KU LEUVEN

M.Sc. Statistics and Data Science
Faculty of Sciences

Exploring Unbalanced Data in Restricted Kernel Machines

Jierong Wen
Ye Liu

Supervisor:
Prof. Dr. Ir. Johan A. K. Suykens
Affiliation *ESAT-STADIUS*

Mentor:
Ir. Bram De Cooman
Ir. Sonny Achten
Affiliation *ESAT-STADIUS*

Thesis presented in fulfillment of the requirements for the degree of Master of Science in Statistics and Data Science

Academic year 2023-2024

Preface

First and foremost, we extend our sincerest gratitude to our mentors, Bram De Cooman and Sonny Achten, whose continuous support has been invaluable throughout this academic year. Their deep insights and constructive feedback have profoundly shaped this work, providing a strong foundation for our exploration and discovery. We would also like to thank ourselves for our dedication and perseverance in this process. It was our combined efforts and commitment that brought this thesis to fruition. We faced numerous challenges, but together, we navigated through them all with resilience and determination. Finally, we thank Prof. Johan Suykens for promoting this thesis, the assessors for reading this work and others for providing us with encouragement and support throughout our research.

Jierong Wen & Ye Liu

Contents

Contents ii

Abstract iv

List of Figures v

List of Tables viii

List of Algorithms x

Abbreviations xi

Division of Contributions xii

1 Introduction 1

2 Literature Review 4

2.1 Deep generative learning 4

2.1.1 Generative learning vs. discriminative learning 4

2.1.2 State-of-the-art: VAEs and GANs 5

2.2 Kernel-based learning 7

2.3 Theoretical framework of Generative restricted kernel machine . 8

2.3.1 A brief overviews on PCA and Kernel PCA 9

2.3.2 Basic formulation of RKM 12

2.3.3 Basic formulation of Generative RKM 16

2.3.4 Related variants of Generative RKM 20

2.4 Learning from unbalanced data 21

2.4.1 Re-sampling and re-weighting technique 21

2.4.2 Unbalanced data in supervised and unsupervised learning 23

2.4.3 Unbalanced data in generative learning 24

3 Problem Statement 25

4 Proposed Methods 28

4.1 Mitigating data imbalance in supervised setting 28
4.1.1 Inverse frequency sampling 28
4.1.2 Inverse frequency sampling in the framework of Gen-RKM 29
4.1.3 Extension to conditional generation in Gen-RKM 30
4.2 Mitigating data imbalance in unsupervised setting 31
4.2.1 Ridge leverage score sampling 32
4.2.2 Incorporating RLS sampling within the framework of Gen-RKM 33
4.2.3 Extension: Other measurement for imbalance sampling . 37

5 Experiments and Results **39**
5.1 Experiment set-up 39
5.1.1 Datasets 39
5.1.2 Evaluation process & metrics 41
5.1.3 Hyperparameter details 42
5.2 Experiments on inverse frequency sampling under supervised setting 44
5.2.1 Unbalanced MNIST/Fashion MNIST 44
5.2.2 Effect of inverse frequency sampling on conditional generation in Gen-RKM 46
5.3 Experiments on weighted sampling schemes under unsupervised setting 47
5.3.1 Synthetic ring/grid dataset 48
5.3.2 Unbalanced 012-MNIST 49
5.3.3 Unbalanced MNIST 50
5.3.4 Unbalanced Fashion MNIST 52
5.3.5 Additional studies 53
5.4 Summary and discussion 57

6 Conclusions **60**

Bibliography **62**

Appendices **71**

A Supplementary material for Chapter 4 **72**
A.1 Derivation of the conditional distribution of GMM 72

B Additional results of Chapter 5 **74**
B.1 Generated Samples: Comparison Between Vanilla Gen-RKM and Gen-RKM with RLS Sampling Adaptation 74

Abstract

Unbalanced data has always been a challenge in the domain of machine learning for a long time. While this issue has been extensively studied in supervised learning, it continues to present difficulties in generative learning, especially in scenarios where data distributions are highly unbalanced. Such imbalances can significantly compromise the efficacy of generative models by introducing biases towards over-represented modes, thereby reducing the diversity of the generated outputs. In this thesis, a comprehensive examination of resampling strategies aimed at addressing data imbalances is conducted within the framework of the Generative Restricted Kernel Machine (Gen-RKM). Specifically, three sampling methods, inverse frequency sampling, rigid leverage score (RLS) sampling, and isolation forest score (Iforest) sampling, are evaluated across various unbalanced datasets. The results indicate that inverse frequency sampling performs well in a supervised setting where data labels are available. Meanwhile, both RLS sampling and Iforest sampling effectively enhance generation diversity in an unsupervised context, with RLS sampling showing superior performance over other sampling strategies when utilized with appropriate feature maps.

List of Figures

2.1 Schematic diagram of vanilla GAN architecture 7

3.1 Schematic representation of Gen-RKM. $\phi(.)$ is the feature map network, while $\psi(.)$ corresponds to the pre-image network. $\mathbf{W}$ is the interconnection matrix yielded from KPCA, mapping feature representations to the latent space $\mathcal{L}$. Latent points can be remapped back to the feature space via the transpose of the transformation matrix $\mathbf{W}^\top$. 25

3.2 Visualizations of latent spaces of Gen-RKM (first row) and VAE (second row) trained on balanced (first column) and unbalanced 012-MNIST datasets (second columns). The minority mode is depicted in green color. Sizes of latent space of both Gen-RKM and VAE are set to 10, and only the first two dimensions are visualized. 27

3.3 Generated samples from the balanced/unbalanced 012-MNIST dataset by Gen-RKM (first two figures) and VAE (last two figures). The first and the third figures display the generations from a balanced dataset, while the second and the fourth figures show the generations from an unbalanced dataset. 27

4.1 A toy example illustrates using RLS sampling can help capture minority modes in the data. The orange curve depicts the PDF of a Gaussian mixture distribution with 1 majority mode in the middle and 2 minority modes on two sides, where the PDF is defined as $p = 0.05\mathcal{N}(-10, 1) + 0.9\mathcal{N}(0, 1) + 0.05\mathcal{N}(10, 1)$. Blue points are a collection of random samples from that pdf, where the height of each point corresponds to the RLS for that individual point. RLSs are computed based on a Gaussian kernel with $\gamma = 10^{-3}$ and $\sigma = 3$. One can observe that samples from minority modes generally have higher RLSs, thus sampling based on the RLSs could contribute to balance the data. This figure is reproduced from [77]. 33

5.1 2D synthetic data: Ring(left) and Grid(right) 40
5.2 Number of generated samples in each class for the unbalanced MNIST dataset: Vanilla Gen-RKM (top left) vs. Gen-RKM with inverse frequency sampling (top right). For the Fashion MNIST dataset: Vanilla Gen-RKM (down left) vs. Gen-RKM with inverse frequency sampling (down right). A rebalancing effect is visible 45
5.3 Random generation of unbalanced MNIST dataset: vanilla Gen-RKM (top right) and Gen-RKM with inverse frequency sampling (top left). Fashion MNIST dataset: vanilla Gen-RKM (down right) and Gen-RKM with inverse frequency sampling (down left) 45
5.4 Conditional generations produced by multi-view Gen-RKM on the unbalanced MNIST dataset (top row) and the unbalanced Fashion MNIST dataset (bottom row). Figures in left column is generated by vanilla Gen-RKM, and figures in the right column are generated by Gen-RKM adapted with inverse frequency sampling. Images in each column of each figure are generated conditionally on a particular label. Imbalance ratios for both datasets are set to 0.1. 47
5.5 Number of samples in each mode from RING dataset: Original RING(left), Vanilla Gen-RKM (middle) and Gen-RKM with RLS sampling (right). 48
5.6 Visualizations of random generations on synthetic datasets: Vanilla Gen-RKM on RING (top left), Gen-RKM with RLS sampling on RING (top right), Vanilla Gen-RKM on GRID (down left) and Gen-RKM with RLS sampling on GRID (down right). . . . 49
5.7 Number of generated samples per mode from different RKM models on the unbalanced 012-MNIST dataset. Each row indicates that the dataset is generated with a specific balance ratio. Each column corresponds to a different model used. The minority mode in each figure is depicted in red color and, the majority modes are depicted in blue. 51
5.8 Number of generated samples per mode from different RKM models on the unbalanced MNIST dataset. 53
5.9 Number of generated samples per mode from different RKM models on the unbalanced Fashion MNIST dataset. 54

5.10 Visualizations of the latent spaces of two Gen-RKMs trained on the unbalanced Fashion MNIST dataset (imbalance ratio is 0.1). Only the first two principal components in the latent spaces are displayed. The left figure is developed based on a vanilla Gen-RKM, whereas Gen-RKM in the right figure is trained with the RLS sampling (class) manner. Latent representations of minority modes are colored as red, while those of the majority classes are in blue. Note that the size of each point in the right figure indicates the frequency of being oversampled. 55

B.1 Generated images from the unbalanced 012-MNIST dataset by vanilla RKM (left) and RLS (class) sampling adapted RKM (right). Images are highlighted by a red border if classified as minority modes. 74

B.2 Generated images from the unbalanced MNIST dataset by vanilla RKM (left) and RLS (class) sampling adapted RKM (right). Images are highlighted by a red border if classified as minority modes. 75

B.3 Generated images from the unbalanced Fashion MNIST dataset by vanilla RKM (left) and RLS (class) sampling adapted RKM (right). Images are highlighted by a red border if classified as minority modes. 75

List of Tables

5.1 Hyperparameter settings for Gen-RKM 43
5.2 Details of network architectures used in Gen-RKM. All convolutional layers and transposed convolutional layers have stride 2 and padding 1. Pre-image map is the reverse of feature map architecture, except that a sigmoid activation function is implemented for the output layer [8]. 43
5.3 Results of experiments on 2D synthetic datasets (Experiments are replicated over 10 runs with random initialization). 48
5.4 Results of experiments on the unbalanced 012-MNIST dataset under different imbalance ratios. Experiments are replicated over 5 runs with random initializations. Means and standard deviations (enclosed in parentheses) for each metric are reported. 50
5.5 Results of experiments on the unbalanced MNIST dataset under different imbalance ratios. Experiments are replicated over 3 runs with random initializations. Means and standard deviations (enclosed in parentheses) for each metric are reported. "Minority mean" refers to the average number of samples generated from each minority mode. 52
5.6 Results of experiments on unbalanced Fashion MNIST dataset. Experiments are replicated over 3 runs with random initializations. Means and standard deviations (enclosed in parentheses) for each metric are reported. "Minority mean" refers to the average number of samples generated from each minority mode. . 53
5.7 Ablation study over the impact of different pre-trained classifiers on the performance of RLS-RKM (class) on the unbalanced 012-MNIST dataset. Minority mode is highlighted in bold, with the imbalance ratio set to 0.1. The results are averaged over 5 runs of replicated experiments. 54
5.8 Results of comparison to VAE and RLS-VAE on the unbalanced 012-MNIST dataset with imbalance ratio 0.1. RLSs are computed by explicit feature map based on a pre-trained classifier. . 56

5.9 Training times in seconds (averaged over 3 runs) of different weighted sampling implementations in Gen-RKM. For the unbalanced 012-MNIST, the maximum epoch number is set to 100, while the maximum epoch number in unbalanced MNIST and Fashion MNIST is 150. The batch size is uniformly set to 328 across all cases. All experiments are conducted using a single NVIDIA GeForce RTX 3080 GPU (12GB). 56
5.10 Ablation study over the significance of mini-batch sampling and final step resampling in RLS-RKM. This experiment is conducted on the unbalanced 012-MNIST dataset with imbalance ratio of 0.05. RLSs are computed by an explicit feature map based on a pre-trained classifier. 57

List of Algorithms

1 Training and generation algorithm for one-view Gen-RKM (Dual) [8] 18
2 inverse frequency sampling for one-view Gen-RKM (Dual) . . 29
3 Conditional generation algorithm in Gen-RKM 31
4 RLS Sampling in Gen-RKM with explicit feature map based on a pre-trained classifier 35
5 RLS Sampling in Gen-RKM with a shared feature map 36

Abbreviations

DRKM Deep Restricted Kernel Machine.

ELBO Evidence Lower Bound.

FID Fréchet Inception Distance.

GAN Generative Adversarial Network.

Gen-RKM Generative Restricted Kernel Machine.

GMM Gaussian Mixture Model.

KL Divergence Kullback-Leibler Divergence.

KPCA Kernel Principal Component Analysis.

PCA Principal Component Analysis.

RBF Radial Basis Function.

RKM Restricted Kernel Machine.

RLS Ridge Leverage Score.

Stiefel RKM Stiefel-Restricted Kernel Machine.

SVD Singular Value Decomposition.

SVM Support Vector Machine.

UMAP Uniform Manifold Approximation and Projection.

VAE Variational Autoencoder.

Division of Contributions

Section	Author
1 Introduction	Jierong Wen
2.1 Deep generative learning	Jierong Wen
2.2 Kernel-based learning	Jierong Wen
2.3 Theoretical framework of Generative restricted kernel machine	Jierong Wen
2.4 Learning from unbalanced data	Ye Liu
3 Problem Statement	Ye Liu
4.1 Mitigating data imbalance in supervised setting	Common
4.2 Mitigating data imbalance in unsupervised setting	Common
5.1 Experiment set-up	Jierong Wen
5.2 Experiments on inverse frequency sampling under supervised setting	Common
5.3 Experiments on weighted sampling schemes under unsupervised setting	Common
5.4 Summary and discussion	Jierong Wen
6 Conclusions	Ye Liu

Chapter 1

Introduction

The field of machine learning has witnessed significant advancements in recent years, driven by the explosive growth in data availability and computational power. In today's cutting-edge field of artificial intelligence, generative learning has become a highly popular area of research, especially with the rise of some state-of-the-art generative language AI like ChatGPT. By effectively learning the data distribution in either an explicit or an implicit form, highly realistic samples can be randomly generated [1]. However, despite these technological strides, certain challenges still persist, particularly when modeling unbalanced data distributions, which is a common issue in many real-world scenarios [2].

Unbalanced data, where certain classes or modes are underrepresented, keeps posing a significant challenge to various machine learning or deep learning models. In the context of discriminative learning, the inequality between classes could cause classification models to poorly generalize to minority classes, leading to a poor prediction performance on those underrepresented classes [2], [3]. Currently, the problem of data imbalance in classification models has been well-studied. Some traditional and effective solutions including resampling and reweighting can be implemented during the training phase to rebalance the data.

In the realm of generative learning, data imbalance could also lead to similar detrimental effects as seen in the domain of classification, where the generative model disproportionately favors overrepresented groups while overlooking minority modes [4], [5]. As a result, the frequency and quality of generated samples from minority modes are generally inferior to those of the majority classes. Simply speaking, unbalanced training data results in biased or unfair generation from the generative model. Unlike in the con-

text of classification, there is a significant scarcity of research focused on addressing data imbalance in the setting of generative learning. Although much effort has been devoted to addressing the mode collapse problem in GANs pursuing a more diverse generation, most of these approaches do not consider the setting of unbalanced data [6], [7]. Furthermore, difficulties in correcting data imbalance in generative learning can be concluded in two key points. First, generative models typically involve complex neural network architectures and optimization schemes, making it impractical to directly apply techniques used for addressing data imbalance in classification. Instead, specific modifications according to different models are often required. Second, generative learning is typically performed in an unsupervised manner, where label information is not provided, making it less straightforward to identify and measure data imbalance.

This thesis aims to address the challenges of biased generation or mode missing brought from unbalanced training data within the framework of Gen-RKM, a relatively novel generative model proposed by Pandey et al [8]. New synergies between kernel principal component analysis and deep neural network architectures have been made in Gen-RKM, thereby unlocking potential new research directions in the field of kernel-based learning. However, the capability of Gen-RKM to handle unbalanced data remains unexplored. Specifically, our work centers on two key research questions:

- How does the generation of Gen-RKM get affected when unbalanced data is given, and what is the nature of this impact?
- How can we extend Gen-RKM to make it generalize well on minority modes in the unbalanced data?

Contribution The contributions of this thesis can be summarized in three main parts:

- We show that data imbalance leads to biased generation in Gen-RKM, identifying that this issue arises from a distorted latent space, where latent points corresponding to minority modes are blurred or intermixed with those of majority classes.
- Different weighted sampling schemes including inverse frequency sampling, RLS sampling and Iforest sampling are incorporated with the Gen-RKM framework in this thesis. A comprehensive set of experiments is conducted and their effectiveness is verified.

– We further extend Gen-RKM's capability to generate samples conditioned on a given label. By combining inverse frequency sampling with conditional generation, we show that it is possible to synthesize minority instances even when the training data is extremely unbalanced.

Thesis Organization The rest of the thesis is structured as follows. Chapter 2 reviews the theoretical foundation of Gen-RKM and discusses some related works in unbalanced data learning. Next, the problem of mode missing or mode collapse in Gen-RKM caused by unbalanced data is formally outlined and analyzed in Chapter 3. After that, Chapter 4 introduces various weighted sampling schemes to tackle data imbalance problems under both supervised and unsupervised settings. In Chapter 5, a comprehensive experimental study is carried out to thoroughly assess the performance of different sampling techniques. Finally, the conclusion of this thesis and some suggestions of future work are given in Chapter 6.

Chapter 2

Literature Review

In this chapter, a comprehensive literature overview is provided. We start with a brief introduction to deep generative learning in Section 2.1 and kernel-based learning in Section 2.2. The basic theoretical formulation of Gen-RKM is then presented in Section 2.3. Lastly, we review related works that address data imbalance in the context of both classification and generative learning in Section 2.4.

2.1 Deep generative learning

Deep generative learning has always been a fascinating frontier in the field of artificial intelligence for the last decade. The reason for that is its remarkable capability to generate almost indistinguishable synthetic samples from real-world objects. Generative models have diverse applications in the real world, including painting creation, speech synthesis, language learning, and many others. Below we will discuss the difference between generative learning and traditional discriminative learning, as well as some state-of-the-art models in generative learning.

2.1.1 Generative learning vs. discriminative learning

The key distinction between generative approaches and discriminative approaches is their different underlying probability inference procedures. Mathematically speaking, given a dataset with feature-label pairs $\{\boldsymbol{x}, \boldsymbol{y}\}$, discriminative learning is interested in modeling the conditional probability of $\boldsymbol{y}$ given feature $\boldsymbol{x}$, namely $p(\boldsymbol{y}|\boldsymbol{x})$. In contrast with that, generative learning tends

to capture the joint probability $p(\boldsymbol{x}, \boldsymbol{y})$, enabling it to generate new samples by random sampling from the learned distributions. Note that it is also very common to just model $p(\boldsymbol{x})$ in generative learning since the information on labels is not always available in practice. In general, generative models need to estimate the overall distribution of the data, whereas discriminative classifiers merely aim to find the optimal decision boundary in the data space. This fundamental distinction explains why training generative models is generally more computationally expensive than training discriminative models.

2.1.2 State-of-the-art: VAEs and GANs

Generative learning has gone through significant advancements over recent decades thanks to the developments in deep neural network architectures, with various types of generative models emerging. Among these, Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) have achieved great success for their widespread adoption. Below, we provide a brief introduction to these two models. One can refer to [9] for more detailed overviews on other types of generative models.

Variational Autoencoders: VAE is a type of latent variable model combining the probabilistic framework with an encoder-decoder architecture, first proposed by Kingma and Welling in 2013 [10]. Let input data be $\boldsymbol{x} \in \mathcal{X}$ and latent variables be $\boldsymbol{z} \in \mathcal{Z}$ with prior distribution $p(\boldsymbol{z})$ (usually pre-defined as a Gaussian distribution). Since the joint distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ is implicitly defined in VAE, the conditional distribution of $\boldsymbol{x}$ given latent variable $\boldsymbol{z}$ can be denoted as $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ where $\boldsymbol{\theta}$ is the parameter vector. A natural parameters estimations procedure is via maximizing marginal log-likelihood of $\boldsymbol{x}$, where

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z}. \tag{2.1}$$

However, optimizing on marginal log-likelihood is intractable because of the existence of an integral part. To tackle this problem, Kingma et al. proposed using evidence lower bound (ELBO) to approximate the exact marginal like-

lihood, which can be written as

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \log \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} \\ &\geq \int q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log p(\boldsymbol{z}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})] \\ &= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}_{\text{log-likelihood}} - \underbrace{D_{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})]}_{\text{KL divergence}} = \mathcal{L}_{ELBO}.\end{aligned} \tag{2.2}$$

$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is known as proposal distribution (or recognition model) to approximate the true prior and $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is often called generation model, corresponding to encoder and decoder part in VAE respectively. The first term (log-likelihood term) in the training objective can be approximated by

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] \approx \frac{1}{N} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}^{(i)}) \tag{2.3}$$

with $\{\boldsymbol{z}^{(i)}\}_{i=1}^{N}$ sampled from proposal distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. Kingma et al. suggested using a reparameterization trick where $\boldsymbol{z}^{(i)} = g_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{\epsilon}^{(i)})$ with $\epsilon^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$. $g_{\boldsymbol{\phi}}(.)$ is a differentiable and deterministic function that ensures the differentiability of the stochastic sampling process, allowing gradient-based optimization schemes can be effectively used for training VAEs.

Generative Adversarial Networks: GANs were first introduced by Goodfellow et al. in 2014 [6], bringing revolutionary impacts to the field of generative learning due to their exceptional ability to generate highly realistic images through adversarial training schemes. Unlike in VAE, GAN learns the underlying distribution of data without explicit density estimation, which means we do not have to parametrically pre-define the distribution of data. A GAN typically consists of two multilayer neural networks, namely a discriminator $\mathcal{D} : \mathbb{R}^d \to \{0, 1\}$ and a generator $\mathcal{G} : \mathbb{R}^l \to \mathbb{R}^d$, where d is the dimension of input data and l is the dimension of latent space. Discriminator is simply a binary classifier that distinguishes real data from fake data (i.e. data generated by generator), whereas generator is for image synthesis given a noise input $\boldsymbol{z}$. The general architecture of GAN is shown in figure 2.1. The training objective in vanilla GAN is written as

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{\boldsymbol{x} \sim p_{\mathcal{X}}(\boldsymbol{x})}[\log \mathcal{D}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\mathcal{Z}}(\boldsymbol{z})}[\log(1 - \mathcal{D}(\mathcal{G}(\boldsymbol{z})))], \tag{2.4}$$

where probability of discriminator giving the right classification is maximized in the first expectation term while generator is trained to fool the discriminator. The whole optimization problem can be intuitively regarded as a min-max game with two players, each side wants to defeat the other. The global optimality of 2.4 has been proven to be $p_{\mathcal{X}}(\boldsymbol{x}) = p_{\mathcal{G}}(\boldsymbol{x})$, which means the distribution of fake samples and the distribution of real samples are identical under optimality condition [6].

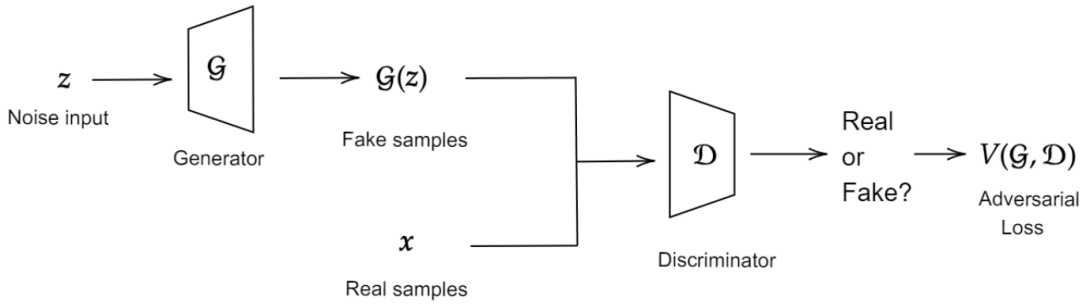

Figure 2.1: Schematic diagram of vanilla GAN architecture

2.2 Kernel-based learning

Kernel methods were first introduced by Vapnik (1995), gaining great attention through their successful application onto Support Vector Machines (SVMs) [11], [12]. Kernel-based learning extends traditional linear approaches to non-linear scenarios by projecting input data into a high-dimensional feature space using a non-linear mapping function $\varphi(.)$. The core motivation behind that is one can expect linearly separable patterns in the feature map space, thus traditional linear techniques can learn more complex patterns from non-linear data after feature mapping. One great challenge is that features may lie in very high dimensions, possibly infinite, which makes computing $\varphi(.)$ very expensive or even intractable.

Thanks to Mercer's theorem, it is possible to implement kernel methods without explicitly knowing the form of $\varphi(.)$ making them both powerful and computationally convenient. More precisely, given input data $\{\boldsymbol{x}_i\}_{i=1}^N$ with $\boldsymbol{x} \in \mathbb{R}^d$, Mercer's theorem states that there exists a feature mapping $\varphi : \mathbb{R}^d \to \mathcal{H}$ and a kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that

$$K(\boldsymbol{x}, \boldsymbol{z}) = \varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{z}), \tag{2.5}$$

if and only if $K(.,.)$ is a positive-definite function [13]. The above theorem implies that kernel K can be written as the dot product of two data points, and the collection of kernels on all pairs of training examples in the data set is defined as the kernel matrix (or Gram matrix) where

$$\mathbf{K} = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_N, \boldsymbol{x}_1) & K(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}. \tag{2.6}$$

The computational problem can be then addressed by pre-computing the kernel matrix since the size of the kernel matrix is fixed on $N \times N$. The general procedure of working on kernel matrix $\mathbf{K}$ is usually referred to as *kernel trick*.

Some representative applications of kernel methods in machine learning include Support Vector Machines (SVMs) [12], spectral clustering [14], Kernel Principal Component Analysis (KPCA) [15], Gaussian Processes [16], and etc. Beyond that, inspired by the powerful representation learning capabilities of deep learning architectures, there is growing interest in finding new synergies between traditional kernel methods and deep neural networks [17], [18]. One notable example is Deep Restricted Kernel Machines (DRKM), which encapsulates KPCA, SVMs and possibly some other kernel learning techniques into one unified deep learning architecture via conjugate feature duality [19]. One can refer to [20], [21] for more detailed overviews and explanations on kernel-based learning.

2.3 Theoretical framework of Generative restricted kernel machine

The theoretical framework of Generative restricted kernel machine (Gen-RKM) algorithm is discussed in this section. Starting by reviewing PCA and KPCA in subsection 2.3.1, the general restricted kernel machine (RKM) framework is then illustrated in subsection 2.3.2. Next, subsection 2.3.3 showcases the basic training and generation algorithm of Gen-RKM under both single-view and multi-view scenarios. Besides, some related variants of Gen-RKM are introduced in subsection 2.3.4.

2.3.1 A brief overviews on PCA and Kernel PCA

Basic formulation of PCA Principal Component Analysis (PCA) is a widely-used unsupervised algorithm for dimensionality reduction while preserving as much variability as possible in the data [22], [23]. Intuitively, PCA searches for a linear transformation such that maximizes the variance of the projected data in a lower-dimensional subspace. Orthonormal vectors that span this subspace after projection is often called *principal components* (PCs). In the mathematical setting, one can formulate PCA as a constrained optimization problem. Consider a data matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{d \times N}$ (here we assume data has already been centered for simplicity), the *maximum variance formulation* [22] of PCA is given by

$$\begin{aligned} \max_{\boldsymbol{u}_i} \quad & \boldsymbol{u}_i^\top \mathbf{C} \boldsymbol{u}_i \\ \text{s.t} \quad & \|\boldsymbol{u}_i\|_2^2 = \boldsymbol{u}_i^\top \boldsymbol{u}_i = 1, \end{aligned} \tag{2.7}$$

where $\boldsymbol{u}_i \in \mathbb{R}^d$ is the i^{th} principal component and $\mathbf{C}$ is the sample covariance of the data ,defined as

$$\mathbf{C} = \frac{1}{N} \mathbf{X}\mathbf{X}^\top = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top. \tag{2.8}$$

Solving the Karush-Kuhn-Tucker (KKT) conditions of this constrained optimization problem, one can obtain the optimal conditions as the eigen-decomposition form on covariance matrix $\mathbf{C}$:

$$\mathbf{C}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i, \quad i = 1, \ldots, r. \tag{2.9}$$

where λ_i corresponds to the Lagrange multiplier. That indicates that the optimal principal components are the eigenvectors corresponding to the r largest eigenvalues $\lambda_1, \ldots, \lambda_r$ solved from eigen-decomposition problem. An alternative formulation of PCA is based on *minimum reconstruction error* [22]. We denote a set of orthonormal directions as matrix $\mathbf{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r]$, the reconstructed data point in PCA is simply $\tilde{\boldsymbol{x}}_n = \mathbf{U}\mathbf{U}^\top \boldsymbol{x}_n$, which leads to the following straightforward reconstruction error minimization problem:

$$\begin{aligned} \min_{\mathbf{U}} \quad & \frac{1}{N} \sum_{n=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|_2^2 \\ \text{s.t} \quad & \mathbf{U}^\top \mathbf{U} = \mathbf{I}. \end{aligned} \tag{2.10}$$

It is easy to verify that the optimal solution for 2.10 is equivalent to 2.9.

Basic formulation of Kernel PCA PCA is limited in only capturing the linear patterns in the data which makes it fail to generalize well on more complex non-linear data. Schölkopf et al. [15] extended the vanilla PCA algorithm to a non-linear setting based on kernel methods. Consider a non-linear mapping function $\boldsymbol{\phi}(.) : \mathbb{R}^d \rightarrow \mathcal{H}$, the basic idea of kernel principal component analysis (KPCA) is to perform conventional PCA on the mapped data in the feature space. The sample covariance matrix after feature map now becomes

$$\mathbf{C}_{\boldsymbol{\phi}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^\top = \frac{1}{N}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top, \tag{2.11}$$

and its corresponding eigen-decomposition problem is given by

$$\mathbf{C}_{\boldsymbol{\phi}}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i, \quad i = 1, \ldots, r. \tag{2.12}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{x}_1), \boldsymbol{\phi}(\boldsymbol{x}_2), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_N)]$. We usually refer to the formulation of eigen-decomposition on covariance matrix as the *primal* form because function $\boldsymbol{\phi}(.)$ needs to be explicitly defined under this case. To avoid directly working on explicit feature map function $\boldsymbol{\phi}(.)$, Schölkopf et al. reformulated this problem by cleverly using integral operator kernel functions [15]. Starting from equation (2.11), an equivalent form can be achieved by multiplying $\boldsymbol{\phi}(\boldsymbol{x}_l)^\top (l = 1, \ldots, N)$ on both sides :

$$\boldsymbol{\phi}(\boldsymbol{x}_l)^\top \cdot \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^\top \boldsymbol{u}_i = \lambda_i(\boldsymbol{\phi}(\boldsymbol{x}_l)^\top \cdot \boldsymbol{u}_i). \tag{2.13}$$

Moreover, one can write the eigen-vectors $\boldsymbol{u}_i$ in the form of linear combination of $\boldsymbol{\phi}(\boldsymbol{x})$ where

$$\boldsymbol{u}_i = \sum_{m=1}^{N} h_{im}\boldsymbol{\phi}(\boldsymbol{x}_m). \tag{2.14}$$

Combining equations (2.14) with (2.13), the key eigenvector equation can be expressed in terms of kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)\top\boldsymbol{\phi}(\boldsymbol{x}_j)$:

$$\frac{1}{N}\sum_{n=1}^{N} K(\boldsymbol{x}_l, \boldsymbol{x}_n)\sum_{m=1}^{N} h_{im}K(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{m=1}^{N} h_{im}K(\boldsymbol{x}_l, \boldsymbol{x}_n). \tag{2.15}$$

After simplifying, one can obtain a neat expression

$$\mathbf{K}\boldsymbol{h}_i = \lambda_i N \boldsymbol{h}_i \tag{2.16}$$

where coefficients vector $\boldsymbol{h}_i = [h_{i1}, h_{i2}, \ldots, h_{im}]^\top$ and $\mathbf{K}$ is the gram matrix of kernel function $K(.,.)$. It is obvious that equation (2.13) is exactly the eigen-decomposition problem on kernel matrix $\mathbf{K}$, hence one can solve the KPCA problem given a high dimensional feature space without demanding computational costs. Notice that when a linear kernel function is provided, KPCA will be reduced to conventional PCA.

LS-SVM formulation of Kernel PCA Least Squares Support Vector Machine (LS-SVM) is a modification on the formulation of the original SVM optimization problem introduced by Suykens et al. [13], [24]. The key difference between LS-SVM formulation and basic SVM formulation is in two-fold. First, inequality constraints in SVM is translated to equality constraints by introducing an error variable $\boldsymbol{e}_i$. Secondly, a squared loss term on $\boldsymbol{e}_i$ is included in the objective converting quadratic programming (QP) problem to solving a set of linear equations [13]. The computation is greatly simplified thanks to LS-SVM form while most of key advantages of traditional SVM are preserved, such as the primal-dual representation [25].

Not limited to SVM classifiers, LS-SVM also provides an alternative formulation for KPCA that can be written as

$$\begin{aligned} \min_{\mathbf{W}, \boldsymbol{e}_i} \quad & \frac{\eta}{2}\mathbf{W}^\top\mathbf{W} - \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{e}_i^\top \Lambda^{-1} \boldsymbol{e}_i \\ \text{s.t} \quad & \boldsymbol{e}_i = \mathbf{W}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) \quad \forall i = 1, \ldots, N \end{aligned} \tag{2.17}$$

with $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_r)$ and transformation matrix $\mathbf{W}$. This optimization problem is analogous to one class modeling problem in the context of SVM where only a zero target value is considered. One can recover the optimal solution to the eigen-problem on kernel matrix $\mathbf{K}$ (equation (2.16)) by solving the KKT conditions. In case $\boldsymbol{\phi}(\boldsymbol{x})$ is not zero-mean, centering before performing PCA is nontrivial. The error variable (or score variable) in problem (2.17) then becomes $\boldsymbol{e}_i = \mathbf{W}^\top(\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{\mu}_\phi)$ [13].

Preimage problem in Kernel PCA One important use case in KPCA is denoising, i.e. removing unwanted noisy patterns by projecting training data onto subspace spanned by eigenvectors. Furthermore, denoising requires mapping the projected data back to the original input space, namely recon-

structing the data. Reconstruction in linear PCA is straightforward because only simple matrix multiplication is involved. However, reconstructing data in Kernel PCA is way more complicated due to the implicit nature of feature map function $\boldsymbol{\phi}(.)$, no exact solution exists for reversing the feature map. The general challenge of approximating the reconstructed data in kernel-based algorithms is known as *pre-image* problem [26], [27]. To tackle this problem, various approaches have been proposed in the last few decades, including fixed-point iterations [27], multidimensional scaling-based techniques [28], and many others.

2.3.2 Basic formulation of RKM

Suykens further extended LS-SVM formulations of different kernel machines to the framework of Restricted Kernel Machine (RKM), allowing a more intuitive architecture with visible-hidden unit representations [19]. In our work, RKM is limited to KPCA with RKM representations.

As the most fundamental building block of RKM framework, we start by introducing the conjugate feature duality.

Lemma 1 (Conjugate feature duality)**.** *Given a diagonal matrix* $\Lambda = diag(\lambda_1, \ldots, \lambda_S)$ *,for all* $\boldsymbol{e}, \boldsymbol{h} \in \mathbb{R}^s$*, we have*

$$\frac{1}{2}\boldsymbol{e}^\top \Lambda^{-1}\boldsymbol{e} + \frac{1}{2}\boldsymbol{h}^\top \Lambda \boldsymbol{h} \geq \boldsymbol{e}^\top \boldsymbol{h}. \tag{2.18}$$

Proof. Fenchel's inequality states that for any pair of convex conjugate functions f and f^*, the following inequality holds:

$$f(x) + f^*(y) \geq x^\top y \tag{2.19}$$

where the definition of conjugate function on $f(x)$ is given by $f^*(y) = \sup_x(y^\top x - f(x))$ [29]. We define the following function

$$f(\boldsymbol{e}) = \frac{1}{2}\boldsymbol{e}^\top \Lambda^{-1}\boldsymbol{e} \tag{2.20}$$

with its corresponding convex conjugate

$$f^*(\boldsymbol{h}) = \sup_{\boldsymbol{e}}(\boldsymbol{e}^\top \boldsymbol{h} - \frac{1}{2}\boldsymbol{e}^\top \Lambda^{-1}\boldsymbol{e}). \tag{2.21}$$

To find the supremum, we compute the stationary points of $f^*(\boldsymbol{h})$ with respect to $\boldsymbol{e}$:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{e}}(\boldsymbol{e}^\top \boldsymbol{h} - \frac{1}{2}\boldsymbol{e}^\top \Lambda^{-1}\boldsymbol{e}) = \boldsymbol{h} - \Lambda^{-1}\boldsymbol{e} = 0 \\ \boldsymbol{e} = \Lambda \boldsymbol{h}. \end{aligned} \tag{2.22}$$

Substituting $\boldsymbol{e} = \Lambda\boldsymbol{h}$ back into $f^*(\boldsymbol{h})$:

$$\begin{aligned} f^*(\boldsymbol{h}) &= (\Lambda\boldsymbol{h})^\top \boldsymbol{h} - \frac{1}{2}(\Lambda\boldsymbol{h})^\top \Lambda^{-1}(\Lambda\boldsymbol{h}) \\ &= \frac{1}{2}\boldsymbol{h}^\top \Lambda \boldsymbol{h}. \end{aligned} \tag{2.23}$$

Conjugate feature duality can be justified if we apply Fenchel-Young inequality:

$$\begin{aligned} f(\boldsymbol{e}) + f^*(\boldsymbol{h}) &\geq \boldsymbol{e}^\top \boldsymbol{h} \\ \frac{1}{2}\boldsymbol{e}^\top \Lambda^{-1}\boldsymbol{e} + \frac{1}{2}\boldsymbol{h}^\top \Lambda \boldsymbol{h} &\geq \boldsymbol{e}^\top \boldsymbol{h}. \end{aligned} \tag{2.24}$$

□

$\boldsymbol{h}$ can be viewed as hidden units in the RKM framework. One can then deduce an upper bound of the LS-SVM objective by replacing the squared loss term on $\boldsymbol{e}$, while new hidden units are introduced and optimality conditions remain unchanged. Suykens refers to the general process of introducing hidden units via the above inequality as *conjugate feature duality*.

Kernel PCA under the framework of RKM We now investigate the RKM representation of KPCA. Applying the conjugate feature duality on the objective of LS-SVM formulation of KPCA (equation 2.17), we will achieve the training objective function of RKM:

$$\begin{aligned} \mathcal{J}_{\text{KPCA}} &= \frac{\eta}{2}\mathbf{W}^\top \mathbf{W} - \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{e}_i^\top \Lambda^{-1}\boldsymbol{e}_i \quad (\text{s.t. } \boldsymbol{e}_i = \mathbf{W}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)) \\ &\leq -\sum_{i=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \mathbf{W}\boldsymbol{h}_i + \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{\eta}{2}\mathbf{W}^\top \mathbf{W} = \mathcal{J}_{\text{RKM}}. \end{aligned} \tag{2.25}$$

Stationary points of $\mathcal{J}_{\text{RKM}}$ is characterized by

$$\begin{cases} \dfrac{\partial \mathcal{J}_{\text{RKM}}}{\partial \boldsymbol{h}_i} = 0 \implies \Lambda \boldsymbol{h}_i = \mathbf{W}^\top \boldsymbol{\phi}(\boldsymbol{x}_i), \quad \forall i \\ \dfrac{\partial \mathcal{J}_{\text{RKM}}}{\partial \mathbf{W}} = 0 \implies \mathbf{W} = \dfrac{1}{\eta} \displaystyle\sum_{i=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_i) \boldsymbol{h}_i^\top. \end{cases} \tag{2.26}$$

By eliminating either variable $\boldsymbol{h}_i$ or $\mathbf{W}$, one can arrive at different eigen-decomposition problems corresponding to primal-dual representation :

$$\begin{cases} \text{Eliminating } \boldsymbol{h}_i \implies \dfrac{1}{\eta} \mathbf{K} \mathbf{H}^\top = \mathbf{H}^\top \Lambda \quad \text{(Dual)} \\ \text{Eliminating } \mathbf{W} \implies \dfrac{1}{\eta} \mathbf{S}_{\boldsymbol{\Phi}} \mathbf{W} = \Lambda \mathbf{W} \quad \text{(Primal)} \end{cases} \tag{2.27}$$

where matrix $\mathbf{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N]$ is the collection of all latent variables and matrix $\mathbf{S}_{\boldsymbol{\Phi}} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ is proportional to the sample covariance matrix.

Connections with RBM Restricted Boltzmann Machine (RBM) is a type of energy-based stochastic neural network, consisting of visible and hidden units as the basic building blocks [30], [31]. The key difference between RBM and general BM is that there is no direct connections between units within the same layer, i.e. no interconnections within the group of visible units or hidden units. Computation in RBM is relatively more efficient thanks to this restriction. Denote the set of (binary) visible variables as $\boldsymbol{v} \in \{0,1\}^d$ and the set of (binary) hidden variables of $\boldsymbol{h} \in \{0,1\}^s$, the energy function of RBM is given by

$$\mathcal{J}_{\text{RBM}}(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^\top \mathbf{W} \boldsymbol{h} - \boldsymbol{c}^\top \boldsymbol{v} - \boldsymbol{a}^\top \boldsymbol{h} \tag{2.28}$$

where $\mathbf{W}$ acts as the weighting matrix (or interconnection matrix) from layer to layer and $\boldsymbol{c}$, $\boldsymbol{a}$ are the bias terms. The joint distribution function of RBM is obtained by normalizing the exponential of the negative energy function :

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z(\boldsymbol{v}, \boldsymbol{h})} \exp\left(-\mathcal{J}_{\text{RBM}}(\boldsymbol{v}, \boldsymbol{h})\right). \tag{2.29}$$

The normalization term is often known as *partition function* which is given by $Z(\boldsymbol{v}, \boldsymbol{h}) = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\left(-\mathcal{J}_{\text{RBM}}(\boldsymbol{v}, \boldsymbol{h})\right)$. If comparing the above energy function with the training objective in RKM (equation (2.25)), one can observe that the interaction term $-\boldsymbol{v}^\top \mathbf{W} \boldsymbol{h}$ appears in both RKM and RBM objective, capturing how the states of the visible units influence the states of the hidden units and vice versa. Apart from a similar energy function form,

the notation "R" in "RKM" also refers to no hidden-to-hidden or visible-to-visible interconnections existing in the model architecture. There are also some key differences between RBM and RKM. First, units in RBM are typically binary-valued, while RKM models the underlying patterns within real-valued variables. Although there exist several works [32]–[34] that attempt to generalize classical RBM to modeling real-valued distribution, difficulty in training and inference is still a big issue [33]. Second, training in RKM is based on minimizing the empirical loss using a classical gradient descent approach, whereas RBM tends to maximize the likelihood via k-step contrastive divergence (CD-k) algorithm [35].

Deep RKM RKM is not limited to a single-layer architecture; multiple RKMs can be coupled using the property of conjugate feature duality to form a unified deep architecture [19]. More specifically, consider the KPCA training objective under RKM framework in layer l:

$$\mathcal{J}_{\text{RKM}_l} = -\sum_{i=1}^{N} \boldsymbol{\phi}_l(\boldsymbol{h}_i^{(l-1)})^\top \mathbf{W}_l \boldsymbol{h}_i^{(l)} + \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{h}_i^{(l)\top} \Lambda_l \boldsymbol{h}_i^{(l)} + \frac{\eta}{2}\mathbf{W}_l^\top \mathbf{W}_l. \quad (2.30)$$

The training objective for coupled KPCAs is then given by simply summation of objectives from different layers:

$$\mathcal{J}_{\text{DRKM}} = \sum_{l=1}^{L} \mathcal{J}_{\text{RKM}_l}. \quad (2.31)$$

The input of each layer is the output of the previous layer (i.e. latent variables yielded by previous KPCA operation), notice that $\boldsymbol{h}^{(0)}$ refers to the original training data for the starting layer. Based on the framework of DRKM, Tonin et al. [36] introduced orthogonality constraints on the latent variables to encourage disentanglement as well as to simplify the optimization process. The general theoretical framework of the deep KPCA methodology is summarized in [37]. Furthermore, recent studies [37] have empirically shown that deep KPCA can achieve better disentanglement capability compared to the state-of-the-art InfoVAE [38] due to its effective hierarchical learning process.

2.3.3 Basic formulation of Generative RKM

Generative kernel PCA Schreurs and Suykens [39] proposed a generative version of KPCA, by defining the following generation objective function:

$$\mathcal{J}_{\text{KPCA-gen}} = -\boldsymbol{\phi}(\boldsymbol{x}^*)^\top \widehat{\mathbf{W}} \boldsymbol{h}^* + \frac{1}{2}\boldsymbol{\phi}(\boldsymbol{x}^*)^\top \boldsymbol{\phi}(\boldsymbol{x}^*) \tag{2.32}$$

with a new generated data point $\boldsymbol{x}^*$ and its corresponding hidden variable $\boldsymbol{h}^*$. The estimated interconnection matrix from the training phase is denoted by $\widehat{\mathbf{W}}$. By solving the stationary point, one can show the generated data is given by the equation:

$$\frac{\partial \mathcal{J}_{\text{KPCA-gen}}}{\partial \boldsymbol{\phi}(\boldsymbol{x}^*)} = 0 \implies \boldsymbol{\phi}(\boldsymbol{x}^*) = \widehat{\mathbf{W}}\boldsymbol{h}^* = (\frac{1}{\eta}\sum_{i=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_i)\boldsymbol{h}_i^\top)\boldsymbol{h}^*. \tag{2.33}$$

New $\boldsymbol{h}^*$ is randomly sampled from a fitted Gaussian distribution on the trained hidden units. Then a natural question arises: how to tackle the pre-image problem, i.e. how to recover $\boldsymbol{x}^*$ from the form $\boldsymbol{\phi}(\boldsymbol{x}^*)$ when feature map function is implicitly defined? As mentioned in the previous section, the exact solution may not exist due to the implicit nature of $\boldsymbol{\phi}(.)$. Schreurs and Suykens [39] showed a possible solution by implementing kernel smoothing approach. Through multiplying any training data point after feature map $\boldsymbol{\phi}(\boldsymbol{x}_j)^\top$ for $\forall j \in \{1, \ldots, N\}$, one can obtain the following relation:

$$\boldsymbol{k}_{\boldsymbol{x}^*} = \frac{1}{\eta}\mathbf{K}\mathbf{H}^\top \boldsymbol{h}^* \tag{2.34}$$

where $\boldsymbol{k}_{\boldsymbol{x}^*} = [K(\boldsymbol{x}_1, \boldsymbol{x}^*), \ldots, K(\boldsymbol{x}_N, \boldsymbol{x}^*)]^\top$ is a collection of similarity measures between new data point and remaining training data points in the feature space. The estimation of $\boldsymbol{x}^*$ from kernel smoothing is given by

$$\hat{\boldsymbol{x}}^* = \frac{\sum_{j=1}^{N_r} \tilde{K}(\boldsymbol{x}_j, \boldsymbol{x}^*)\boldsymbol{x}_j}{\sum_{j=1}^{N_r} \tilde{K}(\boldsymbol{x}_j, \boldsymbol{x}^*)} \tag{2.35}$$

where $\tilde{K}(.,.)$ refers to the scaled version of similarity measures and $N_r < N$ needs to be specified beforehand indicating the number of closest (or most similar) points based on the kernel function.

Generative RKM Pandey et al. [8] introduced a novel framework for generative learning, combining KPCA with deep neural network architectures.

To address the pre-image problem, they implemented an alternative approach using a deep convolutional neural network as the feature map function, denoted by $\boldsymbol{\phi}(.)$ and parameterized by $\boldsymbol{\theta}$. Additionally, another convolutional neural network $\boldsymbol{\psi}(.)$, typically the inversion of the neural network used in the feature map, parameterized by $\boldsymbol{\zeta}$, is employed as the pre-image map. An intuitive link between autoencoders and Gen-RKM emerges when considering the feature map as the encoder part and the pre-image map as the decoder part, that is Gen-RKM can be viewed as an autoencoder with Kernel PCA in the bottleneck layer. In this specific architecture, disentanglement is encouraged thanks to the mutually uncorrelated eigenvectors from KPCA operation.

During the training phase, the network parameters are updated in a mini-batching scheme by jointly minimizing the reconstruction errors and objective for KPCA. More specifically, the training objective in Gen-RKM is defined as follows:

$$\begin{aligned} \mathcal{J}_{\text{Gen-RKM}} &= \underbrace{\mathcal{J}_{\text{RKM}} + \frac{c_{\text{stab}}}{2}\mathcal{J}_{\text{RKM}}^2}_{\text{stabilization}} + \underbrace{\frac{\gamma}{N}\mathcal{J}_{\text{recon}}}_{\text{reconstruction error}} \\ \text{s.t.} \quad \mathcal{J}_{\text{recon}} &= \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{x}_i))\|_2^2. \end{aligned} \tag{2.36}$$

Notice that a stabilized version of the training objective on KPCA is employed in practice since the minus sign term will possibly lead to loss explosion as suggested by Sukyens [19]. Pandey et al. [8] have proved that the optimality conditions remain unchanged under this transformation. The constants c_{stab} and γ act as the regularization parameters in the training objective. The final optimization problem is then given by

$$\min_{(\boldsymbol{\theta},\boldsymbol{\zeta})} \mathcal{J}_{\text{Gen-RKM}}. \tag{2.37}$$

The generation phase in Gen-RKM is basically the same as in KPCA. The only difference is that we first fit a Gaussian mixture model (GMM) over the all trained latent representations, a new latent variable is then randomly sampled from the fitted GMM model. Since feature map and pre-image map is explicitly defined by neural network models, corresponding generated data can be obtained via simple matrix multiplication without effort.

The detailed training and generation algorithm for Gen-RKM is depicted in

algorithm 1. Notice that a final computation step is implemented in practice, performing SVD on the full kernel matrix to get all latent variables and weighting matrix. This step can be highly computationally expensive under the dual form when N (number of data points) is very large since the time complexity of SVD is $\mathcal{O}(N^3)$. Thanks to the primal-dual representation of RKM framework, one can address this issue by solving the KPCA problem in the primal form (eigen-decomposition on $\mathbf{S}_{\mathbf{\Phi}}$, see equation (2.27)). The size of covariance matrix $\mathbf{S}_{\mathbf{\Phi}}$ is fixed on $d_f \times d_f$, where the dimension of feature space d_f is typically way smaller than N, and computation on all latent variables $\mathbf{H}$ just involves simple matrix multiplication. Hence one can choose to perform Gen-RKM algorithm in either primal or dual form depending on different N and d_f, ensuring the scalability of Gen-RKM to large-scale dataset [8], [40].

Algorithm 1 Training and generation algorithm for one-view Gen-RKM (Dual) [8]

Input: training data $\{\boldsymbol{x}_i\}_{i=1}^N$; mini-batch size m;
regularization parameters $\eta, c_{\text{stab}}, \gamma$;
explicit feature map $\boldsymbol{\phi}_{\boldsymbol{\theta}}(.)$ and explicit pre-image map $\boldsymbol{\psi}_{\boldsymbol{\zeta}}(.)$;
dimension of latent space s and number of components for fitting GMM l;

procedure TRAINING
  ▷ *Training loop* ◁
  for each epoch **do**
    for each mini-batch **do**
      Get mini-batch $\{\boldsymbol{x}_i\}_{i\in B}$ with $B \subset \{1, \ldots, N\}$ via uniform sampling
      $\mathbf{\Phi} \leftarrow \boldsymbol{\phi}_{\boldsymbol{\theta}}(\{\boldsymbol{x}_i\}_{i\in B})$
      $\mathbf{K} \leftarrow \mathbf{\Phi}^\top \mathbf{\Phi}$
      $\mathbf{H}, \Lambda \leftarrow \text{SVD}(\mathbf{K})$ ▷ *Eigen-decomposition on kernel matrix*
      $\mathbf{W} \leftarrow \mathbf{\Phi}\mathbf{H}^\top$
      $\widehat{\mathbf{\Phi}} \leftarrow \mathbf{W}\mathbf{H}$
      update $\{\boldsymbol{\theta}, \boldsymbol{\zeta}\} \leftarrow \text{Adam}(\mathcal{J}_{\text{Gen-RKM}})$ ▷ *Update network parameters*
    end for
  end for
  ▷ *Final computation step to get all latent variables on full dataset* ◁
  $\mathbf{\Phi}_{\text{full}} \leftarrow \boldsymbol{\phi}_{\boldsymbol{\theta}}(\{\boldsymbol{x}_i\}_{i=1}^N)$
  repeat step 7-9 on $\mathbf{\Phi}_{\text{full}}$
end procedure
procedure GENERATION
  $p(\boldsymbol{h}) \leftarrow \text{GMM}_l(\mathbf{H})$
  $\boldsymbol{h}^* \sim p(\boldsymbol{h})$ ▷ *Random sampling from the fitted GMM*
  $\boldsymbol{x}_{\text{gen}} \leftarrow \boldsymbol{\psi}_{\boldsymbol{\zeta}}(\mathbf{W}\boldsymbol{h}^*)$
end procedure

Extensions to multi-view learning in Gen-RKM Now we consider the generalized form of Gen-RKM with extension to multi-view learning. *Views* generally refers to different data representations (or modalities), and multi-view learning corresponds to the unified modeling process to learn the common feature spaces or shared underlying pattern from different data sources [41]. For illustration purposes, we will present the two-view setting. One can refer to [40] for the theoretical formulation of a more general multi-view case.

Specifically, given a training dataset with two views : $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $\boldsymbol{y}_i \in \mathbb{R}^p$ (one can view $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ as a feature-label pair). The training objective for two-view data in KPCA under RKM form can be formulated as

$$\begin{aligned}\mathcal{J}_{\text{RKM-MV}} = &- \sum_{i=1}^{N} \boldsymbol{\phi}_{\boldsymbol{x}}(\boldsymbol{x}_i)^\top \mathbf{W}_{\boldsymbol{x}} \boldsymbol{h}_i - \sum_{i=1}^{N} \boldsymbol{\phi}_{\boldsymbol{y}}(\boldsymbol{y}_i)^\top \mathbf{W}_{\boldsymbol{y}} \boldsymbol{h}_i \\ &+ \frac{1}{2} \sum_{i=1}^{N} \boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{\eta_{\boldsymbol{x}}}{2} \mathbf{W}_{\boldsymbol{x}}^\top \mathbf{W}_{\boldsymbol{x}} + \frac{\eta_{\boldsymbol{y}}}{2} \mathbf{W}_{\boldsymbol{y}}^\top \mathbf{W}_{\boldsymbol{y}}\end{aligned} \tag{2.38}$$

where the feature map function for each view is denoted by $\boldsymbol{\phi}_{\boldsymbol{x}}(.) : \mathbb{R}^d \to \mathcal{H}_{\boldsymbol{x}}$ and $\boldsymbol{\phi}_{\boldsymbol{y}}(.) : \mathbb{R}^d \to \mathcal{H}_{\boldsymbol{y}}$ respectively. $\mathbf{W}_{\boldsymbol{x}} \in \mathbb{R}^{d_f \times s}$ and $\mathbf{W}_{\boldsymbol{y}} \in \mathbb{R}^{p_f \times s}$ are the interconnection matrices corresponding to each view where d_f, p_f are the dimensions of feature spaces and s indicates the dimension of latent space (number of selected principal components from KPCA). Likewise, with the one-view setting, different eigenvalue problems associated with primal-dual representations can be obtained by solving the stationary points of $\mathcal{J}_{\text{RKM-MV}}$:

$$\begin{cases} [\frac{1}{\eta_{\boldsymbol{x}}} \mathbf{K}_{\boldsymbol{x}} + \frac{1}{\eta_{\boldsymbol{y}}} \mathbf{K}_{\boldsymbol{y}}] \mathbf{H}^\top = \mathbf{H}^\top \Lambda \quad \text{(Dual)} \\ \begin{bmatrix} \frac{1}{\eta_{\boldsymbol{x}}} \boldsymbol{\Phi}_{\boldsymbol{x}} \boldsymbol{\Phi}_{\boldsymbol{x}}^\top & \frac{1}{\eta_{\boldsymbol{x}}} \boldsymbol{\Phi}_{\boldsymbol{x}} \boldsymbol{\Phi}_{\boldsymbol{y}}^\top \\ \frac{1}{\eta_{\boldsymbol{y}}} \boldsymbol{\Phi}_{\boldsymbol{y}} \boldsymbol{\Phi}_{\boldsymbol{x}}^\top & \frac{1}{\eta_{\boldsymbol{y}}} \boldsymbol{\Phi}_{\boldsymbol{y}} \boldsymbol{\Phi}_{\boldsymbol{y}}^\top \end{bmatrix} \begin{bmatrix} \mathbf{W}_{\boldsymbol{x}} \\ \mathbf{W}_{\boldsymbol{y}} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{\boldsymbol{x}} \\ \mathbf{W}_{\boldsymbol{y}} \end{bmatrix} \Lambda \quad \text{(Primal)}. \end{cases} \tag{2.39}$$

One can see that the dual form is the eigen-decomposition of the weighted summation of kernel matrices, while the primal form considers the general cross-covariance matrix of feature vectors from different views. Pre-image maps are also explicitly defined by two convolutional neural networks for the purpose of reconstruction. The reconstruction loss in multi-view settings is simply generalized to the summation of reconstruction errors from different views. Combining both the KPCA objective and reconstruction loss, one

can employ the same stabilization trick to form the final training objective (equation (2.36)), and the training process stays largely the same as algorithm 1.

As for the generation phase, the following generating objective is considered :

$$\begin{aligned}\mathcal{J}_{\text{MV-RKM-gen}} = &- \boldsymbol{\phi}_{\boldsymbol{x}}(\boldsymbol{x}^*)^\top \widehat{\mathbf{W}}_{\boldsymbol{x}} \boldsymbol{h}^* - \boldsymbol{\phi}_{\boldsymbol{y}}(\boldsymbol{y}^*)^\top \widehat{\mathbf{W}}_{\boldsymbol{y}} \boldsymbol{h}^* \\ &+ \frac{1}{2}\boldsymbol{\phi}_{\boldsymbol{x}}(\boldsymbol{x}^*)^\top \boldsymbol{\phi}_{\boldsymbol{x}}(\boldsymbol{x}^*) + \frac{1}{2}\boldsymbol{\phi}_{\boldsymbol{y}}(\boldsymbol{y}^*)^\top \boldsymbol{\phi}_{\boldsymbol{y}}(\boldsymbol{y}^*).\end{aligned} \tag{2.40}$$

Different views of data share one common latent variable in the multi-view setting. Once $\boldsymbol{h}^*$ is sampled from the trained GMM, generated data under feature spaces is given by

$$\begin{aligned}\boldsymbol{\phi}_{\boldsymbol{x}}(\boldsymbol{x}^*) &= \widehat{\mathbf{W}}_{\boldsymbol{x}} \boldsymbol{h}^* \\ \boldsymbol{\phi}_{\boldsymbol{y}}(\boldsymbol{x}^*) &= \widehat{\mathbf{W}}_{\boldsymbol{y}} \boldsymbol{h}^*,\end{aligned} \tag{2.41}$$

which can be readily reversed back to input data space via explicit pre-image map likewise in the single-view setting.

2.3.4 Related variants of Generative RKM

Stiefel RKM Generative RKM requires performing exact SVD at each iteration which could possibly cause numerical instability during the backpropagation process. Moreover, training Gen-RKM can be rather computationally demanding when both mini-batch size and feature space dimension are large. Pandey et al. [42] proposed a modified version of Gen-RKM, namely Stiefel RKM, to address these limitations. The training objective for Stiefel RKM is restated as

$$\begin{aligned}\min_{\mathbf{U},(\boldsymbol{\theta},\boldsymbol{\zeta})} \mathcal{J}_{\text{St-RKM}} = &\underbrace{\frac{\lambda}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{\psi}_{\boldsymbol{\zeta}}(\mathbf{U}\mathbf{U}^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\|_2^2}_{\text{encoder-decoder reconstruction}} \\ &+ \underbrace{\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \mathbf{U}\mathbf{U}^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2}_{\text{KPCA reconstruction}} \\ &\text{s.t.} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I},\end{aligned} \tag{2.42}$$

where the constraint set enforcing the orthogonality of the matrix is called as *Stiefel manifold* [43]. In the training phase, transformation matrix $\mathbf{U}$ and network parameters $(\boldsymbol{\theta}, \boldsymbol{\zeta})$ are jointly optimized in each iteration. Cayley-Adam algorithm [44] is implemented for optimizing the KPCA reconstruction objection, while the autoencoder objective is minimized via the classical Adam optimizer. This approach avoids the expensive computation of exact SVD during the training process.

Robust RKM Pandey et al. [45] observed that Gen-RKM could be sensitive to the outliers in the training data. A weighting scheme is incorporated with the Gen-RKM framework to combat unwanted bias caused by contamination in the training data as well as impose additional regularization on the loss objective. Assuming a positive-definite diagonal weighting matrix $\mathbf{D} = \text{diag}(d_1, \ldots, d_N)$, the training objective for weighted RKM is defined as

$$\mathcal{J}_{\text{WRKM}} = -\sum_{i=1}^{N} \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top \mathbf{W} \boldsymbol{h}_i + \frac{1}{2} \sum_{i=1}^{N} \mathbf{D}_{ii}^{-1} \boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{\eta}{2} \mathbf{W}^\top \mathbf{W}, \tag{2.43}$$

which corresponds to the eigen-decomposition problem on the weighted kernel matrix (if eliminating the primal variable). The final training objective of robust RKM is composed of the weighted reconstruction error and the weighted KPCA loss after employing the same stabilization trick :

$$\min_{(\boldsymbol{\theta},\boldsymbol{\zeta})} \mathcal{J}_{\text{Rob-RKM}} = \mathcal{J}_{\text{WRKM}} + \frac{c_{\text{stab}}}{2} + \mathcal{J}_{\text{WRKM}}^2 + \frac{\gamma}{N} \mathbf{D}_{ii} \mathcal{J}_{\text{recon}}. \tag{2.44}$$

Pandey et al. [45] proposed computing weighting matrix $\mathbf{D}$ based on Minimum Covariance Determinant (MCD) which is a commonly-used robust estimator against contaminated data.

2.4 Learning from unbalanced data

2.4.1 Re-sampling and re-weighting technique

Re-sampling is normally performed during the data preprocessing phase prior to training a model. It modifies the prior probability of the majority and minority class in the training set to obtain a more balanced number of instances in each class. For unbalanced data, resampling can be divided into two categories, over-sampling and under-sampling. Over-sampling methods balance

input data by increasing the number of samples from the minority classes. It can be done by simply duplicating the samples from the minority classes, or more generally, manually creating new samples for the minority classes based on the existing minority samples. For instance, Chawla et al. initially introduced the widely adopted SMOTE technique [46]. Subsequently, various modifications [47] and extensions to deep learning [48] have been extensively explored and developed. The advantage of over-sampling is that it utilizes all information of samples in the training set. However, there is a risk that minority classes may become overrepresented in the training set, potentially leading to overfitting. Conversely, under-sampling deals with unbalanced data by dropping out samples from the majority class while preserving all the minority class samples. For example, the Random Under-Sampling(RUS) randomly eliminates the samples of the majority class[49]. This method is well-suited for large-scale applications where the majority classes are substantially larger. Reducing the number of training samples can decrease training time and make the learning problem more computationally tractable[50]. However, it is risky to lose information of the samples from the majority class since informative samples may be randomly dropped out. Lima and Pereira(2015) found that instead of dropping samples, reducing the number of irrelevant features also significantly improves the model performance on unbalanced data[51].

Re-weighting methods, also known as cost-sensitive based methods, are an alternative technique to deal with unbalanced data problems. Unlike the resampling which is performed during the data preprocessing phase, re-weighting works during the training phase of model construction by assigning different weights to different samples when calculating training loss. Berardi and Zhang initially proposed cost-sensitive neural networks by assigning different costs to errors in different classes[52]. A classic empirical re-weighting method is to assign the samples of each class with the same weight, such as inverse class frequency[53], [54]. It has been further explored by the class-balanced loss[55], which calculates the effective number of examples as class frequency. Besides these empirical re-weighting methods, there are also automatic re-weighting methods that obtain the weights using learning algorithms. Park et al. proposed influence-balanced loss to re-weight samples by the magnitude of the gradient[56]. Liu et al. also propose a method that updates the sample weights under a certain constraint[57].

Furthermore, ensemble-based classifiers, which are known to improve the performance of a single classifier by combining several base classifiers that

outperform every independent one[58], [59], can also be applied to tackle imbalanced data problems. More techniques focus on modifying the classical algorithms, such as kernel and activation function transformation method[60], [61], task decomposition strategies[62] and objective function transformation, are also well explored.

2.4.2 Unbalanced data in supervised and unsupervised learning

In the case of supervised learning, labeled datasets are used to train algorithms to predict outcomes and recognize patterns. The unbalanced data problem would happen when the number of training samples of the minority classes is much smaller compared to other majority classes. As a result, the trained classifiers tend to choose the majority classes and ignore the minority class by treating them as noise[63], [64]. To deal with these problems, both resampling and cost-sensitive-based re-weighting algorithms mentioned in section 2.4.1 can be implemented.

In unsupervised learning, the issue of unbalanced data is less clearly defined because the training data lacks labels, making it challenging to determine whether the dataset is balanced. In this case, most research efforts are concentrated on identifying distinct patterns within the majority of the data, a process often referred to as anomaly detection or outlier detection[65]. Isolation Forest is an efficient anomaly detection algorithm that is similar to the Random Forest, but it chooses the splitting attribute and split point (value) randomly instead of based on information gain or Gini index. During the tree-building process, if some samples quickly reach the leaf nodes (i.e., the distance from the leaf to the root is short), they are considered likely anomalies[66]. Besides, clustering techniques can also be used for outlier detection. For example, DBSCAN[67] and local outlier factor (LOF)[68] both consider as outliers the samples that have a substantially lower density than their neighbors. Meanwhile, one-class classification methods like SVDD [69] and its extension Deep-SVDD [70] provide a different approach. These methods establish spherical boundaries around samples in the feature space, aiming to minimize the volume of the hypersphere to effectively detect outlier samples.

2.4.3 Unbalanced data in generative learning

From the perspective of generative models, an imbalance in training data can be detrimental because a skewed data distribution may introduce undesirable biases towards overrepresented modes. The phenomenon where generation becomes less diverse due to data imbalance is commonly referred to as mode collapse, and it has been widely studied in the context of GANs [6]. However, the mode collapse problem in GANs is not merely specific to data imbalance; other possible causes include gradient vanishing during training, non-convergence of the adversarial loss, discriminator overpowering, and so on [7]. Of course, some works still exist that specifically address the issue of mode collapse in GANs under the data imbalance scenario. Mariani et al. introduce the BAGAN (Balancing GAN) methodology, through an autoencoder-based initialization strategy, which is capable of generating samples with high quality and high diverse conditioned on minority classes, even when the data is extremely imbalanced [71]. Huang et al. further enhance the training stability of BAGAN by introducing an additional gradient penalty term [72]. An alternative approach is to integrate a self-damaging contrastive learning scheme with GAN architecture, known as Damage GAN [73]. Anaissi et al. have shown that Damage GAN is able to generate images with higher quality and greater diversity on the imbalanced training dataset compared to vanilla GAN, as indicated by lower Inception Score (IS) and Fréchet inception distance (FID) [73]. As for research on VAEs in the context of imbalanced data, a representative work is Jigsaw-VAE, which combats the adverse effect brought by feature imbalance by implementing a permutation-based regularization scheme [74]. For RKM-related work, to the best of our knowledge, no studies have yet discussed the data imbalance issue in RKM.

Chapter 3

Problem Statement

This chapter aims to get a deeper insight into the effect of data imbalance within the Gen-RKM framework. In Gen-RKM, new samples are generated by first sampling from the estimated distribution of the latent space $\mathcal{L}$. The sampled latent point is then reconstructed into image shapes via the transpose of interconnection matrix $\mathbf{W}^\top$ and the pre-image network $\boldsymbol{\psi}$ (shown in Figure 3.1). As described in algorithm 1, the latent representations of the training samples are essentially the mutually orthogonal eigenvectors yielded from KPCA. Therefore, if there is an imbalance existing in the data, the minority modes in the original training dataset would also result in corresponding minority groups in the latent space. Simply speaking, an imbalance in the input dataset would cause an imbalance in the latent space representations, resulting in a biased distribution and, ultimately, a less diverse generation.

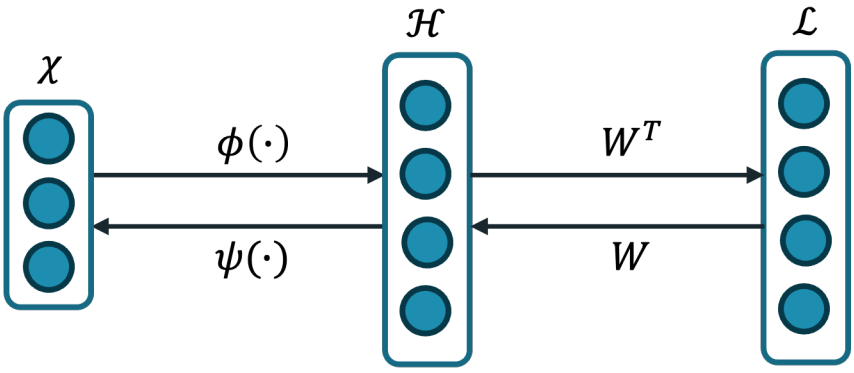

Figure 3.1: Schematic representation of Gen-RKM. $\boldsymbol{\phi}(.)$ is the feature map network, while $\boldsymbol{\psi}(.)$ corresponds to the pre-image network. $\mathbf{W}$ is the interconnection matrix yielded from KPCA, mapping feature representations to the latent space $\mathcal{L}$. Latent points can be remapped back to the feature space via the transpose of the transformation matrix $\mathbf{W}^\top$.

Next, a motivating example on the 012-MNIST dataset is used to illustrate

the data imbalance issue in the Gen-RKM model. We consider an unbalanced version of the MNIST dataset, where only digits 0-2 are included and digit 2 is set to the minority class (see Section 5.1.1 for a more detailed introduction about the dataset used). Figure 3.2 (first row) shows the yielded latent space on the 012-MNIST dataset and the unbalanced version of it in the Gen-RKM setting. In the balanced dataset, well-separated or distinguishable clusters are produced in the latent space, and the corresponding latent representations are (nearly) evenly distributed, indicating each class has a similar influence on the learned latent space. As for the unbalanced scenario, it can be clearly observed that the learned latent space is strongly biased toward the majority groups since latent points from minority modes are mostly blurred or intermixed with the majority modes. The impact of data imbalance is significant on the latent space representation, where minority modes (or classes) are underrepresented, potentially leading to challenges in the generation or reconstruction of the underrepresented groups.

Data imbalance also has a similar effect on VAE as showcased in Figure 3.2 (second row), where the minority classes tend to be biased by the majority classes. That is unsurprising, as Gen-RKM and VAE share similar encoder-decoder architectures, and the data distribution in both models is explicitly defined. The biases in the final generation results brought by data imbalance are confirmed in Figure 3.3, where random generations from both balanced and unbalanced datasets are depicted. One can observe that samples from minority classes are generated much less frequently than those from majority classes.

To extend the capability of Gen-RKM to generalize well to the minority classes in the unbalanced data, a natural idea is to augment the minority classes in the latent space (in other words, synthetically increasing the number of latent points corresponding to minority modes). In this spirit, the generation result should be more diverse. Of course, this approach is equivalent to performing augmentation on the original data, as only a balanced distribution of raw data can result in a balanced distribution in its corresponding latent space. In the next chapter, we will discuss random sampling, a naive yet effective approach for addressing data imbalance. Specifically, we will combine various sampling techniques with Gen-RKM framework to investigate how to tackle the issue of unbalanced data in the generative learning context.

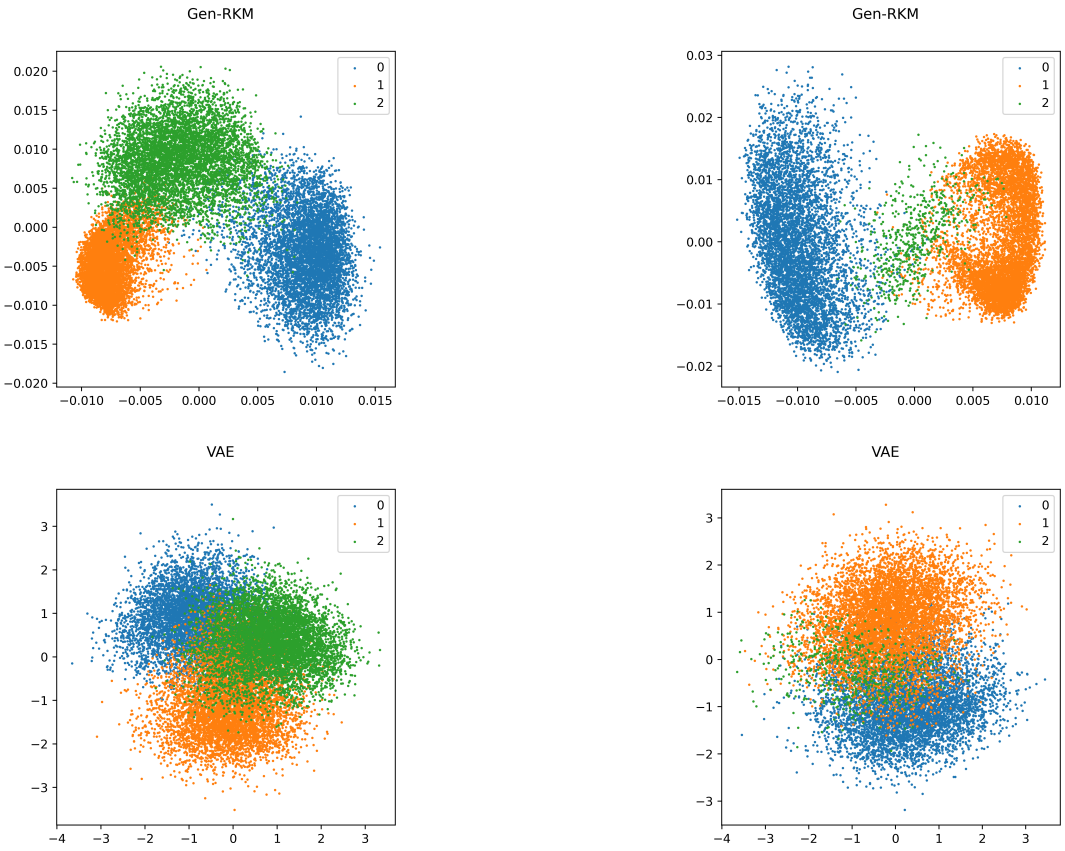

Figure 3.2: Visualizations of latent spaces of Gen-RKM (first row) and VAE (second row) trained on balanced (first column) and unbalanced 012-MNIST datasets (second columns). The minority mode is depicted in green color. Sizes of latent space of both Gen-RKM and VAE are set to 10, and only the first two dimensions are visualized.

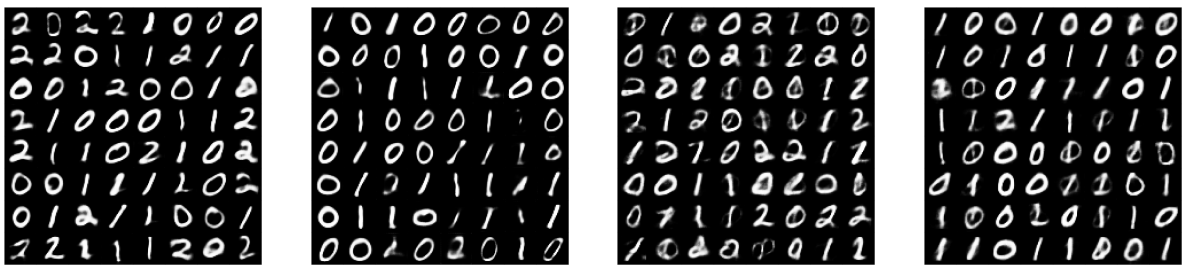

Figure 3.3: Generated samples from the balanced/unbalanced 012-MNIST dataset by Gen-RKM (first two figures) and VAE (last two figures). The first and the third figures display the generations from a balanced dataset, while the second and the fourth figures show the generations from an unbalanced dataset.

Chapter 4

Proposed Methods: Weighted sampling schemes towards diversified generation in Gen-RKM

As discussed in the literature review section, the unbalanced data problem could occur in either supervised or unsupervised scenarios. In this chapter, We propose methods to address the problem within the framework of Gen-RKM under both the two settings.

4.1 Mitigating data imbalance in supervised setting

Normally, classic strategies like class re-sampling or re-weighting (cost-sensitive training) can both be applied to address data imbalance issues in supervised learning settings. However, in the context of the unique architecture of Gen-RKM, where the final latent space is derived by performing SVD on the entire training dataset following the acquisition of the feature map and pre-image map, the re-weighting approach is ineffective. This is because re-weighting on the RKM loss function only affects the weights of the feature map and pre-image map via backpropagation, and does not impact the final SVD process. Consequently, re-sampling methods are primarily investigated in this context.

4.1.1 Inverse frequency sampling

Consider a training dataset $\mathbf{D}_{\text{train}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, where $(\boldsymbol{x}, \boldsymbol{y})$ is the pair of feature and target, $\boldsymbol{x}_i$ the feature vector of the ith sample, $\boldsymbol{y}_i \in (0,1)^c$ the cor-

responding one-hot target vector over c classes, and N the number of samples in the training set. The sampling weights can be obtained simply by calculating the reciprocal of the frequency of the target classes: $w_i = 1/\text{count}(\boldsymbol{y}_i)$. The sampling probability is, therefore, $p_i = w_i / \sum_{i=1}^{N} w_i$. Instead of uniformly sampling data from the training set during the training phase, data points from the minor classes would be sampled with a higher probability than those from the major classes. Therefore, the distribution of training data is modified to achieve a more balanced representation across all classes.

4.1.2 Inverse frequency sampling in the framework of Gen-RKM

The detailed sampling procedure based on inverse frequency sampling in Gen-RKM is illustrated in this subsection. Although inverse frequency sampling is the most classic and straightforward method, adjustments are necessary when applying it to Gen-RKM due to the model's unique architecture. Since the final step of RKM training requires an SVD on the full training set to obtain the latent space, re-sampling is not only needed at the beginning of every mini-batch but also before the last SVD step on the full training set. The detailed training step is shown below in algorithm 2.

Algorithm 2 inverse frequency sampling for one-view Gen-RKM (Dual)

Input: training data $\mathbf{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$; mini-batch size m;
regularization parameters $\eta, c_{\text{stab}}, \gamma$;
explicit feature map $\boldsymbol{\phi}_{\boldsymbol{\theta}}(.)$ and explicit pre-image map $\boldsymbol{\psi}_{\boldsymbol{\zeta}}(.)$;
dimension of latent space s;

▷ *Compute Inverse weight* ◁
$w_i \leftarrow \frac{1}{count(\boldsymbol{y}_i)}$ ▷ *the reciprocal of the frequency of* $\boldsymbol{y}_i$
$p_i \leftarrow w_i / \sum_{i=1}^{n} w_i$
▷ *Training loop* ◁
for each epoch **do**
  for each mini-batch **do**
    $B \leftarrow$ sample a mini-batch of size m with probability $\boldsymbol{x}_i \sim p_i$
    do step 6-11 in algorithm 1
  end for
end for
▷ *Final step to get all latent variables on modified full dataset* ◁
$\mathbf{D}_{\text{resampled}} \leftarrow$ resample a size N dataset from $\mathbf{D}$ with probability $\boldsymbol{x}_i \sim p_i$
repeat step 7-9 in algorithm 1 on $\mathbf{D}_{\text{resampled}}$

4.1.3 Extension to conditional generation in Gen-RKM

Conditional generation involves generating new data samples based on certain attributes. A ubiquitous technique in combating data imbalance is the use of conditional generation with deep generative model. By learning the underlying data distribution via deep architecture, one can augment minority modes (or classes) with highly realistic synthetic data. Both VAE and GAN have their own modified version for conditional modeling, known as CVAE[75] and CGAN[76], respectively. However, the implementation of conditional generation within the framework of Gen-RKM has not yet been explored. Here, We propose a simple yet effective modified generation procedure aimed at achieving structured output representation in Gen-RKM ,as outlined in algorithm 3.

Let $\mathbf{H}$ as a collection of latent variables and $\mathbf{Y}$ as the corresponding class labels for each data point. Recall that generation is based on random sampling from a trained GMM on latent variables in vanilla Gen-RKM:

$$\boldsymbol{h}^* \sim p(\boldsymbol{h}) = \sum_{k=1}^{l} \pi_k \mathcal{N}(\boldsymbol{h}|\boldsymbol{\mu}_{\boldsymbol{h},k}, \boldsymbol{\Sigma}_{\boldsymbol{h},k}). \tag{4.1}$$

Instead of fitting GMM merely on $\mathbf{H}$, we consider modeling the joint PDF of $\boldsymbol{h}$ and $\boldsymbol{y}$ on a concatenated matrix : $\mathbf{H}_{cat} = [\mathbf{H}^\top; \mathbf{Y}^\top]^\top$. To obtain the generation conditioned on a certain class label, it is natural to sample from the conditional distribution $p(\boldsymbol{h}|\boldsymbol{y})$. One can verify that the conditional distribution of a mixture of Gaussian results in another GMM :

$$p(\boldsymbol{h}|\boldsymbol{y}) = \sum_{k=1}^{l} \pi'_k \mathcal{N}(\boldsymbol{h}|\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{h}|\boldsymbol{y},k}). \tag{4.2}$$

with the updated weights for each Gaussian component,

$$\pi'_k = \frac{\pi_k \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_{\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{yy},k})}{\sum_{v=1}^{l} \pi_v \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_{\boldsymbol{y},v}, \boldsymbol{\Sigma}_{\boldsymbol{yy},v})}. \tag{4.3}$$

The more detailed derivation is shown in appendix A.1.

Algorithm 3 Conditional generation algorithm in Gen-RKM

Input: trained explicit pre-image map $\boldsymbol{\psi}_{\boldsymbol{\zeta}}(.)$, interconnection matrix $\mathbf{W}$;
latent variables $\mathbf{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N] \in \mathbb{R}^{d \times N}$;
class labels for each data point $\mathbf{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N] \in \mathbb{R}^{c \times N}$ (in one-hot encoding form, c is the number of classes);
conditioned class label $\boldsymbol{y} \in \mathbb{R}^c$ (in one-hot encoding form);
number of Gaussian components in GMM l;

1: $\mathbf{H}_{cat} \leftarrow [\mathbf{H}^\top; \mathbf{Y}^\top]^\top \in \mathbb{R}^{(d+c) \times N}$
2: $p(\boldsymbol{h}, \boldsymbol{y}) \leftarrow \mathrm{GMM}_l(\mathbf{H}_{cat})$
3: $\boldsymbol{h}^* \sim p(\boldsymbol{h}|\boldsymbol{y})$
4: $\boldsymbol{x}_{\mathrm{gen}} \leftarrow \boldsymbol{\psi}_{\boldsymbol{\zeta}}(\mathbf{W}\boldsymbol{h}^*)$

This modification is inspired by the observation that, in the latent space, the latent variables of different class labels tend to cluster together, as shown in Figure 3.2. Therefore, if we can sample from a selected sub-cluster, we are able to generate data conditioned on a particular label. Notice that this algorithm will fail if latent variables with different labels are intermixed in the latent space. It is advised to include label information as the second view in the training phase before conditional generation because latent representation would become more separated based on their categories, as pointed out in [8].

4.2 Mitigating data imbalance in unsupervised setting

Generative learning is typically conducted under an unsupervised manner, which means no label information is involved during the learning process. However, data imbalance problem might still exist even though it is hard to be clearly identified. More specifically, regions of the data space with fewer data points might be overlooked in the generative model, resulting in biased or unfair generation. Inspired by the successful implementation of ridge leverage scores (RLS) sampling in combating mode collapse problem in GAN [77], we propose to incorporate the RLS sampling as a diversity sampling technique within the framework of Gen-RKM when label information is not available. The basic theoretical aspects of RLS and the modified training algorithm based on RLS sampling will be discussed below

4.2.1 Ridge leverage score sampling

Given a data matrix $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ with size $d \times N$, the classical leverage score for the ith data point is defined as the ith diagonal element value of the projection matrix of $\mathbf{X}$:

$$l_i = \boldsymbol{x}_i^\top (\mathbf{X}\mathbf{X}^\top)^{-1} \boldsymbol{x}_i. \tag{4.4}$$

Leverage score measures the importance or the influence of an individual observation within the dataset. Historically, it has been widely used for model diagnostics and assessment in classical regression models. Ridge regression is a penalized version of linear regression by including an additional L2 regularization term into the loss objective, its corresponding ridge leverage score for the ith observation is given by

$$l_i^R(\gamma) = \boldsymbol{x}_i^\top (\mathbf{X}\mathbf{X}^\top + \gamma \mathbf{I})^{-1} \boldsymbol{x}_i \tag{4.5}$$

where $\gamma > 0$ is known as the regularization parameter in ridge regression. The additional regularization term $\gamma\mathbf{I}$ serves as stabilizing the ill-posed inversion problem in the above expression. Ridge regression can be readily integrated with kernel methods, which is essentially a simplified version of support vector regression [78]. Given a feature map $\boldsymbol{\varphi}(.)$ with its corresponding similarity metric $K(x, y) = \boldsymbol{\varphi}(x)^\top \boldsymbol{\varphi}(y)$, the leverage scores of kernel ridge regression can be expressed in a primal-dual form:

$$l_i^{KR}(\gamma) = \begin{cases} \text{Primal:} & \boldsymbol{\varphi}(\boldsymbol{x}_i)^\top (\mathbf{S}_{\boldsymbol{\varphi}} + \gamma \mathbf{I})^{-1} \boldsymbol{\varphi}(\boldsymbol{x}_i) \\ \text{Dual:} & (\mathbf{K}_{\boldsymbol{\varphi}} (\mathbf{K}_{\boldsymbol{\varphi}} + \gamma \mathbf{I})^{-1})_{ii} \end{cases} \tag{4.6}$$

Where $\mathbf{K}$ is the gram matrix with respect to similarity metric $K(.,.)$ and $\mathbf{S}_{\boldsymbol{\varphi}} = \sum_{i=1}^{N} \boldsymbol{\varphi}(\boldsymbol{x}_i)\boldsymbol{\varphi}(\boldsymbol{x}_i)^\top$ is proportional to the sample covariance matrix. Depending on different data sizes and feature map dimensions, one can choose to leverage either primal or dual form to ensure optimal computational efficiency.

In addition to a regression model diagnostic tool, RLS is crucial for sampling diverse landmarks in low-rank approximation problems, e.g., Nyström approximation for kernel matrices [79], [80]. Schreurs et al. also illustrate the effectiveness of implementing RLS sampling in mitigating generation biases brought by unbalanced training data in the framework of GAN [77]. Once the leverage scores are computed, one can immediately deduce the

sampling probability for each data point via normalization on the leverage scores: $p_i = l_i^{KR} / \sum_{j=1}^{N} l_j^{KR}$. Ideally, minority modes tend to have higher leverage scores, making these points more likely to be oversampled, resulting in a more uniform-like distribution of the resampled data. A motivating example is presented in figure 4.1. One can observe that samples from minority modes generally share higher RLSs and the RLSs for the majority group are very close to zero, thus RLS sampling could automatically oversample minority modes while data points from the majority group are downsampled. In this spirit, problem of imbalance could be resolved by RLS sampling.

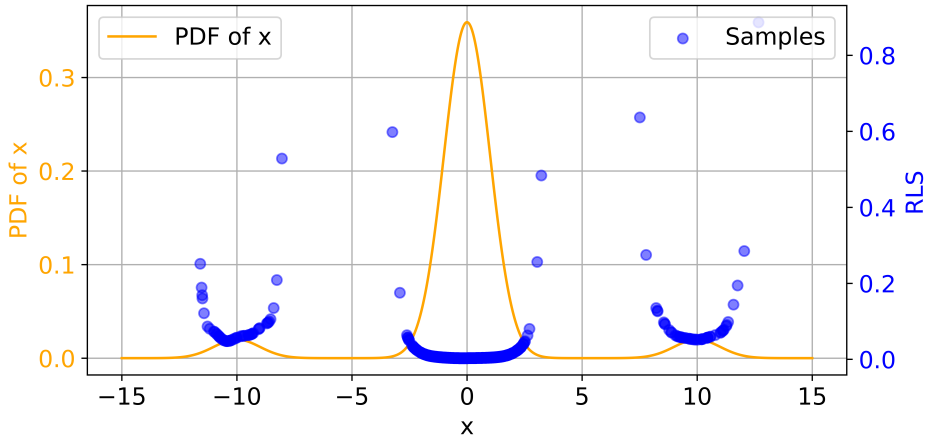

Figure 4.1: A toy example illustrates using RLS sampling can help capture minority modes in the data. The orange curve depicts the PDF of a Gaussian mixture distribution with 1 majority mode in the middle and 2 minority modes on two sides, where the PDF is defined as $p = 0.05\mathcal{N}(-10, 1) + 0.9\mathcal{N}(0, 1) + 0.05\mathcal{N}(10, 1)$. Blue points are a collection of random samples from that pdf, where the height of each point corresponds to the RLS for that individual point. RLSs are computed based on a Gaussian kernel with $\gamma = 10^{-3}$ and $\sigma = 3$. One can observe that samples from minority modes generally have higher RLSs, thus sampling based on the RLSs could contribute to balance the data. This figure is reproduced from [77].

4.2.2 Incorporating RLS sampling within the framework of Gen-RKM

The detailed sampling procedure based on RLSs in Gen-RKM is illustrated in this subsection. The main methodology is primarily developed based on [77], with some modifications applied due to differences in the model setting. We start by discussing two possible options for feature maps of RLS sampling in the framework of Gen-RKM.

Feature map choices To compute RLSs, a feature map function needs to be defined beforehand. Considering our work mostly focuses on learning image data, a more sophisticated similarity metric, typically a deep neural network, is preferred. Here we consider two different approaches for constructing feature maps.

- **Shared explicit feature map with Gen-RKM:** Both Gen-RKM and RLS sampling are kernel based methods, meaning that they each have a feature map function projecting training data into feature space. Therefore the most natural way is to use the encoder part of Gen-RKM directly as feature map in RLS sampling. In this case, computation on RLS and Gen-RKM will share one same explicit feature map network: $\boldsymbol{\varphi}(.) = \boldsymbol{\phi}(.)$ (recall that we denote feature map for Gen-RKM as $\boldsymbol{\phi}(.)$). This approach is preferred when no prior information is available.
- **Explicit feature map based on a pre-trained classifier:** Another possible option is to use the next-to-last layer of a pre-trained classifier, such as ResNet18[81], as the feature extractor. Similar ideas are proposed in [77], [82], where weighted sampling schemes are conducted based on the embeddings extracted from a pre-trained classifier. Some popular deep neural network models pre-trained on ImageNet[83] are readily implemented in `PyTorch` and `TensorFlow`. Notice that there is no need to fine-tune the pre-trained model on the exact training dataset, only forward pass operation is required to get image embeddings.

Techniques for dimension reduction Modern deep neural networks are characterized by their high dimensionality. For example, the output dimension of the next-to-last layer in AlexNet[84] is 4096. Computation on RLSs can be rather demanding when the dimension of feature space is large. To improve computational efficiency, we introduce two dimension reduction techniques for different feature choices, as suggested in [77].

- **Gaussian sketching:** Gaussian sketching is a ubiquitous approach in low-rank approximating large matrices in the field of randomized linear algebra. Let $\mathbf{Z}$ be a Gaussian random matrix of size $d \times k$ where $k << d$, and $\boldsymbol{\varphi}(\mathbf{X}_b)$ be a mini-batch of data with size m after feature map in RLS sampling. The sketched matrix is simply given by multiplying the random matrix:

$$\widetilde{\boldsymbol{\varphi}}(\mathbf{X}_b) = \mathbf{Z}^\top \boldsymbol{\varphi}(\mathbf{X}_b) \in \mathbb{R}^{m \times k}. \tag{4.7}$$

This random projection guarantees that pairwise distances in the dataset are preserved in a much lower embedding space according to Johnson-Lindenstrauss lemma [77]. One can refer to [85] for more technical details about sketching methods.

- **UMAP:** Uniform Manifold Approximation and Projection (UMAP) is a widely used non-linear dimensionality reduction technique developed by McInnes et al. [86]. Compared to Gaussian sketching, UMAP can preserve more complex patterns in the data, though its training process is significantly slower.

Modified training schemes based on RLS sampling Detailed training algorithms based on RLS sampling are outlined in this subsection. For explicit feature map based on a pre-trained classifier, UMAP is preferred for dimension reduction in this case since RLSs are only computed once before the main training loop. The algorithm essentially follows the same principle as the inverse frequency sampling discussed earlier, i.e. performing weighted sampling under each mini-batch as well as resampling the full dataset for the final computation step, as presented in algorithm 4.

Algorithm 4 RLS Sampling in Gen-RKM with explicit feature map based on a pre-trained classifier

Input: training data $\mathbf{D} = \{\boldsymbol{x}_i\}_{i=1}^N$; mini-batch size m;
regularization parameters $\eta, c_{\text{stab}}, \gamma_{RKM}$ for Gen-RKM;
explicit feature map $\boldsymbol{\phi}_{\boldsymbol{\theta}}(.)$ and explicit pre-image map $\boldsymbol{\psi}_{\boldsymbol{\zeta}}(.)$ for Gen-RKM;
pre-trained classifier feature map $\boldsymbol{\varphi}(.)$ and regularization γ_{RLS} for RLS sampling;
dimension reduction size k

▷ *Pre-compute RLS* ◁
$\{\widetilde{\boldsymbol{\varphi}}(\boldsymbol{x}_i)\}_{i=1}^N \leftarrow \text{UMAP}_k(\{\boldsymbol{\varphi}(\boldsymbol{x}_i)\}_{i=1}^N)$ ▷ *get UMAP embeddings*
$l_i \leftarrow \widetilde{\boldsymbol{\varphi}}(\boldsymbol{x}_i)^\top (\mathbf{S}_{\tilde{\varphi}} + \gamma_{RLS}\mathbf{I})^{-1}\widetilde{\boldsymbol{\varphi}}(\boldsymbol{x}_i)$ ▷ *Compute leverage scores*
$p_i \leftarrow l_i / \sum_{j=1}^n l_j$
▷ *Training loop* ◁
for each epoch **do**
  for each mini-batch $B \subset \mathbf{D}$ **do**
    $B \leftarrow$ sample a mini-batch of size m with probability $\boldsymbol{x}_i \sim p_i$
    do step 6-11 in algorithm 1
  end for
end for
▷ *Modified final computation step in Gen-RKM* ◁
$\mathbf{D}_{\text{resampled}} \leftarrow$ resample the full dataset with probability $\boldsymbol{x}_i \sim p_i$
repeat step 7-9 in algorithm 1 on $\mathbf{D}_{\text{resampled}}$

For the RLS sampling using a shared feature map with Gen-RKM, RLSs are recomputed in each iteration since the feature map for RLS is not fixed (i.e. parameters of feature map network need to be updated with every iteration). To alleviate the computational load, Gaussian sketching is implemented to reduce the dimension of the feature map. Sketching is considerably fast even when performed in every iteration, due to the simplicity of matrix multiplication involved. In addition, a two-stage sampling procedure is applied as suggested in [77]. First, a subset of training data is uniformly sampled (we set the size of this subset to be 20 times the mini-batch size), and RLSs are computed only for that core set. In the second stage, a mini-batch is sampled based on the calculated RLSs of this core set for training. The detailed implementation is shown in algorithm 5. Likewise in the previous setting, the final computation step in Gen-RKM is conducted based on the RLS-resampled dataset.

Algorithm 5 RLS Sampling in Gen-RKM with a shared feature map

Input: training data $\mathbf{D} = \{\boldsymbol{x}_i\}_{i=1}^N$; mini-batch size m;
regularization parameters $\eta, c_{\text{stab}}, \gamma_{RKM}$ for Gen-RKM;
explicit feature map $\boldsymbol{\phi}_{\boldsymbol{\theta}}(.)$ and explicit pre-image map $\boldsymbol{\psi}_{\boldsymbol{\zeta}}(.)$ for Gen-RKM;
regularization parameter γ_{RLS} for RLS sampling and dimension reduction size k;

▷ *Training loop* ◁
for each epoch **do**
  for each mini-batch $B \subset \mathbf{D}$ **do**
    ▷ *two-stage sampling procedure* ◁
    $\mathbf{D}_s \leftarrow$ uniformly sample a subset from $\mathbf{D}$ with size $20 * m$
    $\widetilde{\boldsymbol{\phi}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \leftarrow \mathbf{Z}_k^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \quad \text{s.t.} \boldsymbol{x}_i \in \mathbf{D}_s$ ▷ *Gaussian sketching*
    $l_i \leftarrow \widetilde{\boldsymbol{\phi}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top (\mathbf{S}_{\widetilde{\boldsymbol{\phi}}_{\boldsymbol{\theta}}} + \gamma_{RLS}\mathbf{I})^{-1} \widetilde{\boldsymbol{\phi}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$
    $p_i \leftarrow l_i / \sum_{j=1}^{20m} l_j$
    $B \leftarrow$ sample a mini-batch of size m with probability $\boldsymbol{x}_i \sim p_i$
    do step 6-11 in algorithm 1
  end for
end for
▷ *Compute final RLS on full dataset* ◁
$\widetilde{\boldsymbol{\phi}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \leftarrow \mathbf{Z}_k^\top \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \quad \text{s.t.} \boldsymbol{x}_i \in \mathbf{D}$
$l_i^{\text{final}} \leftarrow \widetilde{\boldsymbol{\phi}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top (\mathbf{S}_{\widetilde{\boldsymbol{\phi}}_{\boldsymbol{\theta}}} + \gamma_{RLS}\mathbf{I})^{-1} \widetilde{\boldsymbol{\phi}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$
$p_i^{\text{final}} \leftarrow l_i^{\text{final}} / \sum_{j=1}^{N} l_j^{\text{final}}$
▷ *Modified final computation step in Gen-RKM* ◁
$\mathbf{D}_{\text{resampled}} \leftarrow$ resample a dataset of size N with probability $\boldsymbol{x}_i \sim p_i^{\text{final}}$
repeat step 7-9 in algorithm 1 on $\mathbf{D}_{\text{resampled}}$

4.2.3 Extension: Other measurement for imbalance sampling

As mentioned in Section 2.4.2, many studies focus on anomaly detection in unsupervised learning settings. These experiences can be leveraged to address the issue of imbalanced data within the Gen-RKM framework. In addition to the ridge leverage score, which assesses the importance of individual observations in a dataset, other anomaly detection methods can be utilized to measure the uniqueness or anomaly level of observations. Subsequently, re-sampling can be carried out based on the uniqueness of each sample, similar to the approach used in ridge leverage score sampling.

Isolation forest (Iforest)[66], as mentioned in Section 2.4.2, provides an assessment for the degree of the anomaly of each data point as follows,

$$S(x,n) = 2^{-\frac{h(x)}{c(n)}} \tag{4.8}$$

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{4.9}$$

$$H(k) = \ln(k) + \xi \tag{4.10}$$

where

- $S(x,n)$ represents the anomaly score of data point x, n is the number of samples. The larger the score, the higher the likelihood that x is considered an anomaly or outlier.
- $h(x)$ is the path length required to isolate point x in the isolation forest. Shorter path lengths suggest that x is easier to isolate, thus potentially being an outlier.
- $c(n)$ is the average path length under normal conditions for a sample size n. This calculation takes into account the balance and imbalance factors of a binary search tree.
- $H(k)$ is a harmonic number, typically used to estimate the average path length of binary search trees. And ξ is the Euler-Mascheroni constant, approximately 0.5772156649.

We can easily integrate isolation forest anomaly score sampling (Iforest sampling) into the Gen-RKM framework to address the imbalance data problem by simply replacing steps 3-4 in Algorithm 4 with the computation of anomaly scores using an isolation forest model (see $S(x,n)$ in equation 4.8)

based on the extracted features from the feature map, while maintaining all other steps unchanged. Note that this is only applicable with a pre-trained classifier-based feature map, as it allows the isolation forest to be fitted just once in the algorithm.

Chapter 5

Experiments and Results

In this chapter, various sampling techniques outlined in Chapter 4 are evaluated within the Gen-RKM framework using both synthetic datasets and real-world datasets. We begin with the basic experiment setup, including the datasets used, evaluation tools, and key hyperparameter settings in Section 5.1. After that, we conducted experiments for both the supervised and unsupervised settings to validate their effectiveness. More specifically, Section 5.2 primarily evaluates the performance of inverse frequency sampling in the supervised setting, while also studying and improving the applicability of conditional generation under unbalanced data. Next, Section 5.3 examines the performance of RLS sampling in diversifying generation under the unsupervised manner. In addition, Iforest sampling (i.e., weighted sampling based on anomaly scores outputted from the Isolation Forest algorithm), an extension of RLS sampling, is also included in the comparison. Finally, a series of additional studies are conducted in Section 5.3.5 to gain full insights into the strengths and weaknesses of our proposed methods. The Python code for the experiment part has been made publicly available on GitHub.

5.1 Experiment set-up

5.1.1 Datasets

The following datasets are considered in the experiment part:

Synthetic ring/grid datasets Two unbalanced synthetic datasets are generated following the setting in [77]: an unbalanced ring with 4 minority modes (Ring) and an unbalanced grid (Grid) with 10 minority modes (shown

in Figure 5.1 below). Ring is a mixture of eight two-dimensional isotropic Gaussians in the 2D-plane with means $2.5 \times (\cos(\frac{2\pi}{8}i), \sin(\frac{2\pi}{8}i))$ and standard deviation 0.05 for $i \in 1, ..., 8$. The probability of sampling from the first 4 consecutive Gaussians is only 0.05 times the probability of sampling from the last 4 modes. Grid is a mixture of 25 two-dimensional isotropic normals with a standard deviation of 0.05 and with means on a square grid with spacing 2. The first rectangular blocks of 2×5 adjacent modes are depleted with a factor of 0.05. 2500 samples are generated for each normal mode, thus 125 for each minority mode.

Figure 5.1: 2D synthetic data: Ring(left) and Grid(right)

Unbalanced 012-MNIST/MNIST dataset For the natural dataset from the real world, we opt to employ MNIST as the first benchmark dataset for evaluating the effectiveness of different weighted sampling schemes. MNIST[87] contains 60000 images of handwritten digits varying from 0 to 9, where the size of each image is fixed on 28×28 with single color channel. We mainly consider two unbalanced variants of MNIST in our experiments. The first created dataset, namely unbalanced 012-MNIST, consists of only the digits 0, 1 and 2 where digit 2 is the minority class. Imbalance is artificially introduced by (randomly) reducing the number of digit 2 so that the probability of sampling digit 2 is smaller than that of sampling from other digits. The degree of depletion for digit 2 is determined by a specified imbalance ratio (i.e., ratio of the number of samples in minority class to the number of samples in majority class). The second dataset, unbalanced MNIST, consists of all the digits from 0 to 9. The digits 0, 1, 2, 3, and 4 have been selected as the minority classes, and their numbers are reduced as the same fashion in unbalanced 012-MNIST.

Unbalanced Fashion MNIST dataset Considering the relatively simple

patterns of MNIST dataset, Fashion MNIST [88] is also used in our experiments for a more challenging replacement. Fashion MNIST consists of 60000 images of fashion items (e.g., different types of shoes and clothes) with 10 categories. Likewise in the MNIST setup, all images in Fashion MNIST are gray-scale with size 28×28. Fashion MNIST is generally more complex due to the high variability in shapes and patterns among the images of different real fashion items. The same procedure in the MNIST case is leveraged to create the unbalanced version of Fashion MNIST dataset. Clothing and bag-related categories (T-shirt, Trouser, Pullover, Dress, Coat, Shirt, Bag) are set to minority classes while shoe-related categories (Sandal, Sneaker, Ankle boot) are considered as the majority classes.

5.1.2 Evaluation process & metrics

As for the evaluation part, 10k samples are randomly generated by each trained model. For the 2D synthetic datasets, valid samples are defined within 3 standard deviations of the nearest modes. A mode is covered by a certain model if there are at least 50 valid generated samples within 3 standard deviations of the mode's center. For image-based datasets, a well-trained classification model is employed to identify the labels of generated samples. Since our objective is to assess both the diversity and quality of the generated samples by different models, the following evaluation metrics are leveraged in our experiment.

Number of generated samples per mode We measure mode coverage by counting the number of generated samples under each mode based on the prediction from a trained classifier. Ideally, the distribution of samples per mode should be close to a uniform distribution, which indicates each mode is well captured even though unbalanced training data is given.

KL score A follow-up metric, namely KL score, can be calculated based on the predicted labels of generated samples. KL score is obtained by computing the Kullback–Leibler (KL) divergence between the distribution of the classified labels from the generated samples with a balanced label distribution. We consider KL score as the primary metric to quantify the deviation of the label distribution of the generated samples from a uniform distribution. In the best case, KL score should be very close to zero indicating a balanced generation is acquired. This metric is widely used in various works related to addressing mode collapse problem in GANs [73], [77], [89], [90].

FID So far, the above two metrics are both label-based, which means they merely measure the imbalance under the class level. However, the imbalance could occur in both inter-class and intra-class. Additionally, a well-balanced generation does not necessarily ensure results with high quality. For example, the patterns of the generated samples could be highly diverse, but the overall generation could be considered poor quality due to blurriness, weird distortions, and incorrect proportions. Therefore, we propose to employ Fréchet Inception Distance (FID) [91] as a supplementary evaluation tool for assessing both the diversity and quality of generated images. The nature of FID is to measure the similarity between the distributions of real data and generated data. To compute FID, a pre-trained classifier (normally we use inception-V3) is implemented to extract features of image data, FID is then given by the Fréchet distance of two Gaussian distributions:

$$\text{FID} = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\|^2 + \text{Tr}(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g)^{1/2}) \tag{5.1}$$

where $\boldsymbol{\mu}_r, \boldsymbol{\mu}_g$ are the means and $\boldsymbol{\Sigma}_r, \boldsymbol{\Sigma}_g$ are the covariance matrices of extracted embedding of real and generated images, respectively. The reference statistics (i.e., $\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r$) are computed based on the (class) balanced dataset in all experiments. A low FID score implies that the distribution of generated images closely matches the distribution of balanced training samples, indicating a more diverse and higher-quality generation.

5.1.3 Hyperparameter details

In this section, we outline the key hyperparameters used in the experiment part. Unless otherwise specified, the hyperparameter settings used in the subsequent experiment part will default to those mentioned below.

Hyperparameters for Gen-RKM The basic hyperparameter settings for training Gen-RKM is displayed in Table 5.1. We generally follow the same configurations as those in [8], except that we slightly increase the dimension of the feature map and the mini-batch size. Considering our experiments required re-training the models multiple times, and the sizes of used datasets are relatively large. To maintain more computationally efficient training, Gen-RKM is computed in the primal form across all our experiments (i.e., eigen-decomposition on the covariance matrix), which necessitates $d_f < m$ to guarantee that the covariance matrix is full rank for more stable computations [8].

Hyperparameter	Dataset		
	2D Ring/Grid	012-MNIST	MNIST/Fashion
regularization parameters: $\{\eta, c_{\text{stab}}, \gamma_{\text{RKM}}\}$	{1;1;100}	{1;1;100}	{1;1;100}
dimension of feature map: d_f	128	300	300
dimension of latent space: s	8/25	10	10
mini-batch size: m	256	328	328
maximum epoch number: N_{epoch}	150	100	150
number of Gaussian components in GMM: l	8/25	3	10
learning rate	0.0001	0.0001	0.0001

Table 5.1: Hyperparameter settings for Gen-RKM

Neural network architectures The detailed neural network architectures for the feature map and the pre-image map in Gen-RKM are summarized in Table 5.2. For 2D synthetic data, the architecture includes 3 fully connected layers with ReLU activation functions in between. For MNIST-related datasets, the architecture consists of 2 convolutional layers, and ReLU activations are implemented similarly.

Dataset	Input size	Architecture	
		Feature map	Pre-image map
2D Ring/Grid	2	FC 32 (Linear) ReLu($\alpha = 0.2$) FC 64 (Linear) ReLu($\alpha = 0.2$) FC 128 (Linear)	reverse of fm
012-MNIST/MNIST/Fashion	$28 \times 28 \times 1$	Conv $32 \times 4 \times 4$ ReLu($\alpha = 0.2$) Conv $64 \times 4 \times 4$ ReLu($\alpha = 0.2$) FC 300 (Linear)	reverse of fm

Table 5.2: Details of network architectures used in Gen-RKM. All convolutional layers and transposed convolutional layers have stride 2 and padding 1. Pre-image map is the reverse of feature map architecture, except that a sigmoid activation function is implemented for the output layer [8].

Hyperparameters for RLS sampling Regarding the hyperparameters used in RLS sampling, we generally adopt the same settings as in the original work

of RLS-GAN [77]. More specifically, the ridge regularization parameter is set to 0.0001, and feature maps in kernel ridge regression are reduced to $k = 25$ by either UMAP or Gaussian sketching. However, Instead of using Inception-v3 as mentioned in [77], we empirically observe that RLS sampling with an explicit feature map based on AlexNet could achieve better performance. Therefore, throughout the experiments, the pre-trained classifier-based RLS sampling is computed using the embeddings extracted from the next-to-the-last layer of Alexnet (which has already been pre-trained on the ImageNet dataset). One can refer to Section 5.3.5 for the detailed results of the ablation study on the impact of different pre-trained classifiers on the performance of RLS sampling.

Hyperparameters for Iforest sampling Fine-tuning the parameters in Isolation Forest is challenging and typically requires some labeled data for validation. However, since no prior information regarding labels is provided in a fully unsupervised manner, we use the default hyperparameter settings provided in the `sklearn` package.

5.2 Experiments on inverse frequency sampling under supervised setting

5.2.1 Unbalanced MNIST/Fashion MNIST

Given that inverse frequency sampling transforms the input distribution into a uniform one, there's a definite and notable improvement in the diversity of the generated samples. Figure 5.2 shows the distributions of the generated samples from Gen-RKM and Gen-RKM with inverse frequency sampling using MNIST and Fashion MNIST datasets respectively. The minority classes are shown in red and the normal classes are in blue. The improvement in generation diversity is notable. Figures 5.3 visualizes the random generation of Gen-RKM using the MNIST and FashionMNIST datasets, before and after applying inverse frequency sampling. In these figures, the minority classes are highlighted with red rectangles.

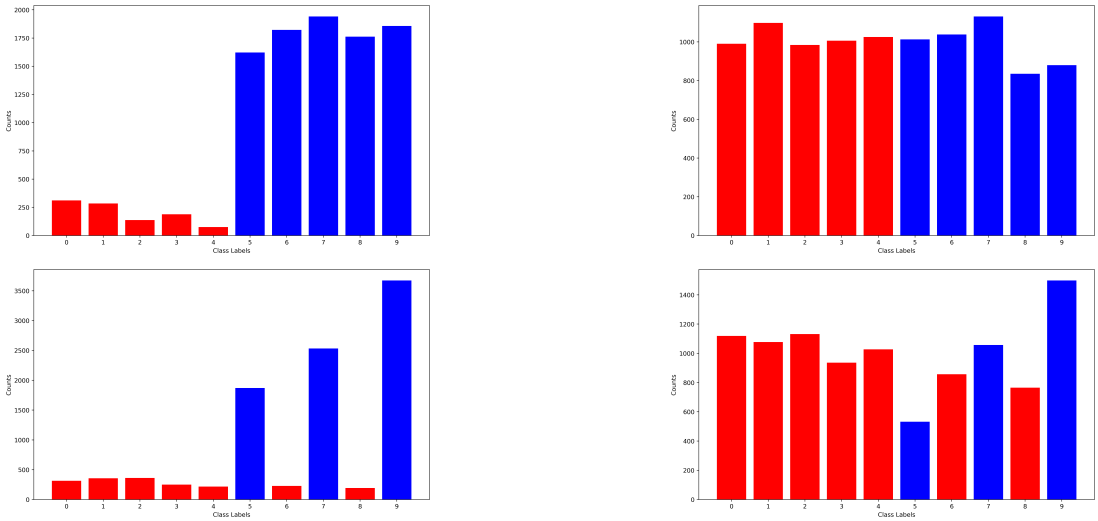

Figure 5.2: Number of generated samples in each class for the unbalanced MNIST dataset: Vanilla Gen-RKM (top left) vs. Gen-RKM with inverse frequency sampling (top right). For the Fashion MNIST dataset: Vanilla Gen-RKM (down left) vs. Gen-RKM with inverse frequency sampling (down right). A rebalancing effect is visible

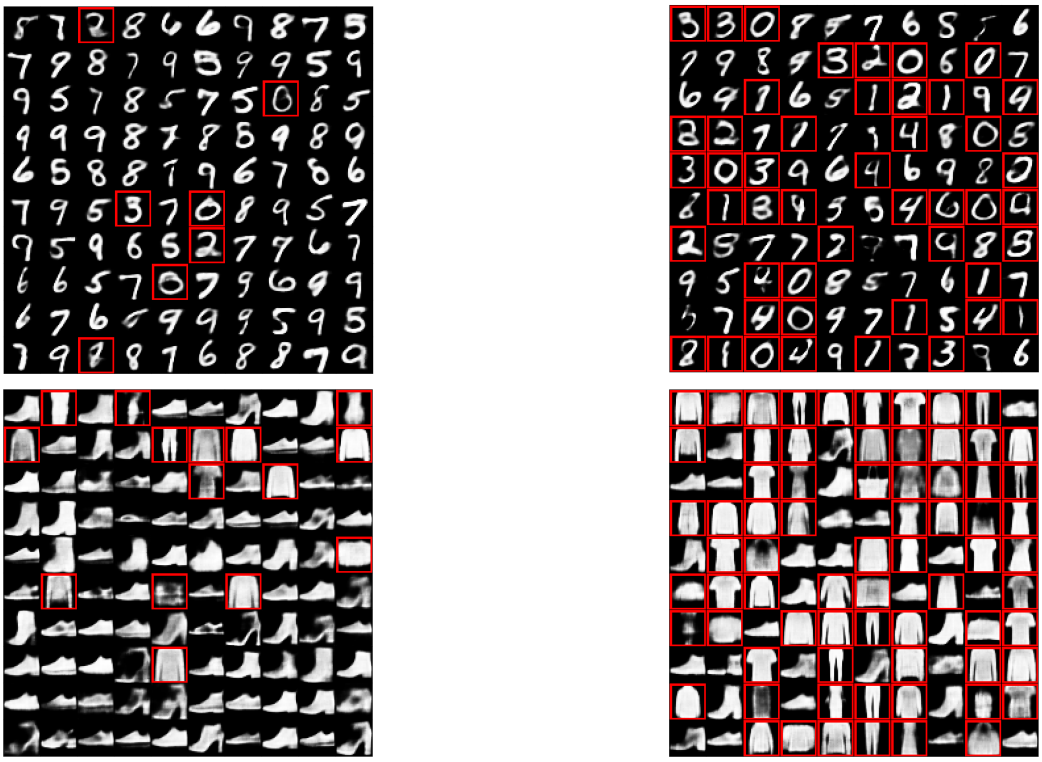

Figure 5.3: Random generation of unbalanced MNIST dataset: vanilla Gen-RKM (top right) and Gen-RKM with inverse frequency sampling (top left). Fashion MNIST dataset: vanilla Gen-RKM (down right) and Gen-RKM with inverse frequency sampling (down left)

5.2.2 Effect of inverse frequency sampling on conditional generation in Gen-RKM

We examine the effectiveness of conditional generation (described in Section 4.1.3) within the framework of Gen-RKM in this section. Conditional generations on unbalanced MNIST and unbalanced Fashion MNIST are visualized in Figure 5.4. Unsurprisingly, the generation quality could be worse if the generation is conditioned on a minority class, and it is likely to incorrectly generate images from other classes. For example, if one inspects the fifth column (generation conditioned on digit 4) in the MNIST case, the generation resembles digit 9 instead of digit 4. This issue is caused by the distorted latent space due to unbalanced data, latent representations of minority modes could be blurred or mixed with other majority modes, leading to failed conditional generation for underrepresented groups. This problem can now be resolved by adapting Gen-RKM with inverse frequency sampling. As shown in the right side of Figure 5.4, correct and clearer conditional generation can be obtained. By cleverly combining inverse frequency sampling with conditional generation, one can augment minority classes even from highly unbalanced training data.

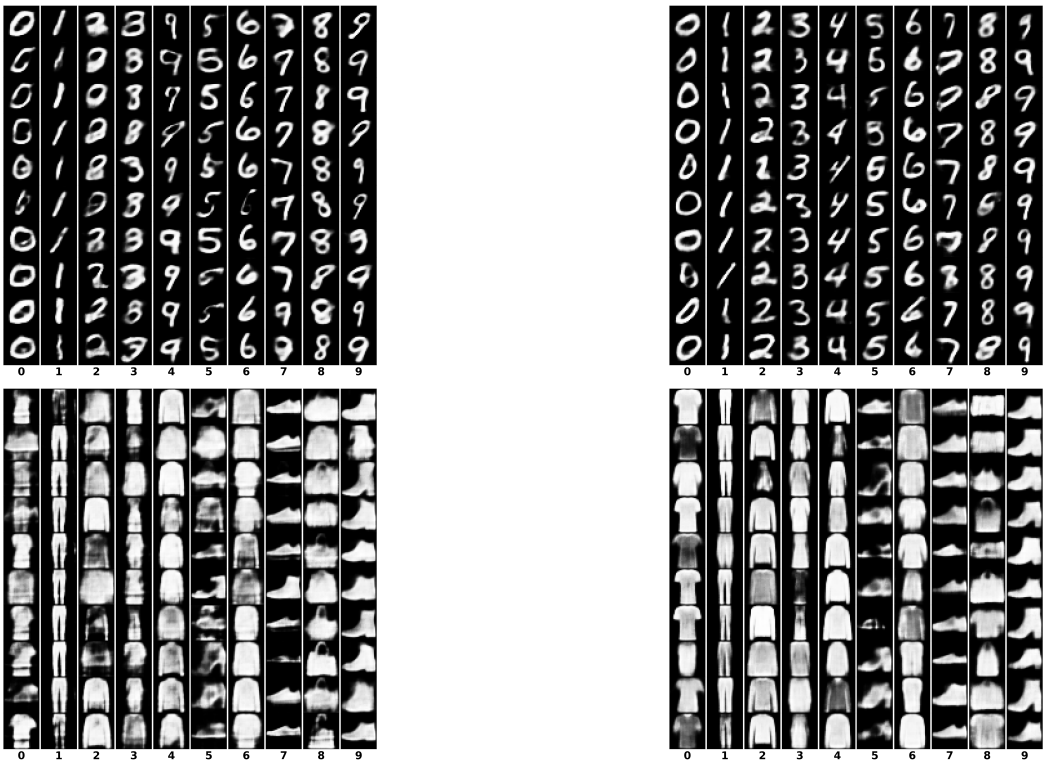

Figure 5.4: Conditional generations produced by multi-view Gen-RKM on the unbalanced MNIST dataset (top row) and the unbalanced Fashion MNIST dataset (bottom row). Figures in left column is generated by vanilla Gen-RKM, and figures in the right column are generated by Gen-RKM adapted with inverse frequency sampling. Images in each column of each figure are generated conditionally on a particular label. Imbalance ratios for both datasets are set to 0.1.

5.3 Experiments on weighted sampling schemes under unsupervised setting

In this section, we investigate the effectiveness of different diversity sampling schemes under unsupervised settings where no label information is provided during the training phase. Notice that label information is still required only for the purpose of evaluation. We start by evaluating RLS sampling in Gen-RKM on a toy 2D dataset in Section 5.3.1. Next, Section 5.3.2 and 5.3.3 describe the results of experiments on MNIST-related datasets. In addition, results on Fashion MNIST under a similar setting are presented in Section 5.3.4. Lastly, some additional studies are conducted in 5.3.5.

5.3.1 Synthetic ring/grid dataset

The generated samples from models trained both with and without RLS sampling (using a shared feature map) are presented in Figure 5.6. It is evident from the visualization that the RKM, when trained using uniform sampling, fails to generate a sufficient number of points for the first four minority modes in the Ring and the first ten minority modes in the Grid. This problem is effectively solved by employing RLS sampling. The improved sampling effectiveness is demonstrated through the comparison of the two generation distributions for the Ring dataset, as shown in Figure 5.5. Meanwhile, Table 5.3 presents the results of the two variants of RLS sampling, the primal form (shared feature map) and the dual form (RBF kernel with $\sigma = 0.15$), on 2D synthetic datasets. Here, "minority mean" refers to the average number of points generated in each minority mode, while "number modes" indicates how many modes have generated more than 50 points. Both the primal and dual forms of RLS sampling enhance mode coverage and substantially increase the generation count in the minority modes.

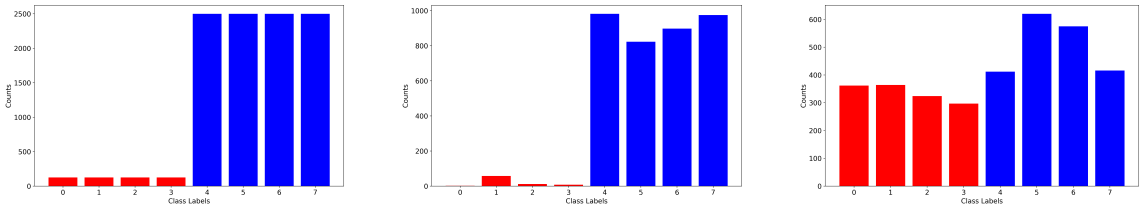

Figure 5.5: Number of samples in each mode from RING dataset: Original RING(left), Vanilla Gen-RKM (middle) and Gen-RKM with RLS sampling (right).

	Ring with 8 modes		Grid with 25 modes	
	Num modes	Minority mean	Num modes	Minority mean
RKM	4.7(0.67)	17.4(8.02)	13.4(0.97)	3.45(0.63)
RLS-RKM(shared)	**7.7(0.48)**	284.0(54.90)	14.2(0.91)	**74.5(3.01)**
RLS-RKM(rbf)	7.0(1.25)	**344.73(43.71)**	**20.02(1.75)**	62.6(3.97)

Table 5.3: Results of experiments on 2D synthetic datasets (Experiments are replicated over 10 runs with random initialization).

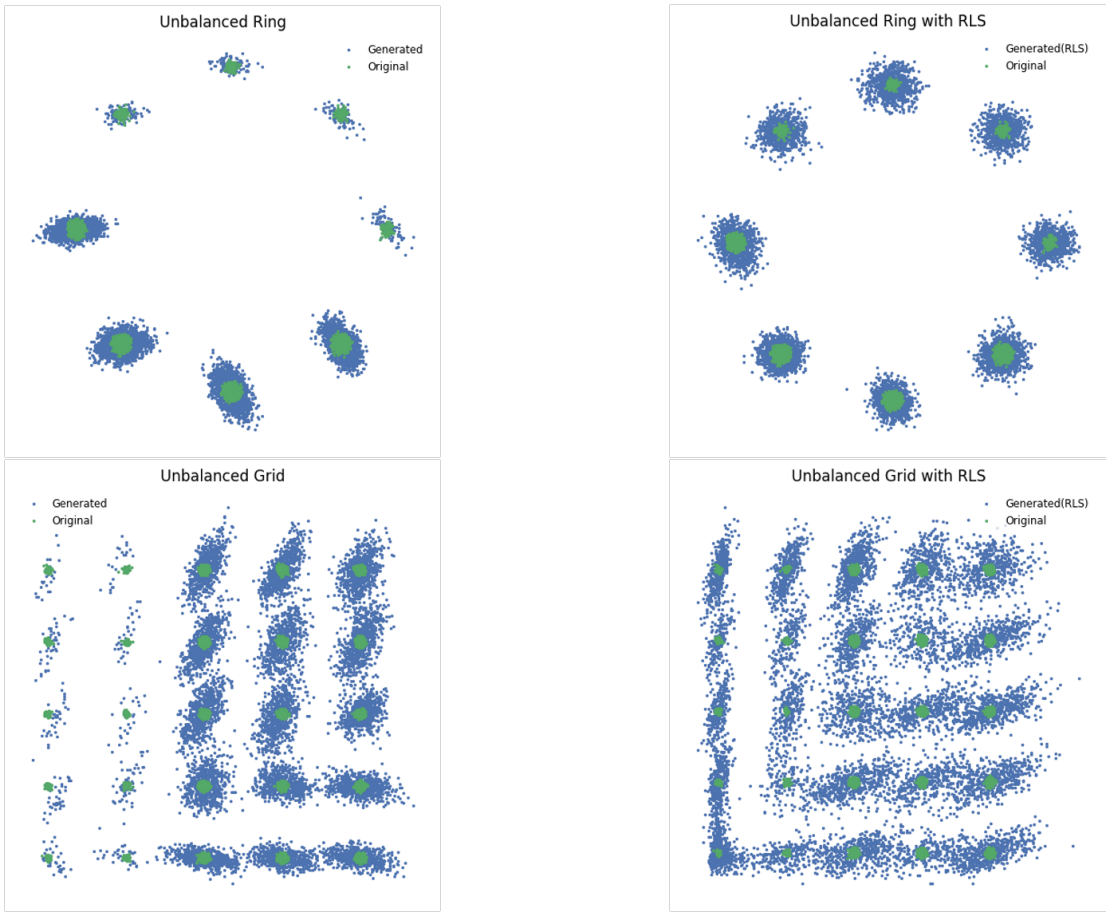

Figure 5.6: Visualizations of random generations on synthetic datasets: Vanilla Gen-RKM on RING (top left), Gen-RKM with RLS sampling on RING (top right), Vanilla Gen-RKM on GRID (down left) and Gen-RKM with RLS sampling on GRID (down right).

5.3.2 Unbalanced 012-MNIST

In this experiment, the performance of different weighted sampling schemes is studied on the unbalanced 012-MNIST dataset, which corresponds to a single minority mode scenario. Specifically, two RLS sampling variants with Gen-RKM are considered: RLSs computed based on an explicit feature map obtained from the last-to-next-layer of a pre-trained classifier (abbreviated as "RLS-RKM (class)"), and RLSs calculated from shared feature map with Gen-RKM (abbreviated as "RLS-RKM (shared)"), as described in Section 4.2.2. In addition, weighted sampling built on isolation forest score (described in Section 4.2.3 and abbreviated as "Iforest RKM") is included. The baseline model in our experiment is Gen-RKM without any modifications but with the same network architecture and hyperparameter setting. In the evaluation process, the mode of each generated sample is identified by a well-trained MNIST classifier based on a ResNet18 architecture, which achieves 99.4%

accuracy on the test set. The results of calculated metrics are presented in Table 5.4 where "Minority" refers to the number of generated samples classified as minority mode, and the distribution of numbers of generated samples under each mode is showcased in Figure 5.7.

It can be observed from Table 5.4 that both RLS sampling and Iforest sampling could improve mode coverage and encourage more diverse generation under various imbalance ratio settings. More specifically, RLS-RKM (class) clearly outperforms all other methods in terms of both KL score and FID. Overall, This experiment demonstrates the immense effectiveness of RLS sampling with a pre-tained classifier as explicit feature map in single minority mode scenario. Samples from the minority group are generated more frequently as shown in Figure B.1, thus a debiased generation is achieved.

Imbalance ratio	Model	Minority	KL score (↓)	FID (↓)
0.05	RKM	75 (±34)	0.37 (±0.02)	64.97 (±2.25)
	RLS-RKM (shared)	410 (±60)	0.32 (±0.02)	74.09 (±3.39)
	RLS-RKM (class)	**1563** (±201)	**0.09** (±0.02)	**55.95** (±4.52)
	IforestRKM	295 (±150)	0.30 (±0.04)	65.22 (±2.79)
0.10	RKM	135 (±76)	0.35 (±0.03)	65.04 (±1.74)
	RLS-RKM (shared)	782 (±64)	0.24 (±0.01)	70.02 (±1.07)
	RLS-RKM (class)	**2259** (±144)	**0.03** (±0.01)	**55.42** (±6.33)
	IforestRKM	863 (±180)	0.18 (±0.03)	59.63 (±3.74)
0.30	RKM	842 (±113)	0.18 (±0.02)	56.98 (±1.54)
	RLS-RKM (shared)	2016 (±171)	0.09 (±0.01)	62.04 (±3.32)
	RLS-RKM (class)	**2868** (±77)	**0.01** (±0.00)	51.95 (±3.60)
	IforestRKM	2206 (±183)	0.04 (±0.02)	**51.77** (±4.33)

Table 5.4: Results of experiments on the unbalanced 012-MNIST dataset under different imbalance ratios. Experiments are replicated over 5 runs with random initializations. Means and standard deviations (enclosed in parentheses) for each metric are reported.

5.3.3 Unbalanced MNIST

It would also be meaningful to investigate how RLS sampling and Iforest sampling perform on a more challenging dataset, one that has multiple minority classes instead of just one. In the unbalanced MNIST dataset, half of the classes are minority classes, namely digits 0-4. Similarly, RLS-RKM

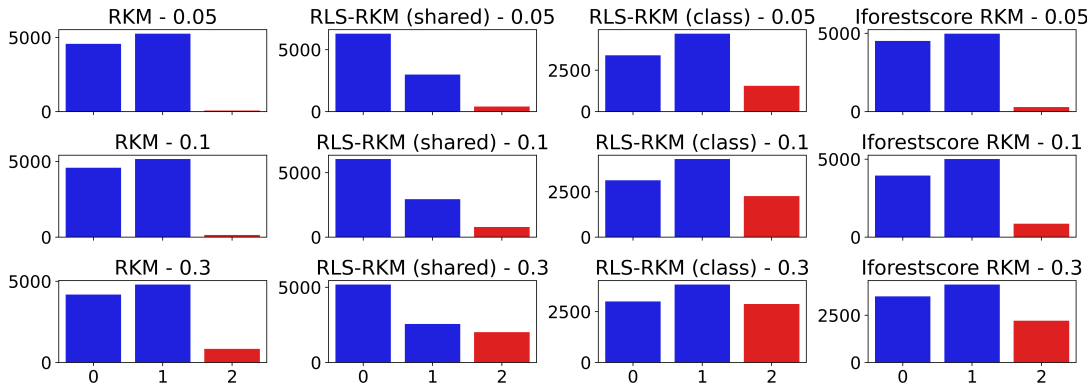

Figure 5.7: Number of generated samples per mode from different RKM models on the unbalanced 012-MNIST dataset. Each row indicates that the dataset is generated with a specific balance ratio. Each column corresponds to a different model used. The minority mode in each figure is depicted in red color and, the majority modes are depicted in blue.

(class) achieves best performance on KL score and FID compared with other approaches as observed from Table 5.5. Resampling schemes can more or less improve mode coverage in unbalanced data; however, when the imbalance ratio is very small (i.e., extremely unblanaced data), the improvement can be significantly diminished. If one looks at the distribution of generated labels in Figure 5.8, it is interesting to find that not all minority classes are oversampled when in RLS-RLM (class) and Iforest RKM. Instead, some particular minority classes might be downsampled. For example, minority classes such as digits 2,3, and 4 are generated even less frequently than before using RLS sampling or Iforest sampling, while only digits 0 and 1 are generated more frequently. This would ultimately result in a more unbalanced distribution of generated samples, as can be observed in the third and the fourth columns of Figure 5.8. Considering the sampling weights in both RLS (class) sampling and Iforest sampling are calculated based on embeddings extracted by a pre-trained classifier. Since this pre-trained classifier was not trained on this specific training dataset, it might fail to distinguish certain minority classes from others. The relatively low uniqueness results in a lower probability of being sampled. As for the RLS-RKM (shared), all minority classes are generally oversampled, but its ability to correct the imbalance is relatively weaker compared to other methods. To conclude, although weighted sampling schemes can more or less improve the diversity and quality of generation given unbalanced data with multiple minority modes, none of our methods could successfully captured all the underrepresented groups in the unbalanced MNIST dataset.

Imbalance ratio	Model	Minority mean	KL score (↓)	FID (↓)
0.05	RKM	113 (±13)	0.49 (±0.02)	39.19 (±1.17)
	RLS-RKM (shared)	179 (±11)	0.42 (±0.01)	40.36 (±0.46)
	RLS-RKM (class)	**342** (±35)	**0.35** (±0.01)	**34.65** (±1.95)
	IforestRKM	144 (±15)	0.49 (±0.02)	39.08 (±1.01)
0.10	RKM	185 (±11)	0.40 (±0.01)	38.89 (±0.18)
	RLS-RKM (shared)	327 (±3)	**0.27** (±0.00)	38.38 (±0.93)
	RLS-RKM (class)	**462** (±15)	**0.27** (±0.01)	**33.68** (±0.97)
	IforestRKM	257 (±7)	0.37 (±0.01)	35.32 (±1.17)
0.30	RKM	478 (±14)	0.15 (±0.01)	34.09 (±1.28)
	RLS-RKM (shared)	629 (±11)	**0.10** (±0.01)	36.01 (±0.97)
	RLS-RKM (class)	**797** (±14)	0.13 (±0.01)	**30.74** (±1.67)
	IforestRKM	547 (±44)	0.15 (±0.02)	32.23 (±0.24)

Table 5.5: Results of experiments on the unbalanced MNIST dataset under different imbalance ratios. Experiments are replicated over 3 runs with random initializations. Means and standard deviations (enclosed in parentheses) for each metric are reported. "Minority mean" refers to the average number of samples generated from each minority mode.

5.3.4 Unbalanced Fashion MNIST

Until now, experiments related to natural image datasets have only conducted on handwritten digits. In this section, we verify whether the use of RLS sampling and Iforest sampling in Gen-RKM could maintain similar performance on the Fashion MNIST dataset. The class of each generated sample is identified by a resnet18-type classifier which is trained up to 90.8% accuracy on the test set. Unlike unbalanced MNIST, most of the categories in unbalanced Fashion MNIST are minority classes, whereas unbalanced MNIST has an equal number of minority and majority classes. The performance results of different sampling approaches are reported in Table 5.6 and in Figure 5.9. Compared to the previous experiment, the performance of almost all methods is slightly better. Additionally, RLS-RKM (class) continues to outperform other methods, achieving the smallest FID and KL score on the unbalanced Fashion MNIST dataset.

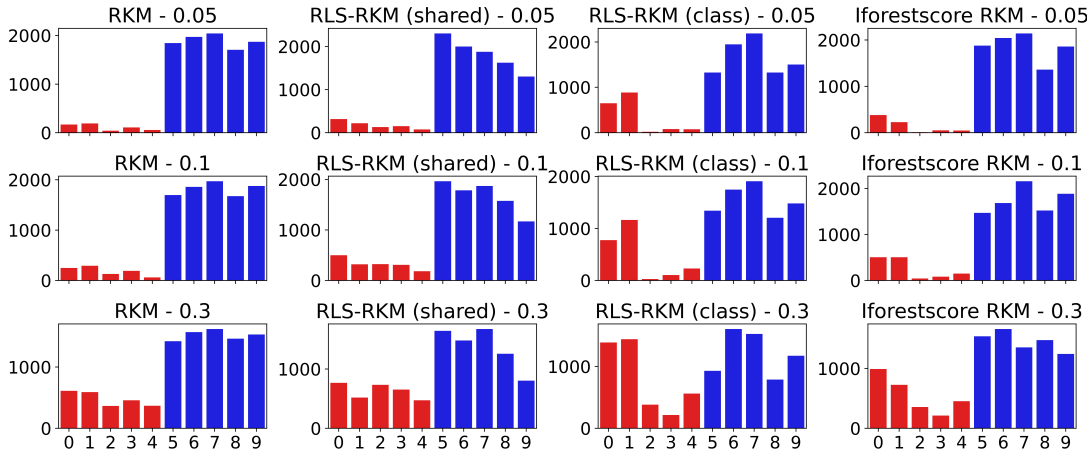

Figure 5.8: Number of generated samples per mode from different RKM models on the unbalanced MNIST dataset.

Imbalance ratio	Model	Minority mean	KL score (↓)	FID (↓)
0.1	RKM	273 (±2)	0.58 (±0.01)	93.02 (±1.12)
	RLS-RKM (class)	**623** (±13)	**0.20** (±0.02)	**70.92** (±0.28)
	RLS-RKM (shared)	437 (±2)	0.38 (±0.02)	80.98 (±2.35)
	Iforest RKM	406 (±48)	0.41 (±0.06)	83.71 (±4.15)

Table 5.6: Results of experiments on unbalanced Fashion MNIST dataset. Experiments are replicated over 3 runs with random initializations. Means and standard deviations (enclosed in parentheses) for each metric are reported. "Minority mean" refers to the average number of samples generated from each minority mode.

5.3.5 Additional studies

Ablation study on different pre-trained classifiers In the original work of RLS-GAN [77], the fixed explicit feature map for computing RLSs is derived from the last-to-the-next layer of an Inception-v3 network. However, the impact of different pre-trained classifiers was not compared. In this Section, an ablation study over the impact of various pre-trained classifiers in RLS-RKM (class) on the unbalanced 012-MNIST dataset is conducted. Specifically, the candidate pre-trained classifiers considered in our experiment include ResNet18[81], ResNet34[81], VGG16[92], AlexNet[84], and Inception-v3[93]. As shown in Table 5.7, we empirically observe that RLS sampling with pre-trained AlexNet as explicit feature map achieves the best performance across all evaluation metrics.

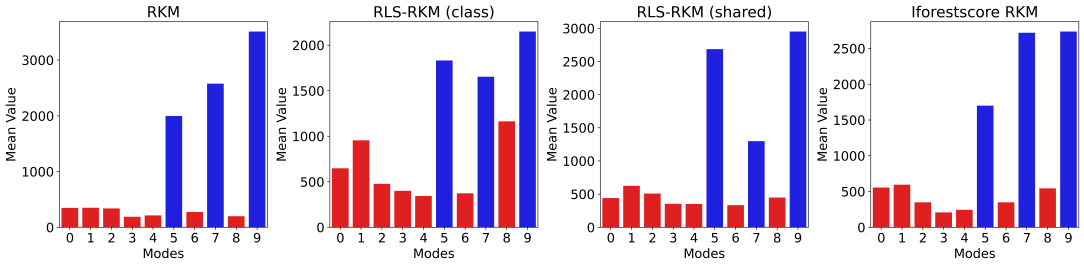

Figure 5.9: Number of generated samples per mode from different RKM models on the unbalanced Fashion MNIST dataset.

Classifier	Mode 1	Mode 2	**Mode 3**	KL score(↓)	FID(↓)
Alexnet[84]	3219 (±218)	4183 (±125)	**2283** (±299)	**0.03** (±0.02)	**52.02** (±4.32)
Inception_v3[93]	3998 (±242)	4496 (±122)	1206 (±278)	0.12 (±0.04)	60.4 (±4.4)
Resnet18[81]	3706 (±188)	4231 (±94)	1738 (±286)	0.06 (±0.02)	57.31 (±3.7)
Resnet34[81]	3328 (±59)	4945 (±177)	1486 (±231)	0.1 (±0.02)	55.34 (±3.0)
vgg16[92]	3765 (±77)	4538 (±183)	1446 (±130)	0.09 (±0.02)	59.53 (±2.11)

Table 5.7: Ablation study over the impact of different pre-trained classifiers on the performance of RLS-RKM (class) on the unbalanced 012-MNIST dataset. Minority mode is highlighted in bold, with the imbalance ratio set to 0.1. The results are averaged over 5 runs of replicated experiments.

Effect of weighted sampling on latent space in Gen-RKM It is also non-trivial to analyze the impact of the use of weighted sampling schemes during training in Gen-RKM on its corresponding latent space. The visualizations of the latent spaces obtained from the standard training process and that from the RLS sampling-adapted training process are shown in Figure 5.10. It is evident that the latent variables corresponding to the minority classes are also augmented in the latent space, and a re-balancing effect is significant. This ensures that the fitted GMM would not overlook these underrepresented groups, thereby reducing the unwanted bias towards majority modes that arise from unbalanced training data in generation. A more diverse and fair generation can thus be obtained in this spirit.

However, augmenting minority classes via oversampling techniques also has some drawbacks. Since correction for imbalance is achieved by oversampling the minority classes, the resampled training data will inevitably contain some duplicate instances. During the KPCA operation, these duplicate data will still yield redundant hidden representations in the latent space, as displayed in the right side of Figure 5.10. The negative effects are in two-fold. First, the smoothness and the continuity of the latent space are no longer guaranteed,

where interpolation between latent points could be problematic. Second, the presence of duplicate latent points might cause the GMM to be overfitted, where duplicate points are assigned higher probabilities. That would increase the risk of data-copying (i.e., directly replicating certain training samples in generation) in the generative models, which has been shown as orthogonal to the problem of mode collapse [94].

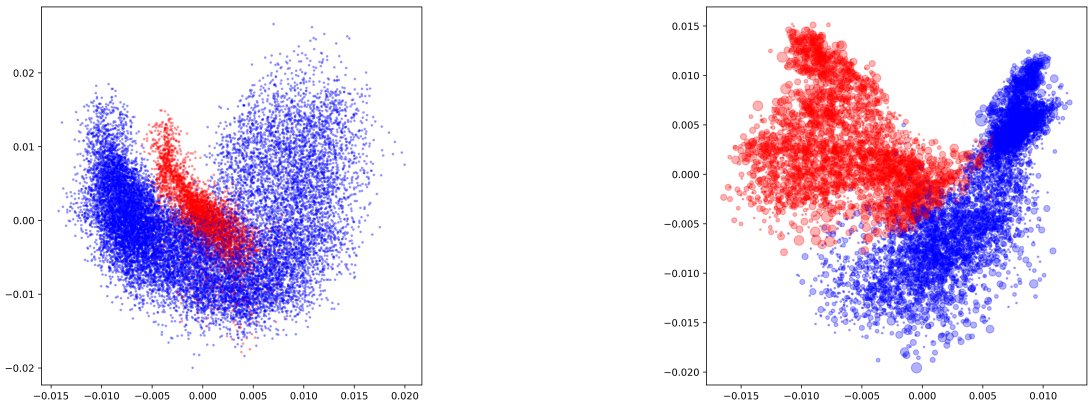

Figure 5.10: Visualizations of the latent spaces of two Gen-RKMs trained on the unbalanced Fashion MNIST dataset (imbalance ratio is 0.1). Only the first two principal components in the latent spaces are displayed. The left figure is developed based on a vanilla Gen-RKM, whereas Gen-RKM in the right figure is trained with the RLS sampling (class) manner. Latent representations of minority modes are colored as red, while those of the majority classes are in blue. Note that the size of each point in the right figure indicates the frequency of being oversampled.

Comparison to VAE and RLS-VAE A qualitative comparison is made in this section between the proposed methods and the standard VAE model. We also include the RLS sampling adapted version of VAE in this experiment to evaluate the effectiveness of RLS sampling across different model settings. The results are displayed in Table 5.8. Note that the same encoder and decoder architecture, along with the hyperparameter settings as outlined in Section 5.1.3 is employed in VAE for a fair comparison. We observe empirically that both FID scores and KL scores are better with the Gen-RKM setting. In addition, the implementation of RLS sampling with a pre-trained classifier as feature map could significantly improve the mode coverage and capturing in both the Gen-RKM and VAE models. However, the use of RLS sampling in VAE could possibly lead to a lower FID score, indicating a poorer generation quality.

Model Name	Mode 1	Mode 2	**Mode 3**	KL Score(↓)	FID(↓)
RKM	4573(±44)	5138(±90)	165(±71)	0.33(±0.02)	61.60(±1.69)
RLS-RKM	3153(±72)	4280(±40)	**2276**(±113)	**0.03**(±0.00)	**56.65**(±1.39)
RLS-VAE	3000(±78)	4955(±198)	1621(±262)	0.10(±0.03)	79.17(±2.70)
VAE	4216(±101)	5409(±154)	176(±14)	0.34(±0.01)	76.39(±1.42)

Table 5.8: Results of comparison to VAE and RLS-VAE on the unbalanced 012-MNIST dataset with imbalance ratio 0.1. RLSs are computed by explicit feature map based on a pre-trained classifier.

Timings of different sampling schemes The training times of Gen-RKM adapted with the proposed sampling schemes are reported in Table 5.9. One can observe that the implementations of both RLS sampling and Iforest sampling in Gen-RKM are more computationally expensive compared to the standard training process. Among these, RLS-RKM (shared) requires the longest training time due to the need for updating RLSs in each iteration.

	Fashion	MNIST	012-MNIST
Number of samples	22200	32467	13261
Iforest RKM	391(±18)	721(±44)	158(±2)
RKM	233(±5)	326(±4)	91(±0)
RLS-RKM (class)	418(±19)	737(±13)	156(±1)
RLS-RKM (shared)	450(±7)	1029(±3)	165(±5)

Table 5.9: Training times in seconds (averaged over 3 runs) of different weighted sampling implementations in Gen-RKM. For the unbalanced 012-MNIST, the maximum epoch number is set to 100, while the maximum epoch number in unbalanced MNIST and Fashion MNIST is 150. The batch size is uniformly set to 328 across all cases. All experiments are conducted using a single NVIDIA GeForce RTX 3080 GPU (12GB).

Significance of mini-batch sampling and final step resampling Recall that in the RLS sampling adapted Gen-RKM, two key modifications based on weighted sampling are introduced during the training phase. Firstly, each mini-batch in every iteration is sampled with replacement according to the probability determined by the normalized RLSs. Secondly, the final computation step in Gen-RKM is performed on the full dataset, which has been resampled based on these RLSs. It is interesting to investigate how the performance of RLS-RKM becomes if we remove one of the above modifications in the algorithm. To conduct this ablation study, two simplified versions of RLS-RKM are considered as follows.

- **RLS-RKM (batch):** Only mini-batch sampling is based on RLSs, while the final computation step remains unmodified (i.e., performing KPCA on the original, unbalanced dataset).
- **RLS-RKM (final):** Only the final computation step uses the resampled full dataset, while each mini-batch is uniformly sampled during the training phase, as usual.

The performance results of these two simplified versions of RLS-RKM are presented in Table 5.10. For RLS-RKM (batch), the performance shows almost no change compared to the vanilla RKM. This is not surprising, because the minority modes are not augmented in the latent space without the final resampling step, which inevitably makes it fail to generate more samples from the minority classes. As for RLS-RKM (final), a slight improvement on the mode coverage can be observed compared to vanilla RKM, but since the encoder and decoder parameters are not exactly trained on the resampled data, the quality of the final generation is actually worse than that of vanilla RKM. In conclusion, this ablation study demonstrates that the modifications on both mini-batch sampling and the final computation step are crucial to the effectiveness of the RLS-RKM algorithm.

Model Name	Mode 1	Mode 2	**Mode 3**	KL score(↓)	FID(↓)
RKM	4623(±106)	5185(±62)	85(±46)	0.36(±0.02)	64.17(±0.28)
RLS-RKM	3363(±196)	4771(±162)	**1558**(±145)	**0.09**(±0.01)	**53.22**(±2.93)
RLS-RKM (batch)	4496(±47)	5249(±67)	166(±17)	0.34(±0.01)	62.68(±2.91)
RLS-RKM (final)	3759(±156)	5111(±132)	679(±186)	0.21(±0.03)	67.11(±6.06)

Table 5.10: Ablation study over the significance of mini-batch sampling and final step resampling in RLS-RKM. This experiment is conducted on the unbalanced 012-MNIST dataset with imbalance ratio of 0.05. RLSs are computed by an explicit feature map based on a pre-trained classifier.

5.4 Summary and discussion

This experiment chapter focuses on evaluating the effectiveness of different weighted sampling schemes implemented during the training of Gen-RKM on unbalanced datasets. Specifically, we conduct different experiments tailored to both supervised and unsupervised scenarios.

For the supervised setting, we assess the applicability of Gen-RKM adapted with inverse frequency sampling on the unbalanced MNIST and Fashion MNIST datasets. We then extend its use to conditional generation in the context of unbalanced data. From the results, we can conclude that

- A notable improvement in the diversity of generation can be observed when applying inverse frequency sampling. All minority modes can be successfully captured, and the distribution of generated samples becomes more uniform and balanced.
- Unbalanced data may impact the effectiveness of conditional generation in Gen-RKM, leading to potentially blurred or incorrect results when generating samples conditioned on certain minority modes. This problem can be resolved by conditional generation based on a Gen-RKM trained with the inverse sampling manner, resulting in improved generation quality.
- The limitation of inverse frequency sampling is quite evident. Since the sampling weights are determined by the inverse of class frequencies, it requires the knowledge of labels. Consequently, the applications of inverse frequency sampling are relatively limited in practice, as label information is often scarce in real-world scenarios.

More delicate experiments are carried out under the unsupervised setting to study the effectiveness of different RLS sampling schemes. Additionally, We also include Iforest sampling as an extension approach to basic RLS sampling. The used datasets vary from synthetic datasets to real-world image datasets. According to the results, the following findings can be concluded:

- For the 2D synthetic dataset, we observe that RLS sampling is effective in preventing mode collapse and improving diversity in generation. However, for more complex datasets with a larger number of modes, such as the 2D grid, some minority modes may not be successfully captured.
- Regarding the unbalanced 012-MNIST dataset where there is only one minority mode, RLS sampling with a pre-trained classifier as the explicit feature map could significantly improve both mode coverage and generation quality under different imbalance ratios.
- A similar observation is made with the unbalanced MNIST and unbalanced Fashion MNIST datasets, where RLS-RKM (class) consistently

outperforms other methods in terms of lower KL score and FID. However, there is a higher chance for the occurrence of mode missing due to the increasing number of minority modes in the datasets.

– A addition comparison to vanilla VAE and RLS-VAE is also conducted. The results indicate that using RLS sampling with VAE leads to a slightly lower FID score, suggesting a decrease in generation quality, despite great improvement in the aspect of mode coverage. Compared to Gen-RKM, the effect of RLS sampling on VAE is less pronounced.

Additionally, some drawbacks of RLS sampling can be identified from the above experiments and a series of additional studies.

– RLS sampling is particularly dependent on the choice of feature map, which means that if the feature map fails to effectively extract information from the unbalanced data (i.e., it cannot distinguish well between minority modes and majority modes), the performance of RLS sampling could be rather poor.

– RLS sampling is rather computationally intensive, especially when using a shared feature map with Gen-RKM. Training could become significantly slower because the RLSs need to be recalculated in each iteration.

– The underlying mechanism behind RLS sampling (or possibly other weighted sampling schemes) leading to a more diverse generation in Gen-RKM is that minority modes in the latent space are also augmented via oversampling. However, this would cause the GMM to overfit, particularly on some repeated latent points, resulting in the generated samples that are very close to the true training samples. The risk of data copying in generation thus increases.

Overall, our experiments show that weighted sampling schemes like inverse frequency sampling and RLS sampling could more or less reduce the problem of mode collapse in Gen-RKM when facing unbalanced training data, leading to a more diverse and fair generation. Nevertheless, some limitations have been discovered, including the need for full label information in inverse frequency sampling, higher computational demands, the risk of data memorization, and reliance on feature maps in RLS sampling, which also suggest potential directions for future research.

Chapter 6

Conclusions

This thesis investigated various sampling techniques for training the Gen-RKM model in scenarios with unbalanced inputs. These sampling strategies aimed at increasing the diversity of the model's generation in an imbalanced input context. Depending on the availability of data labels, the experiments were carried out in both supervised and unsupervised settings.

In the supervised learning context, unbalanced MNIST and unbalanced Fashion MNIST datasets have been specifically created to train the learning models. When comparing the results of inverse frequency sampling with the standard uniform sampling in Gen-RKM, it was obvious that inverse frequency sampling facilitated a more balanced generation of data and increased the generation of samples from minority classes during the training phase. Additionally, inverse frequency sampling demonstrated the capability to selectively generate samples from minority classes when integrated with conditional generation techniques in Gen-RKM.

In the unsupervised context, RLS sampling and Iforest score sampling have been tested and discussed on 2D synthetic datasets, 012-MNIST, MNIST, and FashionMNIST datasets, with each of them having been tested with different imbalance ratios and different feature maps. The results of the experiments indicated that both RLS sampling and Iforest score sampling showed noticeable improvement compared to vanilla Gen-RKM in the diversity of model generation. RLS sampling with implicit feature maps performs well for low-dimensional data. RLS sampling, particularly when using fixed explicit feature maps such as pre-trained classifiers, delivered outstanding outcomes in high-dimensional scenarios on datasets with a limited number of modes. For datasets consisting of a large number of modes, such as those containing 10 or 25 modes, RLS sampling still outperformed the baseline and

other sampling strategies, even though the results were not as impressive as those achieved with simpler datasets.

In addition, the risk of these over-sampling-based methods has been discussed, which provided insights for numerous future studies. We summarize the suggestions for future works as follows:

- Instead of simply increasing the probability of sampling the minority, resampling the data in a more advanced way (e.g., applying differentiable augmentation for over-sampled data [95]) to avoid the risk of data copying/over-fitting in the generation phase.
- As the performance of RLS sampling is particularly dependent on the choice of feature map, finding more advanced feature extractors is intriguing.
- Given the training of Gen-RKM with RLS sampling is quite computationally expensive, especially using shared feature maps, adaptions with faster RLS approximation schemes in the RKM framework are subject to future research.
- Currently, our work is limited to Gen-RKM framework. RLS sampling (or other possible weighted sampling schemes) could also be extended to Stiefel RKM[42] in the future.

Bibliography

[1] J. M. Bernardo, M. J. Bayarri, J. O. Berger, *et al.*, "Generative or discriminative? getting the best of both worlds," *Bayesian statistics*, vol. 8, no. 3, pp. 3–24, 2007.

[2] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds., Boston, MA: Springer US, 2005, pp. 853–867, ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_40.

[3] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, ISSN: 1558-2191. DOI: 10.1109/TKDE.2008.239.

[4] K. Choi, A. Grover, T. Singh, R. Shu, and S. Ermon, "Fair generative modeling via weak supervision," presented at the International Conference on Machine Learning, PMLR, 2020, pp. 1887–1898, ISBN: 2640-3498.

[5] W. Gerych, K. Hickey, L. Buquicchio, *et al.*, "Debiasing Pretrained Generative Models by Uniformly Sampling Semantic Attributes," *Advances in Neural Information Processing Systems*, vol. 36, pp. 45 083–45 101, Dec. 15, 2023.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014.

[7] Y. Kossale, M. Airaj, and A. Darouichi, "Mode Collapse in Generative Adversarial Networks: An Overview," in *2022 8th International Conference on Optimization and Applications (ICOA)*, Oct. 2022, pp. 1–6. DOI: 10.1109/ICOA55659.2022.9934291.

[8] A. Pandey, J. Schreurs, and J. A. K. Suykens, "Generative Restricted Kernel Machines: A framework for multi-view generation and disentangled feature learning," *Neural Networks*, vol. 135, pp. 177–191, Mar. 1, 2021, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.12.010.

[9] J. Xu, H. Li, and S. Zhou, "An Overview of Deep Generative Models," *IETE Technical Review*, vol. 32, no. 2, pp. 131–139, Mar. 4, 2015, ISSN: 0256-4602. DOI: 10.1080/02564602.2014.987328.

[10] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes." arXiv: 1312.6114 [cs, stat]. (Dec. 10, 2022), [Online]. Available: http://arxiv.org/abs/1312.6114 (visited on 03/18/2024), pre-published.

[11] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer, 2000, ISBN: 978-1-4419-3160-3 978-1-4757-3264-1. DOI: 10.1007/978-1-4757-3264-1.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1, 1995, ISSN: 1573-0565. DOI: 10.1007/BF00994018.

[13] J. A. K. Suykens, *Least Squares Support Vector Machines*. River Edge, NJ: World Scientific, 2002, ISBN: 981-238-151-1.

[14] A. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, vol. 14, MIT Press, 2001.

[15] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural Networks — ICANN'97*, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, Eds., Berlin, Heidelberg: Springer, 1997, pp. 583–588, ISBN: 978-3-540-69620-9. DOI: 10.1007/BFb0020217.

[16] C. E. Rasmussen, "Gaussian Processes in Machine Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds., Berlin, Heidelberg: Springer, 2004, pp. 63–71, ISBN: 978-3-540-28650-9. DOI: 10.1007/978-3-540-28650-9_4.

[17] Y. Cho and L. Saul, "Kernel Methods for Deep Learning," in *Advances in Neural Information Processing Systems*, vol. 22, Curran Associates, Inc., 2009.

[18] X. Chen, X. Peng, J.-B. Li, and Y. Peng, "Overview of Deep Kernel Learning Based Techniques and Applications.," *J. Netw. Intell.*, vol. 1, no. 3, pp. 83–98, 2016.

[19] J. A. K. Suykens, "Deep Restricted Kernel Machines Using Conjugate Feature Duality," *Neural Computation*, vol. 29, no. 8, pp. 2123–2163, Aug. 2017, ISSN: 0899-7667. DOI: 10.1162/neco_a_00984.

[20] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, Jun. 2008, ISSN: 0090-5364, 2168-8966. DOI: 10.1214/009053607000000677.

[21] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Jun. 5, 2018, ISBN: 978-0-262-25693-3. DOI: 10.7551/mitpress/4175.001.0001.

[22] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. Springer, 2006, vol. 4.

[23] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, and A. Hooman, "An Overview of Principal Component Analysis," *Journal of Signal and Information Processing*, vol. 4, no. 3, pp. 173–175, 3 Aug. 1, 2013. DOI: 10.4236/jsip.2013.43B031.

[24] J. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, Jun. 1, 1999, ISSN: 1573-773X. DOI: 10.1023/A:1018628609742.

[25] J. A. K. Suykens, C. Alzate, and K. Pelckmans, "Primal and dual model representations in kernel-based learning," *Statistics Surveys*, vol. 4, pp. 148–183, none Jan. 2010, ISSN: 1935-7516. DOI: 10.1214/09-SS052.

[26] P. Honeine and C. Richard, "Preimage Problem in Kernel-Based Machine Learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 77–88, Mar. 2011, ISSN: 1558-0792. DOI: 10.1109/MSP.2010.939747.

[27] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and De-Noising in Feature Spaces," in *Advances in Neural Information Processing Systems*, vol. 11, MIT Press, 1998.

[28] J.-Y. Kwok and I.-H. Tsang, "The pre-image problem in kernel methods," *IEEE Transactions on Neural Networks*, vol. 15, no. 6, pp. 1517–1525, Nov. 2004, ISSN: 1941-0093. DOI: 10.1109/TNN.2004.837781.

[29] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004, ISBN: 0-521-83378-7.

[30] N. Zhang, S. Ding, J. Zhang, and Y. Xue, "An overview on Restricted Boltzmann Machines," *Neurocomputing*, vol. 275, pp. 1186–1199, Jan. 31, 2018, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.09.065.

[31] A. Fischer and C. Igel, "An Introduction to Restricted Boltzmann Machines," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo, Eds., Berlin, Heidelberg: Springer, 2012, pp. 14–36, ISBN: 978-3-642-33275-3. DOI: 10.1007/978-3-642-33275-3_2.

[32] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 1, 2006, ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527.

[33] N. Wang, J. Melchior, and L. Wiskott, "An analysis of Gaussian-binary restricted Boltzmann machines for natural images.," presented at the ESANN, Citeseer, 2012.

[34] M. Ranzato, A. Krizhevsky, and G. Hinton, "Factored 3-Way Restricted Boltzmann Machines For Modeling Natural Images," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, Mar. 31, 2010, pp. 621–628.

[35] M. Á. Carreira-Perpiñán and G. Hinton, "On Contrastive Divergence Learning," in *International Workshop on Artificial Intelligence and Statistics*, PMLR, Jan. 6, 2005, pp. 33–40.

[36] F. Tonin, P. Patrinos, and J. A. K. Suykens, "Unsupervised learning of disentangled representations in deep restricted kernel machines with orthogonality constraints," *Neural Networks*, vol. 142, pp. 661–679, Oct. 1, 2021, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2021.07.023.

[37] F. Tonin, Q. Tao, P. Patrinos, and J. A. K. Suykens, "Deep Kernel Principal Component Analysis for multi-level feature learning," *Neural Networks*, vol. 170, pp. 578–595, Feb. 1, 2024, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2023.11.045.

[38] S. Zhao, J. Song, and S. Ermon, "InfoVAE: Balancing Learning and Inference in Variational Autoencoders," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 5885–5892, 01 Jul. 17, 2019, ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33015885.

[39] J. Schreurs and J. A. Suykens, "Generative Kernel PCA.," presented at the ESANN, vol. 2018, 2018, pp. 129–134.

[40] S. Achten, A. Pandey, H. De Meulemeester, B. De Moor, and J. A. K. Suykens. "Duality in Multi-View Restricted Kernel Machines." arXiv: 2305.17251 [cs]. (Jul. 6, 2023), [Online]. Available: http://arxiv.org/abs/2305.17251 (visited on 04/04/2024), pre-published.

[41] X. Yan, S. Hu, Y. Mao, Y. Ye, and H. Yu, "Deep multi-view learning methods: A review," *Neurocomputing*, vol. 448, pp. 106–129, Aug. 11, 2021, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.03.090.

[42] A. Pandey, M. Fanuel, J. Schreurs, and J. A. K. Suykens, "Disentangled Representation Learning and Generation With Manifold Optimization," *Neural Computation*, vol. 34, no. 10, pp. 2009–2036, Sep. 12, 2022, ISSN: 0899-7667. DOI: 10.1162/neco_a_01528.

[43] H. D. Tagare, "Notes on optimization on stiefel manifolds," *Yale University, New Haven*, 2011.

[44] J. Li, L. Fuxin, and S. Todorovic. "Efficient Riemannian Optimization on the Stiefel Manifold via the Cayley Transform." arXiv: 2002.01113 [cs, stat]. (Feb. 3, 2020), [Online]. Available: http://arxiv.org/abs/2002.01113 (visited on 04/03/2024), pre-published.

[45] A. Pandey, J. Schreurs, and J. A. K. Suykens, "Robust Generative Restricted Kernel Machines Using Weighted Conjugate Feature Duality," in *Machine Learning, Optimization, and Data Science*, G. Nicosia, V. Ojha, E. La Malfa, *et al.*, Eds., Cham: Springer International Publishing, 2020, pp. 613–624, ISBN: 978-3-030-64583-0. DOI: 10.1007/978-3-030-64583-0_54.

[46] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, ISSN: 1076-9757. DOI: 10.1613/jair.953.

[47] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 878–887, ISBN: 978-3-540-31902-3.

[48] D. Dablain, B. Krawczyk, and N. V. Chawla, "Deepsmote: Fusing deep learning and smote for imbalanced data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 9, pp. 6390–6404, 2023. DOI: 10.1109/TNNLS.2021.3136503.

[49] M. A. Tahir, J. Kittler, K. Mikolajczyk, and F. Yan, "A multiple expert approach to the class imbalance problem using inverse random under sampling," in *Multiple Classifier Systems: 8th International Workshop, MCS 2009, Reykjavik, Iceland, June 10-12, 2009. Proceedings 8*, Springer, 2009, pp. 82–91.

[50] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006. DOI: 10.1109/TKDE.2006.17.

[51] R. F. Lima and A. C. M. Pereira, "A fraud detection model based on feature selection and undersampling applied to web payment systems," in *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 3, 2015, pp. 219–222. DOI: 10.1109/WI-IAT.2015.13.

[52] V. L. Berardi and G. P. Zhang, "The effect of misclassification costs on neural network classifiers," *Decision Sciences*, vol. 30, no. 3, pp. 659–682, 1999.

[53] C. Huang, Y. Li, C. C. Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[54] Y.-X. Wang, D. Ramanan, and M. Hebert, "Learning to model the tail," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[55] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-Balanced Loss Based on Effective Number of Samples," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 9260–9269. DOI: 10.1109/CVPR.2019.00949.

[56] S. Park, J. Lim, Y. Jeon, and J. Y. Choi, "Influence-balanced loss for imbalanced visual classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 735–744.

[57] Y. Liu, B. Cao, and J. Fan, "Improving the accuracy of learning example weights for imbalance classification," in *International Conference on Learning Representations*, 2022.

[58] P. C. Lane, D. Clarke, and P. Hender, "On developing robust models for favourability analysis: Model choice, feature sets and imbalanced data," *Decision Support Systems*, vol. 53, no. 4, pp. 712–718, 2012.

[59] B. Krawczyk and G. Schaefer, "An improved ensemble approach for imbalanced classification problems," in *2013 IEEE 8th international symposium on applied computational intelligence and informatics (SACI)*, IEEE, 2013, pp. 423–426.

[60] V. Raj, S. Magg, and S. Wermter, "Towards effective classification of imbalanced data with convolutional neural networks," in *Artificial Neural Networks in Pattern Recognition: 7th IAPR TC3 Workshop, ANNPR 2016, Ulm, Germany, September 28–30, 2016, Proceedings 7*, Springer, 2016, pp. 150–162.

[61] D. Wu, Z. Wang, Y. Chen, and H. Zhao, "Mixed-kernel based weighted extreme learning machine for inertial sensor based human activity recognition with imbalanced dataset," *Neurocomputing*, vol. 190, pp. 35–49, 2016.

[62] W. Xiao, J. Zhang, Y. Li, and W. Yang, "Imbalanced extreme learning machine for classification with imbalanced data distributions," in *Proceedings of ELM-2015 Volume 2: Theory, Algorithms and Applications (II)*, Springer, 2016, pp. 503–514.

[63] Y. Lu, H. Guo, and L. Feldkamp, "Robust neural learning from unbalanced data samples," in *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, IEEE, vol. 3, 1998, pp. 1816–1821.

[64] Y. Murphey, H. Guo, and L. Feldkamp, "Neural learning from unbalanced data: Special issue: Engineering intelligent systems (guest editor: László monostori)," *Applied Intelligence*, vol. 21, Sep. 2004. DOI: 10.1023/B:APIN.0000033632.42843.17.

[65] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[66] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*, IEEE, 2008, pp. 413–422.

[67] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, 1996, pp. 226–231.

[68] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[69] D. M. Tax and R. P. Duin, "Support vector domain description," *Pattern recognition letters*, vol. 20, no. 11-13, pp. 1191–1199, 1999.

[70] L. Ruff, R. A. Vandermeulen, N. Görnitz, *et al.*, "Deep one-class classification," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 4393–4402.

[71] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi. "BAGAN: Data Augmentation with Balancing GAN." arXiv: 1803.09655 [cs, stat]. (Jun. 5, 2018), [Online]. Available: http://arxiv.org/abs/1803.09655 (visited on 04/30/2024), pre-published.

[72] G. Huang and A. H. Jafari, "Enhanced balancing GAN: Minority-class image generation," *Neural Computing and Applications*, vol. 35, no. 7, pp. 5145–5154, Mar. 1, 2023, ISSN: 1433-3058. DOI: 10.1007/s00521-021-06163-8.

[73] A. Anaissi, Y. Jia, A. Braytee, M. Naji, and W. Alyassine, "Damage GAN: A Generative Model for Imbalanced Data," in *Data Science and Machine Learning*, D. Benavides-Prado, S. Erfani, P. Fournier-Viger, Y. L. Boo, and Y. S. Koh, Eds., Singapore: Springer Nature, 2024, pp. 48–61, ISBN: 978-981-9986-96-5. DOI: 10.1007/978-981-99-8696-5_4.

[74] S. Asgari Taghanaki, M. Havaei, A. Lamb, A. Sanghi, A. Danielyan, and T. Custis. "Jigsaw-VAE: Towards Balancing Features in Variational Autoencoders," arXiv e-prints. (May 1, 2020), [Online]. Available: https://ui.adsabs.harvard.edu/abs/2020arXiv200505496A (visited on 05/21/2024), pre-published.

[75] K. Sohn, H. Lee, and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015.

[76] M. Mirza and S. Osindero. "Conditional Generative Adversarial Nets." arXiv: 1411.1784 [cs, stat]. (Nov. 6, 2014), [Online]. Available: http://arxiv.org/abs/1411.1784 (visited on 07/02/2024), pre-published.

[77] J. Schreurs, H. De Meulemeester, M. Fanuel, B. De Moor, and J. A. K. Suykens, "Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks," in *Machine Learning, Optimization, and Data Science*, G. Nicosia, V. Ojha, E. La Malfa, *et al.*, Eds., Cham: Springer International Publishing, 2022, pp. 466–480, ISBN: 978-3-030-95470-3. DOI: 10.1007/978-3-030-95470-3_35.

[78] V. Vovk, "Kernel Ridge Regression," in *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, B. Schölkopf, Z. Luo, and V. Vovk, Eds., Berlin, Heidelberg: Springer, 2013, pp. 105–116, ISBN: 978-3-642-41136-6. DOI: 10.1007/978-3-642-41136-6_11.

[79] M. Fanuel, J. Schreurs, and J. Suykens, "Diversity Sampling is an Implicit Regularization for Kernel Methods," *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 1, pp. 280–297, Jan. 2021. DOI: 10.1137/20M1320031.

[80] S. McCurdy, "Ridge Regression and Provable Deterministic Ridge Leverage Score Sampling," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018.

[81] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[82] C. Zhang, H. Kjellstrom, and S. Mandt, "Determinantal Point Processes for Mini-Batch Diversification," in *CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE (UAI2017)*, Corvallis: Auai Press, 2017.

[83] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[84] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012.

[85] D. P. Woodruff, "Sketching as a Tool for Numerical Linear Algebra," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, Oct. 28, 2014, ISSN: 1551-305X, 1551-3068. DOI: 10.1561/0400000060.

[86] L. McInnes, J. Healy, and J. Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." arXiv: 1802.03426 [cs, stat]. (Sep. 17, 2020), [Online]. Available: http://arxiv.org/abs/1802.03426 (visited on 07/31/2024), pre-published.

[87] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, ISSN: 1558-2256. DOI: 10.1109/5.726791.

[88] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms." arXiv: 1708.07747 [cs, stat]. (Sep. 15, 2017), [Online]. Available: http://arxiv.org/abs/1708.07747 (visited on 08/06/2024), pre-published.

[89] N. Yu, K. Li, P. Zhou, J. Malik, L. Davis, and M. Fritz, "Inclusive GAN: Improving Data and Minority Coverage in Generative Models," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 377–393, ISBN: 978-3-030-58542-6. DOI: 10.1007/978-3-030-58542-6_23.

[90] H. De Meulemeester, J. Schreurs, M. Fanuel, B. De Moor, and J. A. K. Suykens, "The Bures Metric for Generative Adversarial Networks," in *Machine Learning and Knowledge Discovery in Databases. Research Track*, N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read, and J. A. Lozano, Eds., Cham: Springer International Publishing, 2021, pp. 52–66, ISBN: 978-3-030-86520-7. DOI: 10.1007/978-3-030-86520-7_4.

[91] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[92] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." arXiv: 1409.1556 [cs]. (Apr. 10, 2015), [Online]. Available: http://arxiv.org/abs/1409.1556 (visited on 08/09/2024), pre-published.

[93] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[94] R. Bhattacharjee, S. Dasgupta, and K. Chaudhuri, "Data-copying in generative models: A formal framework," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23, vol. 202, Honolulu, Hawaii, USA: JMLR.org, Jul. 23, 2023, pp. 2364–2396.

[95] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han, "Differentiable Augmentation for Data-Efficient GAN Training," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 7559–7570.

Appendices

Appendix A

Supplementary material for Chapter 4

A.1 Derivation of the conditional distribution of GMM

Given that the latent variable $\boldsymbol{h}$ and its corresponding label $\boldsymbol{y}$ (in the form of one-hot encoding). We assume that their joint PDF is modeled by a mixture of multivariate Gaussians:

$$p(\boldsymbol{h}, \boldsymbol{y}) = \sum_{k=1}^{l} \pi_k \mathcal{N}(\boldsymbol{h}_{cat} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{A.1}$$

where $\boldsymbol{h}_{cat}$ is the concatenated vector of $\boldsymbol{h}$ and $\boldsymbol{y}$. Recall that the conditionals or the marginals of a multivariate Gaussian results in another Gaussian distribution. More specifically, for a multivariate Gaussian distribution $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{u}, \boldsymbol{\Sigma})$ with the following partition

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}, \tag{A.2}$$

the conditional distribution is given by

$$p(\boldsymbol{x}_a | \boldsymbol{x}_b) = \mathcal{N}(\boldsymbol{x}_a | \boldsymbol{\mu}_{a|b}, \boldsymbol{\Sigma}_{a|b}), \tag{A.3}$$

where

$$\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1} (\boldsymbol{x}_b - \boldsymbol{\mu}_b) \\ \boldsymbol{\Sigma}_{a|b} &= \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1} \boldsymbol{\Sigma}_{ba}. \end{aligned} \tag{A.4}$$

As for the marginal, it is simply given by $p(\boldsymbol{x}_a) = \mathcal{N}(\boldsymbol{x}_a | \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa})$.

Back to the setting of GMM, since conditional of each Gaussian is another Gaussian, the conditional distribution of $\boldsymbol{h}$ given a certain class label is then

characterized by another mixture of Gaussians:

$$p(\boldsymbol{h}|\boldsymbol{y}) = \sum_{k=1}^{l} \pi'_k \mathcal{N}(\boldsymbol{h}|\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{h}|\boldsymbol{y},k}). \tag{A.5}$$

To compute the updated weights π'_k for each component Gaussian, one needs to apply Bayes' rule:

$$\begin{aligned} p(\boldsymbol{h}|\boldsymbol{y}) &= \frac{p(\boldsymbol{h},\boldsymbol{y})}{p(\boldsymbol{y})} \\ &= \sum_{k=1}^{l} \frac{\pi_k \mathcal{N}(\boldsymbol{h}_{cat}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{p(\boldsymbol{y})} \\ &= \sum_{k=1}^{l} \frac{\pi_k \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_{c,k}, \boldsymbol{\Sigma}_{\boldsymbol{y}\boldsymbol{y},k}) \mathcal{N}(\boldsymbol{h}|\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{h}|\boldsymbol{y},k})}{p(\boldsymbol{y})} \\ &= \sum_{k=1}^{l} \underbrace{\frac{\pi_k \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_{\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{y}\boldsymbol{y},k})}{\sum_{v=1}^{l} \pi_v \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_{\boldsymbol{y},v}, \boldsymbol{\Sigma}_{\boldsymbol{y}\boldsymbol{y},v})}}_{=\pi'_k} \mathcal{N}(\boldsymbol{h}|\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{y},k}, \boldsymbol{\Sigma}_{\boldsymbol{h}|\boldsymbol{y},k}). \end{aligned} \tag{A.6}$$

Appendix B

Additional results of Chapter 5

B.1 Generated Samples: Comparison Between Vanilla Gen-RKM and Gen-RKM with RLS Sampling Adaptation

B.1.1 Unbalanced 012-MNIST

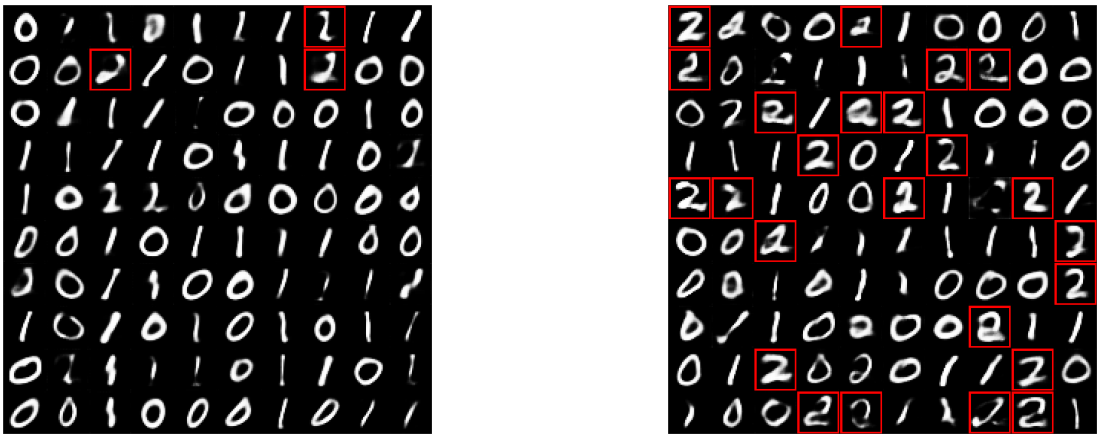

Figure B.1: Generated images from the unbalanced 012-MNIST dataset by vanilla RKM (left) and RLS (class) sampling adapted RKM (right). Images are highlighted by a red border if classified as minority modes.

B.1.2 Unbalanced MNIST

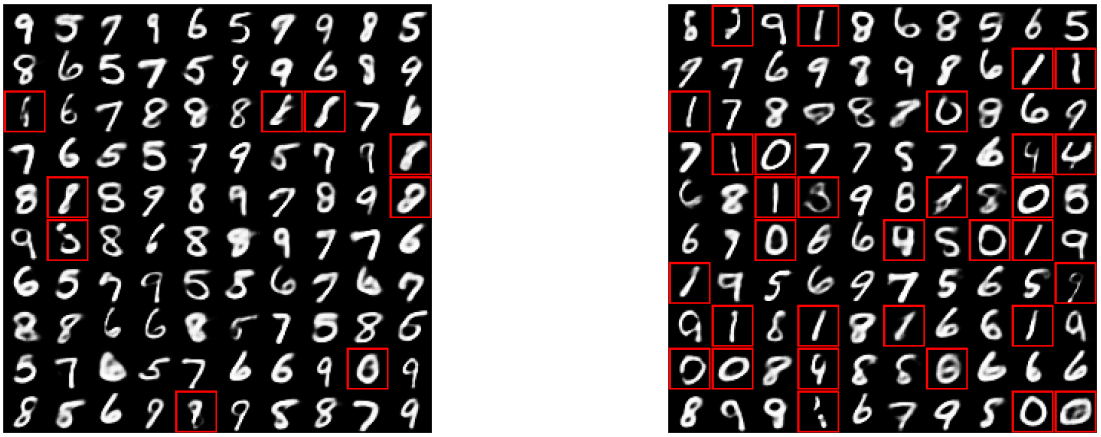

Figure B.2: Generated images from the unbalanced MNIST dataset by vanilla RKM (left) and RLS (class) sampling adapted RKM (right). Images are highlighted by a red border if classified as minority modes.

B.1.3 Unbalanced Fashion MNIST

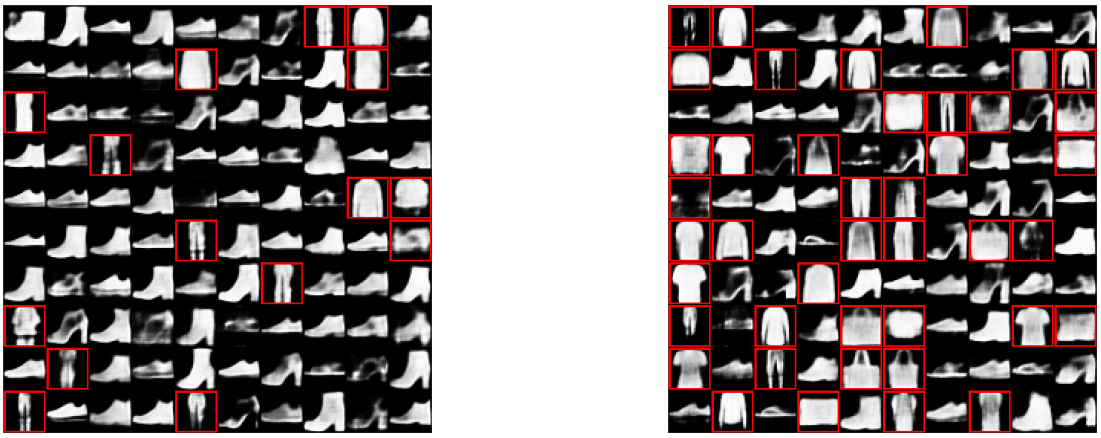

Figure B.3: Generated images from the unbalanced Fashion MNIST dataset by vanilla RKM (left) and RLS (class) sampling adapted RKM (right). Images are highlighted by a red border if classified as minority modes.

AFDELING
Straat nr bus 0000
3000 LEUVEN, BELGIË
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
www.kuleuven.be