC:	48.40 ↑	68.10 ↑	51.69 ↑	62.75 ↑	90.13	91.51 ↑	49.60 ↑	45.93 ↑

	Stages	Pima	Thyroid	Satellite	Optdigits	Ionosphere	Wine	SMTP
osPCA1	Time:	1.50	9.02	17.72	15.67	0.95	0.25	252.87
	AUC:	80.37	40.42 ↑	26.48 ↑	54.66 ↑	69.95 ↑	86.95 ↑	10.52 ↑
	Time:	1.58	9.24	16.17	16.49	1.20	0.24	252.88
	AUC:	75.10	45.96 ↑	42.43 ↑	57.19 ↑	61.24 ↑	60.86 ↑	17.25 ↑
	Time:	1.85	9.62	17.01	18.57	0.90	0.24	282.63
	AUC:	73.26	53.49 ↑	45.65 ↑	54.24 ↑	53.67 ↑	62.31 ↑	11.41 ↑
	Time:	1.85	9.33	18.49	19.32	0.87	0.25	302.31
	AUC:	72.00	52.91 ↑	47.32 ↑	51.65 ↑	50.92 ↑	67.11 ↑	18.37 ↑
	Time:	1.95	9.54	19.59	20.16	0.89	0.28	321.40
	AUC:	71.34	55.80 ↑	48.08 ↑	51.09 ↑	49.74 ↑	72.35 ↑	24.72 ↑
osPCA2	Time:	0.15	3.51	14.58	11.32	0.12	0.01	2101.3
	AUC:	50.94 ↑	49.93 ↑	49.94 ↑	49.95 ↑	48.88 ↑	47.82 ↑	49.99 ↑
	Time:	2.04	17.35	42.41	33.80	0.95	0.25	5346.3
	AUC:	56.10 ↑	57.00 ↑	54.03 ↑	46.14 ↑	57.02 ↑	58.69 ↑	58.03 ↑
	Time:	2.32	25.88	68.78	54.53	1.06	0.26	9979
	AUC:	59.42 ↑	64.59 ↑	56.33 ↑	40.87 ↑	61.58 ↑	67.63 ↑	68.00 ↑
	Time:	2.81	33.53	69.17	73.20	1.07	0.34	14416
	AUC:	60.02 ↑	64.78 ↑	57.23 ↑	38.97 ↑	62.84 ↑	72.14 ↑	69.13 ↑
	Time:	3.21	42.30	122.71	93.13	1.18	0.40	18678
	AUC:	61.05 ↑	67.19 ↑	57.36 ↑	36.60 ↑	64.51 ↑	75.04 ↑	68.82 ↑

6.2.3.2 Experiments for iMondrian – Real Data

I selected eight various datasets with different characteristics from the outlier detection datasets [108] and one very large dataset, CICIDS 2017 [13]. In CICIDS data, I only used the data of Wednesday and excluded its one categorical feature. The datasets have different sample size, dimensionality, and percentage of outliers. The names of these datasets are listed in Tables 6.12 and 6.13.

Batch Experiments: I compared iMondrian forest with iForest, LOF (with $k = 10$), and one-class SVM (with RBF kernel). The experiments were performed with 10-fold cross validation except for the wine dataset where I used two folds due to small sample size. The average Area Under ROC Curve (AUC) and time of algorithms over the ten folds are reported in Table 6.12. The results are reported for both training and test subsets of data. In most datasets, iMondrian outperforms iForest, LOF, and SVM. In the three datasets Pima, thyroid, and SMTP, the method iForest is slightly better; although, the difference is not significant. In time, iForest is mostly better than iMondrian but its accuracy is often less.

Online Experiments: For the online experiments, I divided datasets into five stages using stratified sampling and introduced the streaming data to the algorithms where each new point was accumulated to previous data. The AUC of a stage is for scores up to that stage. WBC was not used here because there was such a small relative portion of outliers it made the stratified sampling not possible. I compared iMondrian forest with incremental LOF (with $k = 10$) [11], osPCA1 [145], and osPCA2 [86], reported in Table 6.13. The results of CICIDS on osPCA methods are not reported as they did not perform in a reasonable time on these datasets. The AUC of iMondrian forest reported for every stage is the rate for recalculated scores of the available data. In the first stage of osPCA1 and osPCA2, I used decremental PCA approach with oversampling [86]. In different datasets, iMondrian forest has stable performance in different stages which shows its stability over the streaming data. In most cases, iMondrian outperforms all the baseline methods. In terms of time, iMondrian outperforms osPCA1 and osPCA2.

6.3 Summary of the Chapter

In this chapter, I reported the experimental results on the proposed algorithms in data reduction. In dimensionality reduction, I reported the simulation results on WFDA, RDA, SSIM kernel, ISCA, LLISE, QQE, backprojection, Fisher losses, and BUT/BUNCA. In numerosity reduction, I reported the results on PSA, IRMD, CAD, and iMondrian. I showed that the proposed methods are effective in performance, e.g., accuracy, recognition, embedding, reduction, etc.

Chapter 7

Conclusions and Future Directions

7.1 Summary of the Thesis

In this thesis, I proposed several algorithms for data reduction in machine learning and data science. Data reduction can be divided into two main categories which are dimensionality reduction and numerosity reduction. Dimensionality reduction can be categorized into feature extraction and feature selection. Numerosity reduction is branched into prototype selection and prototype generation. This thesis concentrated on feature extraction and prototype selection for data reduction (see Fig. 1.1).

Dimensionality reduction methods can be divided into three categories, i.e., spectral, probabilistic, and neural network-based methods (see Fig. 1.2). Spectral methods have a geometrical point of view and are mostly reduced to the generalized eigenvalue problem. In spectral methods, WFDA, RDA, and image quality aware embedding were proposed. SSIM kernel, ISCA, and LLISE were proposed for image quality aware embedding. I also proposed quantile-quantile embedding as a probabilistic method in which the user can choose the distribution of embedding. Backprojection, Fisher losses, and dynamic triplet sampling using Bayesian updating were other proposed methods in the neural network-based methods. Backprojection is for training shallow networks with a projection-based perspective in manifold learning. In that algorithm, the data are project to a layer and the labels are backprojected (reconstructed) from the last layer to that layer and these two are optimized to match. Two Fisher losses, i.e. FDT and FDC losses, were also proposed for training Siamese triplet networks for increasing and decreasing the inter- and intra-class variances, respectively. I proposed two triplet mining methods, BUT and BUNCA, which are based on Bayesian updating and draw triplet samples stochastically rather than sampling from the existing data instances.

Numerosity reduction methods can be either based on variance, geometry, or ensemble methods (see Fig. 1.2). I proposed PSA and IRMD as the variance-based methods which rank the data points using inter-/intra-class variances and matrix factorization, respectively. In the proposed CAD method, as a geometry-based method, the points are assumed to be the vertices of polyhedron. I also proposed iMondrian forest as an ensemble method. This method, which was not my main project and I collaborated in it, is a novel hybrid of isolation and Mondrian forests.

I also proposed some applications for data reduction in medical image analysis, image processing, and computer vision (see Fig. 1.2). For medical image analysis, I utilized the FDT and FDC losses as well as the BUT and BUNCA triplet mining approaches for embedding the histopathology image patches. I also proposed offline/online triplet mining using extreme distances for histopathology embedding. For image processing and computer vision applications, I proposed Roweisfaces and Roweisposes in the fields of face recognition and 3D action recognition, respectively. These two methods used the proposed RDA subspace learning method. I also proposed anomaly landscape and anomaly path using the proposed CAD method and employed them for image denoising. I reported extensive experiments, on different datasets, to show the effectiveness of the proposed algorithms. I showed that the proposed dimensionality reduction methods can extract informative features for better separation of classes or different patterns, e.g., separation of tumorous and normal histopathology tissues for better cancer diagnosis. I also showed that the proposed numerosity reduction methods can be very useful for ranking data instances based on their importance. Hence, they can help reduce data volumes without a significant drop in performance of machine learning and data science methods.

7.2 Conclusions and Discussions

In this thesis, I tackled different open problems in data reduction for machine learning, in both dimensionality reduction and numerosity reduction.

The open problems in dimensionality reduction which were tackled in the thesis were as follows:

1. Most of the manifold learning methods, such as PCA, LLE, and SNE, use MSE or ℓ_2 norm in their formulation. However, in image fidelity assessment, it is shown that

MSE is not promising enough [134]. SSIM [135, 133] is an effective method for image quality assessment. I proposed SSIM kernel, ISCA, and LLISE for image quality aware manifold embedding.

2. The dimensionality reduction methods either do not specify the distribution of the embedded data points in the embedded space and merely focus on preserving the local or global distances, such as Isomap [126], MVU [136], and Sammon mapping [113], or they only restrict the distribution in the embedded space to be a specific distribution, such as SNE [69] and t-SNE [94]. I proposed QQE to give a freedom to the user for choosing the embedding distribution.

3. There exist several spectral dimensionality reduction methods which are based on the generalized eigenvalue problem [42]. These methods belong to a general optimization problem for eigenvalue decomposition. I proposed RDA as a generalized subspace learning method.

4. The supervised subspace learning methods see the pairs of classes with the same eye; although, the distances and confusion of classes are different from each other. Some classes are closer to each other and harder to separate while some classes are apart. I proposed WFDA to tackle this problem by weighting the distances of pairs of classes where the more confused (closer) classes are assigned larger weights to be emphasized for separation.

5. In terms of increasing amd decreasing the inter- and intra-class variances, respectively, the triplet [118] and contrastive [65] losses have similarity with FDA. The proposed Fisher losses, FDT and FDC losses, are fusions of FDA and triplet/contrastive losses.

6. The sampling based triplet mining methods in the literature sample the triplets from the existing embedded data instances [139] so it does not use the stochastic information of the embedding space. I proposed BUT and BUNCA for dynamic triplet sampling.

The open problems in numerosity reduction and anomaly detection which I tackled in this proposal are as follows:

1. Most of the numerosity reduction methods are proposed merely for classification [28]. The proposed PSA, IRMD, and CAD are task agnostic usable for all classification, regression, and clustering.

2. Most of existing numerosity reduction methods do not rank the data and only retain a subset of data by removing the rest of data. Ranking the instances can be useful which are applicable in our proposed PSA, IRMD, and CAD.

3. There is a gap in the literature on numerosity reduction and anomaly detection for methods based on geometry. The proposed CAD, iCAD, and their kernel variants fill this gap by assuming every point to be a vertex of a polyhedron using its neighbors. The more anomalous point has more polyhedron curvature.

4. Isolation forest is an existing method for batch anomaly detection and Mondrian forest is an existing method for online random forest. Isolation forest is not capable of handling streaming data. The proposed iMondrian forest is a fusion of these two methods which can handle both offline and online (streaming) data.

An important conceptual contribution of this thesis was giving an organization and taxonomy to data reduction, dimensionality reduction, and numerosity reduction. I found the gaps in different parts of the proposed taxonomy and proposed novel algorithms to fill those gaps and open problems this taxonomy led to. In terms of new proposed algorithms, I would like to highlight the three most significant and novel approaches. The first is proposing RDA which generalized subspace learning methods, such as PCA, FDA, and SPCA, to be special cases of a general family of algorithms. The second is the idea of using SSIM in manifold and subspace learning by introducing the concept of image structure manifold. In other words, I combined the two worlds of image quality assessment and machine learning which opened a new field of research. The third is the novel geometrical perspective to anomaly detection and proposing the concept of anomaly paths which can be also useful for image denoising. All of the proposed feature extraction methods can be used to help improve the performance of classification, regression, or clustering. Moreover, the proposed numerosity reduction methods help remove the redundant or misleading data instances. Both proposed dimensionality and numerosity reduction find insights from data by the classical or theoretical methods.

7.3 Advantages and Costs of the Proposed Methods

It is known in mathematics, so as in life, that gaining something results in loosing something else. Therefore, all methods have some cost or drawback. In the following, I explain the costs of the proposed methods in this thesis:

- WFDA: The solution of FDA is a generalized eigenvalue problem [42]. WFDA, finds the weights for better separation of confused classes. However, the cost of WFDA is having iterative optimization which is slower than optimization of traditional FDA.

- RDA: The solution of RDA is a generalized eigenvalue problem [42]. It is a generalized family of infinite number of subspace learning methods. However, because of the limits of the generalized eigenvalue problem, it cannot handle very large volumes of data.

- SSIM kernel: Many machine learning and manifold learning methods can use the proposed SSIM kernel. As calculation of SSIM is patch-wise in images [135, 133], calculation of the SSIM kernel is time consuming. This problem is not caused by my method but it exists in all image filtering methods, including convolutional networks [62]. In image filtering methods, such as calculation of the SSIM kernel, we need to move pixel by pixel to calculate the desired measurement.

- ISCA: ISCA is the formulation of PCA but with SSIM distance rather than ℓ_2 norm. The solution of PCA is an eigenvalue problem (see [35]) while the solution of ISCA is found iteratively by ADMM. Therefore, ISCA makes PCA suitable for the human visual system and image structure subspace learning but at the cost of slower optimization in finding the subspace.

- LLISE: LLISE is the formulation of LLE but with SSIM distance rather than ℓ_2 norm. The solution of LLE is an eigenvalue problem (see [37]) while the solution of LLISE is found iteratively by ADMM. Therefore, LLISE makes LLE suitable for the human visual system and image structure manifold learning but at the cost of slower optimization in unfolding the manifold.

- QQE: The proposed QQE gives the choice of embedding distribution to users while the existing manifold learning methods do not do that. The optimization of QQE uses the diagonal quasi-Newton's method, which is iterative, and takes some time to converge because of this. Moreover, the time complexity of QQE is $\mathcal{O}(n^3)$.

- Backprojection: The proposed backprojection is a novel training algorithm for neural networks. In one of his recent seminars, Geoffrey Hinton, one of the important machine learning researchers, has mentioned that it is time to develop new training algorithms for neural nets after the tremendous development of backpropagation. Backprojection gives projection-based insight to networks. It is moderately faster than backpropagation because it updates weights layer by layer and not weight by weight as in backpropagation. However, backprojection could be more difficult to extend to convolutional networks. More investigation is needed to extend it to convolutional layers.

- Fisher losses: The proposed Fisher losses combine FDA from classic machine learning and triplet loss in Siamese network from modern deep learning. It finds an embedding space which increases and decreases the inter- and intra-class variances, respectively. Compared to triplet and contrastive losses, the Fisher losses do not have any large costs. The only difficulty being that dealing with the last linear layer makes it a little more complicated than the triplet and contrastive losses.

- BUT and BUNCA: The triplets for training Siamese networks can be sampled in one of these ways:

  - Sampled randomly from classes
  - Sampled by extreme distances of points
  - Sampled by distribution but from existing points
  - Sampled stochastically from distributions of classes

  The first three methods of triplet sampling already exist in the literature but the fourth one is novel and done by the proposed BUT and BUNCA which sample triplets for training Siamese networks. The cost of the proposed dynamic triplet sampling methods is the overhead of updating the distributions of classes in the embedding space which makes neural network training a little slower.

- PSA: The proposed PSA algorithm considers different trade-offs for finding the most informative data instances. The RANSAC iterations and calculation of scatters in PSA take time because of the large number of iterations. PSA can be improved further to be faster on large volumes of data.

- IRMD: The proposed IRMD finds the most informative instances based on similarity of data points to the most informative directions of data. It cannot handle very large volumes of data due to limitations of matrix factorization for decomposing large matrices.

- CAD: The proposed CAD and iCAD are among the first numerosity reduction methods based on geometry and topology of data, to the best of my knowledge. Its cost is calculation of cosine or kernels between the neighbors of every point.

- iMondrian: The proposed iMondrian combines the two existing forests, isolation forest and Mondrian forest, which were for batch anomaly detection and online classification/regression, respectively. iMondrian extends isolation forest for online anomaly detection using the Mondrian forest formulation. Its cost is a little more complicated construction of trees in iMondrian compared to isolation forest.

Note that since the proposed tools are mostly used for data reduction as pre-processing which comes before the classification, regression, or clustering tasks, they can be applied off-line; hence, their computational complexity is not usually the deciding factor.

7.4 When to Use the Proposed Methods

In this section, I explain when the reader can use the proposed methods in this thesis and when the proposed methods are not applicable.

7.4.1 When Do We Use the Proposed Methods?

This thesis concentrated on data reduction in machine learning and data science. It provides a taxonomy for data reduction and divides data reduction into dimensionality reduction and numerosity reduction (see Table 1.1 and Figs. 1.1 and 1.2). Consider the matrix of data where the data instances can be staked row-wise or column-wise in this matrix. If the data instances are row-wise in this matrix, the rows and columns correspond to instances and features (dimensions), respectively. In terms of data reduction, one can reduce the number of rows or columns in the data matrix. Hence, we can have numerosity reduction, dimensionality reduction, or both. The concepts and methods introduced in this thesis can be used for data reduction to make the data matrix smaller for better data representation, class separations, storage efficiency, improving learning running time, etc. I concentrated on feature extraction and prototype selection in this thesis (see Fig. 1.1).

This thesis is very useful especially when the data matrix is large and one wants to reduce data for better data representation, class separations, storage efficiency, and improving running time of learning algorithms. The proposed methods can be used as pre-processing for feature extraction or informative instance selection before classification, regression, or clustering tasks.

The approach of dimensionality reduction methods depends on having or not having labels. Most of the supervised manifold learning methods use the idea of FDA proposed by

Ronald A. Fisher a century ago [25]. They all try to increase and decrease the inter-class and intra-class variances, respectively. In this way, the classes become more separated while they collapse to smaller data clouds in the embedding space. Most of the unsupervised manifold learning methods have the idea of local fitting while thinking to global structure of data which was proposed by Sam T. Roweis [110, 114] twenty years ago. They try to preserve the locality of neighborhoods in the embedding space hoping that the global structure of data is preserved when the manifold is unfolded. My proposed dimensionality reduction methods in this thesis mostly have these approaches.

As explained before, dimensionality reduction methods can be categorized into spectral, probabilistic, and neural network-based approaches. The numerosity reduction methods can be divided into variance-based, geometry-based, and ensemble methods. In this thesis, I found the gaps and open problems in each of these categories and proposed novel algorithms to fill those gaps. Those gaps were mentioned in Section 7.2.

As a conclusion, this thesis is useful to reduce the data matrix in terms of either numerosity or dimensionality for better data representation. An example of this usage is gigapixel images, such as WSI of histopathology, which I have worked on in thesis as well. The gigapixel images are very huge in terms of dimensionality, i.e., number of pixels. One should divide a gigapixel image into many patches and work on the image patches rather than the whole image for the computational reasons. This patching results in a huge number of patches which require numerosity reduction. Moreover, working in pixel space of image patches is not recommended because a small shift in image can ruin everything. One should learn an embedding space for the image patches and work on patches in that space. To summarize, before patching, dimensionality reduction is sensed to be required because of huge number of pixels. After patching, numerosity reduction is required for the large number of patches. Also, dimensionality reduction is needed to extract features because working in pixel space is not recommended for the explained reason before.

7.4.2 When Are the Proposed Methods not Applicable?

As was explained in Section 7.3, the proposed methods have some shortcomings. Therefore, there are some cases for which the proposed methods are not applicable. The spectral dimensionality reduction methods, including the proposed WFDA, RDA, SSIM kernel, ISCA, and LLISE, are not applicable for very large volume of data. The direct spectral methods cannot handle huge dimensionality of data such as huge images with high number of pixels. The dual spectral methods, however, cannot handle high number of instances. If we have either huge dimensionality or numerosity, one can choose the direct or dual methods appropriately; however, if both numerosity and dimensionality of data are huge, the spectral method are not applicable at all.

The time complexity of QQE is $\mathcal{O}(n^3)$ and as such it cannot be used for very large number of data instances. The current version of the proposed backprojection cannot be used for convolutional networks. Moreover, its current version is applicable for only shallow networks. The proposed methods for deep networks, including FDT, FDC, BUT, and BUNCA, require enough number of data instances to be trained without overfitting [34]. Deep networks, except some recent few-shot learning methods, require enough data instances so this issue applies to all deep networks [62]. Hence, my proposed deep methods are not well suited to the datasets containing only a small number of instances.

In PSA, the RANSAC iterations and the summations in scatter calculations cause the bottlenecks for time complexity [30]. Although PSA can be slightly modified to improve its complexity [30], its current version cannot handle huge number of data instances. IRMD also has the same problem which is because of limitations of matrix factorization when dealing with huge matrices. Note that using dual methods for some of the factorization-based subspace learning methods, such as PCA and SPCA, can handle data matrices with either small dimensionality or numerosity. However, if both dimensionality and numerosity are huge, the proposed IRMD cannot be applied. The proposed CAD and iCAD work properly on different data types but increasing the number of neighbors may increase the run time. Having small sub-graphs of data rather than having a connected graph may require a large number of neighbors. Hence, CAD and iCAD may not be applicable or efficient for data with not connected sub-graphs. The proposed iMondrian is suitable for different datasets but it may be very slow in constructing trees for very large number of instances.

7.5 Other Open Problems and Future Directions

There exist some other open problems which I leave for possible future research. In the following, I enumerate and explain some possible open problems:

- Open problems in the proposed methods for dimensionality reduction:
    1. SSIM kernel, ISCA, and LLISE are some example methods for image quality aware embedding. Other methods can be proposed, by replacing the ℓ_2 or Frobenius norm with SSIM distance, for learning the image structure manifold.

2. There exist several possible future directions for QQE. The first future direction is to improve the time complexity of QQE which is $\mathcal{O}(n^3 + ndk)$. Since the complexity of QQE is $\mathcal{O}(n^3)$, dealing with a large number of instances would be a challenge for this initial version. Handling out-of-sample data is another possible future direction for QQE. QQE uses the least squares problem which is not very robust, so another possibility would be to investigate high breakdown estimators for robust regression to make QQE more robust and faster.

3. Backprojection was proposed for training shallow feedforward neural networks. One can investigate extending this method for convolutional networks as well as deep networks.

4. Triplet sampling and mining can be further improved for better stochastic dynamic sampling. Our proposed BUT and BUNCA assume a Gaussian distribution for the embedding of every class. This is because the Gaussian distribution is the most common distribution according to the central limit theorem. One can consider multivariate Gaussian or mixture of other distribution [41] for embedding of every class.

- Open problems in the proposed methods for numerosity reduction:

  1. Because of the essence of matrix decomposition, it is not feasible for large volume of data. One can extend the proposed IRMD algorithm for large volume of data.

  2. The proposed CAD method using polyhedron curvature, as well as the proposed anomaly path (see Eq. (5.6)), may be used to propose a curvature preserving manifold embedding algorithm.

  3. As a future direction for iMondrian forest, one can develop the idea of isolation-based anomaly detection in other online tree structures such as binary space partitioning forest [23] which is based on the binary partitioning process.

Moreover, note that the proposed data reduction tools are used as pre-processing which mostly can be used before the classification, regression, and clustering tasks; therefore, the computational complexity of the proposed methods is not a big issue because they can be mostly done off-line. Although the computational complexity of some methods have been evaluated (e.g., see [52] for QQE, [30] for PSA, and [93] for iMondrian), we will analyze the improve the computation complexity of other proposed methods in the future work.

References

[1] Julien Ah-Pine. Normalized kernels as similarity indices. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 362–373. Springer, 2010.

[2] Jonathan L Alperin. *Local representation theory: Modular representations as an introduction to the local representation theory of finite groups*, volume 11. Cambridge University Press, 1993.

[3] Shaeela Ayesha, Muhammad Kashif Hanif, and Ramzan Talib. Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59:44–58, 2020.

[4] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371, 2011.

[5] Mohammad Hossein Basiri, Benyamin Ghojogh, Nasser L Azad, Sebastian Fischmeister, Fakhri Karray, and Mark Crowley. Distributed nonlinear model predictive control and metric learning for heterogeneous vehicle platooning with cut-in/cut-out maneuvers. In *59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020.

[6] Peter N Belhumeur, João P Hespanha, and David J Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):711–720, 1997.

[7] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[8] Yoshua Bengio, Jean-françcois Paiement, Pascal Vincent, Olivier Delalleau, Nicolas L Roux, and Marie Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. In *Advances in neural information processing systems*, pages 177–184, 2004.

[9] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[12] Dominique Brunet, Edward R Vrscay, and Zhou Wang. On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing*, 21(4):1488–1499, 2012.

[13] Canadian Institute for Cybersecurity. Intrusion detection evaluation dataset (CICIDS2017). https://www.unb.ca/cic/datasets/ids-2017.html, 2017.

[14] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations (ICLR)*, 2016.

[15] Lee AD Cooper, Elizabeth G Demicco, Joel H Saltz, Reid T Powell, Arvind Rao, and Alexander J Lazar. Pancancer insights from the cancer genome atlas: the pathologist's perspective. *The Journal of pathology*, 244(5):512–524, 2018.

[16] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. Chapman and Hall/CRC, 2000.

[17] René Descartes. Progymnasmata de solidorum elementis. *Oeuvres de Descartes*, X:265—276, 1890.

[18] Subhra Sankar Dhar, Biman Chakraborty, and Probal Chaudhuri. Comparison of multivariate distributions using quantile–quantile plots and related tests. *Bernoulli*, 20(3):1484–1506, 2014.

[19] Shengyong Ding, Liang Lin, Guangrun Wang, and Hongyang Chao. Deep feature learning with relative distance comparison for person re-identification. *Pattern Recognition*, 48(10):2993–3003, 2015.

[20] Dheeru Dua and Casey Graff. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2017.

[21] George S Easton and Robert E McCulloch. A multivariate generalization of quantile-quantile plots. *Journal of the American Statistical Association*, 85(410):376–386, 1990.

[22] Chris Ellis, Syed Zain Masood, Marshall F Tappen, Joseph J LaViola, and Rahul Sukthankar. Exploring the trade-off between accuracy and observational latency in action recognition. *International Journal of Computer Vision*, 101(3):420–436, 2013.

[23] Xuhui Fan, Bin Li, and Scott Anthony Sisson. Binary space partitioning forests. In *22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3022–3031, 2019.

[24] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[25] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[26] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[27] Benjamin Fruchter. *Introduction to factor analysis.* Van Nostrand, 1954.

[28] Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE transactions on pattern analysis and machine intelligence*, 34(3):417–435, 2012.

[29] Samuele Gasparrini, Enea Cippitelli, Ennio Gambi, Susanna Spinsante, Jonas Wåhslén, Ibrahim Orhan, and Thomas Lindh. Proposal and experimental evaluation of fall detection solution based on wearable and depth data fusion. In *International Conference on ICT Innovations*, pages 99–108. Springer, 2015.

[30] Benyamin Ghojogh. Principal sample analysis for data ranking. In *Canadian Conference on Artificial Intelligence*, pages 579–583. Springer, 2019.

[31] Benyamin Ghojogh and Mark Crowley. Principal sample analysis for data reduction. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pages 350–357. IEEE, 2018.

[32] Benyamin Ghojogh and Mark Crowley. Instance ranking and numerosity reduction using matrix decomposition and subspace learning. In *Canadian Conference on Artificial Intelligence*, pages 160–172. Springer, 2019.

[33] Benyamin Ghojogh and Mark Crowley. Linear and quadratic discriminant analysis: Tutorial. *arXiv preprint arXiv:1906.02590*, 2019.

[34] Benyamin Ghojogh and Mark Crowley. The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial. *arXiv preprint arXiv:1905.12787*, 2019.

[35] Benyamin Ghojogh and Mark Crowley. Unsupervised and supervised principal component analysis: Tutorial. *arXiv preprint arXiv:1906.03148*, 2019.

[36] Benyamin Ghojogh, Mark Crowley, and Fakhri Karray. Addressing the mystery of population decline of the rose-crested blue pipit in a nature preserve using data visualization. *arXiv preprint arXiv:1903.06671*, 2019.

[37] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Locally linear embedding and its variants: Tutorial and survey. *arXiv preprint arXiv:2011.10925*, 2020.

[38] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Multidimensional scaling, sammon mapping, and isomap: Tutorial and survey. *arXiv preprint arXiv:2009.08136*, 2020.

[39] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Stochastic neighbor embedding with gaussian and student-t distributions: Tutorial and survey. *arXiv preprint arXiv:2009.10301*, 2020.

[40] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey. *arXiv preprint arXiv:2101.00734*, 2021.

[41] Benyamin Ghojogh, Aydin Ghojogh, Mark Crowley, and Fakhri Karray. Fitting a mixture distribution to data: tutorial. *arXiv preprint arXiv:1901.06708*, 2019.

[42] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Eigenvalue and generalized eigenvalue problems: Tutorial. *arXiv preprint arXiv:1903.11240*, 2019.

[43] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Fisher and kernel Fisher discriminant analysis: Tutorial. *arXiv preprint arXiv:1906.09436*, 2019.

[44] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Hidden markov model: Tutorial. *engrXiv*, 2019.

[45] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Image structure subspace learning using structural similarity index. In *International Conference on Image Analysis and Recognition*, pages 33–44. Springer, 2019.

[46] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Locally linear image structural embedding for image structure manifold learning. In *International Conference on Image Analysis and Recognition*, pages 126–138. Springer, 2019.

[47] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Principal component analysis using structural similarity index for images. In *International Conference on Image Analysis and Recognition*, pages 77–88. Springer, 2019.

[48] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Roweis discriminant analysis: A generalized subspace learning method. *arXiv preprint arXiv:1910.05437*, 2019.

[49] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Anomaly detection and prototype selection using polyhedron curvature. In *Canadian Conference on Artificial Intelligence*, pages 238–250. Springer, 2020.

[50] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Backprojection for training feedforward neural networks in the input and feature spaces. In *International Conference on Image Analysis and Recognition*, pages 16–24. Springer, 2020.

[51] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Generalized subspace learning by Roweis discriminant analysis. In *International Conference on Image Analysis and Recognition*, pages 328–342. Springer, 2020.

[52] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Quantile-quantile embedding for distribution transformation, manifold embedding, and image embedding with choice of embedding distribution. *arXiv preprint arXiv:2006.11385*, 2020.

[53] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Roweisposes, including eigenposes, supervised eigenposes, and Fisherposes, for 3D action recognition. *arXiv preprint arXiv:2006.15736*, 2020.

[54] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Theoretical insights into the use of structural similarity index in generative models and inferential autoencoders. In *International Conference on Image Analysis and Recognition*, pages 112–117. Springer, 2020.

[55] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Weighted Fisher discriminant analysis in the input and feature spaces. In *International Conference on Image Analysis and Recognition*, pages 3–15. Springer, 2020.

[56] Benyamin Ghojogh, Hoda Mohammadzade, and Mozhgan Mokari. Fisherposes for human action recognition using Kinect sensor data. *IEEE Sensors Journal*, 18(4):1612–1627, 2018.

[57] Benyamin Ghojogh, Hadi Nekoei, Aydin Ghojogh, Fakhri Karray, and Mark Crowley. Sampling algorithms, from survey sampling to Monte Carlo methods: Tutorial and literature review. *arXiv preprint arXiv:2011.00901*, 2020.

[58] Benyamin Ghojogh, Ali Saheb Pasand, Fakhri Karray, and Mark Crowley. Quantized Fisher discriminant analysis. *arXiv preprint arXiv:1909.03037*, 2019.

[59] Benyamin Ghojogh, Maria N Samad, Sayema Asif Mashhadi, Tania Kapoor, Wahab Ali, Fakhri Karray, and Mark Crowley. Feature selection and feature extraction in pattern analysis: A literature review. *arXiv preprint arXiv:1905.02845*, 2019.

[60] Benyamin Ghojogh, Milad Sikaroudi, Sobhan Shafiei, Hamid R Tizhoosh, Fakhri Karray, and Mark Crowley. Fisher discriminant triplet and contrastive losses for training Siamese networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[61] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2005.

[62] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[63] Jianping Gou, Yuanyuan Yang, Zhang Yi, Jiancheng Lv, Qirong Mao, and Yongzhao Zhan. Discriminative globality and locality preserving graph embedding for dimensionality reduction. *Expert Systems with Applications*, 144:113079, 2020.

[64] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[65] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 2, pages 1735–1742. IEEE, 2006.

[66] Ji Hun Ham, Daniel D Lee, Sebastian Mika, and Bernhard Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *International Conference on Machine Learning*, 2004.

[67] 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.

[68] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[69] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.

[70] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[71] Prateek Jain and Purushottam Kar. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–336, 2017.

[72] Neal Jean, Sherrie Wang, Anshul Samar, George Azzari, David Lobell, and Stefano Ermon. Tile2vec: Unsupervised representation learning for spatially distributed data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3967–3974, 2019.

[73] Oscar Jimenez-del Toro, Sebastian Otálora, Mats Andersson, Kristian Eurén, Martin Hedlund, Mikael Rousson, Henning Müller, and Manfredo Atzori. Analysis of histopathology images: From traditional machine learning to deep learning. In *Biomedical Texture Analysis*, pages 281–314. Elsevier, 2017.

[74] Ian Jolliffe. *Principal component analysis*. Springer, 2011.

[75] Michael I. Jordan. The conjugate prior for the normal distribution. Technical report, University of California, Berkeley, 2010.

[76] Khamisi Kalegele, Hideyuki Takahashi, Johan Sveholm, Kazuto Sasai, Gen Kitagata, and Tetsuo Kinoshita. On-demand data numerosity reduction for learning artifacts. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 152–159. IEEE, 2012.

[77] Shivam Kalra, HR Tizhoosh, Sultaan Shah, Charles Choi, Savvas Damaskinos, Amir Safarpoor, Sobhan Shafiei, Morteza Babaie, Phedias Diamandis, Clinton JV Campbell, et al. Pan-cancer diagnostic consensus through searching archival histopathology images using artificial intelligence. *NPJ Digital Medicine*, 3(1):1–15, 2020.

[78] Jakob Nikolas Kather, Johannes Krisam, Pornpimol Charoentong, Tom Luedde, Esther Herpel, Cleo-Aron Weis, Timo Gaiser, Alexander Marx, Nektarios A Valous, Dyke Ferber, et al. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. *PLoS medicine*, 16(1), 2019.

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

[80] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[81] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations (ICLR)*, 2013.

[82] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[83] Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. Mondrian forests: Efficient online random forests. In *Advances in neural information processing systems*, pages 3140–3148, 2014.

[84] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database. http://yann.lecun.com/exdb/mnist/, 1998. Accessed: Jan 2018.

[85] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[86] Yuh-Jye Lee, Yi-Ren Yeh, and Yu-Chiang Frank Wang. Anomaly detection via online oversampling principal component analysis. *IEEE transactions on knowledge and data engineering*, 25(7):1460–1470, 2013.

[87] Bo Li, Yan-Rui Li, and Xiao-Long Zhang. A survey on Laplacian eigenmaps based manifold learning methods. *Neurocomputing*, 335:336–351, 2019.

[88] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[89] Chuan Liu, Wenyong Wang, Meng Wang, Fengmao Lv, and Martin Konan. An efficient instance selection algorithm to reconstruct training set for support vector machine. *Knowledge-Based Systems*, 116:58–73, 2017.

[90] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[91] Marco Loog, Robert PW Duin, and Reinhold Haeb-Umbach. Multiclass linear dimension reduction by weighted pairwise Fisher criteria. *IEEE transactions on pattern analysis and machine intelligence*, 23(7):762–766, 2001.

[92] Rohit Lotlikar and Ravi Kothari. Fractional-step dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):623–627, 2000.

[93] Haoran Ma, Benyamin Ghojogh, Maria N Samad, Dongyu Zheng, and Mark Crowley. Isolation Mondrian forest for batch and online anomaly detection. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020.

[94] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[95] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. In *International Conference on Learning Representations (ICLR)*, 2016.

[96] John I Marden. Positions and QQ plots. *Statistical Science*, 19(4):606–614, 2004.

[97] Steenn Markvorsen. Curvature and shape. In *Yugoslav Geometrical Seminar, Fall School of Differential Geometry, Yugoslavia*, pages 55–75, 1996.

[98] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29), 2018.

[99] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop*, pages 41–48. IEEE, 1999.

[100] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017.

[101] Kevin P Murphy. Conjugate Bayesian analysis of the Gaussian distribution. Technical report, University of British Colombia, 2007.

[102] Hadi NekoeiQachkanloo, Benyamin Ghojogh, Ali Saheb Pasand, and Mark Crowley. Artificial counselor system for stock investment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9558–9564, 2019.

[103] R Wayne Oldford. Self-calibrating quantile–quantile plots. *The American Statistician*, 70(1):74–90, 2016.

[104] Daniel Otero, Davide La Torre, Oleg V Michailovich, and Edward R Vrscay. Alternate direction method of multipliers for unconstrained structural similarity-based optimization. In *International Conference Image Analysis & Recognition*, pages 20–29. Springer, 2018.

[105] Daniel Otero and Edward R Vrscay. Unconstrained structural similarity-based optimization. In *International Conference Image Analysis and Recognition*, pages 167–176. Springer, 2014.

[106] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[107] Parisa Abdolrahim Poorheravi, Benyamin Ghojogh, Vincent Gaudet, Fakhri Karray, and Mark Crowley. Acceleration of large margin metric learning for nearest neighbor classification using triplet mining and stratified sampling. In *Conference on Vision and Intelligent Systems (CVIS)*, 2020.

[108] Shebuti Rayana. Outlier detection data sets. http://odds.cs.stonybrook.edu/, 2019.

[109] Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[110] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[111] Daniel M Roy and Yee Teh. The Mondrian process. *Advances in neural information processing systems*, 21:1377–1384, 2008.

[112] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[113] John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409, 1969.

[114] Lawrence K Saul and Sam T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, 4:119–155, 2003.

[115] Bernhard Schölkopf. The kernel trick for distances. In *Advances in neural information processing systems*, pages 301–307, 2001.

[116] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[117] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[118] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[119] Milad Sikaroudi, Benyamin Ghojogh, Fakhri Karray, Mark Crowley, and Hamid R Tizhoosh. Batch-incremental triplet sampling for training triplet networks using Bayesian updating theorem. In *International Conference on Pattern Recognition*. IEEE, 2020.

[120] Milad Sikaroudi, Benyamin Ghojogh, Fakhri Karray, Mark Crowley, and Hamid R Tizhoosh. Offline versus online triplet mining based on extreme distances of histopathology patches. In *15th International Symposium on Visual Computing (ISVC)*. Springer, 2020.

[121] Milad Sikaroudi, Benyamin Ghojogh, Fakhri Karray, Mark Crowley, and HR Tizhoosh. Magnification generalization for histopathology image embedding. In *IEEE International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2021.

[122] Milad Sikaroudi, Amir Safarpoor, Benyamin Ghojogh, Sobhan Shafiei, Mark Crowley, and Hamid R Tizhoosh. Supervision and source domain impact on representation learning: A histopathology case study. In *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2020.

[123] David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning Proceedings 1994*, pages 293–301. Elsevier, 1994.

[124] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of machine learning research*, 2(Nov):67–93, 2001.

[125] Sriram Ganapathi Subramanian, Jaspreet Singh Sambee, Benyamin Ghojogh, and Mark Crowley. Decision assist for self-driving cars. In *Canadian Conference on Artificial Intelligence*, pages 381–387. Springer, 2018.

[126] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[127] The Digital Technology Group. AT&T laboratories cambridge. http://cam-orl.co.uk/facedatabase.html, 1994. Accessed: June 2017.

[128] Hamid R Tizhoosh. Opposition-based learning: a new scheme for machine intelligence. In *International Conference on Computational Intelligence for Modelling, Control and Automation*, volume 1, pages 695–701. IEEE, 2005.

[129] Isaac Triguero, Joaquín Derrac, Salvador Garcia, and Francisco Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(1):86–100, 2012.

[130] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[131] Laurens Van Der Maaten. Learning a parametric embedding by preserving local structure. In *Artificial Intelligence and Statistics*, pages 384–391, 2009.

[132] Yi-Qing Wang. An analysis of the Viola-Jones face detection algorithm. *Image Processing On Line*, 4:128–148, 2014.

[133] Zhou Wang and Alan C Bovik. Modern image quality assessment. *Synthesis Lectures on Image, Video, and Multimedia Processing*, 2(1):1–156, 2006.

[134] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[135] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[136] Kilian Q Weinberger, Fei Sha, and Lawrence K Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 106. ACM, 2004.

[137] D Randall Wilson and Tony R Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000.

[138] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421, 1972.

[139] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

[140] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27. IEEE, 2012.

[141] Hong Xuan, Abby Stylianou, and Robert Pless. Improved embeddings with easy positive triplet mining. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2474–2482, 2020.

[142] M-H Yang, Narendra Ahuja, and David Kriegman. Face recognition using kernel eigenfaces. In *Proceedings of the IEEE international conference on image processing*, volume 1, pages 37–40. IEEE, 2000.

[143] Ming-Hsuan Yang. Kernel Eigenfaces vs. kernel Fisherfaces: Face recognition using kernel methods. In *Proceedings of the fifth IEEE international conference on automatic face and gesture recognition*, pages 215–220, 2002.

[144] Jieping Ye. Least squares linear discriminant analysis. In *Proceedings of the 24th international conference on machine learning*, pages 1087–1093. ACM, 2007.

[145] Yi-Ren Yeh, Zheng-Yi Lee, and Yuh-Jye Lee. Anomaly detection via over-sampling principal component analysis. In *New Advances in Intelligent Decision Technologies*, pages 449–458. Springer, 2009.

[146] Xu-Yao Zhang and Cheng-Lin Liu. Confused distance maximization for large category dimensionality reduction. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 213–218. IEEE, 2012.

[147] Xu-Yao Zhang and Cheng-Lin Liu. Evaluation of weighted Fisher criteria for large category dimensionality reduction in application to Chinese handwriting recognition. *Pattern Recognition*, 46(9):2599–2611, 2013.

[148] Xiaoming Zhao and Shiqing Zhang. Facial expression recognition using local binary patterns and discriminant kernel locally linear embedding. *EURASIP journal on Advances in signal processing*, pages 1–9, 2012.

APPENDICES

Appendix A

Publications in My PhD

A list of my publications is also available in my Google Scholar page: Click here

* The stars on the names mean equal contribution and the order of starred names is completely random.

There are some other publications, published during my PhD, which were not related to my PhD or were left from my master's. Those publications are not listed here.

A.1 Published Papers

1. B. Ghojogh*, M. Sikaroudi*, F. Karray, M. Crowley, H.R. Tizhoosh. "Magnification Generalization for Histopathology Image Embedding." IEEE International Symposium on Biomedical Imaging (ISBI) 2021. [Link to paper] Ref: [121]

2. B. Ghojogh*, M. Sikaroudi*, F. Karray, M. Crowley, H.R. Tizhoosh. "Batch-Incremental Triplet Sampling for Training Triplet Networks Using Bayesian Updating Theorem." IEEE International Conference on Pattern Recognition (ICPR) 2020. [Link to paper] Ref: [119]

3. B. Ghojogh*, P.A. Poorheravi*, V. Gaudet, F. Karray, M. Crowley, 2020. "Acceleration of Large Margin Metric Learning for Nearest Neighbor Classification Using Triplet Mining and Stratified Sampling." Conference on Vision and Intelligent Systems (CVIS), Published in Journal of Computational Vision and Imaging Systems, 2020. [Link to paper] Ref: [107]. Awarded the best paper award of conference.

4. B. Ghojogh*, M. Sikaroudi*, Amir Safarpoor, F. Karray, M. Crowley, H.R. Tizhoosh. "Offline versus Online Triplet Mining based on Extreme Distances of Histopathology Patches." 15th International Symposium on Visual Computing (ISVC), Springer, 2020. [Link to paper] Ref: [120]

5. B. Ghojogh*, H. Ma*, M. N. Samad*, D. Zheng, M. Crowley. "Isolation Mondrian Forest for Batch and Online Anomaly Detection." IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020. [Link to paper] Ref: [93]

6. M. H. Basiri, B. Ghojogh, N. L. Azad, S. Fischmeister, F. Karray, M. Crowley. "Distributed Nonlinear Model Predictive Control and Metric Learning for Heterogeneous Vehicle Platooning with Cut-in/Cut-out Maneuvers." IEEE International Conference on Decision and Control (CDC), 2020. [Link to paper] Ref: [5]

7. B. Ghojogh*, M. Sikaroudi*, A. Safarpoor*, S. Shafiei, M. Crowley, H. R. Tizhoosh. "Supervision and Source Domain Impact on Representation Learning: A Histopathology Case Study." International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2020. [Link to paper] Ref: [122]

8. B. Ghojogh, M. Sikaroudi, S. Shafiei, H. R. Tizhoosh, F. Karray, M. Crowley. "Fisher Discriminant Triplet and Contrastive Losses for Training Siamese Networks." International Joint Conference on Neural Networks (IJCNN), IEEE, 2020. [Link to paper] Ref: [60]

9. B. Ghojogh, F. Karray, M. Crowley. "Generalized Subspace Learning by Roweis Discriminant Analysis." In International Conference on Image Analysis and Recognition, Springer, Cham, 2020. [Link to paper] Ref: [51]

10. B. Ghojogh, F. Karray, M. Crowley. "Backprojection for Training Feedforward Neural Networks in the Input and Feature Spaces." In International Conference on Image Analysis and Recognition, Springer, Cham, 2020. [Link to paper] Ref: [50]

11. B. Ghojogh, F. Karray, M. Crowley. "Theoretical Insights into the Use of Structural Similarity Index In Generative Models and Inferential Autoencoders." In International Conference on Image Analysis and Recognition, Springer, Cham, 2020. [Link to paper] Ref: [54]

12. B. Ghojogh, M. Sikaroudi, H. R. Tizhoosh, F. Karray, M. Crowley. "Weighted Fisher Discriminant Analysis in the Input and Feature Spaces." In International Conference on Image Analysis and Recognition, Springer, Cham, 2020. [Link to paper] Ref: [55]

13. B. Ghojogh, F. Karray, M. Crowley. "Anomaly Detection and Prototype Selection Using Polyhedron Curvature." In Canadian Conference on Artificial Intelligence, pp. 238-250. Springer, Cham, 2020. [Link to paper] Ref: [49]

14. B. Ghojogh, F. Karray, M. Crowley. "Locally Linear Image Structural Embedding for Image Structure Manifold Learning." In International Conference on Image Analysis and Recognition, Springer, Cham, pp. 126-138, 2019. [Link to paper] Ref: [46]

15. B. Ghojogh, F. Karray, M. Crowley. "Principal Component Analysis Using Structural Similarity Index for Images." In International Conference on Image Analysis and Recognition, Springer, Cham, pp. 77-88, 2019. [Link to paper] Ref: [47]

16. B. Ghojogh, F. Karray, M. Crowley. "Image Structure Subspace Learning Using Structural Similarity Index." In International Conference on Image Analysis and Recognition, Springer, Cham, pp. 33-44, 2019. [Link to paper] Ref: [45]

17. B. Ghojogh, M. Crowley. "Instance ranking and numerosity reduction using matrix decomposition and subspace learning." In Canadian Conference on Artificial Intelligence, Springer, Cham, pp. 160-172, 2019. [Link to paper] Ref: [32]

18. B. Ghojogh. "Principal sample analysis for data ranking." In Canadian Conference on Artificial Intelligence, Springer, Cham, pp. 579-583, 2019. [Link to paper] Ref: [30]

19. B. Ghojogh*, H. NekoeiQachkanloo*, A. Saheb Pasand*, M. Crowley "Artificial Counselor System for Stock Investment." Proceedings of the AAAI Conference on Artificial Intelligence, vol 33, 2019. [Link to paper] Ref: [102]

20. B. Ghojogh, M. Crowley. "Principal sample analysis for data reduction." 2018 2nd International Conference on Big Knowledge (ICBK). IEEE, pp. 350-357, 2018. [Link to paper] Ref: [31]

21. B. Ghojogh*, S. Ganapathi Subramanian*, J. Singh Sambee*, M. Crowley. "Decision Assist for Self-driving Cars." In Canadian Conference on Artificial Intelligence. Springer, Cham, pp. 381-387, 2018. [Link to paper] Ref: [125]

A.2 In-Progress Book

1. B. Ghojogh, M. Crowley, F. Karray, A. Ghodsi. "Elements of Dimensionality Reduction and Manifold Learning." to be submitted to a prestigious publication.

A.3 Pre-print Papers

Many of these pre-prints are going to be chapters of an upcoming book in manifold learning and dimensionality reduction. Some of the pre-prints are also under review.

1. B. Ghojogh, A. Ghodsi, F. Karray, M. Crowley. "Factor Analysis, Probabilistic Principal Component Analysis, Variational Inference, and Variational Autoencoder: Tutorial and Survey." arXiv preprint arXiv:2101.00734, 2021. [Link to paper]. Ref: [40]

2. B. Ghojogh, F. Karray, M. Crowley. "Quantile-Quantile Embedding for Distribution Transformation, Manifold Embedding, and Image Embedding with Choice of Embedding Distribution." arXiv:2006.11385 (2020). [Link to paper] Under review. Ref: [52]

3. B. Ghojogh*, H. Nekoei*, A. Ghojogh*, F. Karray, M. Crowley. "Sampling Algorithms, from Survey Sampling to Monte Carlo Methods: Tutorial and Literature Review." arXiv preprint arXiv:2011.00901, 2020. [Link to paper] Under review. Ref: [57]

4. B. Ghojogh, A. Ghodsi, F. Karray, M. Crowley. "Locally Linear Embedding and its Variants: Tutorial and Survey." arXiv preprint arXiv:1908.09288, 2020. [Link to paper] Ref: [37]

5. B. Ghojogh, A. Ghodsi, F. Karray, M. Crowley. "Stochastic Neighbor Embedding with Gaussian and Student-t Distributions: Tutorial and Survey." arXiv preprint arXiv:2009.10301, 2020. [Link to paper] Ref: [39]

6. B. Ghojogh, A. Ghodsi, F. Karray, M. Crowley. "Multidimensional Scaling, Sammon Mapping, and Isomap: Tutorial and Survey." arXiv preprint arXiv:2009.08136, 2020. [Link to paper] Ref: [38]

7. B. Ghojogh, F. Karray, M. Crowley. "Roweisposes, Including Eigenposes, Supervised Eigenposes, and Fisherposes, for 3D Action Recognition." arXiv preprint arXiv:2006.15736 (2020). [Link to paper] Ref: [53]

8. B. Ghojogh, F. Karray, M. Crowley. "Hidden Markov Model: Tutorial." (2019). [Link to paper] Ref: [44]

9. B. Ghojogh, F. Karray, M. Crowley. "Roweis Discriminant Analysis: A Generalized Subspace Learning Method." arXiv preprint arXiv:1910.05437, 2019. [Link to paper] Ref: [48]

10. B. Ghojogh*, A. Saheb Pasand*, F. Karray, M. Crowley. "Quantized Fisher Discriminant Analysis." arXiv preprint arXiv:1909.03037, 2019. [Link to paper] Ref: [58]

11. B. Ghojogh, Fakhri Karray, M. Crowley. "Fisher and Kernel Fisher Discriminant Analysis: Tutorial." arXiv preprint arXiv:1906.09436, 2019. [Link to paper] Ref: [43]

12. B. Ghojogh, M. Crowley. "Unsupervised and Supervised Principal Component Analysis: Tutorial." arXiv preprint arXiv:1906.03148, 2019. [Link to paper] Ref: [35]

13. B. Ghojogh, M. Crowley. "Linear and Quadratic Discriminant Analysis: Tutorial." arXiv preprint arXiv:1906.02590, 2019. [Link to paper] Ref: [33]

14. B. Ghojogh, M. Crowley. "The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial." arXiv preprint arXiv:1905.12787, 2019. [Link to paper] Ref: [34]

15. B. Ghojogh*, M. N. Samad*, S. Asif Mashhadi*, T. Kapoor*, W. Ali*, F. Karray, M. Crowley. "Feature selection and feature extraction in pattern analysis: A literature review." arXiv preprint arXiv:1905.02845, 2019. [Link to paper] Ref: [59]

16. B. Ghojogh, F. Karray, M. Crowley. "Eigenvalue and generalized eigenvalue problems: Tutorial." arXiv preprint arXiv:1903.11240, 2019. [Link to paper] Ref: [42]

17. B. Ghojogh, M. Crowley, F. Karray. "Addressing the Mystery of Population Decline of the Rose-Crested Blue Pipit in a Nature Preserve using Data Visualization." arXiv preprint arXiv:1903.06671, 2019. [Link to paper] Ref: [36]

18. B. Ghojogh, A. Ghojogh, M. Crowley, F. Karray. "Fitting a mixture distribution to data: tutorial." arXiv preprint arXiv:1901.06708, 2019. [Link to paper] Ref: [41]