Advancements in Memory-Efficient and Variance-Optimized Pairwise Learning, Extended to Nonlinear Modeling and Spiking Neural Networks

Advancements in Memory-Efficient and Variance-Optimized Pairwise Learning, Extended to Nonlinear Modeling and Spiking Neural Networks

by

Hilal AlQuabeh

Literature of thesis submitted to the
Deanship of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Machine Learning

Deprtement of Machine Learning
Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)

Examining Committee Membership

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

Supervisor:	Dr. Bin Gu Assistant Professor, Department of Machine Learning Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
External Examiner:	Dr. Boyu Wang Assistant Professor, Department of Computer Science Western University
Internal Member:	Dr. Karthik Nandakumar Associate Professor, Department of Computer Vision Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
Internal Member:	Dr. Martin Takac Associate Professor and Deputy Department Chair Department of Machine Learning Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
Internal Member:	Dr. Zhiqiang Xu Assistant Professor, Department of Machine Learning Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)

Author's Declaration

I hereby declare that I am the author of this thesis, with chapter 2 based on the paper [6], chapter 3 derived from the paper [4], chapter 4 based on the submitted work titled "Manifold Metric Learning with Spiking Neural Networks", authored by Hilal AlQuabeh, Velibor Bojkovic, Dilshod Azizov, and Bin Gu, and chapter 5 adapted from the paper [71].

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents an exploration of online pairwise learning, extending the traditional boundaries through innovative algorithms and the application of Spiking Neural Networks (SNNs) for deep metric learning. The initial chapters introduce and refine advanced online gradient descent (OGD) algorithms tailored for pairwise learning in non-*i.i.d.* environments, showing their scalability and efficiency in handling quadratic growth in computational complexity and extending their applicability to kernelized models. Our proposed OGD methods, characterized by sub-linear regret and reduced complexity, demonstrate marked superiority over existing kernel and linear models across various datasets.

Building upon these foundations, the exploration continues into the domain of deep metric learning with SNNs, with attention into the robustness of rate encoding in SNNs, a critical aspect that directly enhances pairwise learning strategies. By bridging the conceptual gap between SNN robustness and pairwise learning, this work illuminates the multifaceted interplay between encoding stability, metric learning effectiveness, and the overarching robustness of learning systems.

In a complementary exploration, proposing a novel approach that takes advantage of the temporal dynamics of binary encoding to achieve high-dimensional data representation with low latency and superior accuracy. This methodology illustrates the potential of temporal dimension utilization within the SNN framework for enhanced discriminative capabilities in the metric space.

Overall, this thesis not only advances the state of the art in online pairwise learning and deep metric learning with SNNs, but also opens new avenues for research by highlighting the importance of robustness in rate encoding and its implications for pairwise learning methodologies. Through a blend of theoretical innovation and empirical validation, we showcase the synergy between these seemingly disparate domains, paving the way for more resilient and accurate machine learning models in complex environments.

Acknowledgements

I express my sincere appreciation to my beloved family members, whose unwavering support and understanding have been instrumental in enabling me to embark on this journey. I extend my heartfelt gratitude to my parents for instilling within me the values of perseverance and determination, laying the foundation for my achievements. Additionally, I am immensely grateful to my wife for her unwavering support and understanding throughout this endeavor. My heartfelt thanks also go to my cherished children, Ebrahim and Harib, whose patience and encouragement have been a constant source of motivation. Lastly, I extend my appreciation to my brothers Odai, Siraj, Batool, Hamza, and Ahmad for their unwavering support and encouragement.

Special thanks to my supervisor, Dr. Bin Gu, for his invaluable guidance and mentorship. I am grateful to my colleagues Bhaskar, Dilshod, Velibor, and Srinivas for their insightful discussions and unwavering support.

This achievement is a culmination of the collective efforts of many, and for that, I am truly thankful.

Dedication

To my beloved children, Ebrahim and Harib.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Pairwise objective paradigms are a very important and essential part of machine learning, for example, differential network in face recognition [51, 84], metric learning [56], bipartite learning [99], multiple kernel learning [101], and maximization of the area under the curve (AUC) [100], are examples of machine learning techniques that make use of pairwise objective functions.

The most difficult aspect of pairwise learning is its scalability, since the sample size expands quadratically with the number of samples, in contrast to pointwise learning. In offline batch training, researchers have proposed classic pair (or double) stochastic gradient descent algorithms *(e.g., PSGD [33])* that at every iteration samples one or more pairs of examples, and update the model based on the stochastic gradient of these losses only. Additional alternatives to SGD include stagewise training [33], where the sample size is gradually extended so that the total number of iterations sufficient to have a statistically optimal solution is minimized at every stage. The generalization of SGD like algorithms is well studied in the literature (e.g., see [1]).

On the other hand, in many cases, online learning is favored since it enables researchers to avoid the challenges of scalability and data availability, in addition to the disadvantages of offline learning. However, in an online setting, the requirement to match the received example with all previous examples poses a challenge in terms of memory and computational complexity. In AUC maximization, for instance, employing sampling techniques like "reservoir sampling" helps circumvent the complexity associated with maintaining online previous pairs. Similarly, an analogous formulation of the AUC convex square loss has been developed, addressing it through the saddle point problem and subsequently minimizing it using stochastic gradient descent [100, 97]. Moreover, it has been shown that utilizing the statistical average and covariance of the data to update the gradient in $O(Td^2)$ [36] is impractical for high-dimensional data in the context of online learning.

Further exploration is needed for general pairwise learning, particularly in the realm of nonlinear online pairwise learning, which has received relatively little research attention.

1.2 Background

1.2.1 Offline Pairwise Learning

In general, a pairwise function is sought $f : \mathcal{X}^2 \mapsto \mathcal{Y}$ where $\mathcal{X} \subset \mathbb{R}^d$ is the feature space and $\mathcal{Y} \subset \mathbb{R}$ is the label space. Furthermore, we suppose that f is a member of the reproducing kernel Hilbert space (RKHS) denoted by $\mathcal{H}$. Let the set of data pairs be defined as $\{z_i = (\mathbf{x}_i, y_i) \in \mathcal{Z} := \mathcal{X} \times \mathcal{Y} | i = 1, \ldots, T\}$ where T is the number of examples. If there exist an algorithm that is capable of learning from pairs, then the performance of the algorithm can be evaluated using a pairwise loss $\ell : \mathcal{H} \times \mathcal{Z}^2 \mapsto R_+$, that estimates the true expected risk (error) of any function, assuming that the data come from an unknown distribution ρ in $\mathcal{Z}$, as follow,

$$\mathcal{E}(f) := \int \int_{Z^2} \ell(f, z, z') d\rho(z) d\rho(z'). \tag{1.1}$$

Since the pairs are constructed by intersections of examples, they could not be *i.i.d.* and uniform convergence analysis do not apply. Similarly, the empirical risk is used to approximate the true risk, with added regularization as follows:

$$\mathcal{E}_T(f) = \frac{1}{T(T-1)} \sum_{i}^{T} \sum_{j \neq i}^{T} \ell(f, z_i, z_j). \tag{1.2}$$

1.2.2 Online Convex Optimization

In offline learning, algorithms process training samples collectively, aiming to generate a hypothesis that minimizes the expected error on a static dataset. Conversely, online learning tackles evolving data by sequentially exploiting large datasets without relying on probabilistic assumptions. The objective shifts from minimizing error to competing against the best agent and minimizing regret.

The evolution of online learning traces back to pivotal developments. From the concept of learning from expert advice to advancements like the Weighted Majority Algorithm and Follow the Regularized Leader, the field witnessed a gradual expansion. However, the seminal development came with online convex optimization (OCO), which introduced a framework where learners sequentially make decisions within a convex decision space, adapting to evolving data streams. This paradigm shift aimed to minimize cumulative loss over time, achieving low regret compared to the best fixed decision in hindsight. In the following, we define first-order conditions such as convexity, Lipschitz continuity, and smoothness, then we define the online learning protocol for pairwise losses.

Definition 1 (Convexity) *A function $\ell : \mathbb{R}^d \to \mathbb{R}$ is said to be convex if, for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$, it satisfies the inequality:*

$$\ell(\lambda x + (1 - \lambda) y) \leq \lambda \ell(x) + (1 - \lambda) \ell(y), \tag{1.3}$$

furthermore, if ℓ is continuously differentiable, then it is convex if and only if it's lower bounded by a linear hyperplane, i.e., for all $x, y \in \mathbb{R}^d$, we have $\ell(y) \geq \ell(x) + \nabla \ell(x)^T(y-x)$.

Convexity ensures that the optimization problem has a single global minimum, making it easier to find the optimal solution. It also guarantees that any local minimum found is also a global minimum, simplifying the optimization process.

Definition 2 (Lipschitz Continuity) *Let $\ell : \mathbb{R}^d \to \mathbb{R}$ be a function. ℓ is said to be Lipschitz continuous with Lipschitz constant G if there exists a constant $G \geq 0$ such that for all $x, y \in \mathbb{R}^d$, the following inequality holds:*

$$|\ell(x) - \ell(y)| \leq G\|x - y\|, \tag{1.4}$$

where $\|\cdot\|$ denotes the Euclidean norm.

Lipschitz continuity bounds the rate of change of the function, which helps in designing efficient optimization algorithms. By restricting how fast the function can change, it provides stability and convergence guarantees for iterative optimization methods, such as gradient descent.

Definition 3 (M-Smoothness) *Let $\ell : \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable function. ℓ is said to be M-smooth if its gradient $\nabla \ell(x)$ is Lipschitz continuous with Lipschitz constant M, i.e., there exists a constant $M \geq 0$ such that for all $x, y \in \mathbb{R}^d$, the following inequality holds:*

$$\|\nabla \ell(x) - \nabla \ell(y)\| \leq M\|x - y\|, \tag{1.5}$$

where $\|\cdot\|$ denotes the Euclidean norm.

Smoothness, characterized by bounded gradients, further enhances optimization efficiency. It allows for faster convergence rates of iterative algorithms, as the gradients provide reliable information about the direction of steepest descent.

Definition 4 *The pairwise Online Convex Optimization Protocol perform the following at each time step $t = 1, 2, \ldots, T$:*

1. *Choose a decision f_t from a convex decision set $\mathcal{A}$.*
2. *Simultaneously, an adversary (or environment) selects an example $z_t \in \mathcal{Z}$.*
3. *Suffer a loss $\sum_{i=1}^{t-1} \ell(f_t, z_t, z_i)$ where $\ell_t : \mathcal{Z}^2 \to \mathbb{R}_+$ is a convex function.*
4. *Observe z_t.*

The goal of online learning is to minimize the discrepancy between its cumulative loss and the loss incurred by the best decision (expert) in the set $\mathcal{A}$. In essence, OCO aims to achieve sublinear regret.

Definition 5 *The regret of an online algorithm is defined as the difference between the cumulative loss over T steps and the loss obtained by the best decision:*

$$\mathcal{R}_T = \sum_{t=1}^{T} \ell(f_t, z_t) - \inf_{f \in \mathcal{A}} \sum_{t=1}^{T} \ell(f, z_t). \tag{1.6}$$

This regret quantifies how much worse off the online algorithm performs compared to the best decision in hindsight, capturing the effectiveness of the algorithm in minimizing loss over time.

1.2.3 Online Gradient Descent

Gradient descent is a very popular approach for a convex and differentiable loss function (e.g., the empirical error minimization ERM). However, in the case of pairwise, the need to compute the gradient of all pairs is very expensive with time complexity $O(T^3)$. On the other hand, online gradient descent in which the algorithm has only sequential access to the data can offer significant savings in terms of computational complexity (in addition to other benefits of online learning). In online gradient descent with pairwise losses, and after receiving an example z_t, where t is the time step, a local error is computed on the new data point paired with all preceding $t-1$ examples and gradient step with step size of η_t, is performed as follows:

$$f_t \leftarrow f_{t-1} - \eta_t \underbrace{\frac{1}{(t-1)} \sum_{i}^{t-1} \nabla \ell(f_{t-1}, z_t, z_i)}_{L_t(f_{t-1})}.$$

Although the online gradient descent has time complexity $O(T^2)$, it is still not practical for large-scale problems. Consequently, a natural way is to have a buffer of size $s < t-1$ in which only a subset of the received examples are kept, consequently reducing the time and memory complexities to $O(Ts)$, and $O(s)$, respectively. The generalization error of the latter approach is studied in the literature (see, e.g., [100] and[52]) and proven to be $O(\frac{1}{s} + \frac{1}{T})$, i.e. the generalization error is only optimal (e.g., $\mathcal{O}(1/\sqrt{T})$) for buffer with at least $s = O(\log T)$.

Recently the work in [94] stated a significant improvement where the author proposed an online pairwise algorithm with optimal generalization with $s = 1$. However, the framework considers only the linear space, and more importantly, the convergence analysis is applied on an offline algorithm which upon sampling a new data point from *i.i.d.* examples, pairs it with the previous sampled example, leaving zero randomness in the buffer. However, the online algorithm regret minimization for a general non-*i.i.d.* setting is not provided explicitly.

Other work considers the non-linearity of real-world data (e.g. see [43]). The burden of kernel machine in pointwise learning is the need to keep the $O(T^2)$ Gramm matrix, and further extra $O(T)$ computations every iteration in training or inference. This extra cost is significantly greater when pairwise learning is included. Alternatively, researchers approximated the Gramm matrix using a matrix of low rank [22].

However, in online setting where the data are not available beforehand, the Nystrom low-rank approximation is not possible without the need to dynamically update the support vectors. The update procedure adds to the algorithm's complexity; instead, a direct random Fourier feature approximation of the kernel function can be used to reduce the complexity of the kennel machine [76]. The minimal number of random Fourier features required for a successful approximation is unclear; nevertheless, in point-wise situations, [68] has resulted the necessity to have $O(T)$, which means the same complexity as the original kernel machine. However, under extra regularity constraints on the kernel used, the number of random features is reduced to $O(\log(T)^2)$ in [7]. Furthermore, recently in [64] the number of random samples was found to be $O(\sqrt{T}\log(T))$ with an importance sampling function. In spite of this, for the pairwise kernel, random Fourier characteristics require additional analysis. In the following chapter, the reader will be taken on an in-depth exploration of the literature approaches to handle the pairwise difficulties that were described earlier.

1.3 Online Pairwise Learning with Kernels

Assume that the kernel associated with $\mathcal{H}$ is assumed to be Mercer, i.e. continuous and positive definite function $k : \mathcal{X}^2 \times \mathcal{X}^2 \mapsto \mathbb{R}^+$ (the shorthand $k_{(\mathbf{x},\mathbf{x}')}(\cdot) := k((\mathbf{x},\mathbf{x}'),(\cdot,\cdot))$ will be used throughout this thesis). Accordingly, the space $\mathcal{H}$ satisfies the following two properties: (i) Reporducding property, i.e., $\langle k_{(\mathbf{x},\mathbf{x}')}, g\rangle = g(\mathbf{x},\mathbf{x}')$ for any $\mathbf{x},\mathbf{x}' \in \mathcal{X}^2$ and $\forall g \in \mathcal{H}$. (ii) The space has all linear combinations of the functions mapping $\{k_{(\mathbf{x},\mathbf{x}')} | (\mathbf{x},\mathbf{x}') \in \mathcal{X}^2\}$ and their completion topology. Pairwise kernels can be constructed by conventional kernels, i.e., given any uni-variate kernel G, then for any $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_1', \mathbf{x}_2' \in \mathcal{X}$ one pairwise kernel is defined as follows:

$$\begin{aligned} &k(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_1',\mathbf{x}_2') = G(\mathbf{x}_1,\mathbf{x}_1') + G(\mathbf{x}_2,\mathbf{x}_2') \\ &- G(\mathbf{x}_1,\mathbf{x}_2') - G(\mathbf{x}_2,\mathbf{x}_1') = \langle G_{x_1} - G_{x_2}, G_{x_1'} - G_{x_2'}\rangle_G. \end{aligned} \tag{1.7}$$

It is clear that the pairwise kernel k defined above is positive semi-definite on $\mathcal{X}^2 \times \mathcal{X}^2$, and therefore it's Mercel kernel on $\mathcal{X} \times \mathcal{X}$ if G does on $\mathcal{X}$ (e.g., see [98]). The kernel approach enables the model to learn the non-linearity of the data without the requirement to actually lift the data to the RKHS $\mathcal{H}$. However, the complexity of the kernel that is added during training and predictions is not trivial. For example, the complexity of the kernel in an online setting with T instances is $O(T^2 s)$ where s is the buffer size. The kernel function approximation by lower dimensional mapping and the kernel matrix (Gram matrix) approximation by budget maintenance are the two primary methods by which researchers reduce the complexity of kernel machines. In the first approach, a lower dimensional mapping is introduced, i.e., $z : \mathbb{R}^d \mapsto R^D$ for special kernel (Mercer) that is capable of approximating the kernel function directly, where d is the input dimension and D is the new space; note that $D << d^2$, i.e.,

$$\mathcal{G}_x(\cdot) \approx z(\mathbf{x})^T z(\cdot). \tag{1.8}$$

In this sense, the model $f : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{Y}$ can be rewritten using the Representer theorem as follows:

$$\begin{aligned} f_\lambda(\bar{\mathbf{x}}, \hat{\mathbf{x}}) &= \sum_{i=1, j\neq i}^{T} a_{i,j} k_{(\mathbf{x}_i, \mathbf{x}_j)}(\bar{\mathbf{x}}, \hat{\mathbf{x}}) = \sum_{i=1, j\neq i}^{T} a_{i,j} \langle G_{x_i} - G_{x_j}, G_{\bar{x}} - G_{\hat{x}} \rangle_G \\ &= \sum_{i=1, j\neq i}^{T} a_{i,j} \langle z(\mathbf{x}_i) - z(\mathbf{x}_j), z(\bar{\mathbf{x}}) - z(\hat{\mathbf{x}}) \rangle = \langle \mathbf{w}, z(\bar{\mathbf{x}}) - z(\hat{\mathbf{x}}) \rangle, \end{aligned} \tag{1.9}$$

with $\mathbf{w} := [z(\mathbf{x}_1) - z(\mathbf{x}_2), \ldots, z(\mathbf{x}_T) - z(\mathbf{x}_{T-1})]a_T \in \bar{\mathcal{H}} := \mathbb{R}^D$ is the model to learn, and $a_T := [a_{1,1}, \ldots, a_{T,T-1}]$ are the coefficients of the Representer theorem. The quality of the approximation of the pointwise kernel G by random Fourier features is studied in the literature (see [76],[7],[64]), however, the approximation of the pairwise kernel k needs further analysis.

The kernel function G is assumed shift invariant and positive definite, thus using Bochner's theorem the kernel can be represented by the inverse Fourier transform of a non-negative measure $p(u)$ as follows:

$$\mathcal{G}(\mathbf{x}, \mathbf{x}') = \mathcal{G}(\mathbf{x} - \mathbf{x}') = \int p(\mathbf{u}) e^{i\mathbf{u}^T(\mathbf{x}-\mathbf{x}')} d\mathbf{u}, \tag{1.10}$$

where $i = \sqrt{-1}$. For example, if the kernel is the Gaussian kernel, i.e., $\mathcal{G}(\mathbf{x} - \mathbf{x}') = \frac{1}{\sqrt{2\pi d}} \exp -(\mathbf{x} - \mathbf{x}')^T[diag(\sigma)](\mathbf{x} - \mathbf{x}')$, we have $p(\mathbf{u})$ from the Fourier transform:

$$p(\mathbf{u}) \propto \mathcal{N}(\mathbf{0}, diag(\sigma)), \tag{1.11}$$

where $\sigma \in \mathbb{R}^d$ is the kernel width. Thus, the kernel G can be given by a Monte Carlo estimate as follows:

$$\hat{\mathcal{G}}(\mathbf{x}, \mathbf{x}') = \frac{2}{D} \sum_{i=1}^{D/2} \cos(\mathbf{u}_i^T(\mathbf{x} - \mathbf{x}')) = \langle z(\mathbf{x}), z(\mathbf{x}') \rangle, \tag{1.12}$$

Where $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, diag(\sigma))$, and $z(\mathbf{x}) := \frac{1}{\sqrt{D/2}}[cos(\mathbf{u}_i^T\mathbf{x}), sin(\mathbf{u}_i^T, \mathbf{x})]_{i=1}^{D/2}$. The quality of the estimation is given by [76] to depend on the number of random features D used to approximate the Fourier transform. Claim 1 [76] provides the bound of the approximation using random Fourier features, i.e. given $x, x' \in \mathcal{X}$, and kernel G defined in $\mathcal{X} \times \mathcal{X}$, we have the following bound on the error of the approximation $\hat{G}$ using D random Fourier features,

$$\mathbb{P}\left(|\hat{\mathcal{G}}(\mathbf{x}, \mathbf{x}') - \mathcal{G}(\mathbf{x}, \mathbf{x}')| \geq \epsilon\right) \leq 2\exp\left(\frac{-D\epsilon^2}{2}\right). \tag{1.13}$$

In pairwise learning, the research on online large-scale kernel machine is very limited. A recent study by [18] considers the maximization of AUC in the Hilbert space but in an offline setting. The research by [68] considered online kernel learning with random Fourier feature approximation, however, convergence requires $O(T)$ random features.

To overcome these limitations, we target large-scale pairwise learning in an online setting and nonlinear models. The nonlinear models are assumed to be lifted to Hilbert space in which the data are linearly separable. The kernel associated with the Hilbert space is assumed to be Mercer, and thus random Fourier features can be implemented to approximate it. The pairwise loss is computed using one pair that is random, and being updated every iteration by the online stream and 1-point buffer, i.e., the memory is limited to unity.

Although online gradient descent has a time complexity $O(T^2)$ [10, 52, 28], it is still not practical for large-scale problems. As a pairwise example, consider an AUC maximization framework where we receive a data point $\mathbf{x}_t, y_t = 1$ at any t, then we pair the point with all previous $\{\mathbf{x}_{t'}, y_{t'} = -1 | 1 < t' < t-1\}$ to find the true loss. However, the main challenge of this approach is the need to calculate the gradients $\nabla L_t(f_{t-1})$ for all training examples received, which increase linearly in t.

1.4 Online Pairwise Learning with Spiking Neural Networks

Spiking Neural Networks (SNNs) represent the third generation of neural network models, distinguishing themselves by mimicking the way biological neural networks operate more closely than their predecessors. Unlike traditional artificial neural networks that process information using continuous values, SNNs incorporate the concept of time into their operating model [29]. Neurons in an SNN communicate by sending discrete events or spikes only when a specific membrane potential is reached, a process that is more akin to the biological phenomena observed in the human brain. This fundamental difference allows SNNs to potentially consume less power and process information more efficiently, which makes them particularly appealing for neuromorphic computing and energy-efficient hardware implementations [54].

Recent advances in the field of SNNs have been driven by improvements in both the theoretical understanding of their dynamics and the practical applications of this knowledge. From a theoretical point of view, significant progress has been made in learning algorithms for SNNs, including spike-timing-dependent plasticity (STDP) and backpropagation methods adapted for event-based learning. These advances have improved the efficiency and accuracy of the training of SNN models, allowing them to tackle more complex tasks. On the practical side, the development of specialized neuromorphic hardware, such as IBM's TrueNorth and Intel's Loihi, has provided a physical platform for implementing SNNs more efficiently. These technologies aim to replicate neural processing more directly, offering pathways to significant reductions in power consumption and increases in computational speed for certain tasks.

Moreover, SNNs have seen increased application in areas where temporal dynamic processing is crucial, such as real-time decision-making systems, autonomous robotics, and dynamic pattern recognition. The ability of SNNs to handle spatio-temporal data makes them uniquely suited for these domains. As research continues, the integration of SNNs with advanced materials science, quantum computing, and sophisticated learning algorithms promises to unlock new computational paradigms [54]. These advances not only pave the way for more powerful and efficient computing architectures but also deepen our understanding of the neural mechanisms underlying human cognition.

1.4.1 Backpropagation in Spiking Neural Networks

Backpropagation through a Leaky Integrate-and-Fire (LIF) neuron, and the broader concept of Backpropagation Through Time (BPTT) for SNNs [13], involve adapting traditional neural network training techniques to the temporal dynamics and binary output nature of SNNs. Let us break down these concepts for a clearer understanding.

The LIF neuron is a simplified model of a biological neuron that integrates incoming signals until a threshold is reached, at which point it fires (sends a spike) and then resets its state. The leaky part refers to the gradual decay of the neuron's accumulated potential over time, mimicking the loss of membrane potential in a biological neuron. The LIF neuron model can be described by the following differential equation:

$$\tau_m \frac{dV(t)}{dt} = -V(t) + RI(t), \tag{1.14}$$

where $V(t)$ is the membrane potential at time t, τ_m is the membrane time constant, R is the membrane resistance, and $I(t)$ is the input current.

Backpropagation through a LIF neuron involves calculating gradients of the loss function with respect to the parameters of the neuron (e.g., weights) despite the non-differentiable spiking mechanism. To enable gradient-based optimization, surrogate gradient methods are often employed [16]. These methods approximate the gradient of the spike function (which is ideally a Dirac delta function or a step function) with a continuous and differentiable function.

BPTT is a method used to train recurrent neural networks (RNNs) and is adapted for SNNs to handle their temporal dynamics [9]. Since SNNs process inputs and generate outputs over time, it is necessary to consider the entire sequence of inputs, internal states, and outputs when computing gradients. BPTT unfolds the network in time, treating each time step as a separate layer in a deep network, and then applies the backpropagation algorithm to compute gradients across these time-unfolded layers.

When applying BPTT to SNNs with LIF neurons, the key steps involve:

- **Forward Pass:** For each time step, the membrane potential and output spikes of the LIF neurons are computed based on the current inputs and the state from the previous time step.
- **Loss Calculation:** the loss function is computed based on the desired output and the actual spikes generated by the network over the sequence.

- **Backward Pass with Surrogate Gradients:** surrogate gradients are used to approximate the gradient of the spiking mechanism and backpropagate the error through the unfolded network to compute gradients with respect to the weights at each time step.

- **Weight Update:** Aggregated gradients across time steps update weights using a gradient descent algorithm.

This process enables the training of SNNs, including those with LIF neurons, using temporal sequences of data and leveraging the network's dynamics for tasks that require processing time-varying inputs. The use of surrogate gradients is a key innovation that allows the integration of spiking mechanisms with the backpropagation algorithm, making it feasible to train SNNs with gradient-based optimization methods [74].

1.4.2 Robustness in Neural Networks

Robustness in neural networks refers to the ability of a model to maintain its performance and provide reliable predictions in the face of input data variations, adversarial examples, or when operating in environments different from those in which it was trained. The concept of robustness is paramount in the deployment of neural networks in real-world applications, where data can be noisy, incomplete, or intentionally perturbed to deceive the model. As neural networks have become more prevalent in critical decision-making areas such as autonomous driving, medical diagnosis, and financial services, the robustness of these models has become a focal point of research and development.

The primary challenges to robustness in neural networks stem from their inherent sensitivity to changes in input data. Small and often imperceptible alterations in the input data can lead to drastically different predictions. This sensitivity is exploited in adversarial attacks, where carefully crafted perturbations cause the model to err. Furthermore, models may exhibit poor generalization to new data distributions, leading to performance degradation in unseen environments. Recent advances in enhancing the robustness of neural networks have focused on several key areas:

- **Adversarial Training:** Incorporating adversarial examples into the training process to improve the model's resilience against such attacks. This method aims to expose the network to a wide variety of perturbations during training, thereby reducing the sensitivity to small input changes.

- **Certifiable Robustness:** Developing methods that provide theoretical guarantees on the robustness of the model. Techniques such as robust optimization and interval bound propagation have been explored to ensure that predictions remain stable within certain perturbation bounds.

- **Regularization Techniques:** Applying regularization methods that encourage the model to learn more generalizable and stable features, which are less likely to be affected by noise or adversarial manipulations. Techniques such as dropout, weight decay, and batch normalization contribute to this goal by preventing overfitting the training data.

- **Robust Architecture Design:** Designing neural network architectures that are inherently more robust to input variations. Research in this area includes exploring different network topologies, activation functions, and layer designs that enhance model stability.

1.4.3 Randomized Smoothing

Randomized smoothing, a technique that constructs a smoothed classifier from a base model using noise injection, offers a promising approach to defending against adversarial attacks. The core of randomized smoothing involves creating a smoothed classifier $\hat{f}$ from a base classifier f and a noise distribution $\mathcal{N}$. Given an input x, the smoothed classifier's prediction is determined by:

$$\hat{f}(x) = \arg\max_{c} \mathbb{P}(f(\mathbf{x} + \boldsymbol{\delta}) = c), \quad \delta \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}), \tag{1.15}$$

where δ represents the noise sampled from a Gaussian distribution with standard deviation σ.

A significant contribution of Cohen [15] is the establishment of a theoretical robustness guarantee, which asserts that for any input $\mathbf{x}$, if $\hat{f}$ classifies $\mathbf{x}$ as class c with high probability, then $\hat{f}$ will classify all points within a radius r of $\mathbf{x}$ as c, where r depends on σ and the confidence level.

1.5 Related Work

Pairwise objective paradigms are a fundamental and crucial component of machine learning. Maximization of the AUC metric, for example, was initially introduced in [37] and evaluates the probability of obtaining a higher weight on the positive label. In other words, given a space $\mathcal{X} \times \mathcal{Y}$ with unknown distribution $\mathcal{P}$, a function f that predicts the labels of random positive and negative samples have an AUC score as follows:

$$AUC(f) := Pr(f(\mathbf{x}) > f(\mathbf{x}')|y = 1, y' = -1) = \mathbb{E}[\mathbb{I}_{f(\mathbf{x}) > f(\mathbf{x}')}|y = 1, y' = -1], \tag{1.16}$$

where the expectation is w.r.t. the distribution $\mathcal{P}$.

Metric learning, which aims to learn a distance function, is another example of a method that uses pairwise loss. The distance function $f : \mathcal{X}^2 \mapsto \mathbb{R}$ measures the similarity between two points and learning is improved by minimizing the surrogate loss defined as:

$$\ell(f, (\mathbf{x}, y), (\mathbf{x}', y')) := (1 + r(y, y')f(\mathbf{x}, \mathbf{x}')_+, \tag{1.17}$$

where $r(y, y') = 1$ if $y = y'$ and $r(y, y') = -1$ otherwise.

The first order method is a fairly common strategy for solving problems involving convex and differentiable loss functions (e.g., gradient descent) [80]. However, when considering pairwise losses, the requirement to compute the gradient of all possible pairs requires a significant amount of time with a gradient complexity of $O(T^3)$. In the published literature, researchers are trying to address this challenge in both online and offline settings. Whereas in an offline setting the algorithm has access to the entire training data, in an online setting the algorithm only has access to the data in a sequential manner. In the next part, the state-of-the-art in online pairwise learning using a buffer is examined, followed by offline pairwise learning.

The gradient descent is a very popular approach for a convex and differentiable loss function, but in the case of pairwise, the need to have a gradient of all pairs is very expensive with time complexity $O(T^3)$. The online gradient descent, on the other hand, has sequential access to the data, and thus upon receiving one data point z_t, with t referring to time step, the local error is constructed by pairing the current data point with all $t-1$ points as follows:

$$\mathcal{L}_t(f_{t-1}) = \frac{1}{(t-1)} \sum_{i}^{t-1} \ell(f_{t-1}, z_t, z_i).$$

Although the online gradient descent has time complexity $O(T^2)$, it is still not practical for large-scale problems. Thus, a natural way is to have a buffer of size $s < t-1$ in which only a subset of the received examples are kept, and consequently this increases the complexity of time $O(Ts)$. For example, the work in [100] is considered the first online algorithm for AUC maximization, which considers the class of linear models, i.e. $\mathcal{H} = \mathbb{R}^d$. And the pairwise surrogate objective of the AUC metric is redefined with f parameterized by w as follows:

$$AUC(\mathbf{w}) := \frac{1}{2}\lambda\|\mathbf{w}\|_2^2 + \sum_{i=1}^{T_+}\sum_{j=1}^{T_-}(1 - \mathbf{w}^T(\mathbf{x}_i^+ - \mathbf{x}_j^-))_+ \tag{1.18}$$

where $T_+ + T_-$ are the number of positive and negative examples respectively, and λ is a non negative regularization parameter. To make the problem efficient for the online algorithm, the objective function is rewritten as follows:

$$\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{t=1}^{T}\mathcal{L}_t(\mathbf{w}), \tag{1.19}$$

where,

$$\mathcal{L}_t(\mathbf{w}) = \mathbb{I}_{y_t=1}h_+^t(\mathbf{w}) + \mathbb{I}_{y_t=-1}h_-^t(\mathbf{w}),$$

and

$$h_+^t(\mathbf{w}) = \sum_{t'=1}^{t-1}\mathbb{I}_{y_t=-1}\ell(\mathbf{w}, \mathbf{x}_t - \mathbf{x}_{t'}),$$

$$h_-^t(\mathbf{w}) = \sum_{t'=1}^{t-1}\mathbb{I}_{y_t=1}\ell(\mathbf{w}, \mathbf{x}_{t'} - \mathbf{x}_t).$$

In other words, at any given time t if algorithm receives a data point $x_t, y_t = 1$, then it's paired with all previous $\{x_{t'}, y_{t'} = -1 | 1 < t' < t-1\}$ to find the true loss by the current w. This enables the author to maintain two buffers, B_+ and B_-, of sizes N_+ and N_-, respectively, and only the buffer items are paired with the current data.

Under the constraint of a finite buffer size, Reservoir Sampling is used to update the buffer to maintain a uniform sample of the original dataset. Finally, it is shown that the regret is sublinear, albeit under the assumption that the buffer data and the model are independent at each iteration step. According to the study by Kar et al. [52], which demonstrated that the model and the data in the buffer can be dependent, this assumption is considered incorrect. Note that the regret bound in Theorem 2 [100] depends on the size of the buffers, but increasing the size of the buffers does not have an effect on regret; however, the work in [52] shows that the buffer size has a significant effect on the regret bound.

In 2013 Gao [28] reformulated the aforementioned AUC objective, but using the square loss function to have the followings:

$$\mathcal{L}_t = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{\sum_{i=1}^{t-1} \mathbb{I}[y_t \neq y_i](1 - y_t \mathbf{w}^T(\mathbf{x}_t - \mathbf{x}_i))^2}{2|\{i \in [t-1] | y_i y_t = -1\}|}. \tag{1.20}$$

Exploring the gradient of the squared loss function, which demonstrates that the covariance and mean of the data are sufficient statistics for the streamed data, eliminates the requirement to access all of the data streamed. Obviously, this method cannot be used for high-dimensional data, since the covariance has a space complexity of $O(d^2)$. Although the approach given by [28] maintains a sublinear regret without requiring the storage of historical data, the restriction of the squared loss function and regularization of l_2, as well as the limited dimension of its input data, are its main drawbacks.

In 2016 Ying et al. [97] have constructed a similar formulation of the AUC convex square loss by making use of the saddle point problem and have reduced it by stochastic gradient descent along the primal variables and stochastic gradient ascent along the dual variables. Although the approach is rather effective, it is only applicable to linear AUC models with square loss. Followed by that [77] which replaced the dual variables solution to be determined by closed-form version, and added non-smooth regularization.

Kar et al. [52] modified the sampling technique instead of reservoir sampling (RS) to modified reservoir sampling (RSx). In particular, instead of a random Bernoulli process with probability s/t to replace one random sample of the buffer with the received data point, RSx performs Bernoulli processes s with probability $1/t$ to replace every buffer point with the received data point as illustrated in figure 1.1. This sampling technique guarantees that the s data points are i.i.d. samples of the preceding stream and, at the same time, independent of w.

However, for buffer loss, [52] showed that at every step, the buffer loss have a uniform convergence to true loss (all-pairs) using Rademacher complexity, i.e. : All-pairs loss $\leq$ buffer loss $+\epsilon$ with probability at least $1-\delta$, which is then trnaslated to derive the regret bound of any online algorithm with buffer loss using RSx sampling as $O\left(\sqrt{\frac{\log\frac{T}{\delta}}{s}}T + \sqrt{T}\right)$. Here, the regret is sublinear only for buffer sizes $s > \log(T)$.

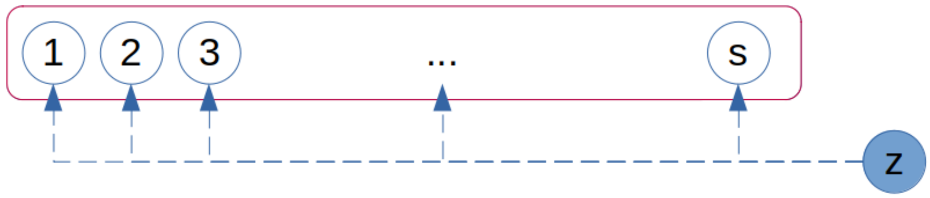


Figure 1.1: Bernoulli Sampling Strategy: Replace each example with probability $p = \frac{1}{t}$, where t is the time step.

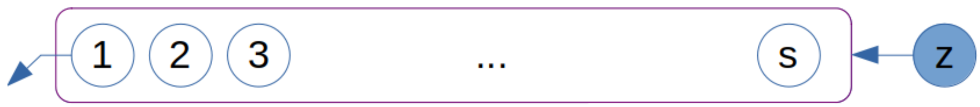


Figure 1.2: FIFO (First-In-First-Out): Updates the buffer by replacing the oldest example with the new one.

Recently [94] claimed that with a buffer of size $s = 1$ i.e., the last received sample as illustrated in figure 1.2, the generalization bound can be optimal. The proof of the claim, using algorithm stability, depends on the assumption that the data received are *i.i.d.* and that at any iteration t we have w_{t-2} and the data in the buffer are independent. Whereas the latter assumption is acceptable, the former assumption is uncommon in online settings, and thus deactivates the second assumption. The buffer plays a critical role in the learning process.

If the euclidean space induced by the inner product l_2 is the model space $\mathcal{H}$, then these models are denoted as linear pairwise models. For example, in the AUC maximization paradigm, since the formulation in (1.16) is neither convex nor differentiable, researchers have approximated the AUC with surrogate convex functions, such as square function and hinge loss.

$$AUC(\mathbf{w}) = \frac{1}{2n^+n^-} \sum_{i,j\neq i}^{T} \ell(f_w(\mathbf{x}_i, \mathbf{x}_j), y_i, y_j)\mathbb{I}[y_i = 1, y_j = -1] + \Omega(\mathbf{w}), \tag{1.21}$$

where $f_w(\mathbf{x}, \mathbf{x}') = \mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_j)$, $w \in \mathbb{R}^d$ is the model to be learned, $\Omega(\mathbf{w})$ is possibly non-smooth but convex regularization, and n^+, n^- are the number of positive and negative examples respectively. Pairwise Stochastic Gradient Descent (PSGD) is a powerful paradigm for machine learning due to its low-cost complexity, flexibility of use, and performance in offline settings with differentiable loss [60]. The fact that the pairwise problem requires $O(T^2)$ instances for the T training data gives PSGD the ability to achieve size-independent complexity, where PSGD iteratively updates the model based on stochastic gradients computed on one or a small number of randomly drawn training samples. Nonetheless, the most important problem with any PSGD based algorithm is determining how to anticipate the performance on unseen data, which can be characterized by the generalization error bound. In the literature, the generalization bound of PSGD algorithms is bounded using two approaches, namely; the uniform convergence [59], and the algorithm stability [93] [61], where the optimal generalization bound $O(1/T)$ is proved for strongly convex loss, with gradient computations $O(T)$. In study by [33], stage-by-stage PSGD is examined, where the training set is increased at each level and the solutions from previous stages are by-based on the next stage as the initial solution.

1.6 Outline of Thesis

Each chapter introduces specific methodologies, theoretical insights, and applications, contributing to the overarching narrative of improving the online pairwise learning with kernelized models, and SNNs in complex learning tasks:

- Chapter 2: *Variance Reduced Online Gradient Descent for Kernelized Pairwise Learning with Limited Memory*: Chapter 2 delves into variance reduction techniques for OGD with buffer in pairwise learning. The chapter presents a comprehensive study of variance reduction strategies tailored to the OGD framework, illustrating how these methods can significantly improve the performance of pairwise learning tasks. Moreover, the chapter also discusses the idea of Random Fourier features to kernelize the model in online pairwise learning.
- Chapter 3: *Limited Memory Online Gradient Descent for Kernelized Pairwise Learning with Dynamic Averaging:* This chapter explores the possibility of having constant buffer OGD algorithm in pairwise learning. The discussion extends to the theoretical requiremnt of buffer based-OGD in a $non-i.i.d.$ setting, offering insights into its convergence properties and practical considerations for implementation.
- Chapter 4: *Manifold Metric Learning with Spiking Neural Networks*: This section introduces the concept of metric learning and its relevance to SNNs, detailing how the distinctive temporal dynamics and binary output nature of SNNs can be leveraged for effective metric learning. Through this exploration, the aim of this chapter is to showcase the potential of SNNs to advance the frontiers of metric learning, offering novel insights and directions for future research.
- Chapter 5:*Provable Robust Spiking Neural Networks for Online Metric Learning*: This chapter shifts the focus to the robustness aspects of SNNs, a critical evaluation metric for assessing the resilience of neural networks against adversarial attacks and perturbations. The discussion encompasses both theoretical frameworks and practical approaches to improving the robustness of SNNs, highlighting their unique advantages and challenges in maintaining integrity under adversarial conditions.

Together, these chapters present a coherent narrative on advancements in online pairwise learning, optimization strategies, and the application of SNNs in robust and metric-based learning frameworks. Discussions blend theoretical contributions with practical insights, paving the way for future explorations and innovations in the field.

1.7 Notations

This section provides an overview of the notations employed throughout this thesis, except where explicitly indicated otherwise.

- SNN: Spiking Neural Network.
- ANN: Artificial Neural Network.
- $\mathbf{x}, \mathbf{w}$: Vectors.
- $\| \cdot \|_{\mathcal{H}}$: Norm in a Hilbert space.
- AUC: Area Under the Receiver Operating Characteristic Curve.
- Pr: Probability.
- $\mathcal{X}$: Input space.
- $\mathcal{Y}$: Label space.
- $\mathcal{E}$: Risk or generalization error.
- Ω: Regularization function.
- k, $\mathcal{G}$: Kernels functions.
- $\mathcal{N}$: Normal distribution.
- $O(\cdot)$: Big O notation to describe the upper bound of algorithms complexity.
- ∇L or ∇f: Gradient of the loss function or any function, respectively.
- τ: Time constant.
- $V[t]$ or $u[t]$: Membrane potential at time t.
- $I[t]$: Input at time t.
- R: Membrane resistance.
- σ: Width of a kernel or distribution.
- B_t: Buffer at time t.
- G: Lipschitz constant.
- M: Smoothness constant.
- $\nabla^2 f$: Second derivative (Hessian) of function f.
- $\mathcal{R}$: Regret.

- ϵ or δ: Radius.
- η, γ: Step sizes.
- Γ: Trace of the covariance matrix.
- $g(t)$: Convexity gap in time t.
- D: Number of random features.
- AOGD: Average Online Gradient Descent.
- SP-Net: SPAM Elastic Net algorithm.
- OGD: Online Gradient Descent.
- S. Kernel: Sparse Kernel algorithm.
- Proj.++: Projection++ algorithm.
- V.R.: Variance Reduction.
- i.i.d.: Independent and Identically Distributed.
- $\mathbb{V}$: Variance.
- κ: Number of clusters.
- $\mathcal{C}_i$: Cluster i.
- $\mathbb{E}$: Expected value.
- h, f: Models or functions.
- sgn: Sign function.
- θ: Model weights.
- Φ: Standard distribution.
- $\mathbb{I}$: Indicator function.
- FGSM: Fast Gradient Sign Method.
- PGD: Projected Gradient Descent.
- GN: Gaussian Noise.
- BPTT: Backpropagation Through Time.
- BPTR: Backpropagation Through Rate.
- $d(a, b)$: Distance between the points a and b.
- t-SNE: t-Distributed Stochastic Neighbor Embedding.

Chapter 2

Variance Reduced Online Gradient Descent for Kernelized Pairwise Learning with Limited Memory

Pairwise learning is essential in machine learning, especially for problems involving loss functions defined on pairs of training examples. Online gradient descent (OGD) algorithms have been proposed to handle online pairwise learning, where data arrive sequentially. However, the pairwise nature of the problem makes scalability challenging, as the gradient computation for a new sample involves all past samples. Recent advances in OGD algorithms have aimed to reduce the complexity of calculating online gradients, achieving complexities lower than $O(T)$ and even as low as $O(1)$. However, these approaches are primarily limited to linear models and have induced variance.

Building on the groundwork laid in Chapter 2, Chapter 3 advances our exploration of scalable solutions for online pairwise learning. Where, the inherent scalability challenges head-on is addressed by proposing a limited memory OGD algorithm. This algorithm not only extends to kernelized models, but also introduces a novel approach to variance reduction. Through strategic sample stratification and the use of a fixed-size buffer, our method optimizes gradient computations, leading to improved sub-linear regret bounds. Our theoretical insights into the relationship between gradient variance and regret, paired with rigorous experimentation, demonstrate a leap in performance, further solidifying the algorithm's place within the realm of online pairwise learning.

In this study, we propose a limited memory OGD algorithm that extends to kernel online pairwise learning while improving the sub-linear regret. Specifically, we establish a clear connection between the variance of online gradients and regret, and construct online gradients using the most recent stratified samples with a limited buffer of size of s representing all past data, which have a complexity of $O(sT)$ and employ random Fourier features $O(\sqrt{T}\log T)$ for kernel approximation. Importantly, our theoretical results demonstrate that the variance-reduced online gradients lead to an improved sub-linear regret bound. The experiments on real-world datasets demonstrate the superiority of our algorithm over both kernelized and linear online pairwise learning algorithms.

2.1 Introduction

Pairwise learning is a machine learning paradigm that focuses on problems where the loss function is defined on pairs of training examples. It has gained significant attention due to its wide range of applications in various domains. For example, in metric learning [56], pairwise learning is used to learn a similarity or distance metric between data points. In bipartite learning [50], it is used to address fairness concerns when making decisions based on two distinct groups. Multiple kernel learning utilizes pairwise learning to combine multiple kernels and enhance the performance of kernel-based methods [30]. AUC maximization involves pairwise learning to optimize the Area Under the ROC Curve, a popular evaluation metric for binary classification [37]. Pairwise differential Siamese networks utilize pairwise learning to compare and classify pairs of samples [51, 84].

Online pairwise learning is an effective approach for real-time decision-making, particularly when dealing with large-scale and dynamic datasets. The process involves sequential processing of data points and updating the model using pairwise examples. One technique that has been explored is online gradient descent, which provides computational efficiency and scalability. However, a drawback of online gradient descent in pairwise learning is its time complexity of $O(T^2)$, where T represents the number of received examples. This is due to the requirement to pair each new data point with all previous points, which leads to significant computational complexity. To address this limitation, researchers have been investigating alternative methods such as buffering and sampling strategies [100, 52, 94]. These approaches aim to reduce computational burden and enable efficient learning on large-scale problems.

Online gradient descent has gained significant attention in the domain of online pairwise learning, leading to the development of various approaches. These methods, including online buffer learning [100, 52], second-order statistic online learning [28], and saddle point-problem methods [97, 77], have all employed linear models. Moreover, there has been limited exploration of non-linear models in this field, particularly with kernelized learning [98, 23]. One notable approach in pairwise learning is online buffer learning, introduced by Zhao et al. [100]. This method uses a finite buffer with reservoir sampling to reduce time complexity to $O(sT)$, where s denotes the buffer size. By storing a subset of the data and ensuring uniform samples within the buffer, this technique effectively alleviates the computational burden. Furthermore, Yang et al. [94] achieved optimal generalization with a buffer size of $s = 1$, marking a significant advancement in the field.

The existing frameworks in the literature have primarily focused on linearly separable data, overlooking the challenges associated with non-linear pairwise learning. Moreover, the online buffer methods proposed so far have not adequately addressed the sensitivity of generalization to the variance of the gradient. This limitation restricts their ability to capture the complexity present in real-world datasets. Moreover, there is a lack of extensive research on non-linear pairwise learning, particularly in the context of kernel approximation. Although non-linear methods provide increased expressive power, the computational cost associated with kernel computation, which scales as $O(T^2)$ [66, 49], poses challenges to their scalability and efficiency in practical applications. In terms of generalization bounds, the analysis of online pairwise gradient descent with buffers and linear models has

Table 2.1: Recent pairwise learning algorithms(where T is the iteration number, d: the dimension, D is random features, V.R. denotes variance reduction, and s is a buffer size), note the time complexity is w.r.t. gradients computations.

Algorithm	**Problem**	**Model**	**Scheme**	**V.R.**	**Time**	**Space**
[33]	AUC	Linear	Offline	NA	$O(T)$	$O(1)$
[73]	AUC	Linear	Online	NA	$O(T)$	$O(1)$
[97]	AUC	Linear	Online	NA	$O(T)$	$O(1)$
[100]	AUC	Linear	Online	No	$O(sT)$	$O(s)$
[28]	AUC	Linear	Online	No	$O(T)$	$O(d)$
[94]	General	Linear	Online	No	$O(T)$	$O(1)$
[52]	General	Linear	Online	No	$O(sT)$	$O(s)$
[66]	General	Kernel	Online	NA	$O(T^2)$	$O(T)$
[49]	AUC	Kernel	Offline	NA	$O(T \log T)$	$O(T^2)$
FPOGD (Ours)	General	Kernel	Online	Yes	$O(\frac{sD}{d}T)$	$O(s)$

been extensively explored in previous studies [90, 52]. These works establish a bound of $O(1/s + 1/\sqrt{T})$ for this approach. However, it is important to note that this bound is only optimal when the buffer size s is approximately $O(\sqrt{T})$, which presents challenges for scenarios where a smaller buffer size is desired. Additionally, the generalization analysis in [94] assumes independent examples in sequential data, disregarding the temporal nature of the data and the possible ordering and correlation among data points. This assumption may lead to inaccurate performance estimation and unreliable convergence guarantees in online learning scenarios. Taken together, these weaknesses highlight the need for further research and development in the field of online pairwise learning to address the limitations of linear frameworks, explore non-linear methods more comprehensively, and overcome the computational challenges associated with kernel computation.

Our approach extends online pairwise learning to handle nonlinear data by incorporating kernelization of the input space. We address the impact of variance on regret through online stratified sampling, selectively updating the model based on cluster relevance. Utilizing random Fourier features, we efficiently estimate the kernel with sub-linear error bound, achieving computational savings without sacrificing performance. By combining kernelization, efficient kernel approximation, and online stratified sampling, our method overcomes linear limitations, handles nonlinear data, and mitigates variance impact, resulting in a robust and effective online pairwise learning approach (Table 2.1). Our main contributions can be summarized as follows:

- We present an online pairwise algorithm for non-linear models with fast convergence. Our algorithm achieves sub-linear regret with a buffer size of $O(s)$.
- We address the impact of variance on regret and propose online stratified sampling to control and improve the regret rate.
- For the case of Gaussian kernel, we approximate the pairwise kernel function using

only $O(\sqrt{T}\log T)$ features in comparison to $O(T)$ in previous works, while maintaining a sub-linear error bound.

- We demonstrate the effectiveness of our proposed technique on numerous real-world datasets and compare it with state-of-the-art methods for AUC maximization. Our methodology showcases improvements across both linear and non-linear models for the majority of the examined datasets.

The following sections are organized as follows. Section 2.2 introduces problem setting, Section 2.4 presents the proposed method, Section 2.5 provides regret analysis, Section 2.6 discusses related work, followed by Section 2.7 producing experimental results, and finally Section 2.8 concludes the chapter.

2.2 Problem Setting

The concept of pairwise learning arises in the context of a subset $\mathcal{X} \subset \mathbb{R}^d$ and a label space $\mathcal{Y} \subset \mathbb{R}$. It can be categorized into two cases:

- Pairwise hypothesis: This case involves learning a pairwise hypothesis, as in metric learning, where the goal is to determine the relationship or distance between pairs of data points in $\mathcal{X}$. In particular, the hypothesis predicts the distance between pairs of instances, i.e. $f : \mathcal{X}^2 \mapsto \mathcal{Y}$, and therefore given n examples, the loss function is a finite sum of $\binom{n}{2}$ terms.

- Pairwise loss function: In this case, the focus is on minimizing a pairwise loss function, such as in AUC (area under curve) maximization. The objective is to optimize the ordering or ranking of pairs of data points based on their labels in $\mathcal{Y}$. In general, the hypothesis is pointwise as in SVM, regression and binary deep classification, i.e. $f : \mathcal{X} \mapsto \mathcal{Y}$, however, the loss function itself represents the probability of correctly predicting the labels of opposing examples.

In our analysis, we specifically investigate pairwise loss functions from both branches. We establish a connection between the pairwise kernel associated with pairwise hypotheses and regular kernels. This enables us to explore the characteristics of pairwise loss functions within the framework of regular kernels. Consider an algorithm that learns from the examples $z_i := (x_i, y_i) \in \mathcal{Z} := \mathcal{X} \times \mathcal{Y}$, where $i \in [T]$ denotes the number of examples. Let f belong to the space $\mathcal{H}$. In this paradigm, the pairwise loss function serves as a performance measure, denoted as $\ell : \mathcal{H} \times \mathcal{Z}^2 \to \mathbb{R}_+$.

Likewise, in online learning with pairwise losses, when a new data point z_t is received, a local error is generated by incorporating the new data point together with all previous $t-1$ points. The local error is then determined on the basis of the chosen pairwise loss function as follows:

$$L_t(f_{t-1}) = \frac{1}{t-1}\sum_{i=1}^{t-1} \ell(f_{t-1}, z_t, z_i). \tag{2.1}$$

The core objective in online pairwise learning is to create an ensemble of models, denoted as $f_1, f_2, \ldots, f_T$, aimed at minimizing the expected risk. Assuming that the data are assigned to a higher-dimensional space where linear separability is achieved, we consider a linear model represented as w. To address the issue of memory requirements, we employ a buffer-based local error denoted as $\bar{L}_t(w_{t-1})$, as defined in Equation 2.2. At each step t, the buffer, denoted B_t, contains a limited number of historical example indices, and the cardinality of the buffer is represented as $|B_t|$ (equivalent to s in the existing literature).

$$\bar{L}_t(w_{t-1}) = \frac{1}{|B_t|} \sum_{i \in B_t} \ell(w_{t-1}, z_t, z_i). \tag{2.2}$$

The buffer plays a critical role in the learning process, being updated at each step using diverse strategies, ranging from randomized techniques like reservoir sampling [100, 52] to non-randomized approaches like FIFO [94]. However, it should be noted that there is a noticeable research gap on the variance implications of these sampling methods, despite their widespread use.

To handle complex real-world data, our pairwise online approach assumes mapping both the hypothesis and the data to a Reproducing Kernel Hilbert Space (RKHS) denoted as $\mathcal{H}$. The associated Mercer pairwise kernel function $k : \mathcal{X}^4 \mapsto \mathbb{R}$ satisfies the reproducing property $\langle k_{(x,x')}, g \rangle = g(x, x')$, where $x, x' \in \mathcal{X}^2$ and $g \in \mathcal{H}$. In the case of pointwise hypothesis but pairwise loss functions, such as AUC loss, the kernel function simplifies to $k : \mathcal{X}^2 \mapsto \mathbb{R}$. The space $\mathcal{H}$ encompasses all linear combinations of functional mappings $\{k_{(x,x')} | (x, x') \in \mathcal{X}^2\}$ and their limit points.

To address the computational complexity of kernelization in the online setting, we utilize random Fourier features (RFF) as an efficient approximation of the Mercer kernel function. RFF provides a lower-dimensional mapping $r(\cdot)$, which approximates the kernel function, with the estimate denoted as $\bar{k}(\cdot)$. This approximation allows us to perform computations using linear operations, significantly reducing computational complexity. The space spanned by the new kernel functions is denoted as $\hat{\mathcal{H}}$. Previous work has studied the error of random Fourier approximation in pointwise and offline settings. In the online setting, the minimum number of random features required to ensure sub-linear regret has been found to be $O(T)$. In our method, we introduce an error bound for pairwise problems using only $O(\sqrt{T} \log T)$ random features (see Section 5 for more details).

2.2.1 Assumptions

Before introducing our main theorems, we outline a set of widely accepted assumptions concerning the properties of the loss function and kernels. These assumptions hold significance in the realm of convex optimization and encompass commonly used loss functions such as squared loss as well as popular kernels like the Gaussian kernel.

Assumption 1 (M-smoothness) *Assume for any $a \in \mathcal{Z}^2 \times \mathcal{H}$, the gradient of the loss function $\nabla \ell(a)$ is M-Lipschitz continuous, i.e. $\forall w, w' \in \mathcal{H}$,*

$$\|\nabla \ell(a) - \nabla \ell(a')\| \leq M \|a - a'\|_2 .$$

Assumption 2 (Convexity) *Assume for any $z, z' \in \mathcal{Z}$, the loss function $\ell(\cdot, z, z')$ is convex function, i.e. $\forall w, w' \in \mathcal{H}$,*

$$\ell(w, z, z') \geq \ell(w', z, z') + \nabla \ell(w', z, z')^T (w - w').$$

Assumption 3 (Finite Kernel) *Assume for any ρ-probability measure on $\mathcal{X}^2$ the positive kernel function $k : \mathcal{X}^2 \times \mathcal{X}^2 \to \mathbb{R}$ is ρ-integrable, i.e. for any $(x, x') \in \mathcal{X}^2$,*

$$\int \int_{\mathcal{X}^2} k((x, x'), (\hat{x}, \hat{x}')) d\rho(\hat{x}) d\rho(\hat{x}') < \infty.$$

2.3 Backgorund

In the analysis of buffer-based pairwise online gradient descent algorithms, two key concepts are essential for understanding the relationship between regret and variance in the proposed method.

2.3.1 Variance of Stochastic Gradient

Let us denote the variance of the stochastic gradient as $\mathbb{V}(u_t)$, with $u_t := \nabla \bar{L}_t(w_{t-1})$ is the gradient based on a finite buffer. The variance is defined as the trace of the covariance matrix, i.e., $\mathbb{V}(u_t) := \mathbb{E}\|u_t - \mathbb{E}u_t\|^2$. The following lemma sheds light on the connection between the variance of the gradient and the distance between the inputs and their corresponding expected values.

Lemma 1 *Assuming that the loss function is M-smooth, as mentioned in assumption 1, let z_i denote the i-th sample drawn from a uniform distribution, i.e., $i \sim Uniform(1, t-1)$. Then, the variance $\mathbb{V}(u_t)$ of the stochastic gradient is bounded by:*

$$\mathbb{V}(u_t) \leq M^2 \mathbb{E}\|x_i - \mathbb{E}x_i\|^2, \tag{2.3}$$

where the expectation is w.r.t. the uniform random variable i.

Proof 1 *Let x_i sampled example from the history of examples, where $i \sim uniform[1, t-1]$, then since the loss function is M-smooth we have:*

$$\|\nabla \ell(w, z_t, (x, y)) - \nabla \ell(w, z_t, (\mathbb{E}x, y))\|_2 \leq M \left\| x - \mathbb{E}x \right\|_2,$$

then by adding and subtracting $\mathbb{E}\nabla \ell(w, z_t, z)$ to LHS after squaring both sides, and denote $\mathbb{E}z = (\mathbb{E}x, y)$ we have,

$$
\begin{aligned}
\Leftrightarrow & \|\nabla\ell(w,z_t,z)-\mathbb{E}\nabla\ell(w,z_t,z)+\mathbb{E}\nabla\ell(w,z_t,z)-\nabla\ell(w,z_t,\mathbb{E}z)\|^2 \le M^2\|x-\mathbb{E}x\|^2 \\
\Leftrightarrow & \|\nabla\ell(w,z_t,z)-\mathbb{E}\nabla\ell(w,z_t,z)\|^2+\|\mathbb{E}\nabla\ell(w,z_t,z)-\nabla\ell(w,z_t,\mathbb{E}z)\|^2 \\
& -2(\nabla\ell(w,z_t,z)-\mathbb{E}\nabla\ell(w,z_t,z))^T(\mathbb{E}\nabla\ell(w,z_t,z)-\nabla\ell(w,z_t,\mathbb{E}z)) \le M^2\|x-Ex\|^2.
\end{aligned} \tag{2.4}
$$

Taking expectation on both sides w.r.t. uniform distribution of i, and rearrange to have:

$$
\begin{aligned}
& \mathbb{E}\|\nabla\ell(w,z_t,z)-\mathbb{E}\nabla\ell(w,z_t,z)\|^2+\mathbb{E}\|\mathbb{E}\nabla\ell(w,z_t,z)-\nabla\ell(w,z_t,\mathbb{E}z)\|^2 \\
& \quad -2\underbrace{\mathbb{E}(\nabla\ell(w,z_t,z)-\mathbb{E}\nabla\ell(w,z_t,z))^T}_{=0}(\mathbb{E}\nabla\ell(w,z_t,z)-\nabla\ell(w,z_t,\mathbb{E}z)) \le M^2\mathbb{E}\|x-Ex\|^2 \\
\Leftrightarrow & \mathbb{E}\|\nabla\ell(w,z_t,z)-\mathbb{E}\nabla\ell(w,z_t,z)\|^2 \le -\|\mathbb{E}\nabla\ell(w,z_t,z)-\nabla\ell(w,z_t,\mathbb{E}z)\|^2+M^2\mathbb{E}\|x-Ex\|^2 \\
& \le M^2\mathbb{E}\|x-Ex\|^2.
\end{aligned} \tag{2.5}
$$

Recall the definition of the variance of the stochastic gradient for final results. This completes the proof.

2.3.2 Regret Bound and Stochastic Gradient Variance

The variance of the stochastic gradient plays a crucial role in determining the regret bound of pairwise online gradient descent algorithms. The following lemma establishes a connection between regret and the variance of the stochastic gradient.

Lemma 2 *With assumption 2, let $[w_t]_{i=1}^T$ be the sequence of models returned by running any buffer-based algorithm for T time-steps using an online sequence of data. If $\bar{w}^* = \arg\min_{w\in\hat{\mathcal{H}}}\sum_{t=2}^T L_t(w)$, and B_t is sampled from the history of the received examples uniformly and independently, then the following holds:*

$$
\sum_{t=2}^T L_t(w_{t-1}) \le \sum_{t=2}^T L_t(\bar{w}^*) + \frac{\|\bar{w}^*\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=2}^T(\mathbb{V}(u_t)+\|\mathbb{E}u_t\|^2), \tag{2.6}
$$

where the expectation is w.r.t. the uniform distribution of buffer examples.

Hence, reducing the variance of the stochastic gradient can improve the regret of buffer online pairwise learning, which can be achieved using online stratified sampling as illustrated in the next section.

Proof 2 *Let $g_t(\cdot) = \frac{1}{|B_t|}\sum_{j\in B_t}\ell(\cdot, z_t, z_j)$ be convex function for all $t \ge 1$ where B_t is the buffer of uniformly sampled i.i.d. history examples. Let $u_t \in \partial g_t(w_{t-1})$. If we take the distance of two subsequent models to the optimal model, we have:*

$$
\begin{aligned}
& \|w_t-\bar{w}^*\|^2-\|w_{t-1}-\bar{w}^*\|^2 = \|w_{t-1}-\eta_t u_t-\bar{w}^*\|^2-\|w_{t-1}-\bar{w}^*\|^2 \\
& = \|w_{t-1}-\bar{w}^*\|^2-2\eta_t u_t^T(w_{t-1}-\bar{w}^*)+\eta_t^2\|u_t\|^2-\|w_{t-1}-\bar{w}^*\|^2 \\
& = -2\eta_t u_t^T(w_{t-1}-\bar{w}^*)+\eta_t^2\|u_t\|^2 \\
& \le -2\eta(g_t(w_{t-1})-g_t(\bar{w}^*))+\eta_t^2\|u_t\|^2,
\end{aligned} \tag{2.7}
$$

where the last inequality implements the Assumption 2, i.e. $-u_t^T(w_{t-1}-\bar{w}^*) \leq -(g_t(w_{t-1})-g_t(\bar{w}^*))$.

Setting the step size $\eta_t = \eta$ *for all* t*, taking the expectation w.r.t. the uniform randomness of the history points, and assume that if* w *is fixed then* $\mathbb{E}g_t(\cdot) = L_t(\cdot)$

$$L(w_{t-1}) - L(\bar{w}^*) \leq \frac{\|w_{t-1} - \bar{w}^*\|^2 - \|w_t - \bar{w}^*\|^2}{2\eta} + \frac{\eta\mathbb{E}\|u_t\|^2}{2}. \tag{2.8}$$

Finally, using the identity $\mathbb{E}\|u_t\|^2 = \mathbb{V}(u_t) + \|\mathbb{E}u_t\|^2$*, summing from* $t = 2$ *to* $t = T$ *and setting* $w_1 = 0$*, would complete the proof.*

2.4 Proposed Method

The proposed method consists of two essential parts that are mutually dependent. Firstly, by mapping non-linearly separable data to the RKHS $\mathcal{H}$, we achieve a transformation that renders the data linearly separable. This mapping serves as the foundation for effectively addressing non-linearity. Secondly, through the strategic implementation of stratified sampling, we can potentially reduce variance, preserve low memory utilization, and achieve sub-linear regret.The initial mapping to the RKHS enables us to seamlessly fulfill the objectives of the second component, ensuring a general efficient approach as illustrated in Figure 2.1.

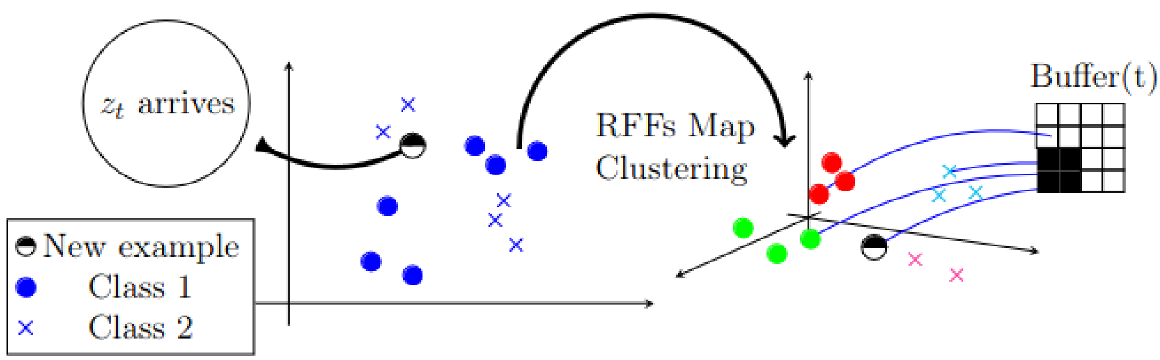


Figure 2.1: The incoming point z_t from the left are transformed through kernelization and RFFs to a new space. Subsequently, they are clustered and added to the buffer using either a FIFO strategy or with a probability-based approach.

2.4.1 Non-Linear Mapping to RKHS

In pairwise online gradient descent algorithms, our goal is to handle non-linearly separable data. To achieve this, we employ a transformation $\Phi : \mathcal{X} \mapsto \mathcal{H}$ that maps the input space $\mathcal{X}$ to a high-dimensional RKHS $\mathcal{H}$. By performing this transformation, we project the non-linearly separable data into a higher-dimensional space where it becomes linearly separable.

However, the computational complexity associated with using explicit kernel computations can be prohibitive. To address this challenge, we leverage the RFFs technique, which allows us to approximate the inner products in the target space $\mathcal{H}$ more efficiently. Instead of directly computing the inner products in $\mathcal{H}$, we map the input space to an approximate space $\hat{\mathcal{H}}$ using a randomized mapping function. This approximation enables us to estimate the inner products of the original data points in the approximated space, rather than in the full high-dimensional space.

By applying this non-linear mapping using RFFs to the RKHS, we effectively handle the non-linearity in pairwise online gradient descent algorithms. This approach facilitates better separation of data points in the transformed space $\hat{\mathcal{H}}$, while maintaining a lower computational complexity of $O\left(\frac{sD}{d}T\right)$, compared to the $O(T^2)$ complexity associated with explicit kernel computations.

2.4.2 Online Stratified Sampling

In addition to the non-linear mapping, we employ stratified sampling to further improve the efficiency and reduce the variance of the stochastic gradient estimates. Online Stratified Sampling (OSS) partitions the input space into balls of radius ϵ and ensures that each ball is represented by a uniform sample at each iteration. By doing so, we achieve several advantages.

First, stratified sampling reduces the variance of the stochastic gradient estimates. By partitioning the input space and uniformly sampling within each partition, we effectively minimize the variance of the stochastic gradient. This is achieved by reducing the expected distance between the sampled variables and their corresponding expected values, as highlighted in the Lemma 1.

Second, stratified sampling preserves low memory utilization. Instead of storing the entire history of the received examples, we maintain a uniform sample from each partition at every step t. This approach reduces the memory requirement while still providing sufficient information to estimate the gradients accurately.

Finally, stratified sampling enables us to achieve sub-linear regret. By reducing variance and preserving low memory utilization, our method ensures efficient exploration and exploitation of the data, leading to improved regret bounds. To this end, we redefine the loss at each time step considering the presence of κ_t partitions, denoted as $\mathcal{C}_j^t$, where each partition has a cardinality of $|\mathcal{C}_j^t|$, and the corresponding gradient is shown in equation 2.9.

$$\nabla \bar{L}_t(w_{t-1}) = \sum_{j=1}^{|B_t|} \frac{|\mathcal{C}_j^t|}{t-1} \nabla \ell(w_{t-1}, z_t, z_{B_t[j]}). \tag{2.9}$$

Note that the gradient mentioned above is unbiased, i.e. $\mathbb{E}\nabla \bar{L}_t(w_{t-1}) = \nabla L_t(w_{t-1})$. To reduce the variance in the estimation of stochastic gradients, we maintain a uniform sample of each partition at every step t, ensuring that $|B|_t = \kappa_t$. This update approach enables us to achieve a lower variance. To accomplish this, our objective is to find the optimal partitions at each time step t, which involves solving the following optimization problem or its upper bound based on the Lemma 1.

$$\begin{aligned}\min_{\mathcal{C}_j^t} \quad \mathbb{V}(u_t) &= \sum_{j=1}^{\kappa_t} \frac{(|\mathcal{C}_j^t|)^2}{(t-1)^2} \mathbb{E}_{j|j\in\mathcal{C}_j^t} \|\nabla \ell(w, z_t, z_j) - \mathbb{E}_{j|j\in\mathcal{C}_j^t} \nabla \ell(w, z_t, z_j)\|^2 \\ &\leq \sum_{j=1}^{\kappa_t} \frac{(|\mathcal{C}_j^t|)^2 M^2}{(t-1)^2} \mathbb{E}_{j|j\in\mathcal{C}_j^t} \|x_j - \mathbb{E}_{j|j\in\mathcal{C}_j^t} x_j\|^2 \\ &= \sum_{j=1}^{\kappa_t} \frac{|\mathcal{C}_j^t| M^2}{(t-1)^2} \sum_{j\in\mathcal{C}_j^t} \|x_j - \mathbb{E}_{j|j\in\mathcal{C}_j^t} x_j\|^2. \end{aligned} \tag{2.10}$$

The objective of our approach is similar to a conventional clustering problem, where $\mathbb{E}_{j|j\in\mathcal{C}_j^t} x_j = \sum_{j\in\mathcal{C}_j^t} \frac{x_j}{|\mathcal{C}_j^t|}$ represents the centroid of partition j (note that partition may be referred to as "cluster" intermittently). Our approach offers an effective solution by simultaneously addressing memory efficiency and variance reduction as certified below.

2.4.3 Certificate of Variance Reduction

Assume that there exist κ_t clusters and denote $\hat{u}_t(\cdot)$ the cluster-based buffer gradient constructed using online stratified sampling 1, and $u_t(\cdot)$ represents the estimate obtained from uniform sampling without online clustering,

$$\begin{aligned}\mathbb{V}(u_t) &= \frac{1}{\kappa_t} \mathbb{E}_i \|\nabla \ell(w, z_t, z_i) - \mathbb{E}_i \nabla \ell(w, z_t, z)\|^2 \\ &= \frac{1}{\kappa_t} \mathbb{E}_i \|\nabla \ell(w, z_t, z_i)\|^2 - \|\mathbb{E}_i \nabla \ell(w, z_t, z)\|^2 \\ &\overset{(a)}{=} \frac{1}{\kappa_t} \mathbb{E}_{\mathcal{C}_j^t} \mathbb{E}_{i|i\in\mathcal{C}_j^t} \|\nabla \ell(w, z_t, z_i | i \in \mathcal{C}_j^t)\|^2 - \|\mathbb{E}_i \nabla \ell(w, z_t, z)\|^2 \\ &= \frac{1}{\kappa_t} \sum_{j=1}^{\kappa_t} \frac{c_j^t}{t-1} \mathbb{E}_{i|i\in\mathcal{C}_j^t} \|\nabla \ell(w, z_t, z_i | i \in \mathcal{C}_j^t)\|^2 - \|\mathbb{E}_i \nabla \ell(w, z_t, z)\|^2 \\ &= \frac{1}{\kappa_t} \sum_{j=1}^{\kappa_t} \frac{c_j^t}{t-1} \mathbb{V}_{i|i\in\mathcal{C}_j^t} \nabla \ell(w, z_t, z_i | i \in \mathcal{C}_j^t) + \|\mathbb{E}_{i|i\in\mathcal{C}_j^t} \nabla \ell(w, z_t, z_i | i \in \mathcal{C}_j^t)\|^2 - \|\mathbb{E}_i \nabla \ell(w, z_t, z)\|^2 \\ &\overset{(b)}{=} \frac{1}{\kappa_t} \sum_{j=1}^{\kappa_t} \frac{c_j^t}{t-1} \mathbb{V}_{i|i\in\mathcal{C}_j^t} \nabla \ell(w, z_t, z_i | i \in \mathcal{C}_j^t) + \|\mathbb{E}_{i|i\in\mathcal{C}_j^t} \nabla \ell(w, z_t, z_i | i \in \mathcal{C}_j^t) - \mathbb{E}_i \nabla \ell(w, z_t, z)\|^2 \\ &\geq \frac{1}{\kappa_t} \sum_{j=1}^{\kappa_t} \frac{c_j^t}{t-1} \mathbb{V}_{i|i\in\mathcal{C}_j^t} \nabla \ell(w, z_t, z_i | i \in \mathcal{C}_j^t) = \mathbb{V}(\hat{u}_t), \end{aligned} \tag{2.11}$$

where equality (a) and equality (b) implements total expectation, i.e. $\mathbb{E}_{\mathcal{C}_j^t} \mathbb{E}_{i|i\in\mathcal{C}_j^t} \nabla \ell(w, z_t, z_i | i \in \mathcal{C}_j^t) = \mathbb{E}_i \nabla \ell(w, z_t, z_i)$. The reduction in variance is influenced by the variances within each partition and the number of examples in it. If each cluster has same number of examples $(t-1)/\kappa_t$, we can observe that the bound becomes $1/\kappa_t$. Note that the maximum reduction in variance is $(t-1)$, which is the case for the full gradient. It is worth noting that the variance reduction assumes comparable variances among clusters. If the clusters have different variances, it is advisable to sample more from the high-variance cluster. This extension can be easily incorporated into our algorithm by considering the running variances of each cluster.

2.4.4 The Algorithm

We introduce the algorithm in 1 for general pairwise learning with kernel approximation using RFFs. The centroid update of the OSS algorithm involves minimizing the upper bound given in Equation 2.10. This minimization can be achieved using the upper bound gradient with respect to the centroid c of partition j, denoted as $c := \mathbb{E}_{j|j\in\mathcal{C}_j^t}[x_j]$. The centroid update is then performed as follows:

$$c \leftarrow c + \eta_c(z_t - c), \tag{2.12}$$

where z_t represents the newly assigned example to partition j, and η_c is the step size. This update ensures that the centroid moves towards the assigned example in order to minimize the upper bound. In the subsequent section, we provide an analysis that decomposes the regret of the algorithm into two distinct components, first, the regret of learning $\bar{w}^* := \arg\min_{w\in\hat{\mathcal{H}}} L(w)$, second, the regret of the kernel approximation using RFFs mapping.

2.5 Regret Analysis

The regret of the online algorithm relative to the optimal hypothesis in the space $\mathcal{H}$, i.e., $w^* = \arg\min_{w\in\mathcal{H}} \sum_{t=2}^{T} L_t(w)$ when running on a sequence of T examples is:

$$\mathcal{R}_{w^*,T} = \sum_{t=2}^{T} L_t(w_{t-1}) - \sum_{t=2}^{T} L_t(w^*), \tag{2.13}$$

where the local all pairs loss $L_t(\cdot)$ is defined in equation 2.1. We can decompose the regret in equation 2.13 by introducing best-in-class hypothesis in the approximated space $\hat{\mathcal{H}}$, e.g: $\bar{w}^* = \arg\min_{w\in\hat{\mathcal{H}}} \sum_{t=2}^{T} L_t(w)$ as follow:

$$\mathcal{R}_{w^*,T} = \underbrace{\sum_{t=2}^{T} L_t(w_{t-1}) - \sum_{t=2}^{T} L_t(\bar{w}^*)}_{T_1} + \underbrace{\sum_{t=2}^{T} L_t(\bar{w}^*) - \sum_{t=2}^{T} L_t(w^*)}_{T_2}.$$

We provide the bound on T_1 in Theorem 2 (Section 2.5.1), and then provide the bound on T_2 in Theorem 3 (Section 2.5.2). Finally, by combining them together, we could provide the main theorem on regret for the Algorithm 1 as follows.

Algorithm 1 Fourier Pairwise Online Gradient Descent (FPOGD)

Require: Initialization $w_1 \in \hat{\mathcal{H}}$, random feature size D, initial example z_1, Fourier feature distribution p of kernel k, step size η, clustering distance ϵ.

$B_1 \leftarrow \{z_1\}$, $C_1 \leftarrow \{z_1\}$
for $t = 2, \ldots, T$ **do**
  Receive new example $(x_t, y_t) \in \mathcal{Z}$
  Sample Fourier feature $\{q_i\}_{i=1}^{D/2} \sim p$
  Map to new space:

$$z_t = (\frac{2}{\sqrt{D}}[\sin(q_i^T x_t), \cos(q_i^T x_t)]_{i=1}^{D/2}, y_t)$$

  Suffer loss

$$g_t(w_{t-1}) = \sum_{j=1}^{|B_t|} \frac{|\mathcal{C}_j^t|}{t-1} \ell(w_{t-1}, z_t, z_{B_t[j]})$$

  Select $v_t \in \partial g_t(w_{t-1})$
  Update model $w_t = w_{t-1} - \eta v_t$
  Update $B_t, C_t = OSS(B_{t-1}, C_{t-1}, \epsilon, z_t)$
end for
Return w_T

Online Stratified Sampling (OSS) with FIFO Buffer Update

Require: Buffer B, Centroids C, Clustering threshold ϵ, new example $z_t = (x_t, y_t)$

if $\min_{c \in C} \|x_t - c\|^2 \leq \epsilon$ **then**
  Select cluster

$$j := \arg\min_{c \in C} \|x_t - c\|^2$$

  Update buffer $B[j] \leftarrow t$
  Update Centroid j using equation 2.12
else
  Add new cluster $C \leftarrow C \cup t$
  Add new examples $B \leftarrow B \cup t$
end if
Return B, C

Theorem 1 *Let $\{z_t \in \mathcal{Z}\}_{t=1}^T$ be sequentially accessed by algorithm 1. Let D be the number of RFFs. And assumptions 1, 2 and 3 hold with M Lipschitz constant. Then, if the step size $\eta \in (0, 1/M]$, the regret bound compared to w^* is bounded with probability at least $1 - 2^8 \left(\frac{\sigma R}{\delta}\right) \exp(-D\delta^2/(4d+8))$ as follows:*

$$\mathcal{R}_{w^*,T} \leq \frac{M}{2} T \|w^*\|_1^2 \delta^2 + \frac{\|\bar{w}^*\|^2}{2\eta} + \eta \sum_{t=2}^{T} \mathbb{V}(u_t), \tag{2.14}$$

where $\|w^*\|_1 = \sum_{i,j\neq i}^T |a_{i,j}^*|$, δ is the kernel approximation error, κ_t is the number of clusters, σ is the kernel width, $\mathbb{V}(u_t) = tr(cov[u_t])$, R is the input diameter.

Remark 1 *Choosing $\eta = \frac{1}{\sqrt{T}}$ and $\delta = T^{-1/4}$ makes the regret bound sub-linear which is optimal. Moreover, if $\mathbb{V}(u_t) = 0$ for all t's, then it is possible to have $\log(T)$ regret by choosing $\eta = \frac{1}{\log(T)}$ and $\delta = \frac{\log(T)}{\sqrt{T}}$, which is similar to the case of full history update.Note that $D = O(\sqrt{T}\log(T))$ in general, but can be as low as $\log^2(T)$ for special kernels, (please refer to section 2.7.5) .*

In the following, we provide the analysis to the upper bounds to T_1 and T_2, respectively.

2.5.1 Regret in the Approximated Space

The choice of buffer updating method, whether randomized (e.g., reservoir sampling) or non-randomized (e.g., FIFO), significantly impacts the analysis, as highlighted by [52] and [90]. To ensure the independence between sampling randomness and data randomness, we begin with a simple FIFO approach, proving the T1 bound in Theorem 2 under the *i.i.d.* assumption. We then introduce reservoir sampling, which uniformly samples the stream without the assumption *i.i.d.*, establishing convergence using the Rademacher complexity of pairwise classes.

Consider the algorithm that has sequential access to the online stream. The following theorem demonstrates that the algorithm achieves optimal regret with memory complexity $O(\kappa_t)$.

Theorem 2 *With assumptions 1 and 2, let $[w_t]_{i=1}^T$ be the sequence of models returned by running Algorithm 1 for T times using the online i.i.d. sequence of data. Then, if $\bar{w}^* = \arg\min_{w\in\hat{\mathcal{H}}} \sum_{t=2}^T L_t(w)$, and $\eta \in (0, 1/M]$, the following holds:*

$$\sum_{t=2}^{T} L_t(w_{t-1}) - \sum_{t=2}^{T} L_t(\bar{w}^*) \leq \frac{\|\bar{w}^*\|^2}{2\eta} + \eta \sum_{t=2}^{T} \mathbb{V}(u_t). \tag{2.15}$$

Remark 2 *If the original space is assumed to be linearly separable (without kernels), then our algorithm has a time complexity of $O(\kappa_T T)$ and offers sub-linear regret with $\eta = 1/\sqrt{T}$. Also, note that for the case of $\epsilon \geq R$ the algorithm is equivalent to [94] and if $\epsilon \leq R/T$ it matches the algorithm in [10]. In particular, if the clustering radius ϵ is large enough, it will result in only one cluster, similar to Yang's approach. Conversely, if ϵ is small (smaller than the distance between any two examples), it will create a cluster for each example, similar to the Boissier approach.*

Remark 3 *In the worst-case scenario, pairwise learning data points are assigned to new clusters or grouped within an epsilon distance from the initial centroid. The maximum number of clusters κ_t at step t can be upper-bounded by considering non-overlapping hyperspheres of radius epsilon in the bounded input space. Given the volume of the input space V_t^d at time step t, the cluster threshold of ϵ, and the gamma function $\Gamma(\cdot)$, we have:*

$$\kappa_t = \min\left\{t-1, \frac{V_t^d \Gamma(d/2+1)}{\pi^{(d/2)}\epsilon^d}\right\}. \tag{2.16}$$

For a hypersphere input space with constant radius R in all time steps, the maximum number of clusters is simplified to $\kappa_t = \min\{t-1, (R/\epsilon)^d\}$. In practice, the actual number of clusters obtained may be lower due to the data distribution (for experimental validations, please refer to section 2.7.4).

This theorem establishes that the algorithm achieves optimal regret. The regret bound ensures that the algorithm's performance closely approaches the optimal solution, considering the Lipschitz continuity and convexity conditions. The FIFO buffer update scheme, combined with the analysis of the expected squared norm of the gradients, contributes to the regret bound.

Randomized Buffer Update In practical online learning scenarios, the assumption of an *i.i.d.* online stream is often impractical, since the data can be dependent. Furthermore, uniform sampling from an online stream is not straightforward, making it challenging to achieve the bound mentioned above when the history examples are not readily available in a memory. To address this, we use buffer update strategies that force data independence in the buffer. Stream oblivious methods are particularly useful, as they separate the randomness of the data from the buffer construction. To ensure an effective buffer update and maintain the desired representation, we adopt reservoir sampling in conjunction with the clustering strategy. This approach treats each partition stream independently. When a new example arrives in cluster $\mathcal{C}_j^t$, the old example is replaced with a probability of $1/|\mathcal{C}_j^t|$, making it difficult to establish a uniform distribution between each cluster. Finally, the bound in Theorem 2 holds with the assumption of model-buffer independence.

Proof 3 *Starting from equation 2.8 with the fact that the cluster-based buffer loss is unbiased of true local loss;*

$$L(w_{t-1}) - L(\bar{w}^*) \leq \frac{\|w_{t-1} - \bar{w}^*\|^2 - \|w_t - \bar{w}^*\|^2}{2\eta} + \frac{\eta\mathbb{E}\|u_t\|^2}{2},$$

and using the M-smoothness of the function g i.e. $L(w_t) \leq L(w_{t-1}) + \mathbb{E}u_t^T(w_t - w_{t-1}) + \frac{M}{2}\|w_t - w_{t-1}\|^2$, *and the update* $w_t = w_{t-1} - \eta_t v_t$,

$$\mathbb{E}L(w_t) \leq \mathbb{E}L(w_{t-1}) - \eta\|\mathbb{E}u_t\|^2 + \frac{\eta^2 M}{2}\mathbb{E}\|u_t\|^2. \tag{2.17}$$

By combining the last two inequalities and considering that the expectation is with respect to uniform sampling, we obtain the following inequality.

$$L(w_t) - L(\bar{w}^*) \leq \frac{\|w_{t-1} - \bar{w}^*\|^2 - \|w_t - \bar{w}^*\|^2}{2\eta} + (\frac{\eta}{2} + \frac{\eta^2 M}{2})\mathbb{E}\|u_t\|^2 - \eta\|\mathbb{E}u_t\|^2.$$

Using the fact that $\mathbb{V}(u_t) = \mathbb{E}\|u_t\|^2 - \|\mathbb{E}u_t\|^2$ *and choosing* $\eta = (0, \frac{1}{M}]$, *we have:*

$$L(w_t) - L(\bar{w}^*) \leq \frac{\|w_{t-1} - \bar{w}^*\|^2 - \|w_t - \bar{w}^*\|^2}{2\eta} + \eta\mathbb{V}(u_t).$$

Finally, summing from $t = 2$ *to* $t = T$ *and setting* $w_1 = 0$ *would complete the proof.*

It is worth noting that our analysis remains valid in both scenarios: the FIFO buffer update, which requires independent examples, and the randomized update of the buffer which does not require online independent examples. Although there is a coupling between the model w_t and the buffer B_t, as the model w_{t-1} incorporates information from the buffer in the previous step, we can still maintain the validity of the analysis considering that this coupling is limited, as demonstrated in previous research (e.g., [100]). Although the gradient is not an unbiased statistic due to this coupling, we argue that the impact on the analysis is minimal.

Moreover, it is important to note that the main difference lies in the buffer size. In the case of coupling, the buffer size must be at least $\log T$, as determined through rigorous analysis using techniques such as Rademacher complexity or covering number (for more details, see [52] and [90]). These analyzes provide a deeper understanding of the underlying mechanisms and further support the validity of our approach.

2.5.2 Regret of RFFs Approximation

The kernel associated with the pairwise hypothesis in space $\mathcal{H}$ is a function defined as $k : \mathcal{X}^2 \times \mathcal{X}^2 \mapsto \mathbb{R}^+$ with a shorthand $k_{(x,x')}(\cdot) := k((x, x'), (\cdot, \cdot))$ and can be constructed given any uni-variate kernel $\mathcal{G}$ for any $x_1, x_2, x'_1, x'_2 \in \mathcal{X}$ as follows:

$$\begin{aligned} k(x_1, x_2, x'_1, x'_2) &= \mathcal{G}(x_1, x'_1) + \mathcal{G}(x_2, x'_2) - \mathcal{G}(x_1, x'_2) - \mathcal{G}(x_2, x'_1) \\ &= \langle \mathcal{G}_{x_1} - \mathcal{G}_{x_2}, \mathcal{G}_{x'_1} - \mathcal{G}_{x'_2} \rangle_{\mathcal{G}}. \end{aligned} \tag{2.18}$$

It is clear that the pairwise kernel k defined above is positive semi-definite on $\mathcal{X}^2$, and therefore it is Mercel kernel if $\mathcal{G}$ does on $\mathcal{X}$ (e.g., see [98]). We further assume that there exists a lower-dimensional mapping $r : \mathcal{X} \mapsto R^D$, such that $\mathcal{G}_x(\cdot) \approx r(x)^T r(\cdot)$.

The quality of the approximation of the pointwise kernel $\mathcal{G}$ by random Fourier features is studied in the literature (see [76],[7],[64]), however, the approximation of pairwise kernel $k(\cdot)$ needs further analysis. Let the kernel function $\mathcal{G}(\cdot)$ be shift invariant and positive definite, thus using Bochner's theorem, it can be represented by the inverse Fourier transform of a non-negative measure p as $\mathcal{G}(x, x') = \mathcal{G}(x - x') = \int p(q) e^{iq^T(x-x')} du$ where $i = \sqrt{-1}$. For example, if the kernel is the Gaussian kernel, the measure is found by the Fourier transform to be $p \propto \mathcal{N}(\mathbf{0}, diag(\sigma))$, where $\sigma \in \mathbb{R}^d$ is the kernel width. In other words, the kernel $\mathcal{G}$ can be approximated using the Monte Carlo method, denoted as $\hat{\mathcal{G}}$, as follows:

$$\hat{\mathcal{G}}(x, x') = \frac{2}{D} \sum_{i=1}^{D/2} \cos(q_i^T (x - x')) = \langle r(x), r(x') \rangle, \tag{2.19}$$

where $q_i \sim \mathcal{N}(\mathbf{0}, diag(\sigma))$, and $r(x) := \frac{1}{\sqrt{D/2}} [cos(q_i^T x), sin(q_i^T, x)]_{i=1}^{D/2}$. The following theorem bounds the random Fourier error in equation (2.14).

Theorem 3 *Given a pairwise Mercer kernel $k_{(x,x')} := k((x, x'), (\cdot, \cdot))$ defined on $\mathcal{X}^2 \times \mathcal{X}^2$. Let $\ell(w, z, z')$ be a convex loss that is Lipschitz smooth with constant M. Then for any $w^* = \sum_{i,j \neq i}^T a^*_{i,j} k_{(x_i, x_j)}$, and random Fourier feature number D we have the following with probability at least $1 - 2^8 \left(\frac{\sigma R}{\delta}\right) \exp(-D\delta^2/(4d + 8))$,*

$$\sum_{t=2}^T L_t(\bar{w}^*) - \sum_{t=2}^T L_t(w^*) \leq \frac{M}{2} T \|w^*\|_1^2 \delta^2, \tag{2.20}$$

where $\|w^*\|_1 = \sum_{i,j\neq i}^{T} |a^*_{i,j}|$.

Remark 4 *Note that* $\|w^*\|_1$ *is controlled by regularization, that is, if there exists more than one optimal solution, then the optimal solution has minimal* $\|w^*\|_1$.

The study in [7] assumes that the true and approximated kernel functions belong to $L^2(\rho)$ i.e. space of square integrable functions (under the assumption 3), with the space $\mathcal{H}$ being dense in $L^2(\rho)$. Before proving the theorem, the following Corollary bounds the error of the pairwise kernel using the main theorem in [76].

Corollary 1 *Given* $x_1, x_2, x'_1, x'_2 \in \mathcal{X}$, *and pairwise kernel* k *defined on* $\mathcal{X}^2 \times \mathcal{X}^2$, *the random Fourier estimation of the kernel has a mistake bounded with probability at least* $1 - 8\exp\frac{-D^2\delta}{2}$ *as follows:*

$$|\hat{k}_{(x_1,x_2)}(x'_1, x'_2) - k_{(x_1,x_2)}(x'_1, x'_2)| \leq \delta.$$

The proof follows from claim 1 in [76] and the definitions of pairwise kernel k, which has four sources of errors.

Proof 4

$$\begin{aligned}
\sum_{t=2}^{T} L_t(\bar{w}^*) - \sum_{t=2}^{T} L_t(w^*) &= \sum_{t=2}^{T} \frac{1}{t-1} \sum_{i=1}^{t-1} \ell(\bar{w}^*, z_t, z_i) - \sum_{t=2}^{T} \frac{1}{t-1} \sum_{i=1}^{t-1} \ell(w^*, z_t, z_i) \\
&= \sum_{t=2}^{T} \frac{1}{t-1} \sum_{i=1}^{t-1} \ell(\bar{w}^*, z_t, z_i) - \ell(w^*, z_t, z_i) \\
&\leq \sum_{t=2}^{T} \frac{1}{t-1} \sum_{i=1}^{t-1} \frac{M}{2} \|\bar{w}^* - w^*\|_2^2, \qquad (2.21)
\end{aligned}$$

where last inequality applies assumption 1, and that fact that $\nabla\ell(w^*, z, z') = 0$ *for any* $z, z' \in \mathcal{Z}$. *Using the Representer theorem and the fact that the space* $\mathcal{H}$ *and* $\hat{\mathcal{H}}$ *are dense in* $L^2(\rho)$ *(space of squared integrable function, under assumption 3). Hence, we can approximate any function in* $\mathcal{H}$ *by a function in* $L^2(\rho)$, *i.e. without loss of generality we assume that* $\bar{w}^* = \sum_{j=1,k\neq j}^{t-1} \alpha^*_{j,k} \hat{k}_{z_t,z_i}$, *then we have* $\|\bar{w}^* - w^*\|_2^2 \leq \|\sum_{j=1,k\neq j}^{t-1} \alpha^*_{j,k}(\hat{k}_{z_t,z_i} - k_{z_t,z_i})\|^2_{L^2(\rho)}$ *using the fact that* $\|\cdot\|_{L^2(\rho)} \geq \|\cdot\|_2$, *finally using the triangle inequality we have,*

$$\begin{aligned}
\sum_{t=2}^{T} L_t(\bar{w}^*) - \sum_{t=2}^{T} L_t(w^*) &\leq \sum_{t=2}^{T} \sup_{x_i,x_t\in\mathcal{X}} \frac{M}{2} \sum_{j=1,k\neq j}^{t-1} \|\alpha^*_{j,k}(\hat{k}_{z_t,z_i} - k_{z_t,z_i})\|^2_{L^2(\rho)} \\
&\overset{a}{\leq} \sum_{t=2}^{T} \frac{M}{2} \sum_{j=1,k\neq j}^{t-1} (\alpha^*_{j,k})^2 \delta^2 \\
&\overset{b}{\leq} \sum_{t=2}^{T} \frac{M}{2} \delta^2 (\sum_{j=1,k\neq j}^{t-1} |(\alpha^*_{j,k})|)^2 \\
&= \frac{M}{2} T \|w^*\|_1^2 \delta^2, \qquad (2.22)
\end{aligned}$$

where inequality (a) implements Corollary 1, inequality (b) use the fact that sum of squares is less than the square of sum, and last equality assumes $\|w^*\|_1 = \sum_{i,j\neq i}^{T} |a^*_{i,j}|$. This completes the proof.

2.6 Related Work

Pairwise scalability poses a challenge in pairwise learning due to the quadratic growth of the problem with the number of samples. To address this issue, researchers have proposed different approaches in the literature. Some examples include offline doubly stochastic mini-batch learning [18, 33, 2, 3], online buffer learning [100, 52, 94], second-order statistic online learning [28], kernelized learning [43, 98], and saddle point-problem methods [97, 77]. Online gradient descent, while having a time complexity of $O(T^2)$ [10, 28], is impractical for large-scale problems. It pairs a data point (x_t, y_t) received at time t with all previous samples $\{x_{t'}, y_{t'} | 1 < t' < t-1\}$ to calculate the true loss. However, computing the gradients for all received training examples, which increases linearly with t, poses a significant challenge. To address this, the work in [100] introduced two buffers, B_+ and B_-, of sizes N_+ and N_-, respectively, using Reservoir Sampling to maintain a uniform sample from the original dataset. Although this approach provides a sub-linear regret bound dependent on buffer sizes, it is limited to AUC maximization with linear models ($\mathcal{H} = \mathbb{R}^d$) and overlooks the effect of buffer size on generalization error. Researchers have also explored the application of saddle point-problem methods to tackle pairwise learning tasks involving metrics such as AUC [97]. By formulating the problem as a saddle point problem and utilizing typical saddle point solvers, this approach achieves a time complexity of $O(T)$ in terms of gradient computations, providing an efficient solution for pairwise learning with reduced computational requirements.

The study by [52] introduces RSx, which replaces buffer samples with incoming data using s Bernoulli processes in $1/t$, ensuring s independent data points. However, achieving optimal generalization in terms of buffer loss requires a prohibitive buffer size of $O(\sqrt{T})$. Conversely, [94] argues for a buffer unity size, but their proof relies on impractical independent data streams and is limited to linear models. The common challenge in previous approaches is the need for *i.i.d.* buffer examples for uniform convergence analysis and model-buffer decoupling. For example, [52] addresses this with modified reservoir sampling, but still requires a buffer size of $O(\sqrt{T})$. Moreover, buffer-based loss becomes less informative as buffer updates become increasingly rare over time. These existing approaches suffer from limitations in computational efficiency, applicability to non-linear models, and a lack of explicit study of the impact of buffer variance on generalization error.

2.7 Experiments

We perform experiments on several real-world datasets, and compare our algorithm to both offline and online pairwise algorithms. Specifically, the proposed method is compared with different algorithms of AUC maximization, with the squared function as the surrogate loss.

Table 2.2: AUC maximization results (average ± standard error)$\times 10^2$ using different batch and online algorithms on different datasets

Dataset	**FPOGD**	**SP-NET**	**OGD**	**S. Kernel**	**Proj++**	**Kar**
diabetes	81.91±0.48	82.03±0.32	82.53±0.31	82.64±0.37	77.92±1.44	79.85±0.28
ijcnn1	**92.32**±0.77	87.01±0.10	83.46±1.25	71.13±0.59	92.20±0.27	83.44±1.21
a9a	**90.03**±0.41	90.02±0.08	88.41±0.42	84.20±0.17	84.42±0.33	77.93±1.55
mnist	**92.85**±0.25	88.57±0.54	88.65±0.34	89.21±0.15	89.82±0.15	84.16±0.15
rcv1	**99.38**±0.20	98.13±0.15	99.05±0.57	96.26±0.35	94.54 ±0.36	97.78 ±0.64
usps	**95.02**±0.84	85.12±0.88	92.88±0.47	91.25±0.84	90.14 ±0.22	91.58±0.25
german	**85.82**±0.24	76.89±2.46	84.20±0.54	80.11±0.44	78.44 ±0.66	84.21±0.45
Reg.	l_2	$l_1 + l_2$	l_2	l_2	l_2	l_2

2.7.1 Experimental Setup

Compared Algorithms. The compared algorithms includes offline and online setting are,

- SPAM-NET [77] is an online algorithm for AUC with square loss that is transformed into a saddle point problem with non-smooth regularization.
- OGD [94], the most similar to our algorithm but with a linear model, that uses the last point every iteration.
- Sparse Kernel [49] is an offline algorithm for AUC maximization that uses the kernel trick.
- Projection ++ [43] is an online algorithm with adaptive support vector set.
- Kar [52] is an online algorithm with a randomized buffer update policy.

Datasets. The datasets used in this study are sourced from the LIBSVM website [14]. Table 2.3 provides an overview of the dataset statistics, including the dataset name, size, feature dimension, and the ratio of negative to positive examples. Non-binary datasets undergo a conversion process into binary by evenly dividing the labels.

Implementation. The experiments were validated for all algorithms through a grid search on the hyper-parameters, employing a three-fold cross-validation. For example, in each algorithm, the step size, denoted as η, was varied within the range of $2^{[-8:-1]}$, providing flexibility for fine-tuning. Similarly, the regularization parameters, represented by λ, were explored in the range of $10^{[-8:-1]}$. In the case of the SPAM-NET algorithm, the elastic-net regularization parameter, denoted as λ_2, was determined by a grid search with values ranging from 10^{-8} to 10^{-1}. To ensure a fair comparison, the use of kernelization is excluded when comparing with linear algorithms. All algorithms were executed five times on different folds using Python, running on a CPU with a speed of 4 GHz and 16 GB of memory.

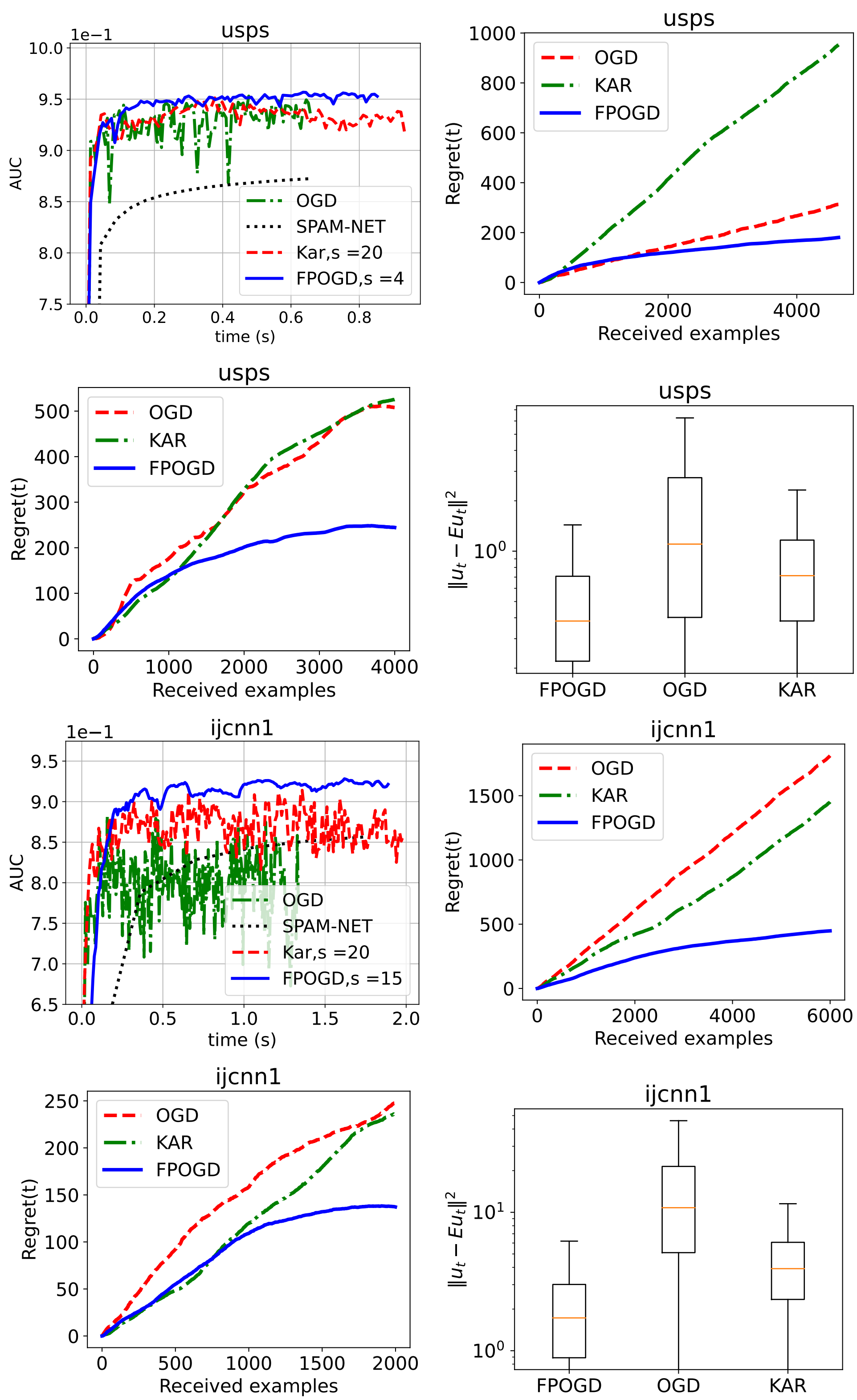


Figure 2.2: The first column presents AUC vs. time. Columns two and three display regret comparisons, with column two focused on $i.i.d.$ datasets and column three on $non-i.i.d.$ datasets. The fourth column provides gradient variance using a 4-size buffer

2.7.2 Experimental Results and Analysis

The effectiveness of our random Fourier pairwise online gradient descent procedure in maximizing the area under the curve (AUC) is confirmed by our results obtained with a squared loss function, as illustrated in Figure 2.2. Table 2.2 clearly shows that our algorithm outperforms both online and offline linear and nonlinear pairwise learning algorithms, yielding enhanced AUC performance, particularly on large-scale datasets. Furthermore, in next section, we provide experimental results that demonstrate the relationship between the number of allowed clusters and the convergence of the algorithm. Additionally, we investigate the impact of the number of random features on the algorithms performance.

2.7.3 Additional Experiments

Table 2.3: Datasets used in the experiments, where N_-/N_+ is the ratio of negative to positive examples.

Dataset	Size	Features	N_-/N_+
diabetes	768	8	34.90
ijcnn1	141,691	22	9.45
a9a	32,561	123	3.15
MNIST	60,000	784	1.0
covtype	581012	54	1.0
rcv1.binary	20,242	47,236	0.93
usps	9,298	256	1.0
german	20,242	24	2.3

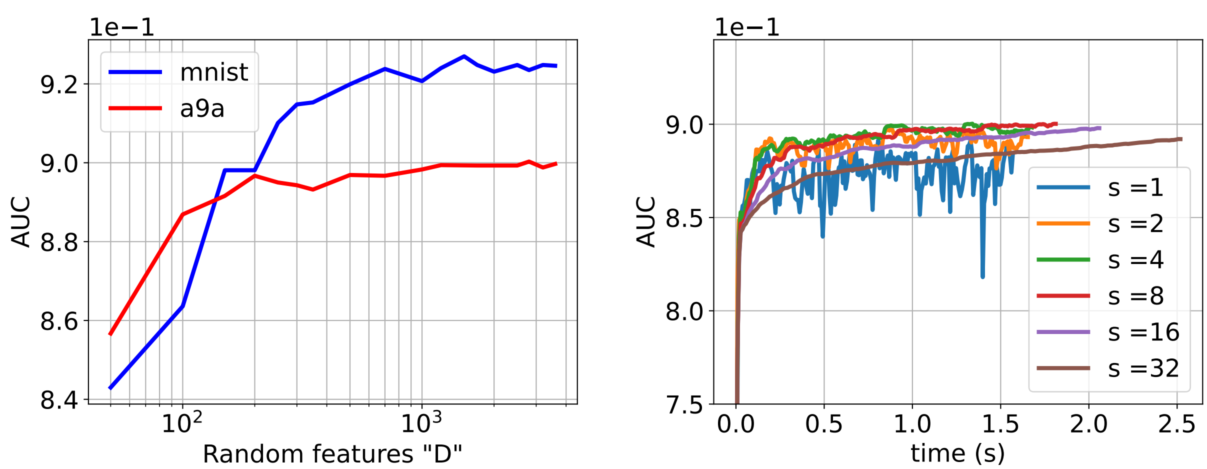


Figure 2.3: On left, AUC versus number of random features D used to approximate the kernel, for a9a and MNIST datasets. On right, AUC versus maximum number of clusters "s" in algorithm 1 for the dataset "a9a".

2.7.4 Number of Clusters

In our algorithm, the number of clusters is determined by the hyper-parameter epsilon. To manage memory requirements and computational costs, we enforce a maximum limit on the number of clusters in our experiments, even if the epsilon theoretically allows for more clusters. This strategic limitation not only ensures resource efficiency, but also contributes to practical applicability.

In the figure below, we shed light on the influence of varying cluster limits on the AUC score, specifically focusing on the "a9a" dataset while using a small epsilon value. The findings reveal that as the number of clusters increases, the AUC score shows a gradual ascent, taking more time to reach its maximum value. Importantly, these results also underscore that, by restricting the number of clusters, we achieve a noticeable reduction in variance while still maintaining a commendable level of performance. This balance between performance and resource management is crucial for the practical implementation of our method.

2.7.5 Number of Random Feature

By utilizing the Mercer decomposition theorem and the properties of eigenvalues, we can derive bounds on the number of random Fourier features needed for different decay rates of the eigenvalues. Specifically, for a decay rate of $1/i$, the sufficient number of features is $D \geq 5T \log 2T$. For a decay rate of R^2/i^{2c}, the sufficient number of features is $D \geq T^{1/2c} \log T$. And for a geometric decay rate of r^i $(r > 1)$, the sufficient number of features is $D \geq \log^2 T$.

In our experiments, we focus on a Gaussian kernel with a constant width of $1/d$, where d is the dimension of the input space. For this kernel, the required number of random features is $O(\sqrt{T} \log(T))$. Figure 1 illustrates the results of our experiments, which show that $D = O(\sqrt{T} \log(T))$ is sufficient for a good approximation of the kernel, as the AUC does not improve significantly beyond this point.

2.7.6 Additional Datasets

In this section, we present a series of figures that highlight the superior performance of our method: Figure 2.6 demonstrates the efficiency and effectiveness of our approach by showcasing the Area Under the Curve (AUC) in relation to time. Figure 2.4 provides information on the gradient variance analysis using a 4-size buffer, highlighting the stability of our algorithm. Finally, Figure 2.5 compares regret values on $i.i.d.$ and $non-i.i.d.$ datasets, showcasing the adaptability and robustness of our method across different data distribution scenarios.

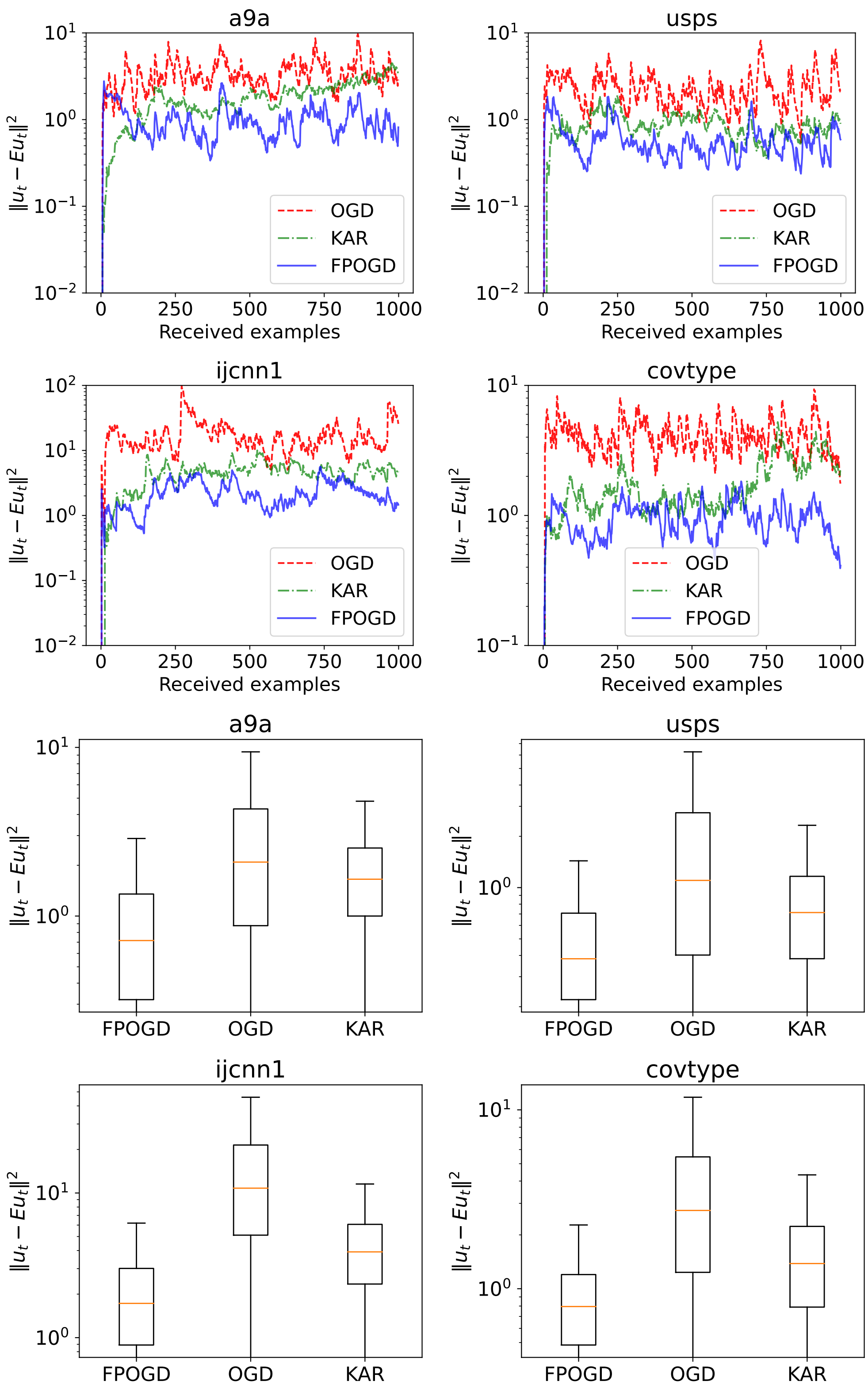


Figure 2.4: This figure investigates gradient variance using a 4-size buffer across four datasets. The top row shows running variances, while the bottom row presents logarithmic box plots. "Received examples" are online inputs at each time step, with u_t as the online stochastic gradient, and Eu_t as the expected gradient.

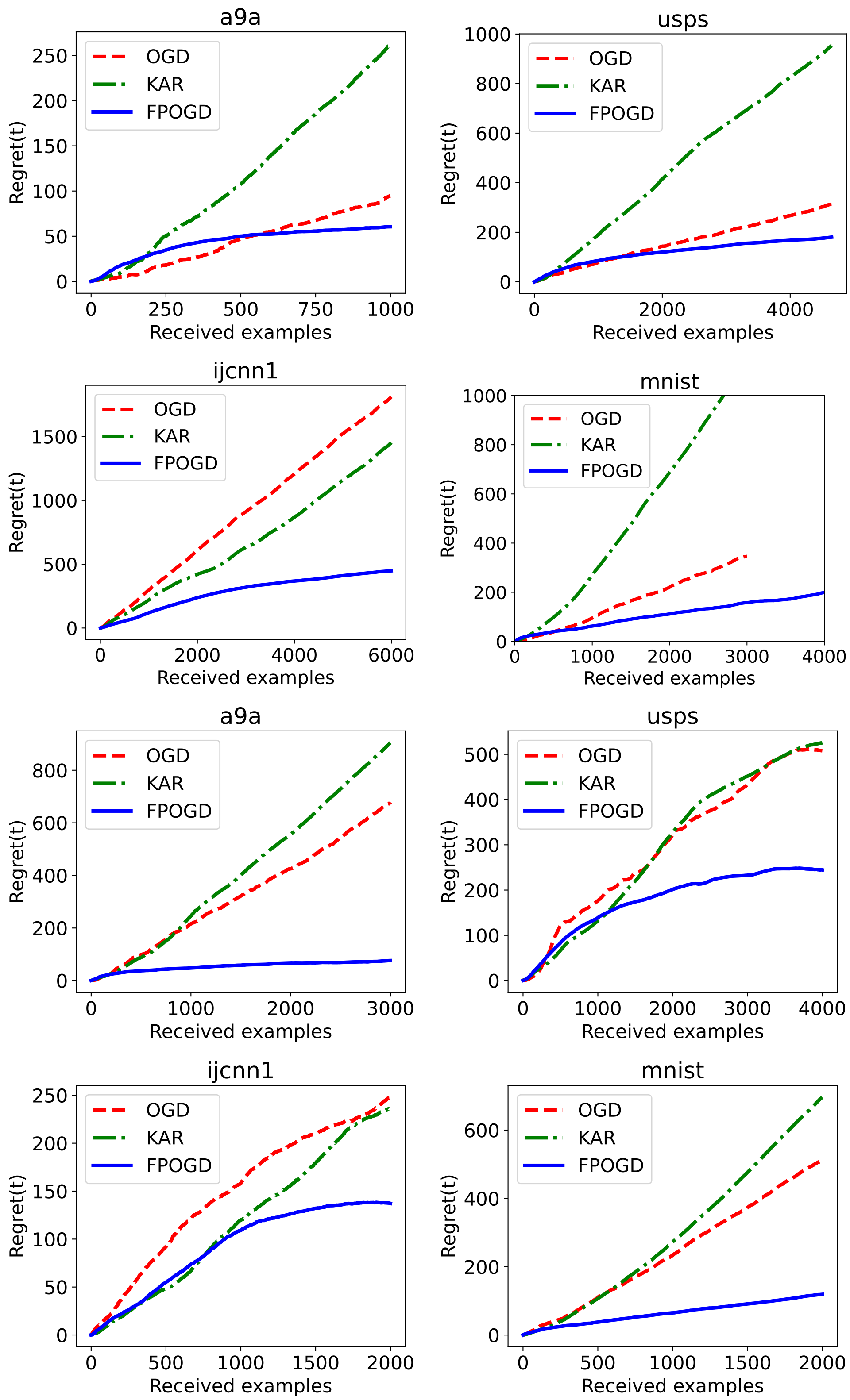


Figure 2.5: This figure illustrates regret, defined as $\sum_t L_t(w_t) - L_t(w^*)$. The first row represents various algorithms applied to the *i.i.d.* datasets, while the second row features show the $non-i.i.d.$ datasets generated by sorting examples based on a single feature.

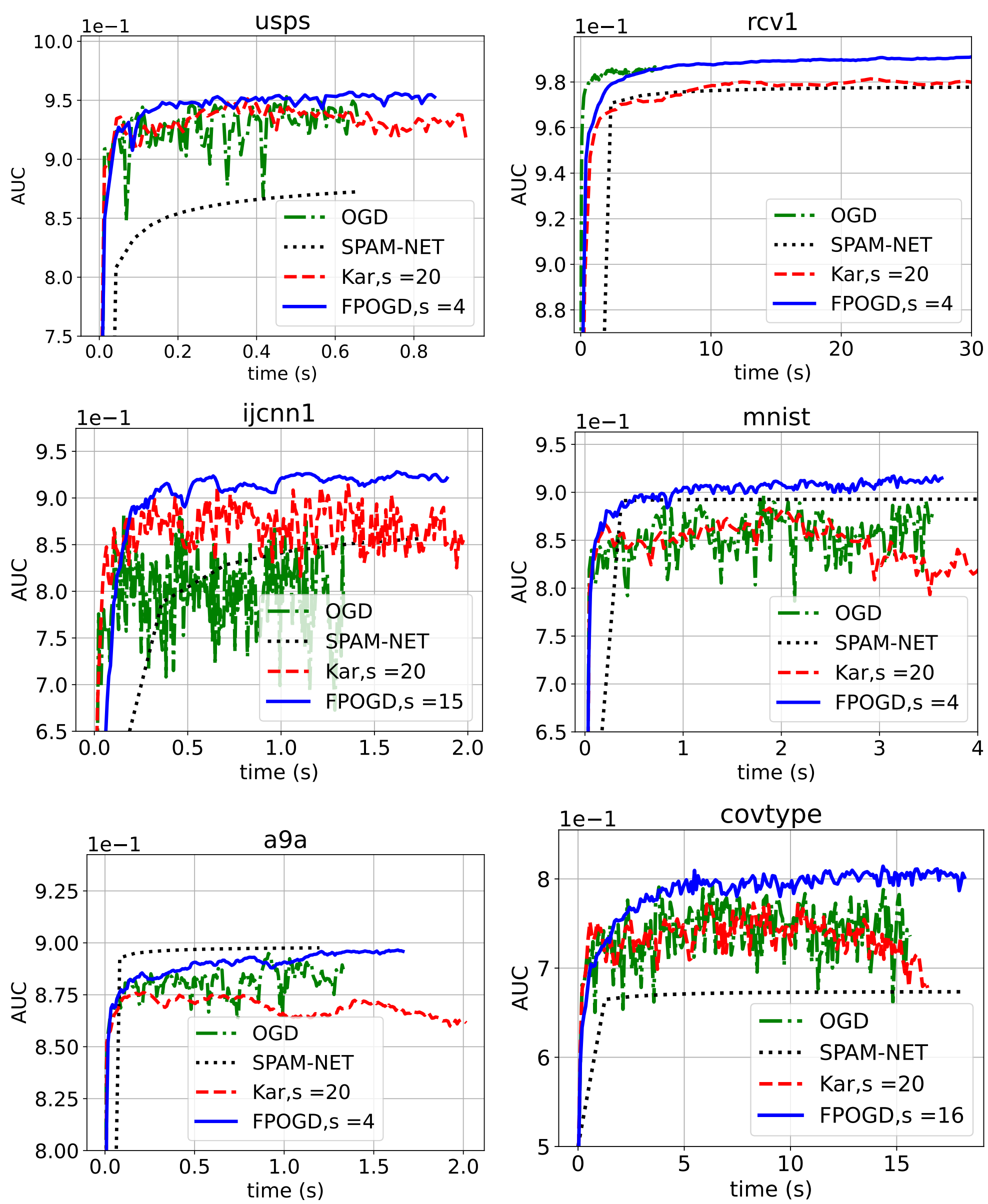


Figure 2.6: The AUC vs. time comparison of the algorithms in different datasets show superior performance of the proposed method.

2.8 Summary

In this chapter, we introduce a lightweight online kernelized pairwise learning algorithm. Our approach involves maintaining an online clustering mechanism and using it to calculate the online gradient based on the current received sample. Additionally, we approximate the kernel function using a random Fourier mapping. As a result, our algorithm achieves a gradient complexity of $O(sT)$ for linear models and $O(\frac{D}{d}sT)$ for non-linear models, where T represents the number of received examples and D denotes the number of random features.

Chapters 4 and 5 take a pivotal turn, applying the principles of pairwise learning within the innovative framework of SNNs. We dive into the theoretical and experimental aspects of SNN robustness under normal loss conditions (e.g., cross-entropy) that echo the overarching theme of enhancing learning methodologies. The insights garnered here not only augment our understanding of SNN encoding mechanisms, but also illuminate potential pathways for integrating these robustness principles into pairwise learning scenarios, suggesting a broader applicability of our findings within and beyond the confines of pairwise learning paradigms.

Chapter 3

Limited Memory Online Gradient Descent for Kernelized Pairwise Learning with Dynamic Averaging

3.1 Introduction

The online buffer techniques introduced so far have not adequately tackled the sensitivity of generalization to *non-i.i.d.* data, particularly in adversarial environments, constraining their capacity to grasp the intricacies inherent in real-world scenarios. Transitioning to the examination of generalization bounds, prior research extensively investigated online pairwise gradient descent with buffers and linear models [90, 52]. These studies establish a bound of $O(1/s + 1/\sqrt{T})$ for this approach. However, it is imperative to acknowledge that this bound attains optimality primarily when the buffer size s approximates $O(\sqrt{T})$. This aspect poses challenges in scenarios where a smaller buffer size is preferred.

Collectively, these vulnerabilities underscore the need for more research and advancement in the field of online pairwise learning. This effort should encompass addressing the limitations of *i.i.d.* frameworks, conducting a more comprehensive exploration of non-linear methods, and surmounting the computational obstacles tied to kernel computation. Our method expands the scope of online pairwise learning to accommodate non-linear data by incorporating kernelization into the input space. We examine the ramifications of *non-i.i.d.* data on regret by evaluating the gradient of both the moving average and a random sample drawn from the past history. Leveraging random Fourier features, we efficiently estimate the kernel while maintaining a sublinear error bound, thus achieving computational efficiency without compromising performance. Through the integration of kernelization, kernel approximation, and dynamic averaging, our approach transcends linear constraints, effectively addresses *non-i.i.d.* data challenges, and mitigates kernel complexity. As a result, we establish a robust online pairwise learning paradigm (as illustrated in Table 3.1). Our principal contributions can be summarized as follows:

- We introduce an online pairwise algorithm for non-linear models, characterized by

Algorithm	Problem	Model	Scheme	i.i.d.	Time Comp.	Space Comp.
[73]	AUC	Linear	Online	Yes	$O(T)$	$O(d)$
[97]	AUC	Linear	Online	Yes	$O(T)$	$O(d)$
[100]	AUC	Linear	Online	Yes	$O(sT)$	$O(ds)$
[28]	AUC	Linear	Online	Yes	$O(T)$	$O(d^2)$
[33]	AUC	Linear	Offline	Yes	$O(T)$	$O(d)$
[94]	General	Linear	Online	Yes	$O(T)$	$O(d)$
[66]	General	kernel	Online	No	$O(T^2)$	$O(dT)$
[49]	AUC	Kernel	Offline	Yes	$O(T \log T)$	$O(dT^2)$
[52]	General	Linear	Online	Yes	$O(sT)$	$O(ds)$
AOGD	General	Kernel	Online	No	$O(\frac{D}{d}T)$	$O(D)$

Table 3.1: Recent pairwise learning algorithms (T is the iteration number, d: the dimension, D is random features, and s is a buffer size), note the time complexity is w.r.t. gradients computations.

the ability to handle *non-i.i.d.* data. In scenarios with a wide margin in the kernel space, our algorithm achieves a sublinear regret with a buffer size of $O(1)$.

- We confront the influence of *non-i.i.d.* data on regret and introduce a random Bernoulli sampling strategy to manage and improve the regret rate.
- In the case of the Gaussian kernel, we approximate the pairwise kernel function utilizing a mere $O(\sqrt{T} \log T)$ feature, a significant reduction compared to $O(T)$ feature in the literature, while keeping a sub-linear regret bound.
- We validate the effectiveness of our proposed methodology across a range of real-world datasets and compare its performance with state-of-the-art methods for AUC maximization.

The subsequent sections are structured as follows: Section 3.2 delineates the problem setting, Section 3.3 explains our proposed method, Section 3.4 presents the analysis of regret, Section 3.5 delves into related work, followed by Section 3.6 presents experimental results, and finally Section 3.7 concludes the chapter.

3.2 Problem Setting

In online learning with pairwise losses, after receiving a new data point z_t, a localized error is used as a performance measure [99, 10]. This error represents the pairing of the new data point with all preceding $t-1$ points. The properties of this localized error depend on the selected pairwise loss function, illustrated as:

$$L_t(w_{t-1}) = \frac{1}{t-1} \sum_{i=1}^{t-1} \ell(w_{t-1}, z_t, z_i), \tag{3.1}$$

where $w \in \mathcal{H}$ represents a linear function mapping, i.e. $f_w(x) = \langle w, x \rangle$. To address memory constraints, [100] introduced a buffer-based mechanism for local error computation denoted as $\bar{L}_t(w_{t-1})$, as explained in Equation 3.2. At each iteration t, the buffer, designated as B_t, accommodates a finite set of historical-example indices, and the size of this buffer is $|B_t|$ (or s in the literature).

$$\bar{L}_t(w_{t-1}) = \frac{1}{|B_t|} \sum_{i \in B_t} \ell(w_{t-1}, z_t, z_i). \tag{3.2}$$

This buffer plays an important role in the learning journey, its contents are updated at each step through diverse strategies that range from randomized methodologies such as reservoir sampling [100, 52] to non-randomized approaches exemplified by FIFO [94]. However, despite the wide adoption of these sampling techniques, there exists a discernible research void in understanding their implications within the context of *non-i.i.d.* online examples. It is imperative to highlight that the issue of buffer size and its update methods becomes pronounced when dealing with adversarial environment scenarios. The choice of buffer size and sampling strategy can significantly affect the model's performance and ability to generalize in the case of *non-i.i.d.* data streams.

In our pursuit of addressing the intricacies inherent in complex real-world data, our online pairwise approach takes into consideration the mapping of both the hypothesis and the data to a Reproducing Kernel Hilbert Space (RKHS), denoted as $\mathcal{H}$. The associated Mercer pairwise kernel function, denoted $k : \mathcal{X}^4 \mapsto \mathbb{R}$, adheres to the reproducing property $\langle k_{(x,x')}, g \rangle = g(x, x')$, where $x, x' \in \mathcal{X}^2$ and $g \in \mathcal{H}$. In particular, for scenarios involving pointwise hypotheses with pairwise losses, such as AUC loss, the kernel function is streamlined to $k : \mathcal{X}^2 \mapsto \mathbb{R}$. The expanse of $\mathcal{H}$ encompasses a spectrum of linear combinations encompassing functional mappings $k_{(x,x')} | (x, x') \in \mathcal{X}^2$ and their limit points.

In our quest to circumvent the computational intricacies associated with kernelization in the online realm, we employ the resourceful apparatus of random Fourier features (RFF) as a pragmatic approximation for the Mercer kernel function. RFF introduces a lower-dimensional mapping $r(\cdot)$, which serves as an approximation of the kernel function, denoted as $\bar{k}(\cdot)$. This approximation empowers us to execute computations through linear operations, culminating in a significant reduction in computational overhead. The domain defined by the novel kernel functions finds its place in the realm of $\bar{\mathcal{H}}$. It is worth mentioning that earlier investigations have explored the precision of random Fourier approximation within pointwise and offline settings. As we navigate the online landscape, the minimum number of random features necessary to ensure sublinear regret has been determined to be $O(T)$. Our method introduces a novel error bound for pairwise predicaments, based on the utilization of merely $O(\sqrt{T} \log T)$ random features (elucidated in Section 5 for comprehensive details).

3.2.1 Assumptions

In addition to the assumptions introduced in chapter 2, we add the following chapter regarding the lipschitz continuity.

Assumption 4 (Lipschitz Continuity) *Assume for any $z, z' \in \mathcal{Z}$, the loss function $\ell(\cdot, z, z')$ is G-Lipschitz continuous, i.e. $\forall w, w' \in \mathcal{H}$,*

$$|\ell(w, z, z') - \ell(w', z, z')| \leq G \|w - w'\|_2 .$$

3.3 Methodology

In light of the core problem we aim to address, namely the computational complexity arising from the necessity to accommodate all preceding examples in the local error computation, our proposed methodology centers around mitigating this challenge. We begin by suggesting the utilization of a moving average of previous examples, thus alleviating the burden of handling the complete historical dataset.

However, it is imperative to note that the simple average approach may lead to misleading outcomes, especially when the data exhibit high variance distributions. Recognizing this potential limitation, we introduce a corrective measure to enhance the reliability of the average-based gradient. This correction mechanism involves introducing a random point from the historical loss, enhancing the robustness of the gradient estimation, and ameliorating the impact of potential average imbalances. This comprehensive strategy effectively addresses the issues associated with handling the entire historical dataset, while also accounting for data distribution complexities that can skew results and, most importantly, does not require the *i.i.d.* assumption. Another parallel research that addresses the high variance in the buffer has recently been proposed [5], however, with $O(s)$ complexity.

Indeed, the loss function we seek to minimize can be formulated as a linear combination of two distinct losses: one based on the moving average and the other on a randomly chosen example. However, it is important to recognize that minimization of this combined loss does not inherently guarantee minimization of true local loss. This discrepancy arises because, at the optimal solution where the gradient is zero, the gradients of both components remain nonzero, depending upon the linear coefficient constant.

To navigate this challenge, an alternate strategy is studied to minimize both losses. In essence, we proceed by iteratively taking steps using the average-based gradient and then employing the random gradient on the transition model derived from the previous step. This alternating optimization strategy accounts for the intricate balance between the two gradient components and fosters a dynamic convergence process. Furthermore, the iterative method ensures that the gradients based on average and random are adequately taken into account in the optimization process. Loss minimization after receiving new examples z_t can be formulated using the average $\bar{z} = (\sum_{i=1}^{t-1} \frac{x_i}{t-1}, y_{t-1} | y_{t-1} \neq y_t)$ and the random sample $\hat{z}$ from $\{z_1, \ldots, z_{t-1}\}$, as follows:

$$\min_w \ell\left(w - \eta_t \nabla \ell(w, z_t, \bar{z}), z_t, \hat{z}\right), \tag{3.3}$$

where η_t is the step size. As detailed in the subsequent section, the magnitude of the step taken in the second update is contingent on both the variance of the data and the smoothness of the loss function. Interestingly, when the variance is negligible, the step taken based on the average gradient is deemed adequate, rendering the random step unnecessary. The objective function can make more sense if we use the second-order Taylor expansion after denoting $\hat{\ell}(\cdot) = \ell(\cdot, z_t, \hat{z})$ and $\bar{\ell}(\cdot) = \ell(\cdot, z_t, \bar{z})$ as follows:

$$\ell(w - \eta_t \nabla \ell(w, z_t, \bar{z}), z_t, \hat{z}) \approx \hat{\ell}(w) - \eta_t \nabla \hat{\ell}(w)^T \nabla \bar{\ell}(w) + \eta_t^2 \nabla \bar{\ell}(w)^T \nabla^2 \hat{\ell}(w) \nabla \bar{\ell}(w). \tag{3.4}$$

Hence, the process of minimizing the objective function involves addressing three integral components. First, we endeavor to minimize the loss associated with randomly selected examples. Then, leveraging the convex nature of the loss function and the positive definiteness of its Hessian, the third term—strictly non-negative in nature—ensures that the gradient of the average-based loss is minimized. Of paramount significance is the role of the second term, which guarantees that the gradients of losses, both average-based and random-based, exhibit a maximum inner product. This intricate interplay is crucial to correcting the moving average gradient, particularly in scenarios involving highly skewed distributions.

To handle the intricate patterns of real-world data, our approach to online pairwise learning involves a fundamental assumption: Both the hypotheses and the data are transformed into a specialized space known as a Hilbert space, denoted as $\mathcal{H}$. This space is equipped with a specific type of kernel, called a Mercer kernel. Essentially, this kernel is a mathematical function that captures the relationships between the data points. This Hilbert space has two key attributes: First, it can reproduce the values of the kernel function, and second, it encompasses various combinations of these kernel functions along with their complete structures.

When we apply traditional buffer-based pairwise algorithms in an online setting, the kernelization process comes with a complexity that grows as $O(T^2 s)$, where s represents the size of the buffer—a rather unwieldy task for larger problems. To address this challenge, we introduce a method that approximates the kernel function using a simpler approach. We assume the existence of a lower-dimensional mapping, represented as $r : \mathcal{X}^2 \mapsto R^D$. This mapping enables us to simplify the calculations of the kernel function, significantly reducing the complexity of the process. This simplified approach is achieved using what are known as random Fourier features, which are derived from the mathematical Fourier transform. Although these techniques have been studied in previous studies, our focus is on adapting them to the online setting, and we establish that as few as $O(\sqrt{T} \log T)$ random features are enough to achieve effective results. Our method is introduced in algorithm 2 for general pairwise learning, and it includes the Gaussian kernel approximation using random Fourier features. In the upcoming section, we dive into the performance of this online algorithm and dissect its regret into two distinct components: the first is a direct result of learning a model from an online stream of examples, and the second arises from the approximation of the kernel function.

3.4 Regret Analysis

The regret incurred by the online algorithm relative to the optimal hypothesis in the original space $\mathcal{H}$, denoted as $w^* = \arg\min_{w \in \mathcal{H}} \sum_{t=2}^{T} L_t(w)$, when applied to a sequence of

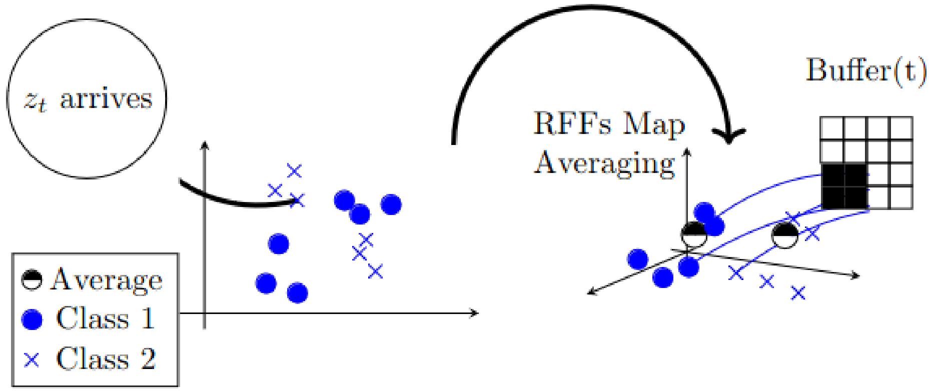


Figure 3.1: The incoming point z_t from the left are transformed through kernelization and RFFs to a new space. Subsequently, they are averaged and added to the buffer along with random point using either a FIFO strategy or with a probability-based approach.

T examples, is formally expressed as:

$$\mathcal{R}_{w^*,T} = \sum_{t=2}^{T} L_t(w_{t-1}) - \sum_{t=2}^{T} L_t(w^*). \tag{3.5}$$

In the context of this equation, the local all-pairs loss $L_t(\cdot)$ is defined according to Equation 3.1. To go deeper, we can decompose the regret represented in Equation 3.5 by introducing the best-in-class hypothesis within the approximated space $\bar{\mathcal{H}}$, denoted as $\bar{w}^* = \arg\min_{w \in \bar{\mathcal{H}}} \sum_{t=2}^{T} L_t(w)$. This decomposition takes the form:

$$\mathcal{R}_{w^*,T} = \underbrace{\sum_{t=2}^{T} L_t(w_{t-1}) - \sum_{t=2}^{T} L_t(\bar{w}^*)}_{T_1} + \underbrace{\sum_{t=2}^{T} L_t(\bar{w}^*) - \sum_{t=2}^{T} L_t(w^*)}_{T_2}. \tag{3.6}$$

In our analysis, we provide a bound for T_1 in Theorem 5, followed by an error bound for T_2 in Theorem 6. By combining these bounds, we derive the regret bound for Algorithm 2 as follows:

Theorem 4 *Let $\{z_t \in \mathcal{Z}\}_{t=1}^{T}$ be sequentially accessed by Algorithm 2. Let D denote the number of random Fourier features in the kernel mapping from the original space $\mathcal{X} \subset \mathbb{R}^d$. Let η be first step size, $\gamma = O(\Gamma M_t \eta)$ the second step size, and G the Lipschitz constant. Then the regret bound compared to $w^* = \arg\min_{w \in \mathcal{H}} \sum_{t=2}^{T} L_t(w)$ is bounded with a probability of at least $1 - 2^8 \left(\frac{\sigma Diam(\mathcal{X})}{\epsilon}\right) \exp(-D\epsilon^2/(4d+8))$, as follows:*

$$\mathcal{R}_{w,T} \leq GT\|w^*\|_1 \epsilon + \frac{\|\bar{w}^*\|^2}{2\eta} + 2\gamma G^2 T, \tag{3.7}$$

where ϵ denotes the kernel approximation error, σ signifies the kernel width, $\Gamma \geq tr(cov[z_t])$, and $M_t : M(w_t)$ stands for the smoothness parameter of ℓ in the neighborhood of w_t.

Algorithm 2 Average Online Gradient Descent (AOGD)

Require: Random feature size D, initial solution $w_1 \in \bar{\mathcal{H}}$, initial average and random example $\bar{z} = \hat{z} \in R^D$, kernel function k and its Fourier transform $P(u)$, probability p, step sizes η_t, γ_t and λ.

Sample Fourier feature $\{u_i\}_{i=1}^{D/2} \sim P(u_i)$

for $t = 2, \dots, T$ **do**

  Receive a new example $z_t = (x_t, y_t) \in \mathcal{Z}$

  Map to the new space:

$$z_t = (\frac{2}{\sqrt{D}}[\sin(u_i^T x_t), \cos(u_i^T x_t)]_{i=1}^{D/2}, y_t)$$

  Suffer loss $\bar{\ell}(w_{t-1}, z_t, \bar{z})$

  Update model $w'_t = w_{t-1} - \eta_t \nabla \bar{\ell}(w_{t-1}, z_t, \bar{z})$

  Suffer loss $\hat{\ell}(w'_t, z_t, \hat{z})$

  Update model $w_t = w'_t - \gamma_t \nabla \hat{\ell}(w'_t, z_t, \hat{z})$

  Update the moving average and the random example,

$$\bar{z} \leftarrow \frac{z * (t-1) + z_t}{t}$$
$$\hat{z} \leftarrow z_t \text{ if Bernoulli(p)} = 1$$

end for

return w_T

Remark 5 *Selecting $\eta = \frac{1}{\sqrt{T}}$ and $\epsilon = \frac{1}{\sqrt{T}}$ results in a sublinear bound, which is optimal. In particular, $D = O(T)$ in general, yet it can be as low as $\sqrt{T}\log T$ for specific kernels. Furthermore, since $\gamma = O(\Gamma M_t \eta)$, the regret depends on the variance of the data and the smoothness of the loss.*

3.4.1 Regret in the Approximated Space

Prior approaches (e.g., [52], [100]) aimed to attain uniform convergence of buffer loss often relied on the assumption of *i.i.d.* examples. These methods also require maintaining uniform samples from historical data through a Bernoulli process with diminishing probabilities that eventually approach zero. Moreover, achieving such uniform convergence required a disentanglement between the model and the buffer to ensure the validity of the expectation. For instance, while [100] employed Reservoir sampling to enforce a buffer *i.i.d.*, they overlooked the interaction between the model and the data. Similarly, [52] utilized modified reservoir sampling and Symmetrization of Expectations to address these challenges. However, their approach required a buffer size of $O(\sqrt{T})$, which is impractical for larger-scale scenarios. In contrast, [94] accomplished sub-linear regret with a buffer size of one, focusing on local loss. However, their assumption of a fully *i.i.d.* training set is not in line with real-world online situations. Our forthcoming theorem obviates the necessity for the *i.i.d.* assumption by combining moving averages and individual random points.

As explained previously, in each iteration, a single moving average is used to determine the direction of the gradient, which establishes a connection to the all-pairs loss without relying on the *i.i.d.* assumption. This is followed by a step involving the gradient of a random example. In particular, this sample should not be uniformly chosen from historical data. If we denote the moving average based loss as $\bar{L}_t(w_{t-1})$, then using the convexity of $\ell(\cdot)$ and the linearity of models in $\bar{\mathcal{H}}$, we have $\bar{L}_t(w_{t-1}) \leq L(w_{t-1})$ or $\bar{L}_t(w_{t-1}) = L(w_{t-1}) - g(t)$ where $g(t) \geq 0$ is Jensen's gap. The gap is first bounded in lemma 3 using the smoothness constant and the variance of the data. Note that since the model is linear in the space $\bar{\mathcal{H}}$, the smoothness assumption holds for loss gradient w.r.t. z. Therefore, we can provide the following bound in Theorem 5.

Theorem 5 *Assume that assumptions 4,1, and 2 hold. Let* $[w_t]_{i=1}^T$ *be the sequence of models returned by running the algorithm 2 for* T *times using the online sequence of data. Then if* $\bar{w}^* = \arg\min_{w\in\bar{\mathcal{H}}} \sum_{t=2}^{T} L_t(w)$, *we have*

$$\sum_{t=2}^{T} L_t(w_{t-1}) \leq \sum_{t=2}^{T} L_t(\bar{w}^*) + \frac{\|\bar{w}^*\|^2}{2\eta} + 2\gamma G^2 T. \tag{3.8}$$

Note that $\Gamma \geq tr(cov[z_t])$ *where* $cov[z_t]$ *is the covariance of the received examples, and* M_t *is the smoothness parameter of the loss function.*

3.4.2 Proof of Theorem 2

Given the linearity of the model, expressed as $f_w(x) = \langle w, x\rangle$, the convexity of the loss function in the weight variable w implies convexity in the input variable. In other words, the linearity of the model suggests that if the loss function is convex with respect to the weight variable w, it will also exhibit convexity with respect to the input variable. This connection further reinforces the relationship between the convexity of the loss function, the linearity of the model, and the convexity properties in both the weight and input variables. Therefore, given the convexity property of the loss function ℓ, upon arrival of a new example z_t, the loss based on the average, denoted as $\bar{z} := (\mathbb{E}[x|y \neq y_t], y) = \left(\sum_{i=1}^{t-1} \frac{x_i}{t-1}, y_{t-1}|y_{t-1} \neq y_t\right)$, can be linked to the local all-pairs loss using Jensen's inequality as follows:

$$\ell(w, z_t, \bar{z}) \leq \frac{1}{t-1} \sum_{t=1}^{t-1} \ell(w, z_t, z_i). \tag{3.9}$$

Alternatively, representing the average-based loss as $\bar{L}(w)$ and the cumulative loss as $L(w) = \frac{1}{t-1} \sum_{t=1}^{t-1} \ell(w, z_t, z_i)$, we introduce a slack variable at $w = w_{t-1}$ as follows:

$$\bar{L}_t(wt - 1) = L(w_{t-1}) - g(t). \tag{3.10}$$

Here, $g(t) \geq 0$ denotes Jensen's gap. This relation establishes a mathematical connection between the average-based loss and the all-pairs loss through the introduction of this slack variable. Furthermore, it should be noted that this gap can be bounded by leveraging the smoothness of the loss function and considering the variance inherent in the received variables, as demonstrated in Lemma 3.

Lemma 3 *Given that the data $\{z_1, \ldots, z_t\}$ are linearly separable and have bounded total variance, i.e. $\Gamma \geq tr(cov[z_t])$ and a loss function with $M_t : M(w_t)$ smoothness parameters at $w = w_t$, the Jensen gap can be bounded as follows:*

$$g(t) \leq \frac{1}{2}\Gamma M_t. \tag{3.11}$$

Proof 5 *Given that the loss function $\ell(w_{t-1})$ is M_t-smooth, and considering the linearity of the model in the approximated space $\bar{\mathcal{H}}$, expressed as $f_w(x) = \langle w, x\rangle$, it follows that if the function is smooth in w, it is also smooth in x (or z in our context). Let $Q := z_1, \ldots, z_{t-1}$ represent the set of historical examples. Define $\bar{z} := (\mathbb{E}[x|y \neq y_t], y)$ where the expectation is taken with respect to the uniform probability over the set Q. Utilizing the smoothness property of ℓ from assumption 1 and considering expectations:*

$$\begin{aligned}
\ell(w_{t-1}, z_t, z) &\leq \ell(w_{t-1}, z_t, \bar{z}) + \nabla_z \ell(w_{t-1}, z_t, \bar{z})^T (z - \bar{z}) + \frac{M_t}{2}(z-\bar{z})^T(z-\bar{z}) \\
\mathbb{E}_z \ell(w_{t-1}, z_t, z) - \ell(w_{t-1}, z_t, \bar{z}) &\leq \mathbb{E}_z \nabla_z \ell(w_{t-1}, z_t, \bar{z})^T (z - \bar{z}) + \mathbb{E}_z \frac{M_t}{2}(z-\bar{z})^T(z-\bar{z}) \\
L(w_{t-1}) - \bar{L}(w_{t-1}) &\leq \mathbb{E}_z \nabla_z \ell(w_{t-1}, z_t, \bar{z})^T (z - \bar{z}) + \mathbb{E}_z \frac{M_t}{2}(z-\bar{z})^T(z-\bar{z}) \\
&= \frac{M_t}{2}\mathbb{E}_z \|z - \bar{z}\|^2 = \frac{M_t}{2}\mathbb{E}_z \sum_i^d (z_i - \bar{z}_i)^2 \\
&= \frac{M_t}{2}\sum_i^d \mathbb{E}_z (z_i - \bar{z}_i)^2 = \frac{M_t}{2}\sum_i^d Var[z_i] = \frac{M_t}{2} tr(cov[z]).
\end{aligned} \tag{3.12}$$

Replace the expression for $g(t)$ to derive the results.

It is important to note that $tr(\text{cov}[z])$ denotes the trace of the covariance matrix related to the variable z (or the feature x of z). We will colloquially refer to this as the total variance. Next, we present the following lemma, which provides an upper bound on the buffer loss (based on the average), before we establish the regret bounds for the all-pairs loss in Theorem 2.

Lemma 4 *Assume that assumptions 4 and 2 hold. Let $[w_t]_{i=1}^T$ be the sequence of functions returned by running Algorithm 1 for T times using the online sequence of data. Then if $\bar{w}^* = \arg\min_{w \in \bar{\mathcal{H}}} \sum_{t=2}^T L_t(w)$, we have the followings:*

$$\sum_{t=2}^T \bar{L}_t(w_{t-1}) \leq \sum_{t=2}^T \bar{L}_t(\bar{w}^*) + \frac{\|\bar{w}^*\|^2}{2\eta} + \gamma G^2 T - \sum_{t=2}^T \frac{\gamma}{\eta}\hat{v}_t^T (w_{t-1} - \bar{w}^*). \tag{3.13}$$

Proof 6 *Let $\bar{\ell}_t(\cdot) = \ell(\cdot, z_t, \bar{z})$ be the loss based on the average of history examples, where $\bar{z} := (\mathbb{E}x|y \neq y_t, y)$ and the expectation is w.r.t. uniform probability over previous examples , and $\bar{v}_t \in \partial\bar{\ell}_t(w_{t-1})$ then the first update is $w'_t := w_{t-1} - \eta_t\bar{v}$. Let $\hat{\ell}_t(\cdot) = \ell(\cdot, z_t, z)$ where z is a random sample from $\{z_1, \ldots, z_{t-1}\}$. Let $\hat{v}_t \in \partial\hat{\ell}_t(w'_t)$. If we take the distance of two subsequent models to the optimal model, we have:*

$$
\begin{aligned}
&\|w_t - \bar{w}^*\|^2 - \|w_{t-1} - \bar{w}^*\|^2 = \|w_{t-1} - \eta_t(\bar{v}_t + \frac{\gamma_t}{\eta_t}\hat{v}_t) - \bar{w}^*\|^2 - \|w_{t-1} - \bar{w}^*\|^2 \\
&= \|w_{t-1} - \bar{w}^*\|^2 - 2\eta_t(\bar{v}_t + \hat{v}_t)^T(w_{t-1} - \bar{w}^*) + \eta_t^2\|\bar{v}_t + \hat{v}_t\|^2 - \|w_{t-1} - \bar{w}^*\|^2 \\
&= -2\eta_t(\bar{v}_t + \frac{\gamma_t}{\eta_t}\hat{v}_t)^T(w_{t-1} - \bar{w}^*) + \eta_t^2\|\bar{v}_t + \frac{\gamma_t}{\eta_t}\hat{v}_t\|^2 \\
&\overset{(a)}{\leq} -2\gamma_t\hat{v}_t^T(w_{t-1} - \bar{w}^*) - 2\eta_t(\bar{\ell}_t(w_{t-1}) - \bar{\ell}_t(\bar{w}^*)) + \eta_t^2\|\bar{v}_t + \frac{\gamma_t}{\eta_t}\hat{v}_t\|^2 \\
&\overset{(b)}{\leq} -2\gamma_t\hat{v}_t^T(w_{t-1} - \bar{w}^*) - 2\eta_t(\bar{\ell}_t(w_{t-1}) - \bar{\ell}_t(\bar{w}^*)) + 2\gamma_t\eta_t G^2,
\end{aligned}
$$

where inequality (a) implements assumption 2, i.e. $-v_t^T(w_{t-1} - \bar{w}^*) \leq -(g_t(w_{t-1}) - g_t(\bar{w}^*))$. And inequality (b) implements Assumption 1, i.e. $\|\nabla\ell(\cdot)\| \leq G$. Setting the step size $\eta_t = \eta$ and $\gamma_t = \gamma$ for all t, we can get:

$$
\begin{aligned}
&\bar{\ell}(w_{t-1}, z_t, \bar{z}) - \bar{\ell}(\bar{w}^*, z_t, \bar{z}) \\
&\qquad \leq \frac{\|w_{t-1} - \bar{w}^*\|^2 - \|w_t - \bar{w}^*\|^2}{2\eta} + \gamma G^2 - \frac{\gamma}{\eta}\hat{v}_t^T(w_{t-1} - \bar{w}^*).
\end{aligned} \tag{3.14}
$$

Finally, if we sum from $t = 2$ to $t = T$ and set $w_1 = 0$.

$$
\begin{aligned}
\sum_{t=2}^{T}\ell(w_{t-1}, z_t, \bar{z}) - \sum_{t=2}^{T}\ell(\bar{w}^*, z_t, \bar{z}) &\leq \frac{\|\bar{w}^*\|^2 - \|w_T - \bar{w}^*\|^2}{2\eta} + \sum_{t=2}^{T}\gamma G^2 - \sum_{t=2}^{T}\frac{\gamma}{\eta}\hat{v}_t^T(w_{t-1} - \bar{w}^*) \\
&\leq \frac{\|\bar{w}^*\|^2}{2\eta} + \gamma G^2 T - \sum_{t=2}^{T}\frac{\gamma}{\eta}\hat{v}_t^T(w_{t-1} - \bar{w}^*).
\end{aligned} \tag{3.15}
$$

Substitute the definition of $\bar{L}_t(\cdot) = \ell(\cdot, z_t, \bar{z})$ to have the results. This completes the proof.

Now we are ready to prove Theorem 2 using the previous two lemmas.

Proof 7 *Starting from the Lemma 2 finals inequality:*

$$
\begin{aligned}
\sum_{t=2}^{T} \bar{L}_t(w_{t-1}) &\leq \sum_{t=2}^{T} \bar{L}_t(\bar{w}^*) + \frac{\|\bar{w}^*\|^2}{2\eta} + \gamma G^2 T - \sum_{t=2}^{T} \frac{\gamma}{\eta} \hat{v}_t^T (w_{t-1} - \bar{w}^*) \\
&\overset{(a)}{\leq} \sum_{t=2}^{T} L_t(\bar{w}^*) + \frac{\|\bar{w}^*\|^2}{2\eta} + \gamma G^2 T - \sum_{t=2}^{T} \frac{\gamma}{\eta} \hat{v}_t^T (w_{t-1} - \bar{w}^*) \\
\sum_{t=2}^{T} L_t(w_{t-1}) - \sum_{t=2}^{T} L_t(\bar{w}^*) &\overset{(b)}{\leq} \frac{\|\bar{w}^*\|^2}{2\eta} + \gamma G^2 T + \sum_{t=2}^{T} \frac{M_t}{2}\Gamma - \frac{\gamma}{\eta} \nabla \ell(w_t', z_t, \hat{z})_t^T (w_{t-1} - \bar{w}^*) \\
&\overset{(c)}{=} \frac{\|\bar{w}^*\|^2}{2\eta} + \gamma G^2 T + \sum_{t=2}^{T} \frac{M_t}{2}\Gamma - \frac{\gamma}{\eta} \nabla \ell(w_t', z_t, \hat{z})^T (w_t' - \bar{w}^* + \eta \nabla \ell(w_{t-1}, z_t, \bar{z})) \\
&\overset{(d)}{\leq} \frac{\|\bar{w}^*\|^2}{2\eta} + \gamma G^2 T + \sum_{t=2}^{T} \frac{M_t}{2}\Gamma - \frac{\gamma}{2\eta M} \|\nabla \ell(w_t', z_t, \hat{z})\|^2 \\
&\quad - \gamma \nabla \ell(w_t', z_t, \hat{z})^T \nabla \ell(w_{t-1}, z_t, \bar{z}) \\
&\overset{(e)}{\leq} \frac{\|\bar{w}^*\|^2}{2\eta} + 2\gamma G^2 T + \sum_{t=2}^{T} \frac{M_t}{2}\Gamma - \frac{\gamma}{2\eta M} \|\nabla \ell(w_t', z_t, \hat{z})\|^2, \quad (3.16)
\end{aligned}
$$

where inequality (a) implements the convexity of the loss, inequality (b) implements Lemma 3, equality (c) implements update $w_t' + \eta \nabla \ell(w_{t-1}, z_t, \bar{z}) = w_{t-1}$, inequality (d) implements the smoothness of the loss, and (e) is the cauchy inequality. Finally choosing the second step size to be $\gamma = O(\Gamma M_t \eta)$ makes the last two terms term cancel out, for any random example $\hat{z}$. Note that $\Gamma \geq tr(cov[z_t])$ where $cov[z_t]$ is the covariance of the received examples, and M_t is the smoothness parameter of the loss function.

3.4.3 Approximation Error for Random Features

The kernel associated with the space $\mathcal{H}$ is a function defined as $k : \mathcal{X}^2 \times \mathcal{X}^2 \mapsto \mathbb{R}^+$ with a shorthand $k_{(x,x')}(\cdot) := k((x, x'), (\cdot, \cdot))$ and can be constructed given any uni-variate kernel $\mathcal{G}$ for any $x_1, x_2, x_1', x_2' \in \mathcal{X}$ as follows:

$$
\begin{aligned}
k(x_1, x_2, x_1', x_2') &= \mathcal{G}(x_1, x_1') + \mathcal{G}(x_2, x_2') - \mathcal{G}(x_1, x_2') - \mathcal{G}(x_2, x_1') \\
&= \langle \mathcal{G}_{x_1} - \mathcal{G}_{x_2}, \mathcal{G}_{x_1'} - \mathcal{G}_{x_2'} \rangle_{\mathcal{G}}.
\end{aligned} \quad (3.17)
$$

The pairwise kernel k defined earlier demonstrates positive semi-definiteness over $\mathcal{X}^2$. This property aligns it with the attributes of a Mercer kernel, as indicated in references such as [98], when $\mathcal{G}$ exhibits such characteristics over $\mathcal{X}$. We further assume that there exists a lower-dimensional mapping $r : \mathcal{X} \mapsto R^D$, such that $\mathcal{G}_x(\cdot) \approx r(x)^T r(\cdot)$

The literature extensively explores the accuracy of approximating the pointwise kernel $\mathcal{G}$ using random Fourier features, as discussed in references such as [76], [7], and [64]. For example, if the kernel is the Gaussian kernel, i.e., $\mathcal{G}(x-x') = \frac{1}{\sqrt{2\pi d}} \exp -(x - x')^T [diag(\sigma)](x - x')$, we have $p(u)$ from the Fourier transform to be $p(u) \propto \mathcal{N}(\mathbf{0}, diag(\sigma))$ where $\sigma \in \mathbb{R}^d$ is the width of the kernel. Thus, kernel $\mathcal{G}$ can be given by a Monte Carlo estimate as follows:

$$\hat{\mathcal{G}}(x, x') \approx \frac{2}{D} \sum_{i=1}^{D/2} \cos(u_i^T (x - x')) = \langle r(x), r(x') \rangle, \tag{3.18}$$

where $u_i \sim \mathcal{N}(0, diag(\sigma))$, and $r(x) := \frac{1}{\sqrt{D/2}} [cos(u_i^T x), sin(u_i^T, x)]_{i=1}^{D/2}$. The quality of the estimation is given by [76] to be dependent on the number of random features D used to approximate the Fourier transform, which can be extended to pairwise kernels as illustrated in the following corroallary.

Corollary 2 *Given $x_1, x_2, x_1', x_2' \in \mathcal{X}$, and the pairwise kernel k defined in $\mathcal{X}^2 \times \mathcal{X}^2$, the random Fourier estimation of the kernel has a mistake bounded with probability at least $1 - 8 \exp \frac{-D^2 \epsilon}{2}$ as follows:*

$$|\hat{k}_{(x_1, x_2)}(x_1', x_2') - k_{(x_1, x_2)}(x_1', x_2')| \leq \epsilon.$$

The following theorem bounds the random Fourier error in equation (3.6) using the results of corollary 2.

Theorem 6 *Given a pairwise Mercer kernel $k_{(x,x')} := k((x, x'), (\cdot, \cdot))$ defined on $\mathcal{X}^2 \times \mathcal{X}^2$. Let $\ell(w, z, z')$ be a convex loss that is Lipschits continuous with constant G. Then for any $w^* = \sum_{i,j \neq i}^T a_{i,j}^* k_{(x_i, x_j)}$, and random Fourier features number D we have the following with probability at least $1 - 2^8 \left(\frac{\sigma Diam(\mathcal{X})}{\epsilon} \right) \exp(-D\epsilon^2/(4d + 8))$,*

$$\sum_{t=2}^{T} L_t(\bar{w}^*) - \sum_{t=2}^{T} L_t(w^*) \leq GT \|w^*\|_1 \epsilon, \tag{3.19}$$

where $\|w^*\|_1 = \sum_{i,j \neq i}^T |a_{i,j}^*|$.

Remark 6 *A trade-off exists between $\|w^*\|_1$ and ϵ when independent support vectors are in the kernel space ($|a_{i,j}^*| > 0$). In such cases, a larger error requires a smaller ϵ, implying a larger D. Regularization can set insignificant support vectors' $a_{i,j}^*$ to zero, a subject for potential future research.*

3.4.4 Proof of Theorem 3

The proof depends on the main claim of [76], we rewrite the results here.

Lemma 5 (Claim 1 [76]) *Given $x, x' \in \mathcal{X}$, and kernel $\mathcal{G}$ defined on $\mathcal{X} \times \mathcal{X}$, we have the following bound on the error of the approximation $\hat{\mathcal{G}}$ using random Fourier characteristics D,*

$$\mathbb{P}\left(|\hat{\mathcal{G}}(x, x') - \mathcal{G}(x, x')| \geq \epsilon \right) \leq 2 \exp\left(\frac{-D\epsilon^2}{2} \right). \tag{3.20}$$

The following Corollary bounds the error of the pairwise kernel defined in equation (1.7) using main theorem in [76].

Corollary 3 *Given $x_1, x_2, x'_1, x'_2 \in \mathcal{X}$, and pairwise kernel k defined on $\mathcal{X}^2 \times \mathcal{X}^2$, the random Fourier estimation of the kernel has an error bounded with probability at least $1 - 8\exp\frac{-D^2\epsilon}{2}$ as follows:*

$$\sup_{x_i, x_t \in \mathcal{X}} |\hat{k}_{(x_1,x_2)}(x'_1, x'_2) - k_{(x_1,x_2)}(x'_1, x'_2)| \leq \epsilon.$$

Now we are ready to prove Theorem 3, where we start by the difference of the loss between the model $\bar{w}^*$ and w^*,

Proof 8

$$\begin{aligned}
\sum_{t=2}^{T} L_t(\bar{w}^*) - \sum_{t=2}^{T} L_t(w^*) &= \sum_{t=2}^{T} \frac{1}{t-1} \sum_{i=1}^{t-1} \ell(\bar{w}^*, z_t, z_i) - \sum_{t=2}^{T} \frac{1}{t-1} \sum_{i=1}^{t-1} \ell(w^*, z_t, z_i) \\
&\overset{a}{\leq} \sum_{t=2}^{T} \frac{1}{t-1} \sum_{i=1}^{t-1} |\ell(\bar{w}^*, z_t, z_i) - \ell(w^*, z_t, z_i)| \\
&\overset{b}{\leq} \sum_{t=2}^{T} \frac{1}{t-1} \sum_{i=1}^{t-1} G\|\bar{w}^* - w^*\|_{L^2(\rho)}, \qquad (3.21)
\end{aligned}$$

where inequality (a) implements the triangle inequality, and inequality (b) applies assumption 1 and $\|\cdot\|_{L^2(\rho)} \geq \|\cdot\|_2$.
Using the Representer theorem and assumption 3, we assume that the space $\mathcal{H}$ and $\bar{\mathcal{H}}$ are dense in $L^2(\rho)$ (the space of square integrable function under the probability distribution of the received examples), then we can approximate any function in $\mathcal{H}$ by a function in $L^2(\rho)$, i.e. without loss of generality we assume that $\bar{w}^* = \sum_{j=1,k\neq j}^{t-1} \alpha^*_{j,k} \hat{k}_{z_t,z_i}$, then we have:

$$\|\bar{w}^* - w^*\|_{L^2(\rho)} = \|\sum_{j=1,k\neq j}^{t-1} \alpha^*_{j,k}(\hat{k}_{z_t,z_i} - k_{z_t,z_i})\|_{L^2(\rho)}, \qquad (3.22)$$

and using the triangle inequality we have:

$$\begin{aligned}
\sum_{t=2}^{T} L_t(\bar{w}^*) - \sum_{t=2}^{T} L_t(w^*) &\leq \sum_{t=2}^{T} \sup_{x_i, x_t \in \mathcal{X}} G \sum_{j=1,k\neq j}^{t-1} \|\alpha^*_{j,k}(\hat{k}_{z_t,z_i} - k_{z_t,z_i})\|_{L^2(\rho)} \\
&\overset{c}{\leq} \sum_{t=2}^{T} G \sum_{j=1,k\neq j}^{t-1} |\alpha^*_{j,k}|\epsilon \leq GT\|w^*\|_1\epsilon, \qquad (3.23)
\end{aligned}$$

where inequality (c) implements corollary 3, and $\|w^*\|_1 = \sum_{i,j\neq i}^{T} |a^*_{i,j}|$.
This completes the proof.

3.5 Related Work

While online gradient descent exhibits a time complexity of $O(T^2)$ [10, 52, 28], this remains impractical for large-scale problems. For example, in AUC maximization, pairing a data point $x_t, y_t = 1$ with all previous $x_{t'}, y_{t'} = -1|1 < t' < t-1$ to determine the true loss poses a key challenge. This approach requires computing gradients $\nabla Lt(ft-1)$ for all training examples received, which grows linearly with t.

Buffer-based methods have been explored in the literature to address this challenge. [100] introduced buffers B_+ and B_-, each of size N_+ and N_-, considering only points within the buffers at each step t. Reservoir Sampling updates the buffers while ensuring uniform sampling. Theorem 1 in [100] establishes sublinear regret related to buffer sizes. However, this work is limited to AUC with linear models and does not explicitly address the impact of buffer sizes on generalization error.

In contrast, [52] proposed modified reservoir sampling (RSx) by performing s Bernoulli processes with probability $1/t$ to replace each buffer point with the received data point. This approach ensures that the s data points are *i.i.d.* samples of the stream and independent of w, with the generalization bound depending on Rademacher complexity and regret. However, optimal generalization requires $s = O(\sqrt{T})$ for buffer loss.

Recently, [94] suggested an optimal generalization with a buffer size of $s = 1$, assuming *i.i.d.* data and independent buffer-data relations. Their work aligns with ours, but we remove the *i.i.d.* assumption for convergence proof and consider non-linear model spaces. Moreover, the next chapter addresses the case of high variance in the buffer.

3.6 Experiments

We perform experiments on several real-world datasets and compare our proposed algorithm to both offline and online pairwise algorithms. Specifically, the proposed method is compared with different pairwise algorithms on the AUC maximization task with the squared function as the surrogate loss.

3.6.1 Experimental Setup

Compared Algorithms. The compared algorithms including offline and online settings, as illustrated below,

- SPAM-NET [77] is an online algorithm for AUC with square loss that is converted to a saddle point problem with non-smooth regularization.
- OGD [94], the most similar to our algorithm but with a linear model, that uses the last point every iteration.
- Sparse Kernel [49] is an offline algorithm for AUC maximization that uses the kernel trick.

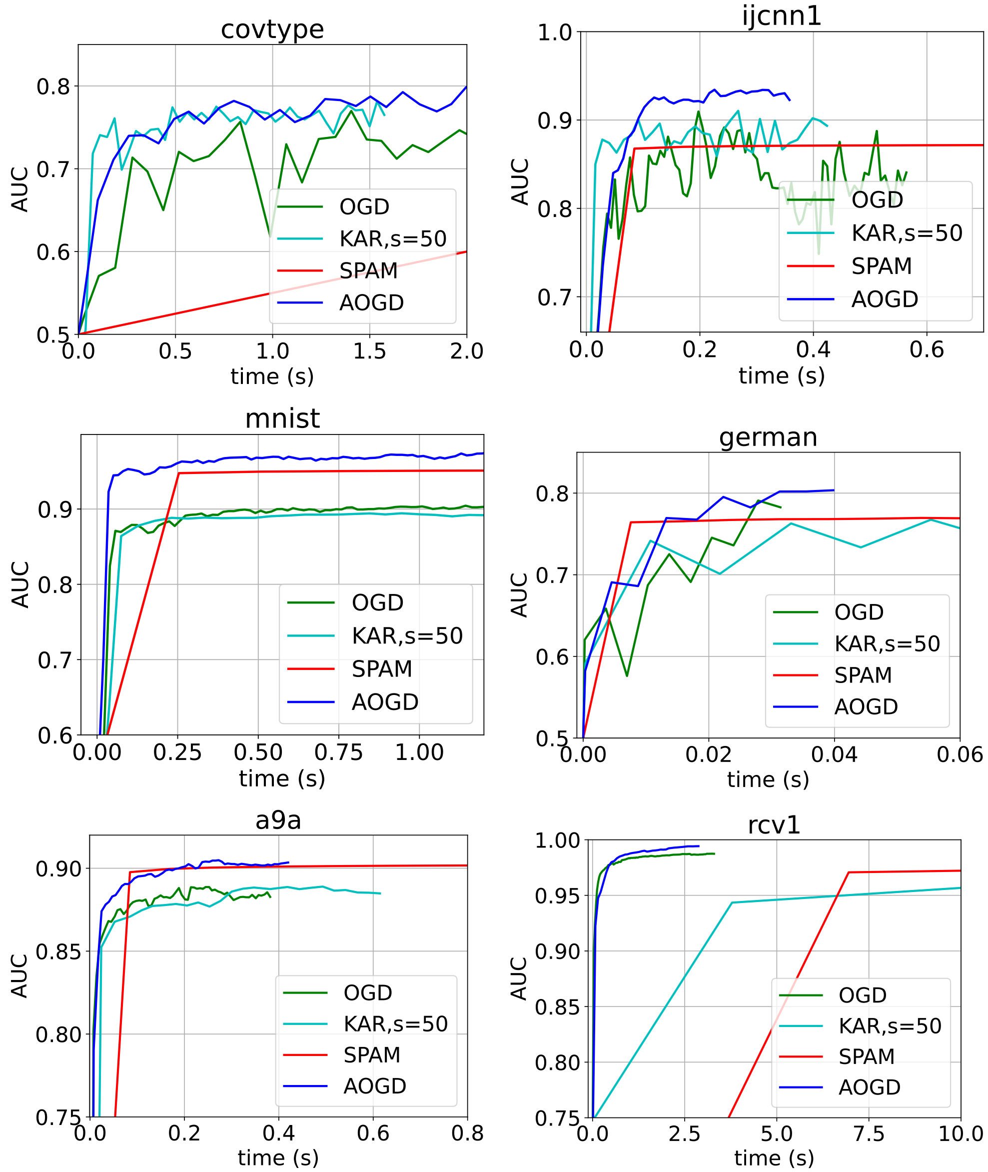


Figure 3.2: The AUC vs. time comparison of the algorithms in different datasets shows superior performance of the proposed method.

Dataset	AOGD	SP-NET	OGD	S. Kernel	Proj.++	Kar
diabetes	**82.68**±0.18	82.03±0.32	82.53±0.31	82.64±0.37	77.92±1.44	79.85±0.28
ijcnn1	**92.25**±0.12	87.01±0.10	83.46±1.25	71.13±0.59	92.20±0.27	83.44±1.21
a9a	90.01±0.05	**90.02**±0.08	88.41±0.42	84.20±0.17	84.42±0.33	77.93±1.55
mnist	**96.98**±0.38	96.57±0.54	89.56±0.34	95.21±0.15	93.82±0.15	89.16±0.14
rcv1	**99.29**±0.23	98.13±0.15	99.05±0.57	**96.26**±0.35	**94.54** ±0.36	97.78 ±0.64
usps	92.76±0.28	85.12±0.88	**92.88**±0.47	91.25±0.84	90.14 ±0.22	91.58±0.25
german	80.75±0.48	76.89±2.46	84.20±0.54	80.11±0.44	78.44 ±0.66	**84.21**±0.45
Reg.	l_2	$l_1 + l_2$	l_2	l_2	l_2	l_2

Table 3.2: AUC maximization results (average ± standard error)$\times 10^2$ using different batch and online algorithms on different datasets

- Projection ++ [43] is an online algorithm with an adaptive support vector set.
- Kar [52] is online algorithm with buffer of size s.

Implementation. The datasets of investigation are available on the LIBSVM website [14]. The experiments are validated for all algorithms by performing a grid search on the hyper-parameters with three-fold cross-validation. For example, in all algorithms the step size $\eta \in 2^{[-8:-1]}$, and the regularization parameters $\lambda \in 10^{[-8:-1]}$. All algorithms have been run five times on different folds on a CPU of 4 GHz speed and 16 GB memory.

3.6.2 Experimental Results and Analysis

Our results on maximizing the area under the curve (AUC) using a squared loss function verify the efficacy of our random Fourier pairwise online gradient descent procedure. Table 2 shows that when compared to online and offline, linear and non-linear pairwise learning algorithms, our algorithm achieves improved AUC performance on large-scale datasets.

Figure 1 displays the AUC performance relative to CPU utilization. The results of ijcnn1 and mnist show that, compared to other state-of-the-art online pairwise learning methods, our algorithm not only has better AUC performance but also has a faster convergence rate. Compared to the two most popular kernel algorithms, Sparse kernel and Projection++, our technique provides superior AUC performance across the board. Projection++ is impractical for large-scale datasets because of its hyper-parameters and the requirement to regularly update a support vector set.

3.7 Summary

In summary, our work introduces a lightweight online kernelized pairwise learning algorithm applicable to both linear and non-linear models. The algorithm efficiently computes the online gradient using a moving average and random examples, with the kernel function approximated through a random Fourier mapping. Achieving a gradient complexity of $O(T)$ for linear models and $O(\frac{D}{d}T)$ for non-linear models, our approach showcases computational efficiency. The incorporation of multiple moving averages, a potential avenue for future research, could further enhance adaptability and robustness. In particular, we address the challenges of *non-i.i.d.* data by dual evaluation of gradients, leveraging random Fourier features for efficient kernel estimation with a sublinear error bound. This integration of kernelization, approximation, and dynamic averaging transcends linear constraints, effectively tackling *non-i.i.d.* data challenges and mitigating kernel complexity.

Chapter 4

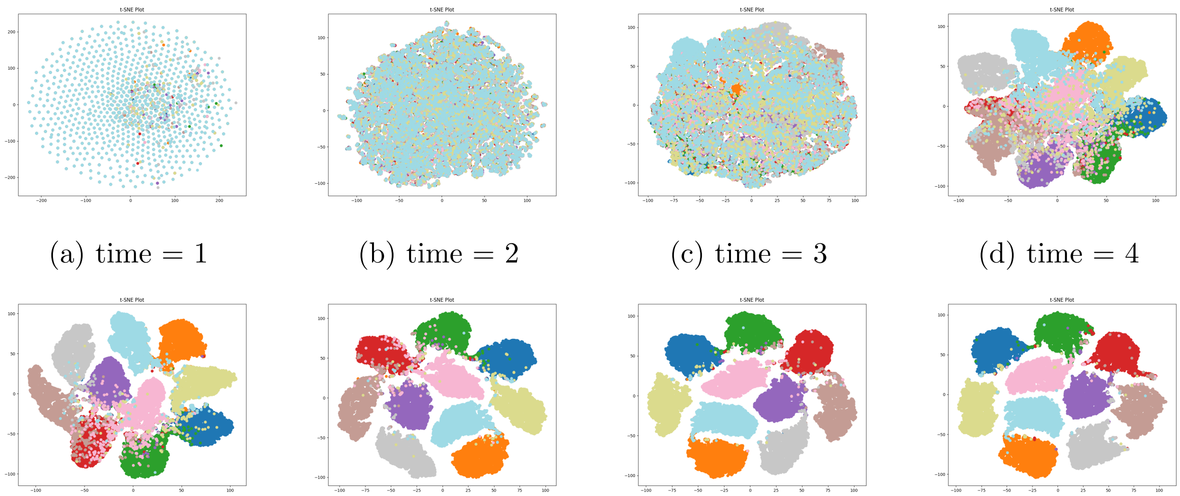
Manifold Metric Learning with Spiking Neural Networks

In this chapter, we introduce a novel strategy for deep metric learning, transitioning from shallow models to deep networks. We leverage spiking neural networks (SNNs) and exploit the temporal dimension to encode high-dimensional data. Our method employs binary encoding with fixed 8-bit steps to efficiently represent integers ranging from 0 to 255, resulting in low latency processing while preserving accuracy. We demonstrate the effectiveness of our approach on benchmark datasets including CIFAR-10, CIFAR-100, and MNIST, achieving superior performance compared to state-of-the-art methods. A key contribution of our work lies in the utilization of the temporal dimension within the SNN framework. By introducing a specialized loss function, we ensure that the temporal dynamics of the network is utilized effectively, treating each time step as a distinct entity. This approach enables the creation of a metric space manifold that captures the intricate relationships between data points, leading to enhanced discriminative capabilities and improved accuracy. Furthermore, our method offers a comparable results of online algorithm (online complexity) to hard-sampling (quadratic complexity) of deep metric learning techniques with SNNs, capitalizing on the unique properties of the moving average-algorithm to achieve superior performance.

4.1 Introduction

Deep metric learning has emerged as a powerful paradigm for learning similarity metrics directly from data, allowing applications such as image retrieval [19], object recognition [67], and clustering [78]. Traditional deep metric learning approaches often rely on standard Artificial Neural Networks (ANNs), which have shown success but are computationally intensive and lack biological plausibility. On the contrary, SNNs have garnered increasing interest due to their potential for low-power, event-driven computation and their ability to emulate the spiking dynamics observed in biological neural systems [29].

The utilization of SNNs in deep metric learning offers several advantages over traditional ANNs. First, SNNs operate on a spike-based communication mechanism, enabling



(a) time = 1 (b) time = 2 (c) time = 3 (d) time = 4

(e) time = 5 (f) time = 6 (g) time = 7 (h) time = 8

Figure 4.1: The t-SNE visualization [87] illustrates the evolution of SNN-based embeddings at each step of the temporal dimension when binary encoding is used. The distinct patterns observed at each step highlight the dynamic nature of the embeddings as they progress through time.

efficient event-driven processing that is well-suited for low-power hardware implementations [95]. This characteristic makes SNNs particularly appealing for applications in resource-constrained environments, such as mobile devices and embedded systems. Furthermore, the temporal dynamics of the spiking activity in SNNs can encode rich information about input data, leading to increased representational capacity and improved discriminative capabilities, as illustrated in Figure 4.1.

In contrast to the fixed processing patterns of traditional neural networks, spiking neurons introduce a dynamic element to their computations. This dynamics, often referred to as inference latency, characterizes its ability to sustain and update their state over time as membrane potential. This potential evolves with each instance of input, typically arriving as weighted spikes. When faced with a static input x, the network must devise a strategy to transform it into a temporal sequence of spikes. This poses an intriguing challenge, prompting the exploration of various encoding techniques. One such method is the direct encoding approach, also known as constant encoding [35]. Here, the value of x is duplicated recurrently T times. This replication essentially treats x as the input potential of the initial layer of neurons, ultimately generating a series of sequential spikes. Alternatively, the rate encoding technique, which draws inspiration from biological phenomena [26]. When x falls within the interval $[0, 1]$, the objective is to approximate its intensity by manipulating the rate of spike generation. Using a Bernoulli distribution, T binary spikes are emitted, their cumulative count serving as an approximation of the input's intensity. Finally, the time-to-first-spike approach offers a unique perspective [48]. Here, a single spike is generated with its precise timing that encodes the intensity of the input pixel.

However, none of the mentioned encoding methods, except the last one, effectively conveys information throughout the temporal dimension. Even though the last method utilizes time as a dimension, it struggles to encode information with minimal latency, posing challenges in the decoding process, especially in the final layer. In this context, we explore an alternative encoding technique that lies between constant encoding and rate encoding. Here, the true information contained in the static input is encoded into a spike train. However, the entire train is required to fully recover the information.

In our study, we introduce an innovative method for deep metric learning that capitalizes on SNNs and fixed-bit binary encoding. This approach enables the efficient encoding of high-dimensional data while maintaining accuracy. Utilizing binary encoding has garnered significant interest in recent times due to its promise for low-latency processing and memory-efficient data storage in the conversion from ANNs to SNNs [63]. However, to the best of our knowledge, our research marks the first dedicated attempt to directly train SNNs using binary encoding. By representing integers using fixed 8-bit steps, binary encoding enables compact and efficient representation of numerical values, making it suitable for real-time inference tasks. The key contributions of our work can be summarized as follows:

- Integration of SNNs with Fixed-Bit Binary Encoding: We propose a novel framework that combines the strengths of SNNs with fixed-bit binary encoding for deep metric learning. By utilizing binary encoding with fixed 8-bit steps, we achieve low-latency processing without sacrificing accuracy, making our approach well-suited for real-time inference tasks.
- Exploitation of Temporal Dynamics: We introduce a specialized loss function that leverages the temporal dynamics of SNNs for deep metric learning. By treating each time step differently and capturing the temporal relationships between data points, we improve the discriminative capabilities of the network and improve performance on benchmark datasets.

4.2 Related Work

The fundamental principle underlying the loss of cross-entropy is to minimize the difference between the true conditional distribution and the distribution inferred by the machine. However, given that the true distribution is frequently unknown, it is commonly approximated using a one-hot vector, which may not consistently provide accurate results. Several research studies have examined the limitations of cross-entropy loss, highlighting its vulnerability to noisy labels [72], susceptibility to adversarial examples [44], and difficulties in generalization, particularly in scenarios beyond traditional classification tasks, such as zero-shot learning [89], where data remain unseen during training.

Deep metric learning has witnessed significant advancements in recent years, with various methods proposed to learn discriminative embeddings directly from data. Contrastive losses are a crucial component in learning representations for various tasks. They can be categorized into pairwise and triplet losses. In pairwise contrastive loss, the objective is to adjust the distance between pairs of samples based on their similarity. By increasing or decreasing this distance, the model learns to differentiate between similar and dissimilar pairs more effectively. On the other hand, triplet loss operates simultaneously on three samples: an anchor, a positive example (similar to the anchor), and a negative example (dissimilar to the anchor). The goal of triplet loss is to minimize the distance between the anchor and the positive example while simultaneously maximizing the distance between the anchor and the negative example. This encourages the model to embed similar samples closer to each other while pushing dissimilar samples apart.

The concept of contrastive losses originated in unsupervised learning settings, where data augmentation techniques are used to create pairs or triplets of samples with known relationships [85]. By training the model to distinguish between these relationships, it learns to capture meaningful features and representations from the data. In a supervised setting, similarity is enforced by the labels of the data. Traditional approaches such as contrastive loss [39], triplet loss [42] and margin-based losses such as margin triplet loss [40] and N-pairs loss [83] have been widely adopted. These methods excel in capturing complex relationships between data points, making them invaluable for tasks like image retrieval, face verification, and person re-identification. However, leveraging pairwise distances incurs a computational cost that scales at least quadratically ($O(n^2)$ with N denote the number of examples) in contrastive losses. In practice, researchers address this challenge through techniques such as hard negative mining. This method involves excluding easy pairs from a batch of examples, thereby optimizing the efficiency of contrastive learning algorithms. Several studies in the field have explored and proposed various strategies to mitigate the computational overhead of pairwise approaches [82, 38, 24, 27].

In recent years, there has been a growing interest in leveraging SNNs for deep learning. SNNs offer the promise of low-power, event-driven computation and are capable of capturing the temporal dynamics inherent in the data. Two main approaches to training SNNs have emerged: surrogate training [74] and conversion training. Surrogate training methods involve updating the weights of the SNNs through a differentiable activation function in the backward loop. In contrast, conversion training methods, as discussed in [46], focus on optimizing SNNs by leveraging pre-trained ANN models.

Despite the potential of SNNs in capturing temporal dynamics, existing encoding methods often neglect the temporal dimension. Popular techniques such as binary encoding and quantization treat each time step equally, disregarding valuable temporal information encoded in the data. This oversight limits the discriminative power of encoding methods, especially when dealing with high-dimensional data. In this paper, we address these limitations by proposing a novel approach to deep metric learning that combines the strengths of SNNs with fixed-bit binary encoding. By harnessing the temporal dynamics of SNNs and incorporating fixed-bit encoding, we aim to achieve high accuracy in the low latency of real-world applications.

4.3 Background

In the field of mathematics, a metric space is a fundamental concept that provides a formal framework for measuring the distances between points within a set. Formally, a metric space is defined as an ordered pair $(\mathcal{X}, M)$, where $\mathcal{X}$ is a set, and M is a metric function that assigns a non-negative real number to each pair of points in $\mathcal{X}$, satisfying certain axioms such as non-negativity, symmetry, and the triangle inequality. In practical applications such as machine learning, where the goal is often to make sense of data and extract meaningful patterns, the notion of distance plays a crucial role. However, traditional metrics may not adequately capture the underlying structure of the data, particularly in cases where the relationship between data points is complex and non-linear.

4.3.1 Supervised Metric Learning

To address this challenge, researchers have turned to the field of metric learning, which aims to learn a distance metric directly from the data. In metric learning, the goal is to map the data to a new space where examples from the same class have a small distance between them, while examples from different classes are far apart. By doing so, we can effectively learn a metric that measures the similarity or dissimilarity between points, taking into account the labels associated with those points. In the realm of deep metric learning, neural networks are used to perform this mapping, harnessing their ability to model complex data distributions and extract high-level features. Deep metric learning methods use neural networks to learn a mapping from the input space to a high-dimensional feature space, where the distance between points corresponds to their similarity or dissimilarity (see Figure 4.2). The reimagined metric space, therefore, becomes optimized for a spectrum of pivotal tasks. Here are a few applications where such a learned metric manifests its utility:

- **Classification:** By grouping similar examples closer together, classification algorithms can make more accurate predictions, especially in scenarios where the geometric boundaries between classes are not immediately obvious in the original feature space.

- **Few-shot and Zero-shot Learning:** Metric learning is particularly beneficial when labeled data are scarce or when predicting unseen classes, as it focuses on learning a generalizable representation of similarity that can transcend the specifics of the training set.

- **Information Retrieval:** In search and recommendation systems, the learned metric can surface relevant items to a query by quantifying relevance through learned distances, thus improving the retrieval quality.

- **Face Recognition and Person Re-identification:** Metric learning frameworks excel in tasks where the objective is to recognize individuals across varying conditions, leveraging subtle but consistent features distilled during training.

- **Learning from Imbalanced Data:** The learned metrics help mitigate the influence of class imbalance by ensuring minority classes are represented in a feature space that emphasizes their distinctiveness from the majority classes.

- **Clustering and Ranking:** The distance metric learned can be used to enhance clustering by bringing semantically similar items closer together and ranking by ensuring that the order reflects true similarity.

In conclusion, the learned metric not only becomes a measure of distance but evolves into a sophisticated tool that enhances various machine learning paradigms. Its influence extends from improving the granularity of classification to reshaping the very core of how similarity is interpreted across diverse applications.

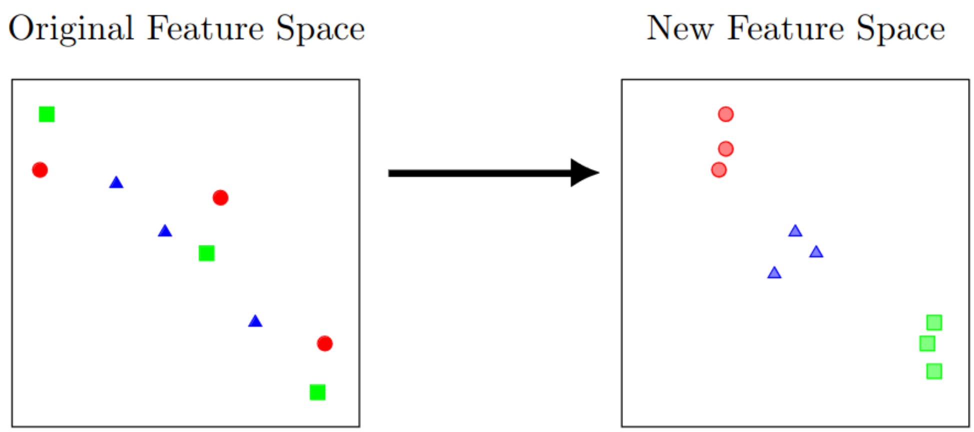


Figure 4.2: Illustration of metric learning using contrastive losses. The plot on the left shows the original feature space, where data points from three classes are randomly scattered. The right plot, after the metric learning process, shows data points from the same classes clustered together with a clear margin m between them. This demonstrates how metric learning optimizes feature spaces to enhance class separability and data representation.

4.3.2 Spiking Neural Networks

In the context of SNNs, which emulate the spiking dynamics observed in biological neural systems, there is growing interest in leveraging their unique properties for metric learning tasks. SNNs offer the potential for low-power, event-driven computation and are capable of capturing the temporal dynamics inherent in the data. By integrating SNNs into the metric learning framework, we can further enhance our ability to learn effective distance metrics that capture spatial and temporal relationships in the data. We adopt the LIF neuron model for our SNN, described by the following discrete dynamics using the Euler method:

$$u[t+1] = \tau V[t] + W^\top s[t], \tag{4.1}$$
$$s[t+1] = H(u[t+1] - V_{\text{th}}), \tag{4.2}$$
$$V[t+1] = u[t+1] - V_{\text{th}} s[t+1], \tag{4.3}$$

where $u[t+1]$ represents the membrane potential before resetting, τ is a time constant, $s[t]$ the output spike, and H is the Heaviside step function. This formulation captures the dynamic response of neurons to incoming spikes over time. Overall, the combination of metric learning techniques with neural networks, including SNNs, holds promise for advancing the state-of-the-art in various machine learning tasks, ranging from classification and clustering to retrieval and recognition. Through ongoing research and development, we can continue to explore the potential of deep metric learning with neural networks to unlock new capabilities in understanding and interpreting complex data.

4.3.3 Training Spiking Neural Networks with Surrogate Gradients

Training SNN presents unique challenges due to the non-differentiable nature of spikes. Unlike traditional neural networks, where the backpropagation algorithm is used straightforwardly thanks to the differentiable activation functions, SNNs require an alternative approach to compute gradients for weight updates. Here, surrogate gradients come into play, enabling the application of gradient-based learning rules to SNNs.

At the core of an SNN's neuron model, such as the Leaky Integrate-and-Fire (LIF) model, is a membrane potential that integrates incoming spikes over time. When this potential crosses a certain threshold, the neuron emits a spike and resets its potential, a process that is inherently non-differentiable. To avoid this, surrogate gradient methods propose a differentiable approximation of the spike function, allowing the gradient of the loss function with respect to the network weights to be estimated.

The surrogate gradient approach involves defining a pseudo-derivative for the spiking threshold function. This pseudo-derivative is non-zero and smooth, providing a pathway for gradients to flow back through the network during training. A commonly used form for the surrogate gradient is a piecewise linear function or a Gaussian function, which approximates the sharp spike at the threshold with a more gradual curve.

The training process with surrogate gradients follows the conventional backpropagation steps but with the critical difference in handling the spike generation mechanism. During the forward pass, the actual spiking behavior is simulated according to the neuron model. However, during the backward pass, the surrogate gradient is used to approximate the derivative of the spike function. This enables the computation of gradients with respect to the neuron's input and weights, facilitating the application of standard optimization algorithms, such as stochastic gradient descent, to update the model parameters.

By using surrogate gradients, SNNs can be trained on a variety of tasks, from classification to temporal pattern recognition, leveraging their temporal dynamics and energy-efficient computation. The method effectively bridges the gap between the rich temporal processing capabilities of SNNs and the powerful optimization machinery developed for traditional neural networks, opening up new avenues for research and application of SNNs in domains where capturing temporal dependencies is crucial.

The use of surrogate gradients thus represents a pivotal development in the training of SNNs, making it possible to harness their potential to process spatio-temporal data while maintaining compatibility with the backpropagation algorithm and its variants.

4.4 Method

Our approach aims to harness the inherent capabilities of SNNs within the realm of deep metric learning. Specifically, we focus on learning a mapping $f : \mathcal{X} \mapsto \mathcal{Z}^T$ from a set of examples $\{z_i | z_i = (x_i, y_i), x_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, \ldots, N\}$. The primary goal is to optimize this mapping to minimize within-class variability and maximize between-class variability, effectively leveraging the metric space to maximize mutual information between labels and embeddings. Given the unique temporal dynamics of SNNs, our method innovatively incorporates the temporal dimension T, proposing a more sophisticated treatment than traditional averaging methods.

4.4.1 Fixed Binary Encoding

Deep metric learning within SNNs aims to learn a space where the similarity between examples is defined not only by spatial proximity, but also through temporal dynamics. Encoding inputs into spike trains introduces a temporal dimension that carries significant information. Given the embeddings in the middle mapping $\mathcal{Z}^T$, with the mapping f, we dive into the nuanced handling of the temporal dimension. Encoding of the inputs in SNN propose the following, for a given pair of examples (x_i, y_i), where the output at time step t is denoted as $f(x)^t$, the contrastive loss is computed as follows:

$$\begin{aligned}\ell((x_i, y_i), (x_j, y_j)|f) =& \mathbb{I}_{y_i = y_j} \left\| \sum^{t} c^t [f(x_i)^t - f(x_j)^t] \right\| \\ &+ \mathbb{I}_{y_i \neq y_j} \left[m - \left\| \sum^{t} c^t [f(x_i)^t - f(x_j)^t] \right\| \right]_+ ,\end{aligned} \tag{4.4}$$

where $m > 0$ is margin parameter, $\mathbb{I}$ is the indicator function, and $[\cdot]_+$ is positive projection. In constant encoding and rate encoding methods, the values c^t are selected to be uniform and collectively sum up to one. However, this uniformity tends to suppress the temporal dimension of conveying meaningful information. Unlike traditional methods that treat temporal sequences in an Euclidean fashion by simple averaging, our approach acknowledges the intricate, manifold-like nature of temporal data. This methodology enables a dynamic representation of embeddings across various temporal states, effectively capturing the evolving characteristics of the data. We explore two methodologies: when the input is encoded via the binary method, where the first spike represents 128, the second 64, and so forth until the last spike represents 1, we explicitly set it as follows:

$$c^t = 2^{8-t} \quad \forall t = 1, 2, \cdots, 8. \tag{4.5}$$

4.4.2 Flexible Binary Encoding

In pursuit of capturing more intricate temporal patterns within our data, we consider an alternative approach to binary encoding where the parameter c is not static, but adaptable. Unlike a fixed encoding scheme, here c is treated as a learnable parameter that is optimized during the training process through gradient descent. This method introduces flexibility and allows the encoding to adjust dynamically, potentially leading to a more nuanced and discriminative representation of the input features over time.

The learning process for c is guided by the backpropagation algorithm, where at each training step t, the value of c is updated according to the gradient of the loss function ℓ with respect to c. This gradient is computed in the context of the differences $f(x_i)^t - f(x_j)^t$ between all input pairs, driving c to evolve in a direction that minimizes the loss:

$$c_k^t \leftarrow c_{k-1}^t - \eta \frac{\partial \ell}{\partial c^t} \quad \forall t = 1, 2, \cdots, T, \tag{4.6}$$

where η represents the learning rate and T is the total number of time steps considered.

This adaptive mechanism is particularly valuable as it grants the model the ability to refine its internal representations over the course of training, thereby enhancing its sensitivity to temporal dynamics as the model learns the encoding parameter.

4.4.3 Online Metric Learning for Classification

We consider the contrastive loss described in equation 4.4, however iterating through all pairs in the set $\mathcal{S}$ results in a computational complexity of $O(n^2)$, rendering it impractical for large datasets. In the literature, the practice of mining negative examples is often utilized, where certain examples are omitted if the classifier readily identifies them. However, the convergence of mining is not assured since the data batch it is applied to is limited. In our experiments, we implement the average online gradient descent algorithm described in Chapter 2. In Chapter 2, our focus was on computing the loss for both new and historical examples by maintaining a dynamic moving average of the history. Yet, with the contrastive losses, the constraint on the data space is not applicable because the embedding space is still being learned. Therefore, we introduce a concept of pseudo moving averages within the embedding space, attracting or distancing new examples based on their labels. To maximize separation, we initialize these moving averages to be orthogonal to each other in 128-metric space, and using the euclidean distance as the metric as illusatred in equation 4.7 with $f = \tilde{h}_\theta$,

$$\ell(h_\theta, \mathbf{x}, y) = \sum_{i=1}^{C} \left\| \tilde{h}_\theta(\mathbf{x}) - \mu_y \right\| + [m - \left\| \tilde{h}_\theta(\mathbf{x}) - \mu_i \right\|]_+ \cdot \mathbb{I}_{i \neq y}, \tag{4.7}$$

where μ_i denote the pseudo moving averages for each class i, and C denote the number of classes. This loss function is designed to minimize the distance between an example's embedding and the moving average of its corresponding class, while simultaneously maximizing the distance from the moving averages of other classes, up to a predefined margin. The loss is very similar to N-paris loss introduced in [83], but with pseudo averages instead of multiple negatives.

After learning the embedding space using the pairwise loss, the classification approach employs a model of a one-layer neural network, utilizing a Softmax loss function for multi-class categorization. The probability that a given input vector $\mathbf{x}$ belongs to class k among a total of K classes is given by the Softmax function:

$$h(\mathbf{x})_k := P(Y = k|\mathbf{x}) = \frac{e^{\tilde{h}(\mathbf{x})^T \beta_k}}{\sum_{j=1}^{K} e^{\tilde{h}(\mathbf{x})^T \beta_j}}, \tag{4.8}$$

where β_k represents the coefficient vector associated with class k. The training objective is to minimize the cross-entropy loss across all N samples in the dataset, which for a single sample is given by:

$$L(\mathbf{x}, y) = -\sum_{k=1}^{K} \mathbb{I}(y = k) \log h(\mathbf{x})_k, \tag{4.9}$$

where $\mathbb{I}(y = k)$ is an indicator function that is 1 if $y = k$ and 0 otherwise, and y is the true class label for the input vector $\mathbf{x}$.

4.5 Experiments

4.5.1 Implementation Details

Our experiments are designed to evaluate the impact of different encoding strategies and metric-based loss functions on the performance of SNN for deep metric learning tasks. We investigate two primary encoding methods: (1) Constant encoding, where input data are replicated T times to match the temporal dimension, and (2) Binary encoding, which transforms each data point into a sequence of 8 spikes based on its integer value ranging from 0 to 255. Furthermore, we explore the efficacy of various metric-based losses, including triplet loss, N-pair loss, and contrastive loss, under different mining strategies, such as hard and semi-hard mining. We conduct our experiments in one architecture, namely VGG19 but with one linear layer. The accuracy is determined by employing the k-Nearest Neighbor (KNN) algorithm with a value of 5 for the parameter k. In addition, we conducted an evaluation using a single linear layer applied to the acquired embeddings to confirm the findings. However, it was found that KNN alone suffices for accurate results. During the training phase, we utilized the surrogate gradient Backpropagation Through Time (BBTT) to facilitate learning within the SNN architecture. In particular, the membrane time constant and threshold were set to one.

Table 4.1: KNN Classifier Accuracy Across Encoding Strategies, Temporal Lengths, and Loss Functions for MNIST, CIFAR-10, and CIFAR-100 Datasets, Measured by Precision at $k = 1$. Precision at $k = 1$ refers to the accuracy of the classifier when only the nearest neighbor is considered to predict the label of a test sample. This metric provides a test of the model's ability to identify the most similar example. By exploring different combinations of encoding strategies, temporal lengths, and loss functions, we aim to uncover configurations that optimize representation efficiency and predictive accuracy, particularly in the challenging context of classifying images from diverse and complex datasets.

Loss	**Online Sampling**			**hard sampling**		
	Triplet	**N-Pairs**	**Margin**	**Triplet**	**N-Pairs**	**Margin**
CIFAR-10						
ANN	89.12	90.02	89.32	88.54	90.00	89.42
Constant T=8	85.84	84.52	85.64	85.20	84.12	85.20
Constant T=16	86.72	87.21	86.56	87.12	86.25	86.2
Constant T=32	87.21	88.12	88.22	88.21	88.7	88.1
rate T=8	72.51	71.15	70.45	71.15	71.25	70.54
rate T=16	75.15	75.18	75.77	74.17	74.98	75.8
rate T=32	78.95	79.1	79.28	79.05	79.07	79.13
Fixed binary T=8	83.38	87.61	87.42	84.6	**87.98**	86.50
Flexible binary T=8	86.08	**87.91**	87.67	86.71	87.82	86.28
CIFAR-100						
ANN	68.38	68.15	69.02	68.02	69.1	69.03
Constant T=8	63.50	64.17	65.35	63.2	63.85	64.15
Constant T=16	64.14	67.57	66.75	65.20	65.15	65.87
Constant T=32	**68.15**	67.02	67.86	**68.35**	66.68	68.01
rate T=8	58.01	59.01	56.98	58.15	57.21	57.58
rate T=16	60.14	59.58	60.95	60.25	59.48	59.34
rate T=32	62.21	61.98	60.92	61.47	62.02	61.57
Fixed binary T=8	64.21	65.47	65.84	63.8	64.67	65.02
Flexible binary T=8	64.24	65.98	66.05	63.84	65.24	65.72
MNIST						
ANN	99.17	99.65	99.35	-	-	-
Constant T=8	98.47	98.28	98.68	98.24	98.65	98.17
Constant T=16	99.10	98.85	98.15	98.98	98.11	99.05
Constant T=32	99.31	98.97	99.24	99.34	99.29	**99.42**
rate T=8	91.25	92.44	93.10	92.47	92.10	91.29
rate T=16	93.25	94.24	94.07	94.22	93.89	93.94
rate T=32	95.47	95.68	95.87	96.02	96.04	96.74
Fixed binary T=8	98.89	99.10	98.58	99.01	98.97	98.75
Flexible binary T=8	**99.20**	98.95	98.84	99.15	98.55	98.64

4.5.2 Encoding Strategies

Constant Encoding: This approach directly replicates the input data T times to form a temporal sequence, facilitating a straightforward integration with the SNN framework without altering the original structure of the data.

Binary Encoding: Data points are converted into a temporal sequence of spikes, where each bit of the 8-bit binary representation triggers a spike, effectively encoding the original value into a temporal pattern.

4.5.3 Loss Functions

To understand the differential impact of loss functions, we examine three key variants, such as the triplet Loss which aims at learning useful embeddings by ensuring that an anchor example is closer to positive examples than to negative examples by a margin as follows;

$$L_{\text{triplet}} = \sum_{(i,j,k)} [d(a_i, p_j) - d(a_i, n_k) + m]_+, \tag{4.10}$$

where a_i represents the anchor input, p_j the positive input, n_k the negative input, d the chosen distance metric, and m the margin. In addition to triplet loss, N-pairs loss is considered, which extends the idea of triplet loss by considering pairs of examples, aiming to distinguish a single positive example from multiple negative examples in a single batch.

4.5.4 Mining Techniques for Contrastive Learning

In the context of metric learning, particularly when employing contrastive losses, the selection of sample pairs plays a crucial role in the training process. Mining techniques are employed to choose informative pairs of examples from the training set to effectively train the model. Among these techniques, hard and semi-hard sample mining are two strategies that have been shown to be effective in improving the performance of deep learning models.

Hard Sample Mining: Hard sample mining involves the selection of sample pairs challenging for the model to correctly classify. These pairs consist of positive pairs that are distant from each other and negative pairs that are closely positioned in the feature space. Specifically, hard positive pairs originate from the same class but lie beyond the current decision margin of the classifier. Conversely, hard negative pairs belong to different classes but fall within the margin. Training on these demanding examples forces the model to learn more discriminative features, as it endeavors to rectify its most challenging classification errors. To achieve this, during training, easy negative samples are replaced by hard negative samples, and easy positive samples are replaced by hard positive samples.

Online Sample Mining: using the new loss introduced in equation 4.7. In the context of deep metric learning, the concept of pseudo moving averages within the embedding space bears resemblance to contrastive learning, albeit without the need for explicit sampling.

To maximize separation, we initialize these moving averages to be orthogonal to each other in the 128-metric space, utilizing the Euclidean distance as the metric. The loss is very similar to N-paris loss introduced in [83], but with pseudo averages instead of multiple negatives.

4.5.5 Classification Accuracy

Table 4.1 illustrates how the accuracy achieved by binary encoding with learnable weights (flexible) exceeds the performance of both constant encoding and fixed binary encoding across two datasets. In particular, constant encoding demonstrates superior performance in the CIFAR-100 dataset. Importantly, the online training has comparable results in all datasets, which shed the light on the effectiveness of the online sampling loss.

In our comparative analysis illustrated in Figure 4.3, we examine the performance of different encoding strategies on model precision and the effects of varying margin values on class separability. Part (a) of the figure evaluates the impact of binary encoding, constant encoding, and rate encoding on precision @ 1. Our results indicate that binary encoding achieves the highest precision, underlining its efficacy in class discrimination. Constant encoding follows, with rate encoding trailing as the least effective method in this context. This hierarchy suggests that the choice of encoding strategy significantly influences the model's ability to accurately classify instances based on a single retrieval attempt.

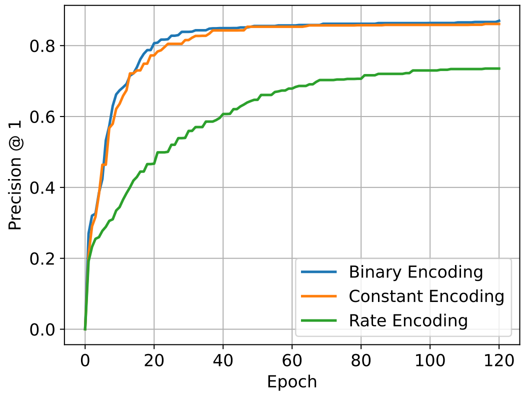
Part (b) of the figure shifts focus to the role of margin values in enhancing the model's performance. It reveals that precision varies with different margin settings, pinpointing an optimal margin that maximizes class separability. This observation highlights the critical importance of margin configuration in metric learning algorithms, directly impacting the model's classification precision.

4.5.6 Embedding Visualization

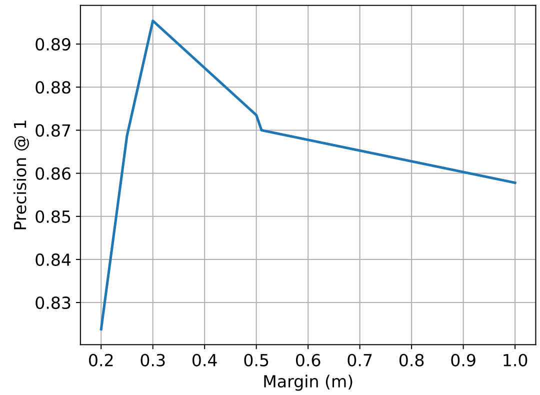
The t-SNE visualization of CIFAR-10 embeddings in Figure 4.4 provides a comprehensive view of the distribution and clustering of the feature representations learned by the model. By reducing the high-dimensional embeddings to a two-dimensional space while preserving the local structures, t-SNE allows for a visually interpretable representation of the data.

In the context of CIFAR-10, which consists of 10 classes of images, the t-SNE visualization can reveal how well the model has learned to differentiate between these classes. Clusters formed by embeddings corresponding to the same class indicate that the model has successfully captured the inherent similarities between images of the same category. On the other hand, the separation between clusters belonging to different classes reflects the discriminative power of the learned features.

Analyzing the t-SNE visualization can also provide insights into potential areas of improvement. For example, if certain classes appear to overlap or are poorly separated in the visualization, it may indicate areas where the model struggles to distinguish between those classes. This information can guide further refinement of the model architecture or training process to enhance its performance on specific classes or categories.



(a) Precision @ 1 for three different encodings.



(b) Precision with various margin values.

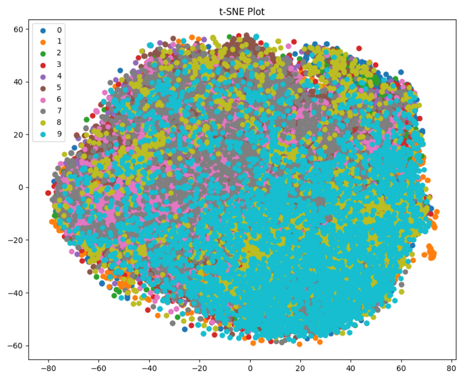
Figure 4.3: Comparative analysis of model performance: (a) explores the impact of three distinct encoding strategies on precision@1, highlighting the effectiveness of each method in discriminating between classes at T=8; (b) investigates precision as a function of different margin values, demonstrating how varying the margin influences the separability of classes within the learned feature space.

4.6 Limitation of Binary Encoding

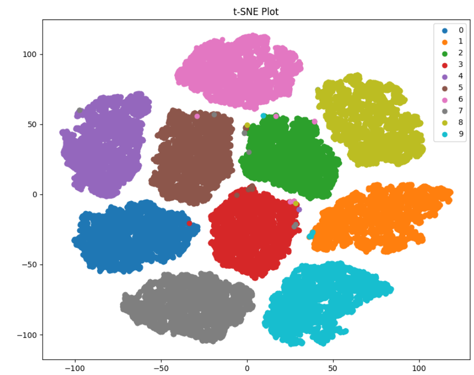
The requirement for inputs to be in the 0-255 range can pose challenges, particularly in scenarios where data augmentation techniques are employed. Data augmentation, a common practice in machine learning, involves artificially expanding the training dataset by applying transformations such as rotation, scaling, or cropping to the input images. However, when using binary encoding, these transformations can alter the input range, potentially exceeding the 0-255 range, and thereby affecting the effectiveness of binary encoding. As a result, maintaining the range 0-255 becomes crucial to ensure the compatibility of binary encoding with data augmentation techniques. The constraint imposed by binary encoding can influence the accuracy of the model, as observed in the CIFAR-10 and CIFAR-100 datasets, particularly compared to models trained using traditional approaches such as ANNs (literature reports 96% and 76% [53]).

4.7 Summary

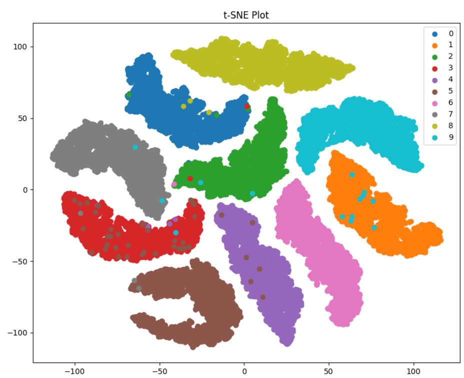
In this study, we embarked on an exploration of deep metric learning within the context of SNNs, evaluating the impact of different data encoding strategies, metric-based loss functions, and neural network architectures on the classification accuracy of KNN classifiers across three benchmark datasets: MNIST, CIFAR-10, and CIFAR-100. Our experiments demonstrated the nuanced interaction between the choice of encoding strategy, whether constant or binary, and the temporal dimension (T), highlighting the significant influence of temporal length on the performance of SNNs.



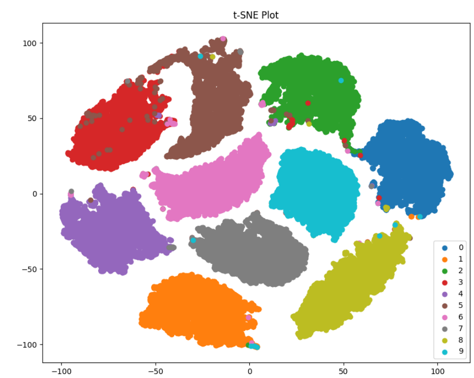
(a) Orignal space



(b) ANN embedding space



(c) SNN-Constant embedding



(d) SNN-Binary embedding

Figure 4.4: The t-SNE visualizations compare the CIFAR-10 dataset before and after mapping with different models and encoding schemes, including ANNs, SNNs with constant encoding, and SNNs with binary encoding.

Comparison of loss functions, specifically triplet, N-pair, and contrastive losses, underlined the importance of selecting an appropriate loss function that complements the encoding strategy and the architecture in use. Our findings suggest that while all loss functions have their merits, their effectiveness is contingent upon the dataset and the encoding strategy employed, and certain combinations yield superior performance over others.

While binary encoding offers advantages in terms of efficiency and memory usage, the strict input range requirement may limit its performance, especially when data augmentation is involved. Future advancements may involve the development of novel encoding techniques that offer greater flexibility and robustness to input variations, thereby enabling more effective integration with data augmentation strategies while maintaining or even surpassing the accuracy of traditional ANNs.

Chapter 5

Provable Robust Spiking Neural Networks for Online Metric Learning

As we explore the properties of deep spiking neural networks for pairwise learning in this chapter, we shift our focus to rate encoding. Introduced in the previous chapter alongside new encoding techniques, rate encoding stands out in research for its potential to enhance the robustness of SNNs. This method converts static inputs into temporal sequences of spikes, which has been experimentally shown to bolster network resilience. Through a combination of theoretical analysis and empirical investigation, we study rate encoding to uncover its influence on strengthening network robustness, building upon the groundwork laid in the preceding chapters.

5.1 Introduction

Similar to the neurons in recurrent neural networks, the spiking neurons possess a temporal dimension (a.k.a. inference latency) and maintain their present state as the membrane potential, which gets updated at each time step after receiving the input in the form of weighted spikes. To process a static input, say $\mathbf{x} \in \mathbb{R}^d$, the network requires a strategy to convert it into a temporal sequence of spikes. A standard encoding technique, called rate-encoding, inspired by biological evidence, is designed to approximate an input $x \in [0, 1]$ with the rate of spikes [25]. It generates T binary spikes $\{z_i\}_{i=1}^T$, using a Bernoulli distribution $z_i \sim Ber(x)$, so that the average number of spikes approximates the intensity of the pixel, $\frac{1}{T}\sum_{i=1}^T z_i \approx x$. It can be noted that among the encoding techniques discussed before, only rate-encoding introduces a systematic noise with the input.

As SNNs aim to replace conventional artificial neural networks (ANN) for energy efficiency, they become exposed to intelligent attacks. Adversarial perturbation of inputs is one such attack, where an input image is altered with a mild perturbation imperceptible to the human eye, but can fool the classifier to make an incorrect prediction [32]. Defending against such attacks is imperative for many safety-critical applications. The work in [81] made some important experimental observations, where they found that the rate-encoded SNNs are adversarially robust compared to the vanilla ANN classifiers and that the robustness of rate-encoding drops with the increase in inference latency, T. They further conjectured that the robustness may be due to the sparse spiking activity. However, [57] observed that constant-encoded SNNs have lower adversarial robustness compared to ANNs, but also have low spiking activity similar to rate-encoded SNNs. The present work is therefore motivated by the need to provide a theoretical proof for the empirical observations of robustness.

We start by observing an intriguing connection between rate-encoding and the randomized smoothing framework [58]. Given a base classifier, the randomized smoothing technique ensures that the smooth classifier obtained as the expected classifier, under noise to the input, is adversarially robust. That is, for a given input $\mathbf{x} \in \mathbb{R}^d$, the smoothed classifier provides a radius within which any perturbation of the input will not result in a change in output. As the guarantee originates from the noise introduced to the input, it becomes a property of the smooth classifier rather than the base classifier. However, [79] observed that it is possible to adversarially train a smooth classifier for better empirical results.

Our Contributions: Minimizing pairwise loss of deep SNN model without the need for sampling (mining) is achieved using the algorithm from chapter 3, with the help of pseudo moving averages in the embedding space.

Considering the rate-encoded network as an estimate of the smoothed classifier to Bernoulli noise enables us to establish the first certified robustness result for SNNs. The observation that rate and constant encoded classifiers correspond respectively to the smooth and base classifiers explains the superior adversarial robustness of rate encoding.

The theoretical results further reveal that the certified radius suffers a drop with increasing latency, confirming the empirical observations. Table 5.2 reports the certified accuracies at different perturbation radii under the l_1-norm for various datasets on state-of-the-art architectures.

To further improve the adversarial robustness of rate-encoded classifiers, we adopted adversarial training methods for rate-encoded classifiers. Since rate-encoding uses stochasticity to introduce Bernoulli noise, we employ the well-known Straight Through Estimator (STE) [8, 96] for back-propagation of gradients to the inputs. Experimental findings demonstrate that adversarially trained rate-encoded networks outperform their vanilla counterpart and other state-of-the-art adversarial training algorithms [21].

This work introduces a method to minimize pairwise loss in deep SNN models without sampling, employing an algorithm from Chapter 2 that leverages pseudo moving averages in the embedding space. This contribution marks a significant advancement in the training efficiency of SNN models.

5.2 Background

5.2.1 Adversarial Attacks with Pairwise Classifiers

Given a dataset $\mathcal{S} := (\mathbf{x}_i, y_i)_{i=1}^n$ where $\mathbf{x}_i \in \mathcal{X}$ represents the inputs and $y_i \in \mathcal{Y}$ represents the class labels, with $\mathcal{Y}$ being the set of possible class labels. A deep neural network is described by the function $h : \Theta \times \mathcal{X} \to \mathcal{Y}$, where Θ denotes the set of model parameters (weights). This section explores the components of pairwise loss used in training classifiers, such as AUC (Area Under the Curve) and contrastive learning losses.

AUC Maximization in Deep Networks: The AUC metric quantifies the performance of a binary classifier in all possible thresholds, particularly useful in scenarios of imbalanced data, where traditional accuracy metrics may be misleading. The complement of the AUC, as defined in Equation 1.16, can be maximized by employing a convex surrogate function (e.g., hinge loss) to approximate the indicator function, with f being a deep neural network parameterized by θ. The loss function is formulated as follows:

$$\ell(h_\theta, (\mathbf{x}, y), (\mathbf{x}', y')) = \max(0, 1 - h_\theta(\mathbf{x}) + h_\theta(\mathbf{x}') | y = 1, y' = -1). \tag{5.1}$$

Deep Metric Learning: Contrastive loss, a widely utilized pairwise loss in deep metric learning, aims to learn embeddings that bring similar examples closer while separating dissimilar ones. Once the embeddings have been established, it becomes possible to train a classifier. Consideration can be given to classifiers that depend on distance metrics, such as KNN, which utilize the embeddings, or to a basic linear model that can be trained swiftly (refer to the experiments for additional information). The contrastive objective is detailed in Equation 5.2 assuming an embedding mapping $\tilde{h} : \Theta \times \mathcal{X} \to \mathbb{R}^D$, although alternative formulations are also available (for examples, see [40] and [53]).

$$\ell(\tilde{h}_\theta, (\mathbf{x}, y), (\mathbf{x}', y')) = \mathbb{I}_{y=y'} \left\| \tilde{h}_\theta(\mathbf{x}) - \tilde{h}_\theta(\mathbf{x}') \right\| + \mathbb{I}_{y \neq y'} \left[m - \left\| \tilde{h}_\theta(\mathbf{x}) - \tilde{h}_\theta(\mathbf{x}') \right\| \right]_+, \tag{5.2}$$

where $m > 0$ is margin parameter, $\mathbb{I}$ is the indicator function, and $[\cdot]_+$ is positive projection.

An adversarial perturbation, denoted by $\boldsymbol{\delta}$, is introduced into an input $\mathbf{x}$ given a classifier h [1] as follows:

$$\begin{aligned} &\arg\min_{\boldsymbol{\delta}} && \|\boldsymbol{\delta}\|_p && (5.3) \\ &\text{subject to} && h_\theta(\mathbf{x} + \boldsymbol{\delta}) \neq h_\theta(\mathbf{x}) && (5.4) \\ & && \boldsymbol{\delta} + \mathbf{x} \in \mathcal{X}. && (5.5) \end{aligned}$$

The objective is to find the smallest perturbation $\boldsymbol{\delta}$ in terms of the p-norm, such that the perturbed input $\mathbf{x} + \boldsymbol{\delta}$ is classified differently by the network compared to the original input

[1] In the case of metric learning, the mapping $\tilde{h}$ is transformed into a soft classifier h through the integration of a small linear layer, which operates as a mapping from the embedding space to the class space, as detailed in the experiments section.

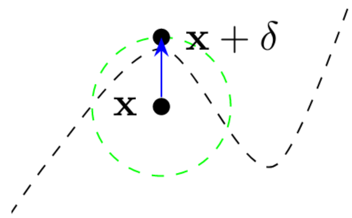


Figure 5.1: Visualization of the adversarial attack.

$\mathbf{x}$, while ensuring that the perturbed input remains within the input space $\mathcal{X}$. Given the challenging nature of solving the optimization problem outlined above, a widely employed approach is to generate such perturbations by maximizing the network loss function over the input perturbations. Specifically, the perturbation is found by solving the following optimization problem:

$$\arg \max_{\|\boldsymbol{\delta}\|_p + \|\boldsymbol{\delta}'\|_p \leq \epsilon} \ell(h_\theta, (\mathbf{x} + \boldsymbol{\delta}, y), (\mathbf{x}' + \boldsymbol{\delta}', y')). \tag{5.6}$$

However, a more straightforward approach might involve perturbing only one example while keeping the other unchanged (in this chapter, we assume $\boldsymbol{\delta}' = 0$.). The constraint $\|\boldsymbol{\delta}\|_p \leq \epsilon$ ensures that the perturbation is bounded by ϵ, limiting the extent of the modification to the input $\mathbf{x}$. This method of creating adversarial perturbations by maximizing the loss function replaces the hard constraint of causing a misclassification with a more tractable objective of maximizing the model's loss, under the assumption that increasing the loss will likely lead to a misclassification (see figure 5.1).

Given the linear characteristics of many components within neural networks, as explained by Goodfellow et al. [31], the loss function can be efficiently approximated using its first-order Taylor expansion, as follows:

$$\arg \max_{\|\boldsymbol{\delta}\|_p \leq \epsilon} \langle \nabla_x \ell(h_\theta, (\mathbf{x}, y), (\mathbf{x}', y')), \boldsymbol{\delta} \rangle = \epsilon \arg \max_{\|\boldsymbol{\delta}\|_p \leq 1} \langle \nabla_x \ell(h_\theta, (\mathbf{x}, y), (\mathbf{x}', y')), \boldsymbol{\delta} \rangle \,, \tag{5.7}$$

where the constant term of the Taylor expansion is omitted as it does not depend on $\boldsymbol{\delta}$. Consequently, the optimal solution is influenced by the dual norm of the gradient. For instance, the Fast Gradient Sign Method (FGSM) seeks to amplify the loss with a single gradient step under the maximum norm budget on $\boldsymbol{\delta}$ ($p = \infty$), yielding the solution:

$$\boldsymbol{\delta} := \epsilon \operatorname{sgn}(\nabla_x \ell(h_\theta, (\mathbf{x}, y), (\mathbf{x}', y'))). \tag{5.8}$$

When applying the Euclidean norm to $\boldsymbol{\delta}$, the solution becomes [70]:

$$\boldsymbol{\delta} := \epsilon \frac{\nabla_x \ell(h_\theta, (\mathbf{x}, y), (\mathbf{x}', y'))}{\|\nabla_x \ell(h_\theta, (\mathbf{x}, y), (\mathbf{x}', y'))\|_2}. \tag{5.9}$$

Finally, for $p = 1$, the solution is sparse (a one-hot vector scaled by ϵ), selected by $\arg\max_i |g_i|$, with g being the gradient w.r.t. input, and signed according to that dimension's sign. As evidenced in [47], sparse perturbations contribute to overfitting in the context of attack problems. To counteract this, the SLIDE approach, introduced by [86], addresses this issue by selecting the top k values from the gradient. Furthermore, for other norms, projections into the feasible set $\mathcal{X}$ are generally performed with minimal concern, as most adversaries inherently reside within this set. However, in the case of $p = 1$, projecting into the feasible set can significantly diminish the effectiveness of optimization, as perturbations are concentrated on sparse dimensions. To address this issue, alternative research has been undertaken to explore simultaneous projections in both the feasible set and the adversarial budget (e.g., see [17]).

For any chosen norm, due to the potential inaccuracy of linear approximation with large values of ϵ, the optimization process typically unfolds over several iterations of gradient ascent. In each iteration, a projection operation, denoted by Π, confines the perturbation within the l_p-norm ball of radius ϵ. When this projection step is applied to an l_∞-ball, the technique is widely recognized as the Projected Gradient Descent (PGD) attack [69].

5.2.2 Adversarial Robustness

To defend a classifier against adversarial attacks, several strategies are proposed that provide empirical robustness to ANNs [75, 34]. One popular defense strategy is adversarial training [69], where a classifier is trained with respect to adversarial examples instead of clean ones. As an optimization to find the worst adversarial perturbation is NP-hard, such a strategy, though empirically adequate, does not guarantee that no adversarial perturbation can exist for a given input.

In the context of SNNs, to improve the empirical robustness of constant-encoded networks, [57] proposed to perturb the input image separately in different time steps instead of supplying the same image repeatedly, thus avoiding any computational burden due to adversarial training. More recently, [21] proposed the adversarial training of constant encoded SNNs with additional regularization, called regularized adversarial training (RAT), which has been shown to provide higher robustness against adversarial attacks.

Certifiable Robustness: Though empirical defenses are practical, they do not ensure the absence of an adversarial perturbation. In practice, empirical defense strategies are often broken by more potent attacks. Given an input $\mathbf{x}$, a classifier h is said to be provably robust under norm l_p attacks if there exists a radius r within which any perturbation to $\mathbf{x}$ does not change the output of the classifier, i.e.,

$$\forall \mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\|_p \leq r \text{ we have, } h(\mathbf{x}') = h(\mathbf{x}). \tag{5.10}$$

Through bounding input/output of the activation and bound propagation, certified training strategies prove whether adversarial perturbation can exist within a particular radius of any input. A recent method that adopts existing literature [92, 88] to SNNs is known as (S-IBP, S-CROWN) [65].

Randomized Smoothing: One of the effective strategies to provide a probabilistic robustness certificate in a classifier agnostic way is through randomized smoothing [58], which gives provable robustness against adversarial attacks restricted under l_1 or l_2 norm.

Randomized smoothing analyses the effect of adding input noise, such as Gaussian or Laplacian, to a base classifier. A classifier that gives class probabilities as output is known as a soft classifier, in contrast to a hard classifier that returns the predicted class label. Given a (soft) base classifier $f : \mathbb{R}^d \to \mathbb{P}(\mathcal{Y})$, the Gaussian smoothing constructs a smooth classifier $g : \mathbb{R}^d \to \mathbb{P}(\mathcal{Y})$:

$$g(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\epsilon} \sim N(0,\sigma^2 \cdot I)} \left[f(\mathbf{x} + \boldsymbol{\epsilon}) \right] = \mathbb{E}_{\mathbf{z} \sim N(\mathbf{x},\sigma^2 \cdot I)} \left[f(\mathbf{z}) \right]. \tag{5.11}$$

Given, $a = \arg\max_{y \in \mathcal{Y}} g(\mathbf{x})_y$, $b = \arg\max_{y \neq a} g(\mathbf{x})_y$ and let p_a, p_b be the corresponding probabilities with their statistical estimates, $\underline{p_a} \leq p_a$ and $p_b \leq \overline{p_b}$, we have from [15]:

$$\arg\max_{y \in \mathcal{Y}} g(\mathbf{x} + \boldsymbol{\delta})_y = a \tag{5.12}$$

$$\forall \boldsymbol{\delta} : \quad \|\boldsymbol{\delta}\|_2 \leq \frac{\sigma}{2}(\Phi^{-1}(\underline{p_a}) - \Phi^{-1}(\overline{p_b})), \tag{5.13}$$

where, Φ^{-1} is the inverse Gaussian CDF. This implies that the smooth classifier, g, when deployed in the test time, can tolerate any adversarial attack with a bounded norm of radius $R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_a}) - \Phi^{-1}(\overline{p_b}))$. As probabilities p_a and p_b are difficult to compute analytically, they are estimated using Monte-Carlo (MC) simulations [15, 79], by evaluating the base classifier on multiple noisy input, i.e.,

$$\hat{g}(\mathbf{x}) \approx \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{z}_i), \qquad \text{where, } \mathbf{z}_i \sim Noise(\mathbf{x}), \tag{5.14}$$

so that the bounds $\underline{p_a} \leq p_a$ and $p_b \leq \overline{p_b}$ hold with high probability.

5.3 Adversarial Robustness using Bernoulli Smoothing

5.3.1 Randomized Smoothing via Bernoulli Noise

We propose to analyze the robustness properties of a classifier that uses Bernoulli noise as the source of randomness. The input to the smoothed classifier g can be an image $\mathbf{x} \in [0,1]^d$, where the values of the individual pixels $x_i \in [0,1]$ are bounded. For Bernoulli smoothing, the individual pixel intensity x_i can be treated as the bias of a Bernoulli random variable, z_i,

$$z_i \in \{0,1\} : \quad z_i \sim Ber(x_i), \tag{5.15}$$

which in vector notation we write as, $\mathbf{z} \sim Ber(\mathbf{x})$. The theorem below proves that we can construct a provably robust classifier g using Bernoulli noise, given a base classifier f.

Theorem 1 *Given a base classifier $f : [0,1]^d \to \mathbb{P}(\mathcal{Y})$, we construct a smooth classifier $g : [0,1]^d \to \mathbb{P}(\mathcal{Y})$, such that*

$$g(\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim Ber(\mathbf{x})} \left[f(\mathbf{z}) \right]. \tag{5.16}$$

Let $p_a = \max_{y\in\mathcal{Y}} g(\mathbf{x})_y$, $p_b = \max_{y\neq a} g(\mathbf{x})_y$, *then,*

$$\forall\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_1 < \frac{p_a - p_b}{2} \quad \arg\max_{y\in\mathcal{Y}} g(\mathbf{x}+\boldsymbol{\delta}) = a. \tag{5.17}$$

As Monte-Carlo simulations estimate the probabilities p_a and p_b, we expect to obtain bounds $p_a \geq \underline{p_a}$ and $p_b \leq \overline{p_b}$ with high probability. Since, $\underline{p_a} - \overline{p_b} \leq p_a - p_b$,

Corollary 1 *We have,*

$$\forall\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_1 < \frac{\underline{p_a} - \overline{p_b}}{2} = \underline{p_a} - 0.5 \quad \arg\max_{y\in\mathcal{Y}} g(\mathbf{x}+\boldsymbol{\delta}) = a,$$

where in the last equality we used the upper bound, $\overline{p_b} := 1 - \underline{p_a}$.

To prove the Theorem 1, we would require to use Definition1 along with lemma 2 and 6.

Definition 1 *A function* $h : \mathbb{R}^d \to \mathbb{R}$ *is said to be L-Lipschitz w.r.t. to the norm* $\|\cdot\|$, *if ,*

$$\forall\mathbf{x}, \mathbf{y} \in \mathbb{R}^d : |h(\mathbf{x}) - h(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|.$$

Equivalently, if h is differentiable, it is L-Lipschitz if and only if,

$$\forall\mathbf{x}, \quad \|\nabla h(\mathbf{x})\|_* \leq L,$$

where $\|\cdot\|_* : \mathbb{R}^d \to \mathbb{R}$ *denotes the dual norm of* $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}$*, and it is defined as:* $\|\mathbf{y}\|_* = \sup_{\|\mathbf{x}\|=1} \mathbf{x}^T\mathbf{y}$

Lemma 2 *Let* $h : \mathbb{R}^d \to \mathbb{P}(\mathcal{Y})$ *be a soft classifier and suppose it is L-Lipschitz with respect to each of the classes* $\mathbf{x} \to h(\mathbf{x})_y$ *and the norm* $\|\cdot\|$*. For a given input* $\mathbf{x}$*, if we have* $p_a = \max_{y\in\mathcal{Y}} h(\mathbf{x})_y$, $p_b = \max_{y\neq a} h(\mathbf{x})_y$, *then,*

$$\forall\boldsymbol{\delta} : \|\boldsymbol{\delta}\| < \frac{p_a - p_b}{2L} \quad \arg\max_{y\in\mathcal{Y}} h(\mathbf{x}+\boldsymbol{\delta}) = a. \tag{5.18}$$

A proof for the above can be found in Lemma 2.1 [62].

Lemma 6 *lemlip For all* y, $g(\mathbf{x})_y$ *is 1-Lipschitz under the* $\|\cdot\|_1$*-norm.*

Proof 9

$$g(\mathbf{x})_y = \mathbb{E}_{z_i \sim Ber(x_i)}\left[f(\langle z_1, z_2, \cdots, z_d\rangle)_y\right] = \sum_{z_i^{(j)} \in \{0,1\}} f(\langle z_1^{(j)}, z_2^{(j)}, \cdots, z_d^{(j)}\rangle)_y \prod_{i=1}^{d} \mathbb{P}(z_i^{(j)}|x_i)$$

$$= \sum_{z_i^{(j)} \in \{0,1\}} f(\langle z_1^{(j)}, z_2^{(j)}, \cdots, z_d^{(j)}\rangle)_y \prod_{i=1}^{d} (x_i)^{z_i^{(j)}} (1-x_i)^{1-z_i^{(j)}}$$

$$= \sum_{i \neq 1, z_i^{(j)} \in \{0,1\}} f(\langle 1, z_2^{(j)}, \cdots, z_d^{(j)}\rangle)_y \, x_1 \prod_{i \neq 1} (x_i)^{z_i^{(j)}} (1-x_i)^{1-z_i^{(j)}}$$

$$+ \sum_{i \neq 1, z_i^{(j)} \in \{0,1\}} f(\langle 0, z_2^{(j)}, \cdots, z_d^{(j)}\rangle)_y \, (1-x_1) \prod_{i \neq 1} (x_i)^{z_i^{(j)}} (1-x_i)^{1-z_i^{(j)}}.$$

Let us now take partial derivative of $g(\cdot)_y$ w.r.t. x_1, which can be generalized to other co-ordinates in a similar fashion:

$$\left|\frac{\partial g_y}{\partial x_1}\right| = \sum_{i \neq 1,\, z_i^{(j)} \in \{0,1\}} \left|\Big(f(\langle 1, z_2^{(j)}, \cdots, z_d^{(j)}\rangle)_y - f(\langle 0, z_2^{(j)}, \cdots, z_d^{(j)}\rangle)_y\Big)\right| \prod_{i \neq 1} (x_i)^{z_i^{(j)}} (1-x_i)^{1-z_i^{(j)}}$$

$$\leq \sum_{i \neq 1,\, z_i^{(j)} \in \{0,1\}} \prod_{i \neq 1} (x_i)^{z_i^{(j)}} (1-x_i)^{1-z_i^{(j)}} \qquad \textit{[diff. of probabilities is less than 1]}$$

$$= 1. \qquad \textit{[using marginal probability]}$$

Thus, we have, $\|\nabla g(\cdot)_y\|_\infty \leq 1$ which following the Definition 1, ensures $g(\cdot)_y$ is $1-$Lipschitz under $\|\cdot\|_1$-norm.

5.3.2 Multi-Bernoulli Smoothing and its Application to SNN

Armed with the proof that the smooth classifier is robust when a single Bernoulli variable is used to encode a pixel x_i, let us now move to the more practical situation of rate-encoding, where T-independent Bernoulli variables encode the input pixel. To fix the notation, consider constant encoding replicating the original input $\mathbf{x} \in \mathbb{R}^d$, T times, creating a temporal encoding that we denote using the notation $[\mathbf{x}]^T$. To be consistent, let us denote a base classifier $f_T : [0,1]^d \to \mathbb{P}(\mathcal{Y})$ that accepts the input without encoding, and further define $f_T(\mathbf{x}) = \tilde{f}_T([\mathbf{x}]^T)$, where the classifier $\tilde{f}_T : [0,1]^{T \times d} \to \mathbb{P}(\mathcal{Y})$ receives the replicated input. The notation of $\tilde{f}_T$ helps us to demonstrate the definition of the corresponding smooth classifier, which receives the replicated input $[\mathbf{x}]^T$ encoded with independent Bernoulli noise. The smooth classifier $g_T : [0,1]^d \to \mathbb{P}(\mathcal{Y})$ can now be defined as:

$$g_T(\mathbf{x}) = \mathbb{E}_{\mathbf{z}_i \sim Ber^T(x_i)}\left[\tilde{f}_T(\langle \mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_d\rangle)\right], \tag{5.19}$$

where, the smooth classifier g_T uses T independent Bernoulli variables $\mathbf{z}_i = \langle z_{i,1}, z_{i,2}, \cdots, z_{i,T}\rangle \in \{0,1\}^T$ to encode each input pixel $x_i \in [0,1]$. The notation $Ber^T(x_i)$ describes the joint Bernoulli distribution of T variables, each with the same bias x_i. We can show now:

Lemma 7 *For all y, $g_T(\mathbf{x})_y$ is $T3^{T-1}$-Lipschitz under $\|\cdot\|_1$-norm.*

Proof 10

$$\begin{aligned}
g_T(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}_i \sim Ber^T(x_i)}\left[\tilde{f}(\langle \mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_d\rangle)\right] \\
&= \sum_{\mathbf{z}_i \in \{0,1\}^T} \tilde{f}(\langle \mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_d\rangle) \prod_{i=1}^{d} \prod_{t=1}^{T} \mathbb{P}(z_{i,t}|x_i) \\
&= \sum_{\mathbf{z}_i \in \{0,1\}^T} \tilde{f}(\langle\langle z_{1,1}, \cdots, z_{1,T}\rangle, \mathbf{z}_2, \cdots \mathbf{z}_d\rangle) \prod_{t=1}^{T} x_1^{z_{1,t}}(1-x_1)^{1-z_{1,t}} \prod_{i\neq 1} \prod_{t=1}^{T} \mathbb{P}(z_{i,t}|x_i).
\end{aligned}$$

Now, let us denote, $c_k(\mathbf{z}_2, \cdots, \mathbf{z}_d) = \sum_{\|\mathbf{z}_1\|_1 = k} \tilde{f}(\langle\langle\cdot\rangle, \mathbf{z}_2, \cdots, \mathbf{z}_d\rangle)_y$, the sum of $\tilde{f}$ for different possible values that $\mathbf{z}_1$ can assume, with the restriction that there are k 1's out of T binary values. Thus,

$$\begin{aligned}
&0 \le c_0(\mathbf{z}_2, \cdots, \mathbf{z}_d) := \tilde{f}(\langle\langle 0, 0, \cdots, 0\rangle, \mathbf{z}_2, \cdots, \mathbf{z}_d\rangle)_y \le \binom{T}{0} \\
&0 \le c_1(\mathbf{z}_2, \cdots, \mathbf{z}_d) := \tilde{f}(\langle\langle 1, 0, \cdots, 0\rangle, \mathbf{z}_2, \cdots, \mathbf{z}_d\rangle)_y + \cdots + \tilde{f}(\langle\langle 0, 0, \cdots, 1\rangle, \mathbf{z}_2, \cdots, \mathbf{z}_d\rangle)_y \le \binom{T}{1} \\
&\qquad\qquad\qquad\qquad \vdots \\
&0 \le c_T(\mathbf{z}_2, \cdots, \mathbf{z}_d) := \tilde{f}(\langle\langle 1, 1, \cdots, 1\rangle, \mathbf{z}_2, \cdots, \mathbf{z}_d\rangle)_y \le \binom{T}{T}
\end{aligned}$$

and, $P(x_2, \cdots, x_d) = \prod_{i\neq 1} \prod_{t=1}^{T} P(z_{i,t}|x_i)$.

So that, we may write after expanding the expectation over the values that variable $\mathbf{z}_1$ may take:

$$\begin{aligned}
g_T(\mathbf{x}) &= \sum_{\mathbf{z}_i \in \{0,1\}^T} \tilde{f}(\langle\langle z_{1,1}, \cdots, z_{1,T}\rangle, \mathbf{z}_2, \cdots \mathbf{z}_d\rangle) \prod_{t=1}^{T} x_1^{z_{1,t}}(1-x_1)^{1-z_{1,t}} P(x_2, \cdots, x_d) \\
&= \sum_{i\neq 1, \mathbf{z}_i \in \{0,1\}^T} \sum_{k=0}^{T} c_k x_1^k (1-x_1)^{T-k} P(x_2, \cdots, x_d).
\end{aligned}$$

Let us denote $h : [0,1] \to \mathbb{R}$ as the function :

$$\begin{aligned}
h_T(x) &= \sum_{k=0}^{T} c_k x^k (1-x)^{T-k} = \sum_{k=0}^{T} c_k x^k \sum_{j=0}^{T-k} \binom{T-k}{j} (-x)^j \\
&= \sum_{k=0}^{T} \sum_{j=0}^{T-k} c_k \binom{T-k}{j} x^{k+j} (-1)^j \\
&= \sum_{l=0}^{T} \sum_{m=0}^{l} c_m \binom{T-m}{l-m} x^l (-1)^{l-m} \quad \textit{[after re-indexing]} \\
\frac{dh_T(x)}{dx} &= \sum_{l=1}^{T} l x^{l-1} \sum_{m=0}^{l} c_m \binom{T-m}{l-m} (-1)^{l-m} \\
&= \sum_{l=1}^{T} l x^{l-1} \sum_{j=0}^{l} c_{l-j} \binom{T-l+j}{j} (-1)^j \quad \textit{[setting, j=l-m]}
\end{aligned}$$

$$\begin{aligned}
&\leq \sum_{l=1}^{T} l x^{l-1} \sum_{j=0}^{l/2} \binom{T}{l-2j} \binom{T-l+2j}{2j} \quad \textit{as,} 0 \leq c_j \leq \binom{T}{j} \\
&= \sum_{l=1}^{T} l x^{l-1} \sum_{j=0}^{l/2} \frac{T!}{(l-2j)!(T-l)!(2j)!} \\
&= \sum_{l=1}^{T} l x^{l-1} \binom{T}{l} \sum_{j=0}^{l/2} \frac{l!}{(l-2j)!(2j)!} \\
&= \sum_{l=1}^{T} l x^{l-1} \binom{T}{l} 2^{l-1} = T \sum_{l=1}^{T} \binom{T-1}{l-1} (2x)^{l-1} = T(1+2x)^{T-1} \leq T3^{T-1}.
\end{aligned}$$

Taking the partial derivative of g w.r.t. x_1,

$$\begin{aligned}
\left| \frac{\partial g_T(\cdot)_y}{\partial x_1} \right| &= \sum_{i \neq 1, \mathbf{z}_i \in \{0,1\}^T} |h'_T(x_1)| \, P(x_2, \cdots, x_d) \\
&\leq T3^{T-1} \sum_{i \neq 1, \mathbf{z}_i \in \{0,1\}^T} P(x_2, \cdots, x_d) \\
&= T3^{T-1}.
\end{aligned}$$

Thus, we have, $\|\nabla g\|_\infty \leq T3^{T-1}$ which following the Definition 1, ensures g is $T3^{T-1}-$ Lipschitz under the norm $\|\cdot\|_1$.

Theorem 3 *For the smooth classifier g_T defined as above, and $p_a = \max_{y \in \mathcal{Y}} g_T(\mathbf{x})_y$, $p_b = \max_{y \neq a} g_T(\mathbf{x})_y$, we have:*

$$\forall \boldsymbol{\delta} : \|\boldsymbol{\delta}\|_1 < \frac{p_a - p_b}{2T\, 3^{T-1}} \quad \arg\max_{y \in \mathcal{Y}} g_T(\mathbf{x} + \boldsymbol{\delta}) = a. \tag{5.20}$$

The proof of the Theorem follows from lemma 2 and 7. It is interesting to note that Theorem 1 is a special case of Theorem 3 when $T = 1$.

Corollary 2 *Similarly to Corollary 1, if we estimate* $p_a \geq \underline{p_a}$ *with high probability, then the robustness guarantee on the perturbation* $\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_1 < \frac{\underline{p_a}-0.5}{T\,3^{T-1}}$*, holds with the same probability.*

One may highlight that the certified radius drops rapidly with T, which, we agree, can perhaps be improved with a tighter bound on the Lipschitzness. Also, the largest l_1-radius on which an input can obtain a robustness guarantee is upper bounded by 0.5, with $T = 1$. To put it in perspective, it allows a change of $\frac{127}{255}$, which, assuming 8-bit encoding of pixels, allows a single pixel to be changed by 127 or 127 pixels to be changed by 1. In comparing the results with Gaussian smoothing, one may find that their guarantee can hold for arbitrary radius. However, certificates for larger radii require a significantly large number of MC simulations. Moreover, such results are impossible for discrete random variables such as Bernoulli.

5.3.3 Prediction and Certified Robustness

Estimation of the smooth classifier $g_T(\mathbf{x})$ requires MC simulations, which corresponds to the evaluation of the base classifier f_T on the rate-encoded input. To evaluate g_T, we use the predict and certify functions described in Algorithms 3 and 4, which are similar to those of [15], with the exception that Gaussian noise is replaced by multi-Bernoulli noise. The sub-routine RateEncode$(f_T, \mathbf{x}, m)$ evaluates the classifier f_T after rate-encoding the input $\mathbf{x}$. The process is repeated m times, and the vector *counts* records the number of times a particular class is predicted. For proof that Algorithm 1 makes an incorrect prediction $\hat{c}_a \neq g_T(\mathbf{x})$ with a probability upper bounded α, refer to Proposition 1 [15] which in turn uses results from [45]. Algorithm 4 describes the certify function, which works in two stages. First, using m_0 MC simulations, it identifies the predicted class $\hat{c}_a$, then to find the lower bound $\underline{p_a}$, it uses other simulations m. The proof that Algorithm 2 makes an incorrect prediction with probability upper bounded by α can also be found in Proposition 2 [15].

Certified Test Accuracy: For each input $\mathbf{x}$, the certify function abstains or returns the predicted class with a radius l_1, within which no adversarial example exists. Two conditions are verified at each input data point to obtain the certified test accuracy of a model at a given radius r. First, the predicted class should match the target label y; second, the radius returned by the certify function should be greater than or equal to r. The fraction of test data that satisfies both conditions without abstaining gives the certified test accuracy of a model. Observe that this is an estimate of the true certified test accuracy, as we only have an estimate of the smooth classifier g_T. However, the estimate can be improved by reducing the error rate α given Algorithms 1 and 2.

Algorithm 3 predict $g_T(\mathbf{x})$

Require: $\tilde{f}_T, \mathbf{x}, m, \alpha$
  $counts \leftarrow \text{RateEncode}(\tilde{f}_T, \mathbf{x}, m)$
  $\hat{c}_a, \hat{c}_b \leftarrow$ top two indicies in $counts$
  $n_a, n_b \leftarrow counts[\hat{c}_a], counts[\hat{c}_b]$
  if $\text{BinomPValue}(n_a, n_a + n_b, 0.5) \leq \alpha$ **then**
    return $\hat{c}_a$
  else
    return abstain
  end if

Algorithm 4 certify g_T around $\mathbf{x}$

Require: $\tilde{f}_T, \mathbf{x}, m_0, m, \alpha$
  $counts0 \leftarrow \text{RateEncode}(\tilde{f}_T, \mathbf{x}, m_0)$
  $\hat{c}_a \leftarrow$ top index in $counts0$
  $counts \leftarrow \text{RateEncode}(\tilde{f}_T, \mathbf{x}, m)$
  $\underline{p_a} \leftarrow \text{LowerConfBound}(counts[\hat{c}_a], m, 1-\alpha)$

  if $\underline{p_a} \geq 0.5$ **then**
    return prediction $\hat{c}_a$, radius $\frac{\underline{p_a}-0.5}{T3^{T-1}}$
  else
    return abstain
  end if

5.4 Adversarial Training of SNN Classifier

The notion of adversarial training can be seen as a min-max problem, where the objective is first maximized with respect to bounded input perturbations and then minimized over the model parameters.

$$\min_{\theta} \frac{1}{|S|} \sum_{\mathbf{x},y \in S} \sum_{\mathbf{x}',y' \in S} \max_{\|\delta\| \leq \epsilon} \ell(h_\theta((\mathbf{x} + \boldsymbol{\delta}), y), (\mathbf{x}', y')). \tag{5.21}$$

Since exactly solving the inner maximization is often not possible, one of the most common approaches of optimization is to alternately solve (approximately) the inner maximization with the help of adversarial attacks and then partially solve the outer minimization using methods such as mini-batch stochastic gradient descent (SGD). Each mini-batch SGD step would require obtaining adversarial images $\mathbf{x} + \delta^*$, computed with respect to the latest network weight that is updated after the SGD step.

As we would like to use the smooth classifier as the final classifier for robustness properties, it is natural to choose $h_\theta = g_{T,\theta}$ in adversarial training, which has been demonstrated to be an effective approach for ANN [79]. However, there are two difficulties. First, we can only estimate the classifier g using $\hat{g}$, as given in Equation 5.14, thus forcing us to use $h_\theta = \hat{g}_{T,\theta}$. For the computational efficiency of the adversarial training process, we use $m = 1, \alpha = 1$ to obtain a prediction of the smooth classifier, which corresponds to a single evaluation of the rate-encoded classifier. In constant encoding, the performance of adversarial training corresponds to setting $h_\theta = f_{T,\theta}$.

Secondly, to carry out the adversarial attack for inner maximization, we need to compute the gradient of $\hat{g}$ with respect to the input, which, following the chain rule, requires back-propagating through the stochastic node that rate-encodes the input $\mathbf{x}$. We employ the Straight Through Estimator [41, 8], which estimates the gradient of a stochastic node assuming that there is an identity function in the backward pass of the node, allowing gradient-based attacks to find adversarial examples.

5.5 Experiments

5.5.1 Experimental Framework

Competitors: We compare the performance of the proposed methods with state-of-the-art adversarial training algorithms in SNN. We refer to the models trained with un-perturbed data as Clean and when perturbed with Gaussian as GN. We generate adversarial images using FGSM and PGD attacks implemented using back-propagation through time (BPTT) with $\epsilon = \frac{8}{255}$ and refer to the corresponding adversarially models by the attack name followed by the abbreviation of encoding, where (C) and (R) stand for constant and rate, respectively. To identify the attacks separately from the corresponding adversarial training algorithms, they are named in small letters, e.g., the FGSM attack assuming a rate-encoded model is referred to as fgsm(R).

In our implementation, we seamlessly integrate an additional linear layer by the LogisticRegression class from the scikit-learn library with a multinomial setup and the Softmax function for multiclass classification. The 'newton-cg' solver was chosen for its efficient handling of the Softmax loss, with a limit of 1000 iterations to ensure rapid convergence. This setup, focusing on fine-tuned embeddings, provides a fast and effective final classification step. Algorithm 5 illustrates the training strategy.

Algorithm 5 Online Adversarial Training with Contrastive Loss

Require: Dataset $\mathcal{S}$, Adversarial budget p, Embedding dimension D, Number of Classes C
Ensure: Trained model with enhanced robustness
  Initialize a pseudo moving average for each class as a basis vector scaled by D, $\{D\mathbf{e}_i \in \mathbb{R}^D | i = 1, \ldots, C\}$.
  for each (x, y) in $\mathcal{S}$ **do**
    Find adversarial perturbation by solving opt. problem 5.7 with budget p
    Compute contrastive loss using Eq. 4.7, with the pseudo moving average.
    Minimize the contrastive loss using gradient-method
    Update pseudo moving averages based on new embeddings
  end for
  Train a one-layer neural network with Softmax for final classification step

Dataset and Network Architecture: We use a standard static image dataset such as CIFAR-10 and CIFAR-100 [55]. They are temporally encoded using constant or rate encoding for the SNN classifier to be trained at $T = 4$. We use the VGGSNN architecture. The VGGSNN is based on VGG-19, where the architecture includes 19 layers with learnable weights; 16 convolutional layers followed by a fully connected layer (classifier's layers were excluded). The convolutional layers are organized into blocks, each followed by max-pooling to reduce dimensionality. [20].

Clean and Robust Accuracy: The clean accuracy of a model is its accuracy on uncorrupted test images. To evaluate the smooth classifier, we employ the Algorithm 3 for prediction, which returns the output of a conventional rate-encoded network when

T=4	CIFAR-10, Constant Encoding				CIFAR-100, Constant Encoding			
Attack	Clean	GN	FGSM(C)	PGD(C)	Clean	GN	FGSM(C)	PGD(C)
clean	85.01	85.41	70.75	67.20	55.54	54.58	44.21	44.6
gn	83.86	84.57	70.40	66.48	55.75	56.25	44.17	44.15
fgsm(C)	40.15	41.21	56.51	46.89	29.74	29.74	30.9	31.3
pgd(C)	**38.14**	**37.10**	**52.75**	**44.58**	**20.18**	**19.74**	**24.15**	**25.7**
T=4, m=1	CIFAR-10, Rate Encoding				CIFAR-100, Rate Encoding			
Attack	Clean	GN	FGSM(R)	PGD(R)	Clean	GN	FGSM(R)	PGD(R)
clean	69.31	67.85	66.71	65.98	53.17	52.17	39.89	39.18
gn	69.57	67.52	66.21	65.88	53.05	52.47	39.17	38.14
fgsm(R)	**44.21**	**45.01**	**53.95**	**52.39**	31.04	33.52	33.74	35.91
pgd(R)	45.14	44.54	54.3	52.71	**26.14**	**26.8**	**29.05**	**29.95**
T=4, m=10	CIFAR-10, Rate Encoding				CIFAR-100, Rate Encoding			
Attack	Clean	GN	FGSM(R)	PGD(R)	Clean	GN	FGSM(R)	PGD(R)
clean	73.36	72.54	70.86	69.87	54.8	53.18	40.18	40.4
gn	73.65	72.1	70.89	70.18	54.62	53.74	40.17	39.5
fgsm(R)	**47.14**	**47.28**	**55.37**	**53.62**	0.3201	34.15	34.15	36.01
pgd(R)	48.55	48.1	55.91	54.27	**27.85**	**27.05**	**30.5**	**30.9**
T=8	CIFAR-10, Constant Encoding				CIFAR-100, Constant Encoding			
Attack	Clean	GN	FGSM(C)	PGD(C)	Clean	GN	FGSM(C)	PGD(C)
clean	85.30	85.47	69.79	69.12	58.74	57.98	47.54	46.8
gn	84.08	84.53	69.58	67.92	57.58	58.15	46.75	46.1
fgsm(C)	42.12	43.80	52.75	46.15	31.2	31.8	32.8	33.4
pgd(C)	**35.78**	**35.31**	**48.48**	**42.17**	**22.1**	**21.5**	**26.4**	**27.5**
T=8, m=1	CIFAR-10, Rate Encoding				CIFAR-100, Rate Encoding			
Attack	Clean	GN	FGSM(R)	PGD(R)	Clean	GN	FGSM(R)	PGD(R)
clean	73.10	72.50	68.93	70.59	57.39	56.14	42.14	41.2
gn	71.35	70.57	70.15	70.35	56.14	53.24	39.69	39.05
fgsm(R)	**42.84**	**43.81**	53.14	54.20	30.14	32.01	33.5	34.1
pgd(R)	44.68	44.39	**51.42**	**49.57**	**25.14**	**26.1**	**28.15**	**29.92**

Table 5.1: Experiments on CIFAR-10 and CIFAR-100 datasets demonstrate that under the strongest attacks, rate encoded classifiers offer superior robustness compared to the constant encoded counterpart. The columns represent different training (adversarial) procedures, while the rows stands for different adversarial attacks.

$m = 1, \alpha = 1$. To demonstrate the smooth classifier, we also make predictions with $m = 10, \alpha = 1$. To avoid the scenario where the classifier abstains from the prediction due to low statistical confidence, we set $\alpha = 1$, making the accuracies comparable to the constant models. The empirical robust accuracy is the prediction accuracy when input images are corrupted with specific adversarial attacks.

5.5.2 Comparison of Empirical Robust Accuracy

Table 5.1 presents a comparison between networks using constant encoding and those employing rate encoding, across various training algorithms. Column-wise, we identify the lowest accuracy observed under any attack for each training approach. This lowest robust accuracy serves as an indicator of the model's empirical resilience when faced with diverse attacks. For example, in CIFAR-10, Clean(R) is 7% more robust than Clean(C), while under adversarial training, FGSM(R), PGD(R) improve the minimum robust accuracy by 5%. For CIFAR-100, the improvement in robust accuracy is 5% for Clean(R), 6% for

T=4	CIFAR-10, Rate Encoding					
$r * 108 =$	0	0.1	0.2	0.3	0.4	0.45
CLEAN(R)	73.1|0	65.44|23.38	61.77|30.28	55.88|39.04	47.8|49.53	37.08|61.82
PGD(R)	70.59|0	64.1 |23.56	59.64|31.0	53.92|39.82	45.52|51.07	36.31|61.94
T=4	CIFAR-100, Rate Encoding					
$r * 108 =$	0	0.1	0.2	0.3	0.4	0.45
CLEAN(R)	41.66|0	35.92|12.34	30.88|21.56	25.94 | 32.68	21.40|43.79	16.52|55.91
PGD(R)	38.1|0	33.2|12.4	29.4|20.6	24.8|31.2	20.5|42.7	15.6|54.9

Table 5.2: presents the certified test accuracy|abstain for rate-encoded SNN models at different l_1-norm radii for various static datasets.

FGSM(R), and 5% for PGD(R), considering predictions using $m = 10$.

5.5.3 Provable Robustness via Certified Accuracy

We provide the empirical certified test accuracy of the rate-encoded smooth classifier for perturbation bounded under l_1-norm radius. As given in Table 5.2, the radius varies between $\left[\frac{0.1}{L(T)}, \frac{0.5}{L(T)}\right]$, with $L(T) = T3^{T-1}$ representing the Lipschitzness of the smooth classifier as in Lemma 7. We use the certify function with parameters $m_0 = 10$, $m = 100$, and $\alpha = 0.01$ to find the certified accuracy of the rate-encoded models. The certified radii obtained are not entirely vacuous. For example, at $T = 4$, we have $L(T) = 108$, so we can obtain a certified accuracy at the radius $\frac{0.45}{108}$ which is larger than $\frac{1}{255}$. Assuming 8-bit pixels allows a single pixel in the image to be changed by 1. There remains also scope to obtain the certificate at lower latencies, as highlighted in the section 5.3.2.

5.6 Applicability for Pointwise trained-Models

The key to applying randomized smoothing across different model training types lies in the ability of the model to produce soft mappings—essentially, probabilities over classes. This characteristic ensures the method's relevance not just for pairwise losses but also for models optimized with pointwise losses.

In essence, the adaptability of randomized smoothing across various loss functions underscores its potential as a versatile technique for improving the robustness of the SNN. By demonstrating its effectiveness in a range of contexts, from traditional classification to pairwise and metric learning scenarios, we highlight the technique's broad applicability in securing models against input perturbations, irrespective of the underlying loss function used during training. In this exploration, we initially approach the robustness of neural networks with an emphasis on models trained using pairwise losses, such as contrastive loss. However, a crucial insight of our study is that the robustness analysis we undertake—predicated on the framework of randomized smoothing—is not confined solely to pairwise or pointwise models.

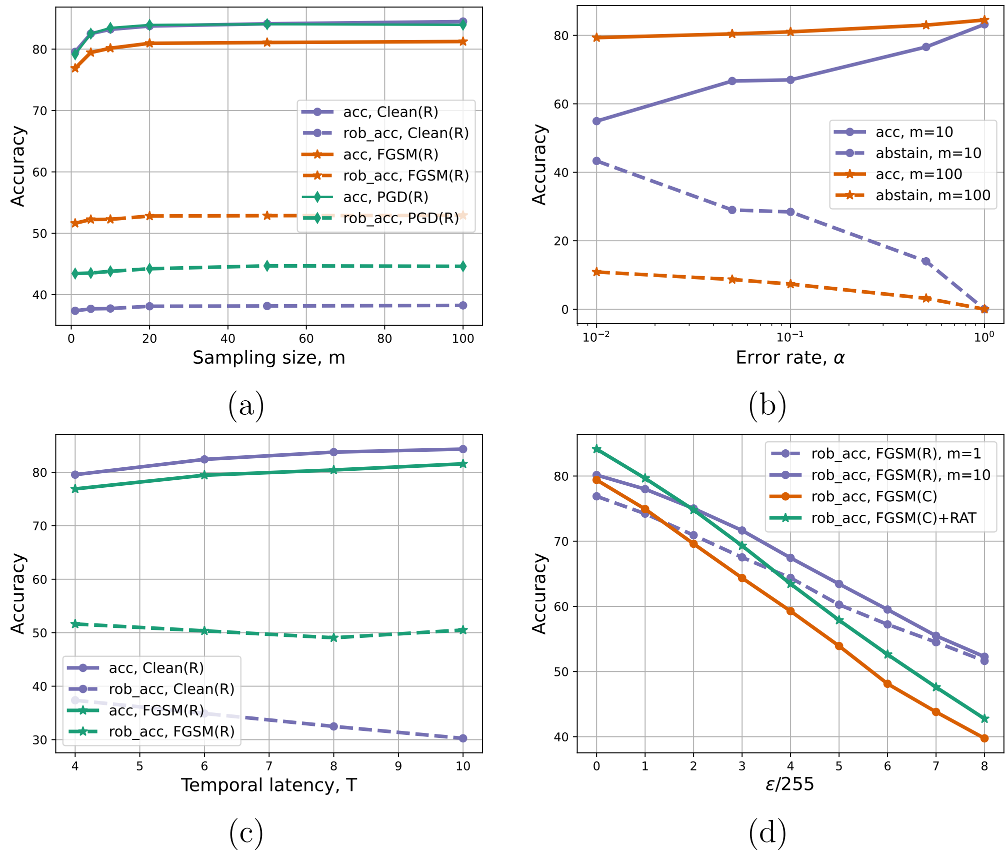


Figure 5.2: Figure (a) shows the clean and robust accuracy of smooth classifiers improve with m and saturates quickly before m=20. The improvement is more significant for clean accuracy compared to robust accuracy. Figure (b) shows the effect of error rate α on the prediction. For $m = 100$, the prediction accuracy does not deteriorate at $\alpha = 10^{-2}$, as it helps to avoid the high abstain rate suffered in the case of $m = 10$. Figure (c) empirically confirms that with a larger T, the clean accuracy of the rate-encoded classifiers improves, while the robust accuracy decreases. Figure (d) compares the robustness of the FGSM(R) model with its competitors, with the radius ϵ of l_∞-norm attacks varying in the range $[\frac{0}{255}, \frac{8}{255}]$. The prediction of the FGSM(R) model with $m = 10$ offers a better robust accuracy for most cases, while the prediction using $m = 1$ beats the competition in the higher range of ϵ.

The realization that both pairwise and pointwise models can be unified under the same robustness analysis framework stems from their shared capability to map inputs to a probability space. This mapping is crucial for the application of randomized smoothing, which assesses the stability of probability outputs in response to input perturbations. Whether the model is trained to evaluate individual data points or pairs, the fundamental requirement for applying our robustness certification methods is the production of these probabilistic outputs. Consequently, we introduce additional experiments with cross entropy loss training, on a variety of datasets in the following section.

T=4	CIFR-10, Constant Encoding				CIFAR-100, Constant Encoding			
Attack	Clean	GN	FGSM(C)	PGD(C)	Clean	GN	FGSM(C)	PGD(C)
clean	92.15	91.7	79.4	79.15	72.01	70.19	54.31	53.38
gn	90.62	91.25	78.84	78.28	66.46	69.64	53.87	53.02
fgsm(C)	10.68	15.95	48.86	48.56	3.19	5.35	26.05	26.49
pgd(C)	**0.1**	**1.24**	**39.75**	**41.65**	**0.04**	**0.45**	**21.12**	**22.61**
fgsm(R)	81.01	85.13	77.33	75.69	45.4	59.23	49.63	50.81
pgd(R)	85.14	85.07	77.42	75.29	55.82	59.73	49.57	50.69
T=4	CIFAR-10, Constant Encoding, RAT				CIFAR-100, Constant Encoding, RAT			
Attack	Clean	GN	FGSM(C)	PGD(C)	Clean	GN	FGSM(C)	PGD(C)
clean	91.29	90.86	84.13	83.33	70.85	68.87	58.1	58.05
gn	88.71	90.72	83.09	82.95	66.05	69.82	57.93	57.34
fgsm(C)	25.09	29.42	54.18	53.50	10.39	13.12	35.62	34.36
pgd(C)	**0.49**	**3**	**42.76**	**44.44**	**0.34**	**1.71**	**27.1**	**29.89**
fgsm(R)	84.62	87.55	81.68	80.90	53.15	64.34	56.31	55.08
pgd(R)	87.41	87.96	81.96	80.58	62.88	64.93	55.85	54.52
T=4, m=1	CIFAR-10, Rate Encoding				CIFAR-100, Rate Encoding			
Attack	Clean	GN	FGSM(R)	PGD(R)	Clean	GN	FGSM(R)	PGD(R)
clean	79.55	79.36	76.89	76.36	50.9	50.89	45.85	46.98
gn	78.62	79.07	76.39	76.77	50.43	50.77	46.4	46.05
fgsm(C)	75.57	75.34	72.03	73.99	48.36	49.12	45.16	45.15
pgd(C)	76.23	76.06	73.03	78.87	47.94	48.95	45.74	45.38
fgsm(R)	43.69	43.31	55.05	55.27	25.75	24.84	31.64	32.47
pgd(R)	**37.37**	**37.4**	**51.63**	**51.59**	**21.98**	**20.72**	**28.56**	**29.99**
T=4, m=10	CIFAR-10, Rate Encoding				CIFAR-100, Rate Encoding			
Attack	Clean	GN	FGSM(R)	PGD(R)	Clean	GN	FGSM(R)	PGD(R)
clean	83.22	83.5	80.15	80.54	55.27	55.83	49.58	50.51
gn	83.54	83.41	80.33	80.41	54.82	55.2	49.84	50.15
fgsm(C)	80.29	79.94	76.41	78.22	52.76	53.6	49.19	48.3
pgd(C)	80.6	80.05	76.73	78.59	52.54	53.73	49.07	48.8
fgsm(R)	44.93	44.89	57.89	57.85	27.61	26.32	33.46	33.88
pgd(R)	**37.74**	**37.58**	**52.27**	**53.08**	**22.79**	**21.5**	**30.03**	**31.05**

Table 5.3: Experiments on CIFAR-10 and CIFAR-100 datasets demonstrate that under the strongest attacks, rate encoded classifiers offer superior robustness compared to the constant encoded counterpart. The columns represent different training (adversarial) procedures, while the rows stands for different adversarial attacks.

5.6.1 Robustness of Cross-entropy trained-Models

Table 5.3 compares the constant encoded network under different training algorithms with the rate-encoded counterpart with cross entropy loss training. We highlight the minimum accuracy against all attacks for each training method, reported column-wise. The minimum robust accuracy represents the empirical robustness of a model when subjected to arbitrary attack. It is found that for constant-encoded models, the pgd(C) attack gives a minimum accuracy, while for the rate-encoded models, pgd(R) is the strongest. It can be observed that RAT improves the robustness of the constant-encoded models. However,

ImageNet-100, T=4	clean	gn	fgsm(C)	pgd(C)	fgsm(R)	pgd(R)
Clean(C)	72.02	71.78	4.98	**0.02**	51.18	61.1
Clean(C) +RAT	65.14	63.66	7.48	**0.06**	51.1	57.22
Clean(R), m=1	62.16	62	52.3	55.44	19.88	**12.06**
Clean(R), m=10	64.18	64.1	58.42	60.24	28.04	**19.86**
SVHN, T=4	**clean**	**gn**	**fgsm(C)**	**pgd(C)**	**fgsm(R)**	**pgd(R)**
Clean(C)	95.36	94.71	26.6	**3.11**	65.61	70.56
Clean(C) +RAT	96.17	95.5	40.43	**4.03**	80.33	78.44
Clean(R), m=1	86.09	85.67	77.49	77.82	43.68	**37.44**
Clean(R), m=10	91.7	91.44	84.51	84.87	46.58	**38.79**

Table 5.4: Experiments on ImageNet-100 and SVHN show that rate-encoded models can offer reasonable clean accuracy and significantly higher robust accuracy against the strongest attacks.

rate-encoded models significantly improve robust accuracy compared to RAT. For example, in CIFAR-10, Clean(R) is 37% more robust than Clean(C), while under adversarial training, FGSM(R), PGD(R) improve the minimum robust accuracy by 9% and 18%, respectively, compared to RAT. For CIFAR-100, the improvement in robust accuracy is 22% for Clean(R), 3% for FGSM(R), and 13% for PGD(R), considering predictions using $m = 10$.

A common criticism of the rate-encoded model is that it offers a lower accuracy on clean images than constant encoding. Fig.5.2(a) shows on CIFAR-10 that the prediction of the rate-encoded models can improve significantly with a better estimation of the smooth model at the test time. The same observation can be made from Table 5.3 where shifting from m=1 to 10 improves the clean accuracy by 4- 5 % across all rate-encoded models. A further defense for rate-encoded models comes from Fig.5.2(d), where we vary the radius of the l_∞-attack between $[\frac{0}{255}, \frac{8}{255}]$, where $\epsilon = 0$, corresponds to clean images. It can be observed that constant encoded models quickly lose accuracy even for small perturbations, showing the relevance of rate-encoded models when there is a possibility of attack. Fig.5.2(c) studies the effect of latency on robustness. Separately training the Clean(R) and FGSM(R) models in various $T \in \{4, 6, 8, 10\}$ and measuring their clean and robust accuracy (against pgd(R)). The results show that, although the clean accuracy improves with latency, the robust accuracy drops, which confirms our theoretical results. In Section 5.6, we provide full results for $T = 8$ to further confirm the same.

Results on other Datasets: We further compare the robustness of constant vs. rate-encoding in larger datasets. Often, for large datasets such as ImageNet, adversarial training can be computationally challenging due to the size of the dataset, which gets multiplied by the temporal dimension of SNN. However, rate-encoding encoded classifiers trained with clean images offer robustness properties and are computationally easier to train. Table5.4 provides empirical robustness results for Imagenet-100 and the SVHN dataset, where Clean(R) obtains significantly higher robust accuracy than other feasible methods.

T=4	CIFAR-10, Rate Encoding					CIFAR-100, Rate Encoding				
$r * 108 =$	0.1	0.2	0.3	0.4	0.45	0.1	0.2	0.3	0.4	0.45
Clean(R)	72.48	67.62	60.29	49.71	37.01	41.8	37.62	32.02	25.36	18.49
PGD(R)	73.23	68.5	61.69	51.58	38.78	41.48	37.31	32.15	25.65	18.68
T=4	ImageNet-100, Rate Encoding					SVHN, Rate Encoding				
Clean(R)	51.38	46.7	41.04	33.54	24.52	81.67	76.85	69.15	56.67	41.18

Table 5.5: presents the certified test accuracy for rate-encoded SNN models at different l_1-norm radii for various static datasets.

5.6.2 Provable Robustness via Certified Accuracy

We provide the empirical certified test accuracy of the rate-encoded smooth classifier for perturbation bounded under l_1-norm radius. As given in Table 5.5, the radius varies between $\left[\frac{0.1}{L(T)}, \frac{0.5}{L(T)}\right]$, with $L(T) = T3^{T-1}$ representing the Lipschitzness of the smooth classifier as in Lemma 7. We use the certify function with parameters $m_0 = 10$, $m = 100$, and $\alpha = 0.01$ to find the certified accuracy of the rate-encoded models. The certified radii obtained are not entirely vacuous. For example, at $T = 4$, we have $L(T) = 108$, so we can obtain a certified accuracy at the radius $\frac{0.45}{108}$ which is larger than $\frac{1}{255}$. Assuming 8-bit pixels allows a single pixel in the image to be changed by 1. There remains also scope to obtain the certificate at lower latencies, as highlighted in the section 5.3.2. Fig. 5.3 shows a comparison between certified accuracy and robust accuracy under attack l_1.

5.6.3 Comparison at Higher Latency

Table 5.6 compares the constant encoded SNN under different training algorithms with their rate encoded counterpart at $T = 8$. Similar to Table 5.3 with $T = 4$, adversarially trained rate-encoding demonstrates superior accuracy compared to their constant encoded counterparts. Also, it can be observed that for most cases in rate-encoded models, the clean accuracy has improved when compared to $T = 4$, while the robust accuracy has dropped. Our theoretical findings for rate-encoded models also confirm that rate-encoded classifiers robustness decreases with higher latency.

5.6.4 Comparison of Empirical Robust Accuracy

We further compare the test fgsm/pgd attacks implemented with different back-propagation techniques such as, Back Propagation Through Time (BPTT) [91], Backward Pass Through Rate (BPTR) [21], Rate Gradient Attack (RGA) [11], and report the corresponding accuracies in the respective order. Table 5.7 shows the experiments conducted on CIFAR-10, where BPTT produces the strongest attacks.

T=8	CIFAR-10, Constant Encoding				CIFAR-100, Constant Encoding			
Attack	Clean	GN	FGSM(C)	PGD(C)	Clean	GN	FGSM(C)	PGD(C)
clean	90.29	92.28	81.71	82.69	73	71.2	55.21	56.96
gn	88.93	91.85	81.26	82.56	68.67	71	53.04	56.47
fgsm(C)	8.45	18.04	48.79	46.33	4.17	6.47	32.39	25.99
pgd(C)	**0.07**	**1.11**	**39.41**	**41.04**	**0.1**	**0.67**	**17.37**	**23.36**
fgsm(R)	71.03	84.77	79.17	80.43	49.36	62.03	52.06	53.39
pgd(R)	75.54	84.49	79.48	81.35	61.36	62.84	51.01	54.55
T=8,m=1	CIFAR-10, Rate Encoding				CIFAR-100, Rate Encoding			
Attack	Clean	GN	FGSM(R)	PGD(R)	Clean	GN	FGSM(R)	PGD(R)
clean	83.77	83.57	80.43	80.64	54.26	48.73	48.78	50.85
gn	76.94	82.86	80.03	79.76	53.34	48.25	48.58	50.41
fgsm(C)	70.49	78.85	76.63	77.05	51.36	42.6	47.87	48.34
pgd(C)	71.76	80.11	77.8	77.86	50.9	44.1	47.46	49.22
fgsm(R)	36.05	41.01	54.42	52.02	24.95	17.74	33.51	28.87
pgd(R)	**32.48**	**31.82**	**49.06**	**51.29**	**21.23**	**14.13**	**33.41**	**28.31**

Table 5.6: Experiments on CIFAR-10 and CIFAR-100 datasets, with $T = 8$, demonstrate that under the strongest attacks, rate encoded classifiers offer superior robustness compared to the constant encoded counterpart. Columns represent different training (adversarial) procedures, while rows represent different adversarial attacks.

5.6.5 Comparison Under Gaussian Noise Attack

The results in Table 5.3 with the gn attack use a Gaussian perturbation: $x + \delta$, where $\delta \sim N(0, \sigma^2)$, with $\sigma = 8/255 \approx 0.031$. The superior results obtained there for the constant encoded models against Gaussian noise attacks can quickly break down as we increase the strength of the attacks, as computed in Table5.8. For example, at $\sigma = 0.1$, CLEAN (R) with m=10, provides 21.77% higher accuracy than CLEAN (C)+RAT.

5.6.6 Comparison Under ANN-SNN Converted Models

The theory of randomized smoothing holds irrespective of the base classifier we choose. That is, whether we use the base classifier f (in Eqn. 1) from an adversarially trained model or a converted model, the corresponding smooth classifier g, will be robust against adversarial perturbation, i.e., $g(x+\delta) = g(x)$. However, if the base classifier is not suitable, the prediction $g(x)$ can be incorrect, leading to poor clean and robust accuracy.

We conducted experiments ANN-to-SNN converted VGG-16 models on the CIFAR-10 dataset using recent conversion technique [12]. We find that under rate-encoding, converted SNNs have low-clean accuracy at small latency T, which is a limitation of the base classifier, which is trained to work with direct inputs, without the additional Bernoulli noise. This can even happen to constant encoded SNN trained on clean images if we add large Gaussian noise to the input (see Table 5.8)

T=4	CIFAR-10, Constant Encoding			
Attack	Clean	GN	FGSM(C)	PGD(C)
clean	92.15	91.7	79.4	79.15
gn	90.62	91.25	78.84	78.28
fgsm(C)	10.68 | 16.08|10.68	15.95 | 22.40 | 15.53	48.86 | 55.79 | 52.73	48.56 | 55.55 | 52.12
pgd(C)	**0.1** | 1.71 | 0.47	**1.24** | 6.64 | 2.76	**39.75** | 49.70 | 47.62	**41.65** | 49.57 | 48.30
fgsm(R)	81.01 | 84.36 | 80.18	85.13 | 88.39 | 83.10	77.33 | 78.08 | 76.78	75.69 | 76.93 | 74.90
pgd(R)	85.14 | 88.84 | 82.49	85.07 | 88.73 | 82.48	77.42 | 78.40 | 77.02	75.29 | 76.46 | 75.13
T=4	CIFAR-10, Constant Encoding, RAT			
Attack	Clean	GN	FGSM(C)	PGD(C)
clean	91.29	90.86	84.13	83.33
gn	88.71	90.72	83.09	82.95
fgsm(C)	25.09 | 25.90 | 15.40	29.42 | 34.45 | 20.74	54.18 | 64.46 | 55.86	53.50 | 63.97 | 56.15
pgd(C)	0.49 | 5.97 | **0.45**	**3** | 15.93 | 3.23	**42.76** | 58.92 | 48.66	**44.44** | 58.78 | 50.05
fgsm(R)	84.62 | 84.06 | 82.94	87.55 | 88.59 | 87.11	81.68 | 82.56 | 81.45	80.90 | 82.53 | 0.63
pgd(R)	87.41 | 88.29 | 85.75	87.96 | 89.15 | 87.41	81.96 | 82.95 | 81.60	80.58 | 82.52 | 80.33
T=4, m=1	CIFAR-10, Rate Encoding			
Attack	Clean	GN	FGSM(R)	PGD(R)
clean	79.55	79.36	76.89	76.36
gn	78.62	79.07	76.39	76.77
fgsm(C)	75.57 | 78.01 | 75.62	75.34 | 77.75 | 75.36	72.03 | 73.01 | 73.00	73.99 | 75.81 | 74.50
pgd(C)	76.23 | 78.73 | 76.19	76.06 | 78.74 | 76.30	73.03 | 74.76 | 73.35	74.87 | 76.62 | 75.15
fgsm(R)	43.69 | 55.97 | 46.02	43.31 | 54.90 | 45.30	55.05 | 62.49 | 56.06	55.27 | 63.42 | 55.77
pgd(R)	**37.37** | 51.71 | 39.48	**37.4** | 50.79 | 40.0	**51.63** | 59.92 | 52.12	**51.59** | 61.21 | 51.99
T=4, m=10	CIFAR-10, Rate Encoding			
Attack	Clean	GN	FGSM(R)	PGD(R)
clean	83.22	83.5	80.15	80.54
gn	83.54	83.41	80.33	80.41
fgsm(C)	80.29 | 82.65 | 80.05	79.94 | 82.04 | 80.37	76.41 | 77.97 | 76.82	78.22 | 79.88 | 78.6
pgd(C)	80.6 | 82.93 | 80.51	80.05 | 82.73 | 80.27	76.73 | 78.87 | 76.76	78.59 | 80.15 | 78.69
fgsm(R)	44.93 | 59.58 | 47.64	44.89 | 57.75 | 47.54	57.89 | 65.94 | 58.63	57.85 | 66.87 | 59.03
pgd(R)	**37.74** | 53.68 | 40.97	**37.58** | 52.2 | 40.46	**52.27** | 62.07 | 53.77	**53.08** | 63.93 | 54.19

Table 5.7: We compare the test fgsm/pgd attacks implemented with different backpropagation techniques such as BPTT, BPTR, and RGA and report the corresponding accuracies in the respective order. The results show that, most often, BPTT produces the most potent attacks.

It is interesting to note that, (i), while the clean accuracy of rate encoding is lower than that of directly trained SNNs, the robust accuracy (minimum accuracy across among the attacks) of rate encoding surpasses that of SNNs with constant encoding. (ii) The clean accuracy of rate encoding keeps improving larger T, but the robust accuracy initially increases (due to better prediction) but eventually drops, possibly due to the effect of T as found in our theory. (T=4: 11.49, T=32:16.36, T=64: 18, T=128: 15.76). A similar observation was made in directly trained SNNs, as given in Table 5.6, since we evaluated direct trained models at T=4 vs T=8. (iii) A similar drop in robust accuracy for constant encoded models may hint at the rate-encoded nature of spikes in general SNN layers, as

σ	0.05	0.1	0.2	0.3	0.4	0.5
CLEAN(C)	86.53	59.58	22.8	14.63	13.14	11.73
CLEAN(C)+RAT	84.3	60.3	26.01	16.01	12.66	11.26
CLEAN(R), m=1	78.38	77.74	72.58	61.77	48.38	35.13
CLEAN(R), m=10	83.13	82.07	77.65	66.26	51.14	36.66

Table 5.8: Under stronger Gaussian noise rate-encdoed SNNs show superior robustness.

	Constant Encoding				Rate encoding			
Attack	**T**=4	32	64	128	4	32	64	128
clean	92.58	95.47	95.47	95.51	11.83	24.12	41.5	63.11
gn	85.84	89.1	88.98	88.97	11.54	22.8	38.35	57.4
fgsm(C)	21.02	14.25	15.85	17.56	11.14	17.43	23.64	28.8
pgd(C)	0.2	0.03	0.02	0.03	11.24	17.56	23.54	26.2
fgsm(R)	73.59	64.54	37.34	14.7	11.49	16.36	18	15.76
pgd(R)	87.46	82.53	58.36	11.9	11.46	17.54	18.94	13.92

Table 5.9: Converted SNN model trained on CIFAR10 dataset

reported in [11].

5.6.7 Training Hyper-parameters and Time

Table 5.10 reports the training hyper-parameters used across the four datasets. Additionally, Table 5.11 reports the time required by each epoch of various adversarial training methods on a single NVIDIA RTX A6000 GPU. For example, the column FGSM (C/R) report that constant and rate encoded adversarial training takes the same amount of time for one epoch.

5.6.8 Comparison with Sparse Adversarial Perturbations

The certified accuracy is further compared with robust accuracy obtained with respect to pgd attack within an adversarial budget l_1. The projection in the l_1 ball is implemented using the code from [17]. Figure 5.3 compares the results on different datasets.

5.6.9 Comparison of Pairwise and Pointwise Robust accuracy

Experimental results reveal a notable disparity in robust accuracy between pairwise and pointwise trained models. Pairwise trained models, utilizing a margin-based approach, exhibit significantly higher robustness against adversarial attacks compared to pointwise

Table 5.10: Training hyper-parameters

	CIFAR-10/100	ImageNet-100	SVHN
Number epochs	200	200	200
Mini batch size	64	64	64
T	4,8	4	4
LIF: β	0.5	1	0.5
LIF: u_0	0	0	0
LIF: u_{th}	1	1	1
Learning Rate	0.1	0.1	0.1
FGSM/PGD/GN: ϵ	8/255	8/255	8/255
PGD (train): η	2/255	na	na
PGD (train) Iteration	4	na	na
PGD (test): η	2.55/255	2.55/255	2.55/255
PGD (test) Iteration	7	7	7

Optimizer: SGD with momentum: 0.9, weight decay: 5×10^{-4}, Rate Scheduler: cosine annealing

Table 5.11: Training time per epoch (in seconds)

	Clean (C/R)	GN (C/R)	FGSM(C/R)	PGD(C/R)
CIFAR-10	65	66	106	150
CIFAR-100	48	48	72	251
Imagenet-100	704	na	na	na
SVHN	95	na	na	na

trained models. For instance, at T=4T=4, pairwise trained models achieve a robust accuracy of 38.14%, while pointwise trained models only reach 0.1%. This highlights the effectiveness of margin-based separation in defending against adversarial examples.

5.7 Discussions

This work provides a theoretical grounding for the empirical observations of the adversarial robustness of the rate-encoded classifiers in pariwise or pointwise setting. Consequently, we improve the classifier's prediction at test time by better estimating the corresponding smooth classifier. In addition, we improve the empirical robustness of rate-encoded classifiers by adversarial training, which uses a novel implementation of gradient-based attacks. The future scope for research remains open to reduce the gap between empirical and theoretical results on the shrinkage of the certified radius with latency and other norm-based attacks.

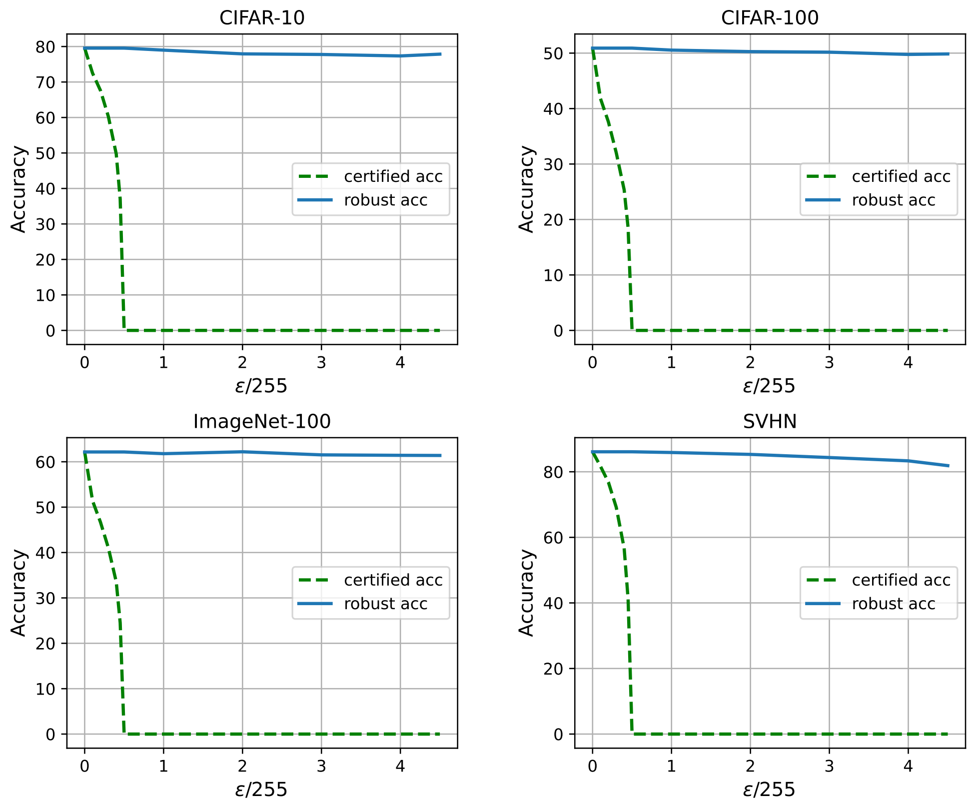


Figure 5.3: We compare the empirical robust accuracy of a CLEAN(R) model under the PGD attack with its certified accuracy across various l_1 radius values and for different datasets. The corresponding gap highlights the scope for improvement in theoretical/experimental results.

Chapter 6

Conclusion and Future Work

This thesis presents an exploration of online kernelized pairwise learning algorithms, their computational efficiency, application in *non-i.i.d.* data scenarios, enhancement of adversarial robustness in classifiers, and investigation of deep metric learning within SNNs. Here, we summarize the core contributions of our study and propose directions for future research.

Contributions and Findings

- In **Chapter 2** we introduced a lightweight online kernelized pairwise learning algorithm designed for both linear and non-linear models. Incorporating an online clustering mechanism, enhancing the algorithm's ability to handle real-time data streams. This modification not only preserved computational efficiency, but also introduced a scalable method to manage gradient complexity across linear and nonlinear models.
- **Chapter 3** expanded upon this by employing a moving average along with random examples, significantly reducing the complexity of the gradient with *non-i.i.d.*. The use of random Fourier mapping for kernel function approximation has proven to be effective in addressing the challenges associated complex data, demonstrating a novel method to mitigate kernel complexity through dual evaluation of gradients and dynamic averaging.
- **Chapter 4** explored deep metric learning in the context of SNNs, assessing the impact of various data encoding strategies, loss functions and architectures. Our experiments across multiple benchmark datasets illuminated the intricate relationships between encoding strategies, temporal dimensions, and performance efficacy. Comparative analysis of loss functions revealed critical considerations for optimizing SNN classification accuracy, suggesting avenues for future development in encoding techniques and loss function selection.
- In **Chapter 5**, our research investigated the adversarial robustness of rate-encoded classifiers, offering both theoretical insights and practical improvements. By focus-

ing on empirical robustness through adversarial training and novel gradient-based attacks, we bridged the gap between theory and practice. This investigation underscores the need for ongoing research to further reduce disparities between empirical and theoretical outcomes, particularly with regard to certified radius and norm-based attacks.

Future Work

The work presented in this thesis lay the groundwork for several promising directions of future research:

- *Enhancing Algorithm Robustness:* Further investigation into multiple moving averages and online clustering mechanisms could offer new insights into increasing the adaptability and robustness of online kernelized pairwise learning algorithms.
- *Bridging Theoretical and Empirical Gaps:* There remains a significant opportunity to narrow the discrepancy between theoretical predictions and empirical observations in adversarial robustness, particularly in the context of rate-encoded classifiers.
- *Advancing Encoding Strategies:* Developing novel data encoding techniques that provide greater flexibility and compatibility with data augmentation strategies could revolutionize the performance of SNNs, potentially surpassing traditional ANN accuracies.
- *Exploring Loss Function Compatibility:* Identifying optimal combinations of loss functions, encoding strategies, and neural network architectures remains a key challenge, with the potential to significantly improve classification performance in SNNs.

In conclusion, our thesis not only contributes to the existing body of knowledge on online kernelized learning algorithms, adversarial robustness, and deep metric learning, but also opens up new pathways for research that could lead to more sophisticated, efficient, and robust machine learning models in the future.

References

[1] Shivani Agarwal and Partha Niyogi. Generalization bounds for ranking algorithms via algorithmic stability. *Journal of Machine Learning Research*, 10(2), 2009.

[2] Hilal AlQuabeh and Aliakbar Abdurahimov. Pairwise learning via stagewise training in proximal setting. *arXiv preprint arXiv:2208.04075*, 2022.

[3] Hilal AlQuabeh, Farha AlBreiki, and Dilshod Azizov. Computational complexity of sub-linear convergent algorithms. *arXiv preprint arXiv:2209.14558*, 2022.

[4] Hilal AlQuabeh, William de Vazelhes, and Bin Gu. Limited memory online gradient descent for kernelized pairwise learning with dynamic averaging. *arXiv preprint arXiv:2402.01146*, 2024.

[5] Hilal AlQuabeh, Bhaskar Mukhoty, and Bin Gu. Variance reduced online gradient descent for kernelized pairwise learning with limited memory. *arXiv preprint arXiv:2310.06483*, 2023.

[6] Hilal AlQuabeh, Bhaskar Mukhoty, and Bin Gu. Variance reduced online gradient descent for kernelized pairwise learning with limited memory. In Berrin Yanıkoğlu and Wray Buntine, editors, *Proceedings of the 15th Asian Conference on Machine Learning*, volume 222 of *Proceedings of Machine Learning Research*, pages 28–43. PMLR, 11–14 Nov 2024.

[7] Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *The Journal of Machine Learning Research*, 18(1):714–751, 2017.

[8] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[9] Albert S Berahas, Liyuan Cao, Krzysztof Choromanski, and Katya Scheinberg. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics*, 22(2):507–560, 2022.

[10] Martin Boissier, Siwei Lyu, Yiming Ying, and Ding-Xuan Zhou. Fast convergence of online pairwise learning algorithms. In *Artificial Intelligence and Statistics*, pages 204–212. PMLR, 2016.

[11] Tong Bu, Jianhao Ding, Zecheng Hao, and Zhaofei Yu. Rate gradient approximation attack threats deep spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7896–7906, 2023.

[12] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

[13] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[14] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[15] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[16] Benjamin Cramer, Sebastian Billaudelle, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, et al. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences*, 119(4):e2109194119, 2022.

[17] Francesco Croce and Matthias Hein. Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers. In *International Conference on Machine Learning*, pages 2201–2211. PMLR, 2021.

[18] Zhiyuan Dang, Xiang Li, Bin Gu, Cheng Deng, and Heng Huang. Large-scale nonlinear auc maximization via triply stochastic gradients. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[19] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (Csur)*, 40(2):1–60, 2008.

[20] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

[21] Jianhao Ding, Tong Bu, Zhaofei Yu, Tiejun Huang, and Jian Liu. Snn-rat: Robustness-enhanced spiking neural network through regularized adversarial training. *Advances in Neural Information Processing Systems*, 35:24780–24793, 2022.

[22] Petros Drineas, Michael W Mahoney, and Nello Cristianini. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(12), 2005.

[23] Changying Du, Changde Du, Guoping Long, Qing He, and Yucheng Li. Online bayesian multiple kernel bipartite ranking. In *UAI*, 2016.

[24] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2018.

[25] Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *arXiv preprint arXiv:2109.12894*, 2021.

[26] JK Eshraghian, M Ward, E Neftci, X Wang, G Lenz, G Dwivedi, M Bennamoun, DS Jeong, and WD Lu. Training spiking neural networks using lessons from deep learning. arxiv. *arXiv preprint arXiv:2109.12894*, 2021.

[27] Bojana Gajić, Ariel Amato, and Carlo Gatta. Fast hard negative mining for deep metric learning. *Pattern Recognition*, 112:107795, 2021.

[28] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. One-pass auc optimization. In *International conference on machine learning*, pages 906–914. PMLR, 2013.

[29] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

[30] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[31] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[32] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *stat*, 1050:20, 2015.

[33] Bin Gu, Zhouyuan Huo, and Heng Huang. Scalable and efficient pairwise learning to achieve statistical accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3697–3704, 2019.

[34] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018.

[35] Wenzhe Guo, Mohammed E Fouda, Ahmed M Eltawil, and Khaled Nabil Salama. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. *Frontiers in Neuroscience*, 15:638474, 2021.

[36] Zheng-Chu Guo, Yiming Ying, and Ding-Xuan Zhou. Online regularized learning with pairwise loss functions. *Advances in Computational Mathematics*, 43(1):127–150, 2017.

[37] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[38] Ben Harwood, Vijay Kumar BG, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2821–2829, 2017.

[39] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4834–4843, 2018.

[40] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. Triplet-center loss for multi-view 3d object retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1945–1954, 2018.

[41] Geofry Hinton. Neural networks for machine learning, 2012. Coursera Video Lectures.

[42] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*, pages 84–92. Springer, 2015.

[43] Junjie Hu, Haiqin Yang, Irwin King, Michael R Lyu, and Anthony Man-Cho So. Kernelized online imbalanced learning with fixed budgets. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[44] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

[45] Kenneth Hung and William Fithian. Rank verification for exponential families. *The Annals of Statistics*, 47(2):758–782, 2019.

[46] Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: Towards optimal mapping from activation values to firing rates. *ICLR*, 2023.

[47] Yulun Jiang, Chen Liu, Zhichao Huang, Mathieu Salzmann, and Sabine Susstrunk. Towards stable and efficient adversarial training against l_1 bounded adversarial attacks. In *International Conference on Machine Learning*, pages 15089–15104. PMLR, 2023.

[48] Roland S Johansson and Ingvars Birznieks. First spikes in ensembles of human tactile afferents code complex spatial fingertip events. *Nature neuroscience*, 7(2):170–177, 2004.

[49] Vishal Kakkar, Shirish Shevade, S Sundararajan, and Dinesh Garg. A sparse nonlinear classifier design using auc optimization. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 291–299. SIAM, 2017.

[50] Nathan Kallus and Angela Zhou. The fairness of risk scores beyond classification: Bipartite ranking and the xauc metric. *Advances in neural information processing systems*, 32, 2019.

[51] Bong-Nam Kang, Yonghyun Kim, and Daijin Kim. Pairwise relational networks for face recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 628–645, 2018.

[52] Purushottam Kar, Bharath Sriperumbudur, Prateek Jain, and Harish Karnick. On the generalization ability of online learning algorithms for pairwise loss functions. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, number 3 in Proceedings of Machine Learning Research, pages 441–449, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[53] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[54] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11270–11277, 2020.

[55] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *ICLR*, 2009.

[56] Brian Kulis et al. Metric learning: A survey. *Foundations and trends in machine learning*, 5(4):287–364, 2012.

[57] Souvik Kundu, Massoud Pedram, and Peter A Beerel. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5209–5218, 2021.

[58] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.

[59] Yunwen Lei, Antoine Ledent, and Marius Kloft. Sharper generalization bounds for pairwise learning. *Advances in Neural Information Processing Systems*, 33:21236–21246, 2020.

[60] Yunwen Lei, Shao-Bo Lin, and Ke Tang. Generalization bounds for regularized pairwise learning. In *IJCAI*, pages 2376–2382, 2018.

[61] Yunwen Lei, Mingrui Liu, and Yiming Ying. Generalization guarantee of sgd for pairwise learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[62] Jerry Li. Lecture 14: Certified defenses iii: Randomized smoothing. https://jerryzli.github.io/robust-ml-fall19/lec14.pdf, 2019. Accessed: 19-09-2023.

[63] Yang Li, Dongcheng Zhao, and Yi Zeng. Bsnn: Towards faster and better conversion of artificial neural networks to spiking neural networks with bistable neurons. *Frontiers in neuroscience*, 16:991851, 2022.

[64] Zhu Li. Sharp analysis of random fourier features in classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7444–7452, 2022.

[65] Ling Liang, Kaidi Xu, Xing Hu, Lei Deng, and Yuan Xie. Toward robust spiking neural network against adversarial perturbation. *Advances in Neural Information Processing Systems*, 35:10244–10256, 2022.

[66] Junhong Lin, Yunwen Lei, Bo Zhang, and Ding-Xuan Zhou. Online pairwise learning algorithms with convex loss functions. *Information Sciences*, 406:57–70, 2017.

[67] Nikos K Logothetis and David L Sheinberg. Visual object recognition. *Annual review of neuroscience*, 19(1):577–621, 1996.

[68] Jing Lu, Steven CH Hoi, Jialei Wang, Peilin Zhao, and Zhi-Yong Liu. Large scale online kernel learning. *Journal of Machine Learning Research*, 17(47):1, 2016.

[69] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[70] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pages 6640–6650. PMLR, 2020.

[71] Bhaskar Mukhoty, Hilal AlQuabeh, Giulia De Masi, Huan Xiong, and Bin Gu. Certified adversarial robustness for rate encoded spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2023.

[72] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

[73] Michael Natole, Jr., Yiming Ying, and Siwei Lyu. Stochastic proximal algorithms for AUC maximization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3710–3719. PMLR, 10–15 Jul 2018.

[74] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[75] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[76] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[77] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016.

[78] Lior Rokach and Oded Maimon. Clustering methods. *Data mining and knowledge discovery handbook*, pages 321–352, 2005.

[79] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*, 32, 2019.

[80] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[81] Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 399–414. Springer, 2020.

[82] Hao Sheng, Yanwei Zheng, Wei Ke, Dongxiao Yu, Xiuzhen Cheng, Weifeng Lyu, and Zhang Xiong. Mining hard samples globally and efficiently for person reidentification. *IEEE Internet of Things Journal*, 7(10):9611–9622, 2020.

[83] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[84] Lingxue Song, Dihong Gong, Zhifeng Li, Changsong Liu, and Wei Liu. Occlusion robust face recognition based on mask learning with pairwise differential siamese network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 773–782, 2019.

[85] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020.

[86] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32, 2019.

[87] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[88] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

[89] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019.

[90] Yuyang Wang, Roni Khardon, Dmitry Pechyony, and Rosie Jones. Generalization bounds for online learning algorithms with pairwise loss functions. In *Conference on Learning Theory*, pages 13–1. JMLR Workshop and Conference Proceedings, 2012.

[91] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[92] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2020.

[93] Zhenhuan Yang, Yunwen Lei, Siwei Lyu, and Yiming Ying. Stability and differential privacy of stochastic gradient descent for pairwise learning with non-smooth loss. In *International Conference on Artificial Intelligence and Statistics*, pages 2026–2034. PMLR, 2021.

[94] Zhenhuan Yang, Yunwen Lei, Puyu Wang, Tianbao Yang, and Yiming Ying. Simple stochastic and online gradient descent algorithms for pairwise learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[95] Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10221–10230, 2021.

[96] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

[97] Yiming Ying, Longyin Wen, and Siwei Lyu. Stochastic online auc maximization. *Advances in neural information processing systems*, 29:451–459, 2016.

[98] Yiming Ying and Ding-Xuan Zhou. Online pairwise learning algorithms with kernels. *arXiv preprint arXiv:1502.07229*, 2015.

[99] Zhen Zhang, Qinfeng Shi, Julian McAuley, Wei Wei, Yanning Zhang, and Anton Van Den Hengel. Pairwise matching through max-weight bipartite belief propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1202–1210, 2016.

[100] Peilin Zhao, Steven CH Hoi, Rong Jin, and Tianbo YANG. Online auc maximization. *ICML*, 2011.

[101] Jinfeng Zhuang, Jialei Wang, Steven CH Hoi, and Xiangyang Lan. Unsupervised multiple kernel learning. In *Asian Conference on Machine Learning*, pages 129–144. PMLR, 2011.