ENHANCING CLASSIFICATION AND RETRIEVAL PERFORMANCE BY MINING SEMANTIC SIMILARITY RELATION FROM DATA

by

Yang Gao

APPROVED BY SUPERVISORY COMMITTEE:

Latifur Khan, Chair

Ding-Zhu Du

Haim Schweitzer

Bhavani Thuraisingham

To my parents, Mr.Gao and Mrs.Wu

ENHANCING CLASSIFICATION AND RETRIEVAL PERFORMANCE BY MINING SEMANTIC SIMILARITY RELATION FROM DATA

by

YANG GAO, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2021

ACKNOWLEDGMENTS

All praise is due to my advisor, Dr.Latifur Khan. I would like to thank him for being a great advisor and for his patience and support in completing this dissertation.

I would like to extend my gratitude to Dr.Bhavani Thuraisingham and Dr.Charu Aggarwal for their effective guidance and valuable comments on my papers.

I would also like to express my deepest love and warmest expression to my father Mr.Gao and my mother Mrs.Wu. The passion and support you have provided me over the years were the greatest gifts anyone has ever given me. Thank you.

I also need to thank my friends, Yi-Fan, Yu, Swarup, Tao, and Bo for their great assistance in my research work and this dissertation.

The research reported herein was supported in part by NSF award DMS-1737978; DGE-2039542; DGE 17236021; OAC-1828467; OAC-1931541; DGE-1906630; ARO award W911-NF-18-1-0249; ONR award N00014-17-1-2995; NSA award H98230-15-1-0271; AFOSR award FA9550-14-1-0173; an endowment from the Eugene McDermott family; NSF FAIN awards DGE-1931800, OAC-1828467, and DGE-1723602; NSF awards DMS-1737978 and MRI-1828467; an IBM faculty award (Research); and an HP grant. Any opinions, recommendations, or conclusions expressed are those of the authors and not necessarily of the aforementioned supporters.

February 2021

ENHANCING CLASSIFICATION AND RETRIEVAL PERFORMANCE BY MINING SEMANTIC SIMILARITY RELATION FROM DATA

Yang Gao, PhD
The University of Texas at Dallas, 2021

Supervising Professor: Latifur Khan, Chair

When describing unstructured data, e.g., images and texts, humans often resort to similarity defining the characteristics of these data in relative terms rather than absolute terms. The subtle differences between such data can be indicated by a human easily while completely describing a single instance of them is a challenging task. For example, in an image retrieval task, to determine if two images are describing the same object, humans may simply ignore the differences in illumination, scaling, background, occlusion, viewpoint and only pay attention to the object itself. On the other hand, describing an image with all its information is hard and unnecessary. Cognitive evidence also suggests that we interpret objects by relating them to prototypical examples stored in our brain. Thus, the similarity is a fundamental property and of great importance in classification and retrieval tasks alike.

Metric learning is the process of determining a non-negative, symmetric, and subadditive distance function $d(a, b)$ that aims to establish the similarity or dissimilarity between objects. It reduces the distance between similar objects and increases the distance between dissimilar objects. From the human's perspective, metric learning can be viewed as determining a function that best matches the user interpretation of the similarity and dissimilarity relation between data items.

In this dissertation, we explore the possibilities to enhance the classification and retrieval performance by mining semantic similarity relations in data via metric learning. Unfortunately, existing metric learning solutions have several drawbacks. First, most metric learning models have a fixed model capacity that cannot be changed for adaption to input data. Second, existing online metric learning models learn a linear metric function which limits the model's expressiveness. Third, they usually require a user-specified margin sensitive to input data and ignore a lot of failure cases during learning. To address these drawbacks, we propose a novel online metric learning framework OAHU that automatically adjusts model capacity based on input data, and introduce an Adaptive Bound Triplet Loss (ABTL) to avoid failure cases during learning. On the other hand, as an important subarea of classification, imbalanced classification is critical to the success of many real-world applications, but few existing solutions have ever considered utilizing data similarity to assist imbalanced learning. Based on this observation, we introduce a novel framework named SetConv, which customizes the feature extraction process for each input sample by considering its semantic similarity relation to the minority class to alleviate the model bias towards the majority classes. We also incorporate metric/similarity learning into a novel open-world stream classifier SIM to handle classifications on open-ended data distribution.

Based on our research, we demonstrate that mining semantic similarity relation in data is critical to improving the performance of real-world classification and retrieval tasks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION [1] [2] [3]

The past two decades have witnessed tremendous growth of unstructured data on the internet, which exist in various forms, e.g., images, texts, videos, etc. Applications such as personalized marketing (Hite et al., 1998) and visual search (Zhang et al., 2018; Zhai et al., 2019) has spurred the demand for technologies to analyze such data. Unfortunately, understanding unstructured data is easy for humans but can be extremely difficult for machines in many cases. For example, given pictures of a product taken from different perspectives, a human can easily tell that all these pictures represent the same product while machines may fail in this task. An important reason is that when describing unstructured data, humans often resort to similarity defining the characteristics of these data in relative terms rather than absolute terms. For example, in a visual search or an image retrieval task, to determine if two images refer to the same object, we may simply ignore their differences in illumination, scaling, background, occlusion, viewpoint, and only focus on the object itself. In fact, describing an image with all its information is hard and unnecessary. Cognitive evidence also suggests that humans interpret objects by relating them to prototypical objects stored in our

[1] This chapter contains material previously published as: Yang Gao, Yi-Fan Li, Yu Lin, Latifur Khan. "SetConv: A New Approach for Learning from Imbalanced Data", In *proceedings of The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1284-1294. 2020. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

[2] This chapter contains material previously published as: Yang Gao, Yi-Fan Li, Swarup Chandra, Latifur Khan, and Bhavani Thuraisingham. "Towards self-adaptive metric learning on the fly." In *The World Wide Web Conference (WWW)*, pp. 503-513. 2019, https://doi.org/10.1145/3308558.3313503. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

[3] This chapter contains material previously published as: ©2019 IEEE. Reprinted, with permission, from Yang Gao, Yi-Fan Li, Bo Dong, Yu Lin, and Latifur Khan. "SIM: Open-World Multi-Task Stream Classifier with Integral Similarity Metrics." In *IEEE International Conference on Big Data (Big Data)*, pp. 751-760. December, 2019. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

brain. Thus, the similarity is a fundamental property and critical to helping the machines understand the semantic meanings of unstructured data in many real-world applications.

Metric/Similarity learning is a learning paradigm that automatically determines a non-negative, symmetric, and sub-additive distance function $d(a, b)$ to establish the similarity or dissimilarity relations between objects. It reduces the distance between similar objects and increases that between dissimilar objects. In other words, metric/similarity learning helps us identify characteristic patterns that reveal the underlying similarity and dissimilarity relations in the unstructured data. That is, given two data instances, a metric/similarity learning model is capable of extracting the most representative features from the input instances and provides a quantitative measure of how similar or dissimilar they are.

We, therefore, explore the possibilities to enhance the classification and retrieval performance by mining semantic similarity relations from data via metric/similarity learning. Specifically, we focus on several common applications including Content-Based Image Retrieval (CBIR), image/text classification, and those derived applications such as face verification, incident detection, and so on.

1.1 Content-Based Image Retrieval (CBIR)

Content-based image retrieval is the problem of searching for digital images in a large database given a query image. "content-based" means that the search analyzes the contents of the image rather than the metadata such as keywords, tags, or descriptions associated with the image.

In content-based image retrieval, there are three key issues i.e., image representation, image organization, and image similarity measurement (Zhou et al., 2017). Image representation originates from the fact that the intrinsic problem of CBIR is image comparison. For ease of comparison, an image is usually transformed into some kind of feature space,

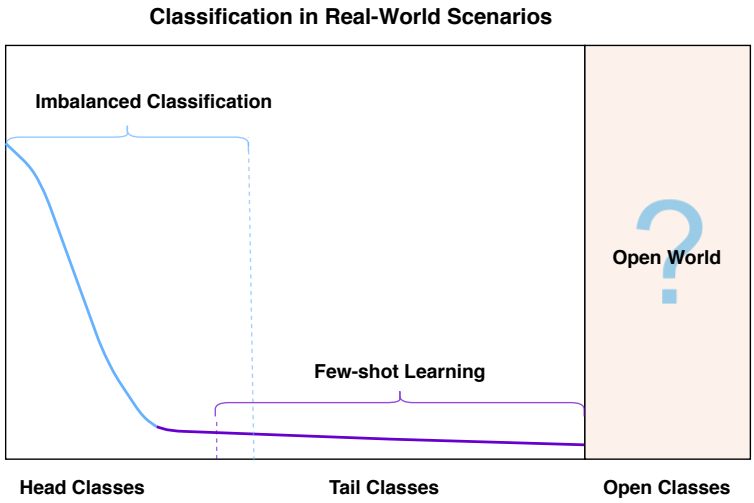


Figure 1.1: Classification in real-world scenarios.

in which the resulting image representation is expected to be descriptive and discriminative to distinguish similar and dissimilar images. More importantly, the representation is also expected to be invariant to various transformations, such as translation, rotation, scaling, illumination change, etc. On the other hand, the similarity measure between images should reflect the relevance in semantics, which, however, is difficult due to the hardness in describing high-level semantic concepts with low-level visual features.

Fortunately, metric/similarity learning can save us from this dilemma. It addresses these issues by forcing the model to find a latent feature space where the representations of similar images are close to each other and those of dissimilar images are far away from each other. Hence, searching for those images similar to a given query image is easy in this latent feature space.

1.2 Classification

In many existing classification settings, the training data and test data are both balanced under a closed-world assumption, e.g., the ImageNet dataset. However, this setting is not a good proxy of the real-world scenario. As shown in Fig. 1.1, in most real-world cases, people

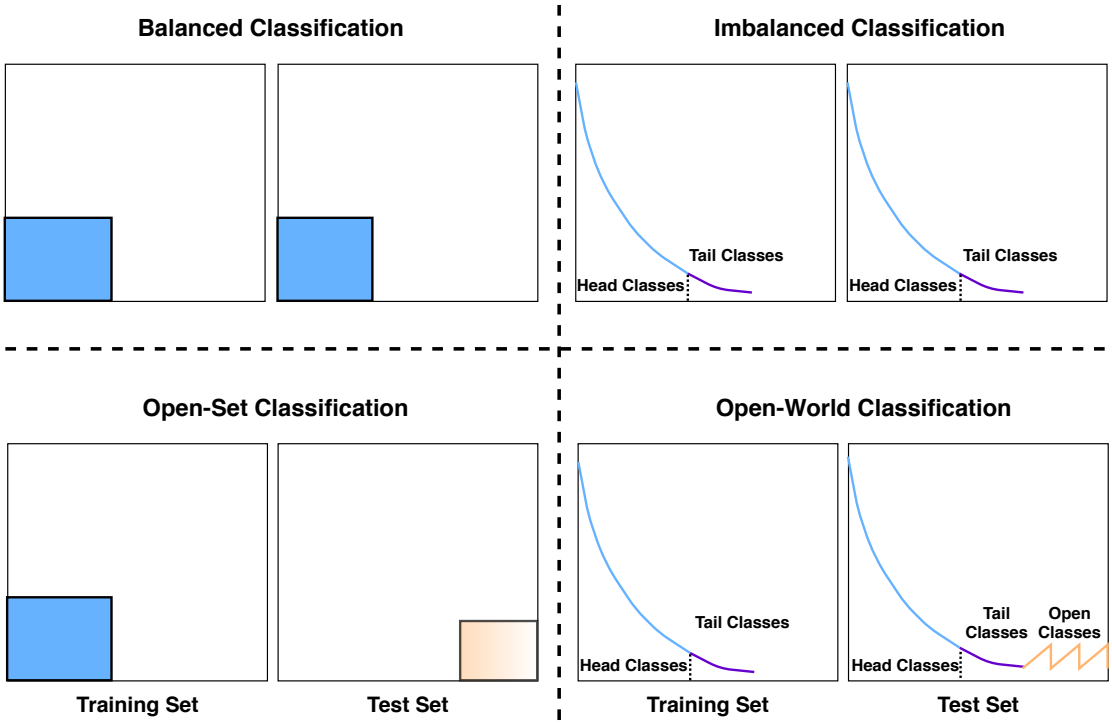


Figure 1.2: The differences between balanced classification, imbalanced classification, open-set classification, and open-world classification.

are bothered by the imbalanced and open-ended distribution from all sorts of datasets: faces, fashion brands, sentiments, network attacks, etc. Therefore, a practical system shall be able to classify among a few common categories and many rare categories, to formulate the concept of a single category from limited samples, and to identify novelty upon an instance of a never seen category. For ease of analysis, based on their prior assumptions, we divide some common classification problems into four categories, i.e., balanced classification, imbalanced classification, open-set classification, and open-world classification.

As shown in Fig. 1.2, in balanced classification, the size of each class in the training or test set is the same. In contrast, the distribution of examples across the known classes is biased or skewed in both training and test sets for imbalanced classification. Open-set classification is a variant of the content-based retrieval problem, where the classes of the test set and training set are disjoint. In this situation, the query data can be classified based

on the neighbors' vote or verified based on a distance threshold in the embedding space. A typical application is face-verification (Masi et al., 2018). In open-world classification, the training set classes are a subset of the test set classes. That is, an open-world classifier is required to not only correctly classify instances of known classes but also identify instances from those categories that were never seen in the training dataset. A notable example of this is the classification of an evolving data stream. In all these classification paradigms, a well-clustered embedding space can significantly boost the model performance. Hence, metric/similarity learning can play a critical role here, since it directly learns from data and automatically finds such a latent feature space where similar data are close together and dissimilar data are far apart. Conversely, the complexity of these classification paradigms also presents a few challenges to metric/similarity learning.

1.2.1 Online Adaptive Metric Learning

Since similarity metrics can significantly facilitate the performance of many large-scale, real-world applications, developing scalable techniques for learning high-quality similarity metrics is becoming more and more important. Most of the existing Online Metric Learning (OML) solutions (Jain et al., 2008a; Jin et al., 2009a; Chechik et al., 2010a; Li et al., 2018a) are unable to precisely measure the non-linear similarity among instances in complex real-world applications because they are designed to learn a pre-selected linear metric from a stream of pairwise or triplet constraints. Moreover, these algorithms often suffer the issue of low constraint utilization, since their learning process focuses only on the "hard" constraints and are biased towards them, leading to the weakened model robustness.

To overcome these model defects, the "Online Adaptive Metric Learning" (OAML) open challenge (Gao et al., 2019) is introduced, i.e., how to learn a metric model that automatically adjusts its model complexity as more constraints are observed in the input stream and maintain high constraint utilization during the learning process? In other words, the

hypothesis space of the learned metric function is dynamic, including functions from pure linear stage to highly non-linear regime.

To address the OAML challenge, our proposed framework OAHU first amends the common ANN architecture by attaching an independent metric embedding layer (MEL) to every hidden layer of the network. The output of a hidden layer is hence the input to its corresponding MEL. Each MEL is associated with a metric weight, measuring its importance in the entire metric model. In other words, OAHU may be regarded as an ensemble of metric models with various complexities sharing the low-level knowledge. We further propose a novel Adaptive-Bound Triplet Loss (ABTL) to eliminate the dependency of existing methods on those "hard" constraints, which improves the constraint utilization. An Adaptive Hedge Update (AHU) method is introduced to update the metric models in the ensemble as well as the associated metric weights. In summary, with all these innovations, OAHU is capable of making full use of input constraints to learn a suitable similarity metric and dynamically adapting its model complexity based on input constraints when necessary.

1.2.2 Imbalanced Classification

Another major challenge is how to alleviate the bias towards the majority classes in imbalanced classification. Solutions proposed in existing studies can be generally divided into three categories (Krawczyk, 2016): (1) *Data-level methods*. These methods employ under-sampling or over-sampling technique to balance the class distributions (Barua et al., 2014; Smith et al., 2014; Sobhani et al., 2014; Zheng et al., 2015). (2) *Algorithm-level methods*. These algorithms concentrate on modifying existing learners to alleviate the model bias towards the majority classes. The most popular branch is the cost-sensitive algorithms, which assign a higher cost on misclassifying the minority class instances (Díaz-Vico et al., 2018). (3) *Hybrid methods*. These approaches attempt to combine the advantages of both data-level and algorithm-level methods to extract their strong points and reduce their weakness (Galar et al., 2012; Wang et al., 2015).

Despite the success of existing solutions on some applications, they have several intrinsic drawbacks. First, for data-level methods, the under-sampling approaches need to remove lots of majority class samples from the dataset, which may lose important information. On the other hand, the over-sampling approaches increase the adverse correlation among samples by introducing a large amount of synthetic minority-class samples into the dataset (Wu et al., 2017). Second, in many cases, it is often difficult to set the actual cost values for cost-sensitive approaches, since they are usually required to be given by expert beforehand or even hard to define in practical scenarios (Krawczyk, 2016). Finally, many open questions remain to be answered in the field of hybrid imbalance learning methods. For example, how to guarantee and utilize the diversity of base learners in an ensemble for prediction when classes are imbalanced is still unclear yet (Wu et al., 2017; Huo et al., 2016).

Unfortunately, directly applying metric/similarity learning may not help in this case. The main reason is that there is a limited number of minority class instances, which restricts the number of constraints reflecting the intra-class similarity that we can produce for these classes. If we over-sample the minority class instances to produce more constraints, we are facing the same issues as those over-sampling methods.

To overcome the above issues, we propose a novel set convolution (SetConv) operation and a new training strategy *episodic training* to assist learning from imbalanced class distributions. To capture the semantic similarity as metric/similarity learning, the proposed SetConv operation directly estimates the convolution kernel weights based on the intra-class and inter-class correlations and applies the learned kernel to extract features from data, which are discriminative among different classes. These features are then compressed into a single class representative that is used for classification. In this way, SetConv helps the model focus on the latent concept not only common to different samples of the same class but also discriminative to other classes. In episodic training, we assign equal weights to different classes in spite of their relative sizes and do not perform re-sampling on data.

During the training process, at each iteration, the model is fed with an episode formed by a set of samples where the class imbalance ratio is preserved. The model is hence encouraged to extract discriminative features even when class distribution is unbalanced. Moreover, the proposed episodic training has no dependence on unknown prior knowledge. The only prior knowledge required is the class imbalance ratio, which can be easily obtained from data in real-world applications.

1.2.3 Open-World Classification

Classical supervised learning typically relies on a closed-world assumption, which says that all the test classes have been observed in the training process. However, the real world is open, dynamic, and full of unknowns. That is, instances of unexpected classes may appear in the test or application data. Figure 1.3 shows examples of crossroad scenes that a commercial self-driving car has to deal with in real life. Apparently, Figure 1.3a and Figure 1.3b provide two common crossroad scenes that can be easily collected to train a self-driving car. However, Figure 1.3c and Figure 1.3d describe two rare crossroad scenes that are hard to see in daily life. Unfortunately, a commercial self-driving car must be able to make correct decisions while facing these rare and difficult cases. Therefore, having a model that is robust to unseen classes and can be continuously updated to incorporate these cases is critical to the success of many practical applications.

Existing stream classifiers usually address the open-world classification problem by first detecting instances from unseen classes (a.k.a novel classes) and then updating the classifier to incorporate information from those instances. The novel class detection mechanisms of previous solutions are based on the existence of strong cohesion and large separation of data in observed feature space. In other words, instances from the same class are closer than those from different classes. Unfortunately, this assumption may not be valid in complex high-dimensional data streams, e.g., a product image stream.

(a) (b)

(c) (d)

Figure 1.3: Examples of common (a,b) and rare (c,d) crossroad scenes that a commercial self-driving car has to deal with.

To address this challenge, we propose a novel framework SIM that is able to perform open-world classification on data streams. In contrast to existing solutions, our method performs similarity/metric learning to actively search for a latent feature space where the strong cohesion and large separation data property holds so that instances from novel classes can be easily detected within this latent space. The metric learning task serves as an auxiliary

task to both the classification and novel class detection tasks, and all these tasks are jointly learned in the proposed SIM framework.

1.3 Contribution of this dissertation

In summary, the contribution of this dissertation is as follows:

1.3.1 Online Adaptive Metric Learning

- We introduce a new framework OAHU designed to address the "Online Adaptive Metric Learning" open challenge. It learns a neural-network-based metric model with adaptive model complexity and full constraint utilization.
- We show the theoretical regret bound of OAHU and prove the optimal range for hyper-parameter selection.
- We empirically demonstrate the superiority of OAHU over existing state-of-the-art solutions on three typical tasks including image classification, face verification, and image retrieval.

1.3.2 Imbalanced Classification

- We propose a novel SetConv operation and a new episodic training strategy to assist learning from imbalanced class distributions.
- SetConv is a data-sensitive convolution operation, which helps the model to customize the feature extraction process for each input sample and potentially improves the model performance.
- SetConv guarantees automatic class balancing. No matter how much data of a class is fed into SetConv, it always produces a single class representative, so that the

subsequent classifier, which takes these class representatives as input, always perceives a balanced class distribution.

- The proposed episodic training strategy has no dependency on unknown prior knowledge. The only prior knowledge required is the class imbalance ratio, which can be easily obtained from data in real-world applications.

1.3.3 Open-World Classification

- We propose a novel framework SIM that actively searches for a latent feature space suitable for both classification and novel class detection on real-world high-dimensional image streams.

- We propose a unified multi-task open-world classifier that is capable of performing metric learning, stream classification, and novel class detection simultaneously.

1.4 Outlines of this dissertation

The rest of the dissertation is organized as follows. Chapter 2 gives a background of approaches discussed in this dissertation. Chapter 3 discusses the approach proposed to address the "online adaptive metric learning" open challenge. Chapter 4 presents our solution for learning from the imbalanced data distribution. Chapter 5 outlines our open-world classifier for learning from the open-end data distribution. Chapter 6 summarizes this dissertation and discusses the future work.

CHAPTER 2

BACKGROUND [1] [2] [3]

In this chapter, we present relevant background information of metric/similarity learning and its derived applications, e.g., imbalanced classification, open-world classification, etc.

2.1 Metric Learning

In this section, we start by introducing the Mahalanobis distance that is widely adopted in Online Metric Learning (OML) algorithms. We then briefly describe representative OML approaches and discuss the drawbacks of existing OML algorithms. Finally, we introduce the open challenge "Online Adaptive Metric Learning" that aims to address these drawbacks.

2.1.1 Mahalanobis Distance

The Mahalanobis distance is a distance measure that incorporates the correlation between features.

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')} \tag{2.1}$$

[1] This chapter contains material previously published as: Yang Gao, Yi-Fan Li, Yu Lin, Latifur Khan. "SetConv: A New Approach for Learning from Imbalanced Data", In *proceedings of The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1284-1294. 2020. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

[2] This chapter contains material previously published as: Yang Gao, Yi-Fan Li, Swarup Chandra, Latifur Khan, and Bhavani Thuraisingham. "Towards self-adaptive metric learning on the fly." In *The World Wide Web Conference (WWW)*, pp. 503-513. 2019, https://doi.org/10.1145/3308558.3313503. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

[3] This chapter contains material previously published as: ©2019 IEEE. Reprinted, with permission, from Yang Gao, Yi-Fan Li, Bo Dong, Yu Lin, and Latifur Khan. "SIM: Open-World Multi-Task Stream Classifier with Integral Similarity Metrics." In *IEEE International Conference on Big Data (Big Data)*, pp. 751-760. December, 2019. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

Table 2.1: Summary of the notations.

Notation	Description
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^d$	Set of d-dimensional real-valued vectors
$\mathbb{R}^{c\times d}$	Set of $c \times d$ real-valued matrices
$\mathbb{S}_+^d$	Cone of symmetric PSD $d \times d$ real-valued matrices
$\mathcal{X}$	Input (instance) space
$\mathcal{Y}$	Output (label) space
$\mathbf{x}$	An arbitrary vector
$\mathbf{M}$	An arbitrary matrix
$\mathbf{M} \succeq 0$	PSD matrix $\mathbf{M}$
$\mathbf{I}$	Identity matrix
$tr(\mathbf{M})$	Trace of matrix $\mathbf{M}$
$\|\| \cdot \|\|_{\mathcal{F}}$	Frobenius norm
$[t]_+ = \max(0, 1-t)$	Hinge loss function

where $\mathbf{x}$ and $\mathbf{x}'$ are random vectors from the same distribution and $\mathbf{M} \in \mathbb{S}_+^d$. Here $\mathbb{S}_+^d$ is the cone of symmetric positive semi-definite (PSD) $d \times d$ real-valued matrices (Bellet et al., 2013).

The Mahalanobis distance $d_{\mathbf{M}}$ is indeed a pseudo-distance: $\forall \mathbf{x}, \mathbf{x}', \mathbf{x}'' \in \mathcal{X}$,

1. Non-negative: $d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') \geq 0$;
2. Identity: $d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}) = 0$;
3. Symmetric: $d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = d_{\mathbf{M}}(\mathbf{x}', \mathbf{x})$;
4. Triangle inequality: $d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}'') \leq d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') + d_{\mathbf{M}}(\mathbf{x}', \mathbf{x}'')$.

If $\mathbf{M}$ is an identity matrix, i.e., $\mathbf{M} = \mathbf{I}$, $d_{\mathbf{M}}$ is equivalent to the well-known Euclidean distance. Otherwise, we can decompose $\mathbf{M} = \mathbf{L}^T\mathbf{L}$, where $\mathbf{L} \in \mathbb{R}^{k\times d}$ and k is the rank of $\mathbf{M}$.

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') &= \sqrt{(\mathbf{x}-\mathbf{x}')^T\mathbf{M}(\mathbf{x}-\mathbf{x}')} \\ &= \sqrt{(\mathbf{x}-\mathbf{x}')^T\mathbf{L}^\mathbf{T}\mathbf{L}(\mathbf{x}-\mathbf{x}')} \\ &= \sqrt{(\mathbf{L}\mathbf{x}-\mathbf{L}\mathbf{x}')^T(\mathbf{L}\mathbf{x}-\mathbf{L}\mathbf{x}')} \end{aligned} \tag{2.2}$$

Thus, a Mahalanobis distance is implicitly equivalent to computing the Euclidean distance after the linear projection of data characterized by $\mathbf{L}$.

2.1.2 Online Metric Learning

In online metric learning, the algorithm receives training constraints one at a time and update at each step the current hypothesis. Two kinds of constraints, i.e., pairwise and triplet constraints, are widely adopted in existing studies. A pairwise constraint consists of two similar or dissimilar instances, while a triplet constraint takes the form (A, B, C), where instance A is similar to instance B but is dissimilar to instance C. Online metric learning algorithms are highly effective to cope with large-scale problems where offline methods typically fail due to time and space limitations.

POLA (Shalev-Shwartz et al., 2004a)

POLA (Pseudo-metric Online Learning Algorithm) is the first online Mahalanobis distance learning approach, which learns a parameter matrix $\mathbf{M}$ and a threshold $b \geq 1$. At each iteration, POLA receives a pair $(\mathbf{x}_i, \mathbf{x}_j, y_{ij})$, where $y_{ij} = 1$ if $\mathbf{x}_i$ is similar to $\mathbf{x}_j$ and $y_{ij} = 0$ otherwise. It performs two successive orthogonal projections:

Step 1: POLA projects the current solution $(\mathbf{M}^{t-1}, b^{t-1})$ onto the set $\{(\mathbf{M}, b) \in \mathbb{R}^{d^2+1} : [y_{ij}\big(d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - b\big) + 1]_+ = 0\}$, which requires the distance between two instances of same (different) labels to be below (above) the threshold b with a margin 1. As a result, an intermediate solution $(\mathbf{M}^{t-\frac{1}{2}}, b^{t-\frac{1}{2}})$ that satisfies this constraint and stays as close as possible to the previous solution can be obtained at this step.

Step 2: POLA projects the intermediate solution $(\mathbf{M}^{t-\frac{1}{2}}, b^{t-\frac{1}{2}})$ onto the set $\{(\mathbf{M}, b) \in \mathbb{R}^{d^2+1} : \mathbf{M} \in \mathbb{S}^d_+, b \geq 1\}$, which produces a new solution $(\mathbf{M}^t, b^t)$ that yields a valid Mahalanobis distance.

LEGO (Jain et al., 2008a)

LEGO (LogDet Exact Gradient Online) is an improved version of POLA (Shalev-Shwartz et al., 2004a), which updates a learned Mahalanobis metric based on LogDet regularization

and gradient descent. It features tighter regret bound, more efficient updates, and better practical performance.

$$d_A(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T A(\mathbf{x}_i - \mathbf{x}_j) \tag{2.3}$$

$$A_{t+1} = \arg\min_{A \succ 0} D(A, A_t) + \eta l\big(d_A(\mathbf{x}_i^t, \mathbf{x}_j^t), y_t\big) \tag{2.4}$$

where $D(A, A_t) = tr(AA_t^{-1}) - \log\det(AA_t^{-1}) - d$ is a regularization function and $\eta > 0$ is the regularization parameter.

RDML (Jin et al., 2009a)

RDML is a flexible version of POLA that does not force the input margin constraint to be satisfied. At each step t, it performs a gradient descent step as follows:

$$\mathbf{M}^t = \pi_{\mathbb{S}_+^d}\big(\mathbf{M}^{t-1} - \lambda y_{ij}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\big) \tag{2.5}$$

where $\pi_{\mathbb{S}_+^d}$ is the projection to PSD cone. The parameter λ is the trade-off between satisfying the pairwise constraint and staying close to the previous matrix $\mathbf{M}^{t-1}$. This update can be performed by resolving a regular convex quadratic program instead of resorting to eigenvalue decomposition like POLA.

OASIS (Chechik et al., 2010a)

In contrast to the approaches discussed above, OASIS learns a bilinear similarity metric of the form:

$$\mathcal{S}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{M} \mathbf{x}_j \tag{2.6}$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is not required to be PSD nor symmetric. The bilinear similarity metric has two advantages (Bellet et al., 2013). First, it is efficiently computable for sparse inputs. If $\mathbf{x}_i$ and $\mathbf{x}_j$ have n_1 and n_2 non-zero elements respectively, $\mathcal{S}_{\mathbf{M}}$ can be computed in $O(n_1 n_2)$ time. Second, $\mathbf{M}$ does not need to be a square matrix. That is, the bilinear similarity can be defined between instances of different dimensions.

$\mathcal{S}_{\mathbf{M}}$ can be optimized using the passive-aggressive family of learning algorithms with a large margin criterion and an efficient hinge loss cost. Specifically, $\mathbf{M}$ is first initialized to be $\mathbf{I}$. Then at each step, the algorithm samples a triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ where $\mathbf{x}_i$ is similar to $\mathbf{x}_j$ but dissimilar to $\mathbf{x}_k$, and solves the following convex problem:

$$\begin{aligned} \mathbf{M}^t &= \arg\min_{\mathbf{M},\xi} \frac{1}{2}||\mathbf{M}^t - \mathbf{M}^{t-1}||_{\mathcal{F}}^2 + C\xi \\ \text{s.t. } & 1 - d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_j) + d_{\mathbf{M}}^2(\mathbf{x}_i, \mathbf{x}_k) \leq \xi \\ & \xi \geq 0 \end{aligned} \tag{2.7}$$

where $C \geq 0$ is the trade-off between satisfying the triplet constraint and staying as close as possible to the previous parameter matrix $\mathbf{M}^{t-1}$.

OPML (Li et al., 2018a)

To achieve a low computational cost when performing online metric learning for large-scale data, OPML adopts a one-pass triplet construction strategy, and employs a closed-form solution to update the Mahalanobis distance metric for new coming samples, leading to a low space (i.e., $O(d)$) and time (i.e., $O(d^2)$) complexity. Specifically, the optimization problem of OPML is defined as:

$$\mathbf{L}_t = \arg\min_{\mathbf{L}} \frac{1}{2}||\mathbf{L} - \mathbf{L}_{t-1}||_{\mathcal{F}}^2 + \frac{\gamma}{2}[1 + ||\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)||_2^2 - ||\mathbf{L}(\mathbf{x}_i - \mathbf{x}_k)||_2^2]_+ \tag{2.8}$$

where $\gamma \in (0, \frac{1}{4})$ is the regularization parameter.

2.1.3 Online Adaptive Metric Learning

Despite their success in some real-world applications, existing OML algorithms have several intrinsic drawbacks. First, most of these algorithms make an attempt to learn a linear similarity metric, e.g., a Mahalanobis or bilinear metric, from the input data. The projection mappings learned by these metrics are purely linear, which is unable to precisely

measure the non-linear semantic similarities among instances in complex real-world applications. Second, the performance of existing OML solutions depends heavily on the quality of input constraints. Only those "hard" constraints within the input contribute to the learning of these models. Here a constraint is considered as "hard" if the dissimilar instances in it are computed as similar by current metric and vice-versa. Hence, existing OML solutions are biased towards these "hard" constraints, thereby weakening their robustness. Last but not least, in practical scenarios, the data are often generated from multiple sources such as Google Search and Twitter, and we have little control over these data sources. Filtering out "hard" constraints from the massive data produced by these sources is computationally expensive and even impossible in a real-time setting. Therefore, a real-world stream usually contains a limited amount of "hard" constraints, and existing OML approaches often suffer the issue of low constraint utilization, which degrades the model performance.

To overcome these model defects, the "Online Adaptive Metric Learning" (OAML) open challenge (Gao et al., 2019) is introduced, i.e., how to learn a metric model that automatically adjusts its model complexity as more constraints are observed in the input stream and maintain high constraint utilization during the learning process? In other words, the hypothesis space of the learned metric function is dynamic, including functions from pure linear stage to highly non-linear regime. We will discuss our proposed solution to this open challenge in Chapter 3.

2.2 Imbalanced Classification

Class imbalance in the datasets can dramatically skew the performance of classifiers, introducing a prediction bias towards the majority class. As a result, it becomes quite difficult for learners to effectively distinguish between the majority and minority classes, especially when the class imbalance is extreme. Moreover, the prediction bias of the learner may lead to severe consequences in situations where the occurrence of false negatives is relatively costlier

than false positives. In this section, we briefly summarize recently published studies focusing on imbalanced learning. These methods can be roughly divided into three general categories, i.e., data-level methods, algorithm-level methods, and hybrid methods (Krawczyk, 2016).

2.2.1 Data-Level Methods

Data-level methods can be further sub-grouped into data-sampling methods and feature-selection methods. The data-sampling methods modify the collection of examples via re-sampling to balance class distributions and can be roughly classified into two categories: (1) Under-sampling methods: these methods balance the distribution between the majority and minority classes by removing the majority class instances from the given dataset, where the removal is largely performed randomly. (2) Over-sampling methods: these methods balance the class distribution by adding minority class samples to the given dataset, where the replication is done either randomly or using a specified algorithmic approach. Feature selection methods may also help select the most influential features (or attributes) that can yield unique knowledge for inter-class discrimination. It reduces the adverse effect of class imbalance on classification performance (Yin et al., 2013; Zheng et al., 2004).

Instance Hardness Threshold (IHT) (Smith et al., 2014) propose to performs under-sampling based on instance hardness. They posit that each instance in a dataset has a hardness property indicating the likelihood that it will be misclassified. For example, outliers are expected to have high instance hardness since a learner will have to overfit to classify them correctly. Those instances with their hardness higher than a user-specified threshold are removed from the dataset. Evolutionary Under-Sampling (EUS) (Triguero et al., 2016) is introduced to deal with the severe class imbalance in big data. EUS provides a fitness function for prototype selection, where the fitness function aims to find a proper balance between under-sampling of training data instances and classification performance. SMOTE (Chawla et al., 2002) is the first synthetic minority over-sampling technique. The

minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. MWMOTE (Barua et al., 2014) first identifies the hard-to-learn informative minority class samples and then uses the weighted version of these samples to generate synthetic samples. Recently, based on k-means clustering and SMOTE, KMEANS-SMOTE (Last et al., 2017) is introduced to eliminate both intra-class and inter-class imbalance while at the same time avoiding the generation of noisy samples.

Addressing class imbalance via feature selection methods is a largely unexplored research area. Zheng et al. (Zheng et al., 2004) investigate feature selection for text categorization on imbalanced data. Their framework selects the positive and negative features separately, and then combine them explicitly afterward. Yin et al. (Yin et al., 2013) provide a new feature selection approach based on class decomposition. Specifically, they partition the majority classes into smaller pseudo-subclasses and assign the pseudo-class labels accordingly. In this process, Hellinger's distance is used as a measure of distribution divergence for feature selection.

Although data-level methods are able to partially address the class imbalance issues, they have some intrinsic drawbacks. For highly unbalanced data, data-sampling methods either discard a large number of samples from the majority class or introduce many synthetic samples into the minority class. It either loses important information (under-sampling) or improperly increases the adverse correlation among samples (over-sampling), leading to performance degradation (Krawczyk, 2016). On the other hand, the extra computational cost of feature-selection methods is an issue of concern when applying them for imbalanced learning (Ali et al., 2015).

2.2.2 Algorithm-Level Methods

Algorithm-level methods attempt to modify existing learners to alleviate their bias towards the majority classes. The most popular branch is the cost-sensitive approach that assigns a

higher cost on incorrectly classifying the minority class instances. Cost-sensitive SVM (Cao et al., 2013a) is an effective wrapper framework that incorporates the evaluation measure, such as AUC and G-mean, into the objective function to improve the classification performance. It simultaneously optimizes the best pair of feature subset, intrinsic parameters, and misclassification cost parameters. In a related study PSOCS-NN (Cao et al., 2013b), the authors propose to replace cost-sensitive SVM with a cost-sensitive neural network, where Particle Swarm Optimization (PSO) is used to train the model. CLEMS (Huang and Lin, 2017) introduces a cost-sensitive label embedding technique that takes the cost function of interest into account. The proposed algorithm approximates the cost information with the distances of the embedded vectors by using the classic multidimensional scaling approach for manifold learning. CS-DMLP (Díaz-Vico et al., 2018) is a deep multi-layer perceptron model utilizing cost-sensitive learning to regularize the posterior probability distribution predicted for a given sample.

Despite the success of cost-sensitive approaches in addressing well-defined imbalanced learning research problems, they may not be suitable for solving practical problems, since they usually depend on a cost matrix, which is often not given by experts beforehand and sometimes even impossible to set in some cases. (Krawczyk, 2016).

2.2.3 Hybrid Methods

Hybrid methods concentrate on combining data-level and algorithm-level methods to extract their strong points and reduce their weaknesses. Building robust and efficient learners by merging data-level solutions with classifier ensembles is quite popular. A typical example is an ensemble model named WEOB2 (Wang et al., 2015). It combines under-sampling based online bagging with adaptive weight adjustment to effectively adjust the learning bias from the majority class to the minority class. Some other works propose hybrid sampling techniques and cost-sensitive approaches. For example, Cost-sensitive MCS (Krawczyk et al.,

2014) is an effective ensemble of cost-sensitive decision trees for imbalanced classification. The base classifiers in this ensemble are constructed based on a given cost matrix and are trained on random feature subspaces to ensure the diversity of the ensemble members.

Unfortunately, there are some open questions that remain to be answered in this field (Krawczyk, 2016). First, there is a lack of a good understanding of diversity in imbalanced learning. For example, is the diversity in majority classes as important as that in minority classes? How to guarantee diversity during the learning process? Second, there is no clue how large ensembles should be constructed. Is there a way to compute the ideal ensemble size for a given imbalanced dataset? Third, there is no clear indicator on how to utilize base learners in an imbalanced ensemble model for prediction. Most of the imbalanced ensemble techniques utilize the majority voting combination method. However, it may not be suitable for imbalanced learning, especially in the case of randomized methods. The individual qualities of base learners trained using sampling methods may differ since they are based on examples with varying difficulties. This property should be considered when combining predictions of base learners to make the final decision.

2.2.4 SetConv

To address the drawbacks of existing solutions, we propose a novel set convolution operation SetConv and a new training strategy episodic training to assist learning from imbalanced data distributions (Gao et al., 2020). Compared to existing studies, our framework has several benefits:

Data-Sensitive Convolution

SetConv explicitly estimates the convolution kernel weights for each input sample based on its relation to the minority class and extracts discriminative features from it for classification

using the learned weights. Thus, SetConv is data-sensitive, which helps the model customize the feature extraction process of input samples, and potentially improves the model performance.

Automatic Class Balancing

At each iteration, no matter how many data instances of a class is fed into the SetConv layer, it always produces a single class representative containing the most discriminative and representative information of these data. The subsequent classifier, which takes these class representatives as input, always perceives a balanced class distribution.

No dependence on unknown prior knowledge

Thanks to episodic training, our framework has no dependence on unknown prior knowledge. The only prior knowledge required is the class imbalance ratio, which can be easily obtained from data in real-world applications.

The proposed SetConv framework will be discussed in details in Chapter 4.

2.3 Open-World Classification

In contrast to the common closed set (or static environment) assumption, in real-world classification tasks, it is usually difficult to collect examples that exhaust all classes when training a classifier. A more realistic scenario is often open and non-stationary, where incomplete world knowledge is given at training time, and unknown classes can be sent to the model at test time. The classifiers are required to not only correctly classify instances from the known classes, but also identify unseen ones. In this section, we describe this essential problem, i.e., *concept evolution*, in the context of open-world stream classification.

2.3.1 Concept Evolution

Concept evolution in stream classification refers to the scenario where instances from novel classes, i.e., classes that are never observed before, occur along the stream. In other words, the classifier is never trained or updated using instances associated with these classes. Existing studies typically address this problem by first detecting those novel class instances, and then updating the classifier to incorporate their information for future predictions.

A series of studies (Masud et al., 2009a, 2011a; Al-Khateeb et al., 2012a; Haque et al., 2016b) have utilized an unsupervised algorithm named as q-NSC to detect novel classes. In q-NSC, a test instance is declared as a filtered outlier or F-outlier, if it falls outside of the decision boundary of a cluster ensemble produced via a clustering algorithm such as K-Means. Then, the model finds the q, c-neighborhood of an F-outlier $\boldsymbol{x}$ ($q, c(\boldsymbol{x})$ for short), which is the set of q instances from class c that are nearest to $\boldsymbol{x}$. The q-NSC score of $\boldsymbol{x}$ is given by:

$$\text{q-NSC}(\boldsymbol{x}) = \frac{\bar{D}_{c_{min},q}(\boldsymbol{x}) - \bar{D}_{c_{out},q}(\boldsymbol{x})}{\max(\bar{D}_{c_{min},q}(\boldsymbol{x}), \bar{D}_{c_{out},q}(\boldsymbol{x}))} \tag{2.9}$$

where $\bar{D}_{c_{out},q}(\boldsymbol{x})$ is the mean distance of an F-outlier $\boldsymbol{x}$ to its q-nearest F-outlier neighbors and $\bar{D}_{c_{min},q}(\boldsymbol{x})$ is the minimum among all $\bar{D}_{c,q}(\boldsymbol{x}), c \in \{\text{Set of existing classes}\}$. The q-NSC score considers both cohesion and separation, and yields a value in $[-1, +1]$. A positive q-NSC score indicates that $\boldsymbol{x}$ is far away from instances of existing classes and is very likely to be from novel classes. Alternatively, Senc-MaS (Mu et al., 2017a) and AMaSC (Zhang et al., 2018a) store the frequent directions of stream data in a global sketch. If an incoming instance is far away from all the stored frequent directions, it is taken as coming from a novel class.

In spite of the success of these methods on open-world stream classification, they rely on the existence of strong intra-class cohesion and large inter-class separation in observed feature space to detect novel classes (Masud et al., 2011b). That is, in the observed feature

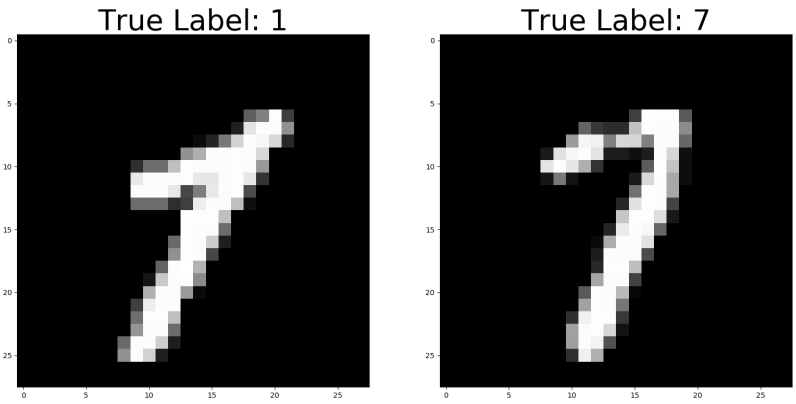


Figure 2.1: Two similar digits 1 and 7.

space, instances from the same class are assumed to be closer than those of different classes, which is referred as intrinsic cohesion and separation assumption. However, this assumption may not be valid in real-world scenarios, especially for high-dimensional sparse data. For example, as shown in Fig. 2.1, in a handwritten digit recognition application, images of digit "1" may look very similar to those of digit "7".

To address this challenge, we propose a semi-supervised algorithm named as SIM that leverages a metric learning mechanism to actively search for a latent feature space suitable for both classification and novel class detection. The unified multi-task open-world classifier in SIM jointly performs metric learning, stream classification, and novel class detection. We discuss SIM in details in Chapter 5.

CHAPTER 3

TOWARDS SELF-ADAPTIVE METRIC LEARNING ON THE FLY [1]

3.1 Approach

In this chapter, we discuss the OAHU framework that aims to address the "Online Adaptive Metric Learning" (OAML) open challenge in details.

3.1.1 Problem Setting

With an input triplet constraint stream $S = \{(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-})\}_{t=1}^{T}$, where $\boldsymbol{x_t} \in \mathcal{R}^d$ (anchor) is similar to $\boldsymbol{x_t^+} \in \mathcal{R}^d$ (positive) but is dissimilar to $\boldsymbol{x_t^-} \in \mathcal{R}^d$ (negative), the objective of OAML is to learn a model $\boldsymbol{F} : \mathcal{R}^d \mapsto \mathcal{R}^{d'}$ such that $||\boldsymbol{F}(\boldsymbol{x_t}) - \boldsymbol{F}(\boldsymbol{x_t^+})||_2 \ll ||\boldsymbol{F}(\boldsymbol{x_t}) - \boldsymbol{F}(\boldsymbol{x_t^-})||_2$. Moreover, the metric model $\boldsymbol{F}$ should have an adaptive complexity so that its hypothesis space is automatically expanded or shrunk as necessary, and can be learned from the input triplet constraint stream in an online fashion with a high constraint utilization rate.

In real-world applications, we may generate the input triplet constraint stream as follows. First, user clicks are tracked and logged, and a few seed triplets are constructed from these logs. Then, transitive closures is applied over these seed triplets to form more triplets. Specifically, if $(\boldsymbol{x_1}, \boldsymbol{x_2})$ and $(\boldsymbol{x_1}, \boldsymbol{x_3})$ are two similar pairs, then $(\boldsymbol{x_2}, \boldsymbol{x_3})$ is also a similar pair. If $(\boldsymbol{x_1}, \boldsymbol{x_2})$ and $(\boldsymbol{x_2}, \boldsymbol{x_3})$ are two similar pairs, then $(\boldsymbol{x_1}, \boldsymbol{x_3})$ is also a similar pair. On the other hand, If $(\boldsymbol{x_1}, \boldsymbol{x_2})$ is a similar pair and $(\boldsymbol{x_1}, \boldsymbol{x_3})$ is a dissimilar pair, then $\boldsymbol{x_2}$ and $\boldsymbol{x_3}$ is dissimilar to each other. If $(\boldsymbol{x_1}, \boldsymbol{x_2})$ is a similar pair and $(\boldsymbol{x_2}, \boldsymbol{x_3})$ is a dissimilar pair, then $\boldsymbol{x_1}$ and $\boldsymbol{x_3}$ is dissimilar to each other. The generated triplets are inserted into the stream in chronological order of creation.

[1] This chapter contains material previously published as: Yang Gao, Yi-Fan Li, Swarup Chandra, Latifur Khan, and Bhavani Thuraisingham. "Towards self-adaptive metric learning on the fly." In *The World Wide Web Conference (WWW)*, pp. 503-513. 2019, https://doi.org/10.1145/3308558.3313503. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

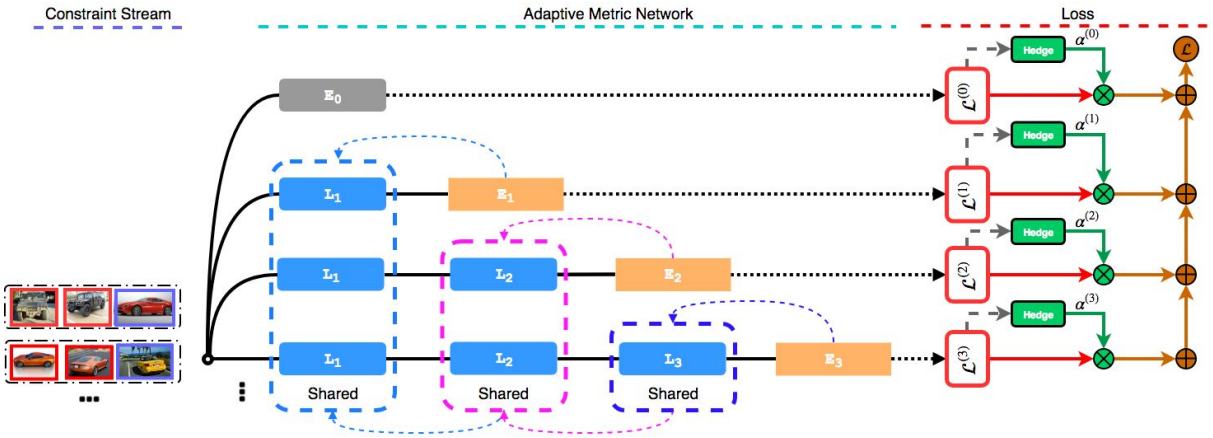


Figure 3.1: Overall architecture of the proposed framework OAHU. The chromatic dashed arrows indicate the gradient backpropagation contributions. Each $L_i \in \{L_1, L_2, \dots\}$ represents a linear transformation layer followed by a ReLU activation. $E_i \in \{E_0, E_1, \dots\}$ is the metric embedding layer of the i_{th} metric model, which is attached to either an input or a hidden layer. Note E_0 represents a pure linear metric model, i.e., a linear transformation from the observed feature space to the metric embedding space.

3.1.2 Overview

To address the online adaptive metric learning challenge, we need to answer an important question: *when* and *how* to change the "complexity" of a metric model in the online learning process? In this section, we discuss our proposed framework OAHU in details, which is a natural solution to this question.

The architecture of the proposed OAHU framework is illustrated in Figure 3.1. Inspired by recent works (Srinivas and Babu, 2016; Huang et al., 2016), we automatically adapt the effective depth of an over-complete network based on input constraints to learn a metric function with appropriate complexity. Given a feedforward neural network with L hidden layers, we attach an independent metric embedding layer to each of the network input and hidden layers. Every embedding layer is a projection to a latent space where similar instances are closer to each other than those dissimilar instances. Thus, in contrast to existing online metric learning solutions that usually learn a linear metric model, the proposed OAHU framework is indeed an ensemble of metric models having various complexities and sharing

low-level knowledge. As shown in Figure 3.1, we use $E_l \in \{E_0, E_1, E_2, \ldots, E_L\}$ to denote the l^{th} metric model in OAHU, which is the sub-network starting from the input layer to the l^{th} metric embedding layer. Note that E_0 represents a linear transformation from the observed feature space to the metric embedding space, and is the simplest model in OAHU. Each metric model E_l is associated with a unique weight $\alpha^{(l)} \in [0, 1]$, which measures the importance of E_l in OAHU.

At step t, for an input triplet constraint $(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-})$, the metric embedding of $\boldsymbol{x_t^*} \in \{\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}\}$ generated by E_l is

$$f^{(l)}(\boldsymbol{x_t^*}) = h^{(l)}\Theta^{(l)} \tag{3.1}$$

where $h^{(l)} = \sigma(W^{(l)}h^{(l-1)})$ $(l \geq 1, l \in \mathbb{N})$ represents the activation of l^{th} hidden layer, and $h^{(0)} = \boldsymbol{x_t^*}$. To reduce the potential model search space and accelerate the training, we explicitly constrain the learned metric embedding $f^{(l)}(\boldsymbol{x_t^*})$ residing on a unit sphere, i.e., $||f^{(l)}(\boldsymbol{x_t^*})||_2 = 1$.

Once the metric embeddings $f^{(l)}(\boldsymbol{x_t^*})$ are obtained, we evaluate the similarity and dissimilarity errors based on these embeddings and produce a local loss $\mathcal{L}^{(l)}$ for metric model E_l. The overall loss introduced by this input triplet is given by

$$\mathcal{L}_{overall}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}) = \sum_{l=0}^{L} \alpha^{(l)} \cdot \mathcal{L}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}) \tag{3.2}$$

Apparently, in addition to the network parameters, our framework OAHU has introduced two sets of new parameters, i.e., $\Theta^{(l)}$ (parameters of l^{th} metric embedding layer) and $\alpha^{(l)}$ (weight of E_l), which have to be learned in the online training process. Therefore, the final optimization problem to solve in OAHU at step t is:

$$\begin{aligned} \underset{\Theta^{(l)}, W^{(l)}, \alpha^{(l)}}{\text{minimize}} \quad & \mathcal{L}_{overall} \\ \text{subject to} \quad & ||f^{(l)}(\boldsymbol{x_t^*})||_2 = 1, \forall l = 0, \ldots, L. \end{aligned} \tag{3.3}$$

To evaluate $\mathcal{L}^{(l)}$, in Section 3.1.3, we introduce a novel Adaptive-Bound Triplet Loss (ABTL). In addition, we update $\Theta^{(l)}$, $W^{(l)}$ and $\alpha^{(l)}$ in an online fashion with a novel Adaptive Hedge Update (AHU) method, which will be discussed in Section 3.1.4.

3.1.3 Adaptive-Bound Triplet Loss

Limitations of Existing Pairwise/Triplet Loss

Existing online metric learning solutions typically optimize a pairwise loss (Shalev-Shwartz et al., 2004b) or a triplet loss (Li et al., 2018b) to learn a similarity metric, which are described in Eq. 3.4 and Eq. 3.5 respectively:

$$L(x_t, x'_t) = \max\{0, y_t(d^2(x_t, x'_t) - b) + 1\} \tag{3.4}$$

$$L(x_t, x_t^+, x_t^-) = \max\{0, b + d(x_t, x_t^+) - d(x_t, x_t^-)\} \tag{3.5}$$

where $y_t = 1$ and $y_t = -1$ indicate that x_t is similar or dissimilar to x'_t respectively, and $b \in \mathcal{R}$ is a user-specified constant margin. In contrast to the triplet loss, which takes both similarity and dissimilarity relations into account, the pairwise loss only focuses on one of these two relations at a time, leading to poor metric quality. On the other hand, an improper margin causes the restriction of a triplet loss to be loose, which may result in many failure cases. In these situations, the triplet loss is evaluated to be zero, but the dissimilar instances are closer than the similar instances. For example, as shown in Figure 3.2, although the positive instance x_t^+ is erroneously closer to the negative instance x_t^- rather than the anchor x_t, the loss corresponding to this triplet (x_t, x_t^+, x_t^-) is 0 since the constraint $b + d(x_t, x_t^+) - d(x_t, x_t^-) \leq 0$ is satisfied. Models optimizing the triplet loss would incorrectly ignore this input triplet constraint. In addition, selecting an appropriate margin for a triplet loss is hard and requires extensive domain knowledge due to its data-sensitive property. Furthermore, as the model improves over time, this margin needs to be increased to avoid failure cases.

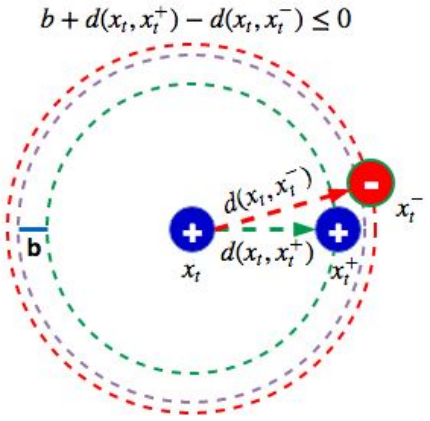


Figure 3.2: A failure case example in traditional fixed-margin triplet loss.

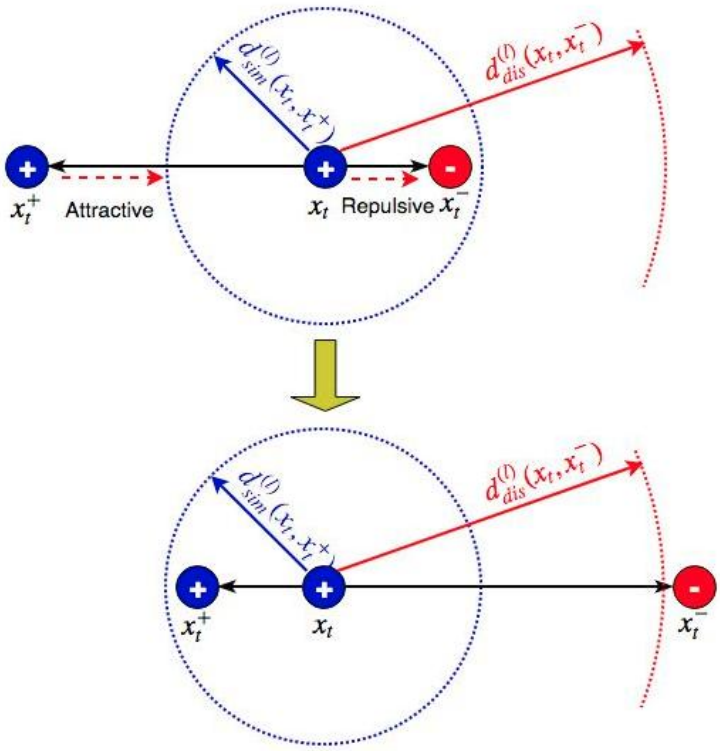


Figure 3.3: Schematic illustration of ABTL.

Observing these drawbacks, we propose an adaptive-bound triplet loss that naturally solves these issues.

Adaptive-Bound Triplet Loss (ABTL)

In our framework, similar instances are required to be mutually attractive to each other, while dissimilar instances are required to be mutually repulsive to each other. Therefore, for any input triplet $(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-})$, we can define an attractive loss $\mathcal{L}_{attr} \in [0, 1]$ between $\boldsymbol{x_t}$ and $\boldsymbol{x_t^+}$, and a repulsive loss $\mathcal{L}_{rep} \in [0, 1]$ between $\boldsymbol{x_t}$ and $\boldsymbol{x_t^-}$. To compute these two losses, for the l^{th} metric model $E_l \in \{E_0, \ldots, E_L\}$, we rely on a distance measure $D^{(l)}(\boldsymbol{x_i}, \boldsymbol{x_j})$ that is defined as follows:

$$D^{(l)}(\boldsymbol{x_i}, \boldsymbol{x_j}) = ||f^{(l)}(\boldsymbol{x_i}) - f^{(l)}(\boldsymbol{x_j})||_2, \quad \forall l = 0, 1, 2, \ldots, L \tag{3.6}$$

where $f^{(l)}(\boldsymbol{x})\Big(||f^{(l)}(\boldsymbol{x})||_2 = 1\Big)$ represents the embedding produced by E_l and $D^{(l)}(\boldsymbol{x_i}, \boldsymbol{x_j}) \in [0, 2]$. For convenience, let $D^{(l)}_{orig}(\boldsymbol{x_1}, \boldsymbol{x_2})$ be the distance between $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ before updating $f^{(l)}$ (i.e., updating $\Theta^{(l)}$ and $\{W^{(j)}\}_{j=0}^{l}$), and $D^{(l)}_{update}(\boldsymbol{x_1}, \boldsymbol{x_2})$ denotes the distance after applying the update.

The main idea of ABTL is illustrated in Figure 3.3. The goal is to find a latent feature space where the distance $D^{(l)}_{update}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ between two similar instances $\boldsymbol{x_t}$ and $\boldsymbol{x_t^+}$ is less than or equal to a similarity threshold $d^{(l)}_{sim}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ and the distance $D^{(l)}_{update}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$ between two dissimilar instances $\boldsymbol{x_t}$ and $\boldsymbol{x_t^-}$ is greater than or equal to a dissimilarity threshold $d^{(l)}_{dis}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$. In this space, both the attractive loss $\mathcal{L}^{(l)}_{attr}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ and repulsive loss $\mathcal{L}^{(l)}_{rep}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$ drop to zero. These constraints are formally presented in Eq. 3.7:

$$\begin{cases} D^{(l)}_{update}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) \leq d^{(l)}_{sim}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) \\ \\ D^{(l)}_{update}(\boldsymbol{x_t}, \boldsymbol{x_t^-}) \geq d^{(l)}_{dis}(\boldsymbol{x_t}, \boldsymbol{x_t^-}) \end{cases} \tag{3.7}$$

Note that $d^{(l)}_{sim}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ and $d^{(l)}_{dis}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$ vary with different input constraints. To determine $d^{(l)}_{sim}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ and $d^{(l)}_{dis}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$, we introduce a user-specified hyper-parameter $\tau > 0$ to

explicitly constrain their ranges. It reduces the model search space and accelerates the training. We denote the upper bound of $d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ as $\mathcal{T}_{sim}^{(l)}$, and the lower bound of $d_{dis}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$ as $\mathcal{T}_{dis}^{(l)}$:

$$\begin{cases} \mathcal{T}_{sim}^{(l)} = \tau \geq d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) \\ \\ \mathcal{T}_{dis}^{(l)} = 2 - \tau \leq d_{dis}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-}) \end{cases} \tag{3.8}$$

Without loss of generality, we formulate $d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ and $d_{dis}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$ as:

$$\begin{cases} d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) = a_1 \mathrm{e}^{D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})} + b_1 \\ \\ d_{dis}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-}) = -a_2 \mathrm{e}^{-D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})} + b_2 \end{cases} \tag{3.9}$$

Here, a_1, a_2, b_1 and b_2 are coefficients need to be determined. We choose this formulation so that $d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ and $d_{dis}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$ are monotonically increasing and decreasing with $D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$, respectively. It means that $d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ preserves the relative similarity among data. If $D_{orig}^{(l)}(\boldsymbol{x_1}, \boldsymbol{x_2}) < D_{orig}^{(l)}(\boldsymbol{x_1}, \boldsymbol{x_3})$, $\boldsymbol{x_1}$ should still be closer to $\boldsymbol{x_2}$ than $\boldsymbol{x_3}$ after updating, i.e., $d_{sim}^{(l)}(\boldsymbol{x_1}, \boldsymbol{x_2}) < d_{sim}^{(l)}(\boldsymbol{x_1}, \boldsymbol{x_3})$. This property is critical to fine-grained similarity comparison and is of tremendous benefit to many practical applications such as image retrieval.

To determine the coefficients a_1, b_1, a_2 and b_2, we require $d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ and $d_{dis}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$ to satisfy the following boundary conditions:

$$\begin{cases} d_{sim}^{(l)}|_{D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})=2} \leq \mathcal{T}_{sim}^{(l)} = \tau \quad d_{sim}^{(l)}|_{D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})=0} = 0 \\ d_{dis}^{(l)}|_{D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})=0} \geq \mathcal{T}_{dis}^{(l)} = 2 - \tau \quad d_{dis}^{(l)}|_{D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})=2} = 2 \end{cases} \tag{3.10}$$

Therefore,

$$\begin{cases} d_{sim}^{(l)}\big(\boldsymbol{x_t}, \boldsymbol{x_t^+}\big) = \frac{\tau}{\mathrm{e}^2 - 1}\big(\mathrm{e}^{D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})} - 1\big) \\ \\ d_{dis}^{(l)}\big(\boldsymbol{x_t}, \boldsymbol{x_t^-}\big) = -\frac{2-(2-\tau)}{1-\mathrm{e}^{-2}}\big(\mathrm{e}^{-D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})} - 1\big) + (2 - \tau) \end{cases} \tag{3.11}$$

Given $d_{sim}^{(l)}$ and $d_{dis}^{(l)}$, the attractive loss $\mathcal{L}_{attr}$ and repulsive loss $\mathcal{L}_{rep}$ are evaluated by

$$\begin{cases} \mathcal{L}_{attr}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) = \max\left\{0, \frac{1}{2-d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})} D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) - \frac{d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})}{2-d_{sim}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})}\right\} \\ \\ \mathcal{L}_{rep}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-}) = \max\left\{0, \frac{-1}{d_{dis}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})} D_{orig}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-}) + 1\right\} \end{cases} \tag{3.12}$$

The local loss $\mathcal{L}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-})$ is simply the average of the attractive loss and the repulsive loss. It measures both the similarity and dissimilarity errors of an input triplet $(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-})$ for the metric model E_l.

$$\mathcal{L}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}) = \frac{1}{2}\Big(\mathcal{L}_{attr}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) + \mathcal{L}_{rep}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})\Big) \tag{3.13}$$

Note that there is an important question that remains to be answered: what is the best value of τ? In fact, the theoretical optimal range of τ is $(0, \frac{2}{3})$, which is proved in Theorem 1. Therefore, the best value of τ can be selected empirically within this range via cross-validation in the experiment.

Theorem 1. *With the proposed adaptive-bound triplet loss, if $\tau \in (0, \frac{2}{3})$, different classes can be separated in the learned metric embedding space.*

Proof. Let $D(c_1, c_2)$ denotes the minimal distance between classes c_1 and c_2, i.e., the distance between two closest instances from c_1 and c_2 respectively. Consider an arbitrary quadrupole $(x_1, x_2, x_3, x_4) \in \mathcal{Q}$ where $\{x_1, x_2\} \in c_1$, $\{x_3, x_4\} \in c_2$, and $\mathcal{Q}$ is the set of all possible quadrupoles generated from class c_1 and c_2. Suppose (x_2, x_3) is the closest dissimilar pair among all possible dissimilar pairs that can be extracted from (x_1, x_2, x_3, x_4) [2]. We first prove that the lower bound of $D(c_1, c_2)$ is given by $\min_{(x_1,x_2,x_3,x_4)\in\mathcal{Q}} D^{(l)}(x_1, x_4) - D^{(l)}(x_1, x_2) - D^{(l)}(x_3, x_4)$. Due to the triangle inequality property of p-norm distance, we have

$$\begin{aligned} D^{(l)}(x_1, x_4) &\leq D^{(l)}(x_1, x_2) + D^{(l)}(x_2, x_4) \\ &\leq D^{(l)}(x_1, x_2) + D^{(l)}(x_2, x_3) + D^{(l)}(x_3, x_4) \end{aligned} \tag{3.14}$$

[2] it can always be achieved by re-arranging symbols of x_1,x_2,x_3 and x_4

Therefore,

$$\begin{aligned} D(c_1, c_2) &= \min_{(x_1,x_2,x_3,x_4)\in\mathcal{Q}} D^{(l)}(x_2, x_3) \\ &\geq \min_{(x_1,x_2,x_3,x_4)\in\mathcal{Q}} D^{(l)}(x_1, x_4) - D^{(l)}(x_1, x_2) - D^{(l)}(x_3, x_4) \end{aligned} \tag{3.15}$$

By optimizing the adaptive-bound triplet loss, the following constraints are satisfied

$$\begin{cases} D^{(l)}(x_1, x_2) \leq d^{(l)}_{sim}(x_1, x_2) \leq \mathcal{T}^{(l)}_{sim} \\ D^{(l)}(x_3, x_4) \leq d^{(l)}_{sim}(x_3, x_4) \leq \mathcal{T}^{(l)}_{sim} \\ D^{(l)}(x_1, x_4) \geq d^{(l)}_{dis}(x_1, x_4) \geq \mathcal{T}^{(l)}_{dis} \end{cases} \tag{3.16}$$

Thus

$$\begin{aligned} D(c_1, c_2) &\geq \min_{(x_1,x_2,x_3,x_4)\in\mathcal{Q}} D^{(l)}(x_1, x_4) - D^{(l)}(x_1, x_2) - D^{(l)}(x_3, x_4) \\ &\geq \mathcal{T}^{(l)}_{dis} - 2\mathcal{T}^{(l)}_{sim} \\ &= 2 - \tau - 2\tau \\ &= 2 - 3\tau \end{aligned} \tag{3.17}$$

if $\tau \in (0, \frac{2}{3})$, we have $3\tau < 2$. Therefore,

$$D(c_1, c_2) \geq 2 - 3\tau > 0 \tag{3.18}$$

Eq. 3.18 indicates that the minimal distance between class c_1 and c_2 is always positive so that these two classes are separated.

□

The Advantages of ABTL

In summary, compared to the existing pairwise or triplet loss, the proposed ABTL has several advantages:

- In contrast to the pairwise loss, it simultaneously evaluates both the similarity and dissimilarity errors to learn a good metric function.
- It preserves the relative similarity among data, which is critical to fine-grained similarity learning. Moreover, it eliminates the failure cases of the triplet loss by learning from every input constraint, leading to a higher constraint utilization rate.
- The theoretical optimal range of its sole hyper-parameter τ is provided.

3.1.4 Adaptive Hedge Update (AHU)

The overall loss $\mathcal{L}_{overall}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-})$ in OAHU is:

$$\begin{aligned}\mathcal{L}_{overall}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}) &= \sum_{l=0}^{L} \alpha^{(l)} \cdot \mathcal{L}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}) \\ &= \sum_{l=0}^{L} \frac{\alpha^{(l)}}{2}\Big(\mathcal{L}^{(l)}_{attr}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) + \mathcal{L}^{(l)}_{rep}(\boldsymbol{x_t}, \boldsymbol{x_t^-})\Big)\end{aligned} \tag{3.19}$$

To learn the metric weights $\{\alpha^{(l)}\}_{l=0}^{L}$, we adopt the Hedge Algorithm (Freund and Schapire, 1997). Specifically, all metric weights are uniformly initialized, i.e., $\alpha^{(l)} = \frac{1}{L+1}, l = 0, 1, 2, \ldots, L$. At each iteration, for each metric model E_l, $\alpha^{(l)}$ is updated based on the local loss suffered by E_l:

$$\alpha_{t+1}^{(l)} \leftarrow \begin{cases} \alpha_t^{(l)} \beta^{\mathcal{L}^{(l)}} & \beta^{\min_l \mathcal{L}^{(l)}} \log \mathcal{L}^{(l)} > \beta - 1 \\ \alpha_t^{(l)}[1 - (1-\beta)\mathcal{L}^{(l)}] & otherwise \end{cases} \tag{3.20}$$

where $\beta \in (0, 1)$ is the discount factor. In this way, we maximize the update of $\{\alpha^{(l)}\}_{l=0}^{L}$ at each step. That is, at step t, if a metric model E_l produces a high local loss $\mathcal{L}^{(l)}$, its associated metric weight $\alpha^{(l)}$ tends to gain as much decrement as possible. In contrast, if a metric model E_l produces a low local loss $\mathcal{L}^{(l)}$, its associated metric weight $\alpha^{(l)}$ will be increased as much as possible. Thus, in the proposed framework OAHU, those metric models with good performance are highlighted and vice versa. However, OAHU may excessively

reduce the weights of deeper models and unfairly favor the shallow models since they converge faster at the beginning of the training process, leading to the model bias issue (Chen et al., 2015; Gülçehre et al., 2016). To address this issue, we force the minimal weight of each metric model to be $\frac{s}{L+1}$, where s is the smooth factor. Hence, after updating $\{\alpha^{(l)}\}_{l=0}^{L}$ based on Eq. 3.20, we set the weights as $\alpha_{t+1}^{(l)} = \max(\alpha_{t+1}^{(l)}, \frac{s}{L+1})$. These metric weights are then normalized so that $\sum_{l=0}^{L} \alpha_{t+1}^{(l)} = 1$.

The parameters $\Theta^{(l)}$ and $W^{(l)}$ can be updated via Online Gradient Descent (OGD):

$$\begin{aligned}\Theta_{t+1}^{(l)} &\leftarrow \Theta_t^{(l)} - \eta \nabla_{\Theta_t^{(l)}} \mathcal{L}_{overall}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}) \\ &= \Theta_t^{(l)} - \eta \frac{\alpha^{(l)}}{2} \Big(\nabla_{\Theta_t^{(l)}} \mathcal{L}_{attr}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) + \nabla_{\Theta_t^{(l)}} \mathcal{L}_{rep}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-}) \Big)\end{aligned} \tag{3.21}$$

$$\begin{aligned}W_{t+1}^{(l)} &\leftarrow W_t^{(l)} - \eta \nabla_{W_t^{(l)}} \mathcal{L}_{overall}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}) \\ &= W_t^{(l)} - \eta \sum_{j=l}^{L} \frac{\alpha^{(j)}}{2} \Big(\nabla_{W_t^{(l)}} \mathcal{L}_{attr}^{(j)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}) + \nabla_{W_t^{(l)}} \mathcal{L}_{rep}^{(j)}(\boldsymbol{x_t}, \boldsymbol{x_t^-}) \Big)\end{aligned} \tag{3.22}$$

Since $\Theta^{(l)}$ are the parameters of the metric embedding layer and are specific to the metric model E_l, updating them is simple and is described in Eq. 3.21. In contrast, as shown in Eq. 3.22, because the deeper metric models in OAHU share the low-level knowledge with the shallower metric models, all those models deeper than l layers contribute to the update of $W^{(l)}$. In summary, the training process of OAHU is outlined in Algorithm 1.

3.1.5 Regret Bound

In general, the Hedge Algorithm (Freund and Schapire, 1997) has a regret bound of $R_T \leq \sqrt{TlnN}$, where N denotes the number of experts and T is the number of trails. In our case, each expert corresponds to a particular metric model in the ensemble and each trial corresponds to a particular input triplet. Therefore, $N = L+1$ is the number of metric models in OAHU and T is the number of input triplets. The average worst-case convergence rate of OAHU is hence $O(\sqrt{ln(N)/T})$. Apparently, as long as sufficient triplets are provided

Algorithm 1 OAHU: Online Metric Learning with Adaptive Hedge Update

Require: Discount Factor $\beta \in (0,1)$; Smooth Factor s; Control Parameter $\tau \in (0, \frac{2}{3})$; Learning rate η; A randomly initialized ANN with L hidden layers that is parameterized by $\Theta^{(l)}$,$W^{(l)}$ and $\alpha^{(l)}$.

Ensure: $\alpha^{(l)}$, $\Theta^{(l)}$ and $W^{(l)}$

Initialize $\alpha^{(l)} = \frac{1}{L+1}, \forall l = 0, 1, \ldots, L$
for $t = 1, 2, \ldots, T$ **do**
  Receive a triplet constraint $(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-})$
  Transform and retrieve the metric embedding $f^{(l)}(\boldsymbol{x_t})$, $f^{(l)}(\boldsymbol{x_t^+})$ and $f^{(l)}(\boldsymbol{x_t^-})$ from each model E_l.
  Evaluate $\mathcal{L}_{attr}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+})$ and $\mathcal{L}_{rep}^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^-})$ (Eq. 3.12), $\forall l = 0, 1, \ldots, L$.
  Compute $L^{(l)}(\boldsymbol{x_t}, \boldsymbol{x_t^+}, \boldsymbol{x_t^-}), \forall l = 0, 1, \ldots, L$ as per Eq. 3.13.
  Update $\Theta_{t+1}^{(l)}, \forall l = 0, 1, 2, \ldots, L$ as per Eq. 3.21.
  Update $W_{t+1}^{(l)}, \forall l = 0, 1, 2, \ldots, L$ as per Eq. 3.22.
  Update $\alpha_{t+1}^{(l)}, \forall l = 0, 1, 2, \ldots, L$ as per Eq. 3.20.
  $\alpha_{t+1}^{(l)} = \max(\alpha_{t+1}^{(l)}, \frac{s}{L+1}), \forall l = 0, 1, 2, \ldots, L$.
  Normalize $\alpha_{t+1}^{(l)}$, i.e., $\alpha_{t+1}^{(l)} = \frac{\alpha_{t+1}^{(l)}}{\sum_{j=0}^{L} \alpha_{t+1}^{(j)}}, \forall l = 0, 1, 2, \ldots, L$.
end for

(i.e., T is sufficiently large), the proposed algorithm OAHU is guaranteed to converge after learning from these triplets.

3.1.6 Application of OAHU

Since OAHU is indeed an ensemble of metric models, different aggregation methods have to be employed to apply it for various practical applications. In this section, we discuss these aggregation methods in detail. We mainly focus on three common tasks, i.e., classification, similarity comparison, and analog retrieval. Our discussion provides only one possible way to deploy OAHU for these tasks, and other aggregation methods may still exist.

Classification

Let $\mathcal{D} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{n}$ represents the training dataset, where $\boldsymbol{x_i} \in \mathcal{R}^d$ and $y_i \in \{1, 2, \ldots, c\}$ denote a training instance and its associated label respectively. For a test instance $\boldsymbol{x} \in \mathcal{R}^d$, we determine its label $y \in \{1, 2, \ldots, c\}$ as follows:

Step 1: We first find k nearest neighbors of $\boldsymbol{x}$ in every metric model E_l of OAHU, where the distance is measured by Eq. 3.6. Hence, $k(L+1)$ instances in total are found and are candidates similar to $\boldsymbol{x}$. For convenience, suppose $\mathcal{N} = \{\{(\boldsymbol{x_j^{(l)}}, y_j^{(l)})\}_{j=1}^{k}\}_{l=0}^{L} = \{\mathcal{N}_l\}_{l=0}^{L}$ represent these candidates, where $\boldsymbol{x_j^{(l)}}$ is the j^{th} neighbor of $\boldsymbol{x}$ found in the metric model E_l and $y_j^{(l)}$ is its associated label. The similarity score between $\boldsymbol{x}$ and $\boldsymbol{x_j^{(l)}}$ is:

$$S(\boldsymbol{x}, \boldsymbol{x_j^{(l)}}) = \mathrm{e}^{-\frac{D^{(l)}(\boldsymbol{x_j^{(l)}}, \boldsymbol{x}) - d_{min}}{d_{max} - d_{min}}} \cdot \alpha^{(l)} \tag{3.23}$$

Here $d_{min} = \min_{\mathcal{N}_l} D^{(l)}(\boldsymbol{x_j^{(l)}}, \boldsymbol{x})$ and $d_{max} = \max_{\mathcal{N}_l} D^{(l)}(\boldsymbol{x_j^{(l)}}, \boldsymbol{x})$.

Step 2: Based on $S(\boldsymbol{x}, \boldsymbol{x_j^{(l)}})$, we can compute the association score of each class c_i as follows:

$$S(c_i) = \sum_{y_j^{(l)} = c_i} S(\boldsymbol{x}, \boldsymbol{x_j^{(l)}}) \tag{3.24}$$

Step 3: The predicted label y of $\boldsymbol{x}$ is simply the class that maximizes the class association score:

$$y = \arg\max_{c_i} S(c_i) \tag{3.25}$$

Similarity Comparison

In the similarity comparison task, we need to determine whether a given pair $(\boldsymbol{x_1}, \boldsymbol{x_2})$ is similar or not. To do it, we first compute a similarity score between $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$, and then examine if this score is above a user-specified threshold $\mathcal{T} \in (0, 1)$. Specifically,

Step 1: For each metric model E_l, we first compute the distance $D^{(l)}(\boldsymbol{x_1}, \boldsymbol{x_2}) \in [0, 2]$, and use $\frac{1}{2} \cdot D^{(l)}(\boldsymbol{x_1}, \boldsymbol{x_2})$ to measure the similarity between $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$.

Step 2: Hence, the overall similarity score or similarity probability between $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ is:

$$P(\boldsymbol{x_1}, \boldsymbol{x_2}) = \sum_{l=0,\ldots,L} \alpha^{(l)} \cdot p_l \tag{3.26}$$

where

$$p_l = \begin{cases} 1 & D^{(l)}(\boldsymbol{x_1}, \boldsymbol{x_2})/2 < \mathcal{T} \\ 0 & Otherwise \end{cases} \tag{3.27}$$

Step 3: The pair $(\boldsymbol{x_1}, \boldsymbol{x_2})$ is similar if $P(\boldsymbol{x_1}, \boldsymbol{x_2}) \geq 0.5$ and is dissimilar otherwise.

Analogue Retrieval

In the analog retrieval task, we'd like to find k items in a database $\mathcal{D}$ that is most similar to a given query item $\boldsymbol{x}$. Indeed, the only difference between the classification task and the analog retrieval task is that we find similar items in both cases but ignore the labeling problem in the latter case.

Step 1: Similar to the classification task, we first find $k(L+1)$ candidates that are similar to $\boldsymbol{x}$ and denote them as $\mathcal{N} = \{\{\boldsymbol{x_j^{(l)}}\}_{j=1}^{k}\}_{l=0}^{L} = \{\mathcal{N}_l\}_{l=0}^{L}$. The similarity score between $\boldsymbol{x_j^{(l)}}$ and $\boldsymbol{x}$ is computed by:

$$Sim(\boldsymbol{x_j^{(l)}}, \boldsymbol{x}) = \mathrm{e}^{-\frac{D^{(l)}(\boldsymbol{x_j^{(l)}}, \boldsymbol{x}) - d_{min}}{d_{max} - d_{min}}} \cdot \alpha^{(l)} \tag{3.28}$$

where $d_{min} = \min_{\mathcal{N}_l} D^{(l)}(\boldsymbol{x_j^{(l)}}, \boldsymbol{x})$ and $d_{max} = \max_{\mathcal{N}_l} D^{(l)}(\boldsymbol{x_j^{(l)}}, \boldsymbol{x})$.

Step 2: These candidates are then sorted in descending order of their similarity scores and only the top k distinct candidates are returned as the retrieval result.

3.1.7 Time and Space Complexity

In general, the computation of OAHU mainly concentrates on the gradient calculation in the ANN-based metric models. Assume that the maximum time to compute the gradient of each layer in the ANN is a constant C. The time complexity of OAHU is hence $O(nL^2C)$,

where L denotes the number of hidden layers in the ANN and n is the number of input triplets. Apparently, OAHU runs in linear time with respect to the input and is indeed an efficient metric learning framework. On the other hand, the space complexity of OAHU is $O\big(d \cdot S_{emb} + L \cdot S_{hidden}(d + S_{emb}) + \frac{L(L-1)}{2} S_{hidden}^2\big)$. Here d is the dimensionality of input data, S_{hidden} is the maximum number of neurons of the hidden layers, and S_{emb} denotes the metric embedding size.

3.2 Evaluation

We evaluate the proposed framework OAHU on three typical tasks, including image classification (i.e., *classification*), face verification (i.e., *similarity comparison*) and image retrieval (i.e., *analogue retrieval*).

3.2.1 Baselines

To verify the effectiveness of our method, we compare OAHU with several state-of-the-art online metric learning algorithms.

- **LEGO** (Jain et al., 2008b) (pairwise): an online metric learning algorithm that learns a Mahalanobis metric based on LogDet regularization and gradient descent.
- **RDML** (Jin et al., 2009b) (pairwise): an online algorithm for regularized distance metric learning that learns a Mahalanobis metric with a regret bound.
- **OASIS** (Chechik et al., 2010b) (triplet): an online algorithm for scalable image similarity learning that learns a bilinear similarity measure over sparse representations.
- **OPML** (Li et al., 2018b) (triplet): a one-pass closed-form solution for online metric learning that learns a Mahalanobis metric with a low computational cost.

3.2.2 Implementation

The proposed OAHU framework is implemented via Python 3.6.2 and PyTorch 0.4.0. For most baseline methods except LEGO and RDML, we adopt the official code released by corresponding authors. Since the fully functional codes of LEGO and RDML are unavailable, we implement these two approaches based on the authors' description in the published paper (Jain et al., 2008b; Jin et al., 2009b). We vary the hyper-parameter setting of competing methods including OAHU with different tasks and will discuss it later for each task.

3.2.3 Image Classification

Dataset and Classifier

In the image classification task, we adopt three public available real-world benchmark image datasets for evaluation, including

- **FASHION-MNIST**[3] (Xiao et al., 2017a): FASHION-MNIST is a dataset of Zalando's article images. It consists of a training set of $60,000$ examples and a test set of $10,000$ examples, where each example is a 28×28 grayscale image associated with a label from 10 classes.

- **EMNIST**[4] (Cohen et al., 2017a): The EMNIST dataset is a collection of handwritten character digits, each of which is converted into a 28×28 grayscale image.

- **CIFAR-10**[5] (Krizhevsky, 2009): The CIFAR-10 dataset consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. To be consistent with Fashion-MNIST and EMNIST datasets, we convert images in CIFAR-10 into gray-scale images via the OpenCV API (Bradski, 2000).

[3]https://github.com/zalandoresearch/fashion-mnist

[4]https://www.nist.gov/itl/iad/image-group/emnist-dataset

[5]https://www.cs.toronto.edu/~kriz/cifar.html

Table 3.1: Description of Datasets

Dataset	Task	# instances	# classes	# features
FASHION-MNIST	Image Classification	70,000	10	784
EMNIST	Image Classification	131,600	47	784
CIFAR-10	Image Classification	60,000	10	1,024
LFW	Face Verification	13,233	5,749	73
CARS-196	Image Retrieval	16,185	196	4,096
CIFAR-100	Image Retrieval	60,000	100	4,096

Table 3.2: Classification performance of competing methods on benchmark image datasets. $\circ/\bullet$ indicates that **OAHU** performs statistically worse or better (0.05 significance level) than the corresponding methods. Both mean and standard deviation of error rates, constraint utilization $\mathcal{U}$ and macro F_1 scores are reported. 0.00 denotes a value less than 0.005.

Dataset		LEGO	RDML	OASIS	OPML	OAHU
FASHION-MNIST	*Error Rate*	0.35±0.01 •	0.22±0.01 •	0.21±0.02 •	0.23±0.00 •	**0.18±0.00**
	F_1	0.64±0.00	0.78±0.00	0.79±0.01	0.78±0.00	**0.81±0.00**
	$\mathcal{U}$	0.58±0.00	0.47±0.01	0.14±0.00	0.30±0.01	**1.00±0.00**
EMNIST	*Error Rate*	0.84±0.00 •	0.64±0.03 •	0.41±0.01 •	0.43±0.01 •	**0.35±0.01**
	F_1	0.14±0.02	0.32±0.05	0.59±0.00	0.51±0.02	**0.64±0.01**
	$\mathcal{U}$	0.66±0.01	0.50±0.00	0.24±0.00	0.45±0.00	**1.00±0.00**
CIFAR-10	*Error Rate*	0.88±0.00 •	0.76±0.01 •	0.79±0.02 •	0.83±0.01 •	**0.68±0.00**
	F_1	0.05±0.02	0.24±0.01	0.20±0.03	0.16±0.00	**0.31±0.00**
	$\mathcal{U}$	0.59±0.00	0.50±0.01	0.47±0.00	0.48±0.04	**1.00±0.00**
win/tie/loss		**3/0/0**	**3/0/0**	**3/0/0**	**3/0/0**	-

A K nearest-neighbor classifier is employed in this task, which is consistent with the aggregation methods discussed in Section 3.1.6. The details of these datasets are listed in Table 3.1.

Experiment Setup and Constraint Generation

In our experiments, we first uniformly shuffle the data instances in each dataset and then divide them into a development set and a test set, where the split ratio is 1 : 1. On the other hand, to find proper values of hyper-parameters for the baselines, we first initialize these hyper-parameters to the values reported by the authors and then fine-tune them via 10-fold cross-validation on the development set. Since OPML has introduced a one-pass triplet construction process to generate input constraints, we simply provide those data in

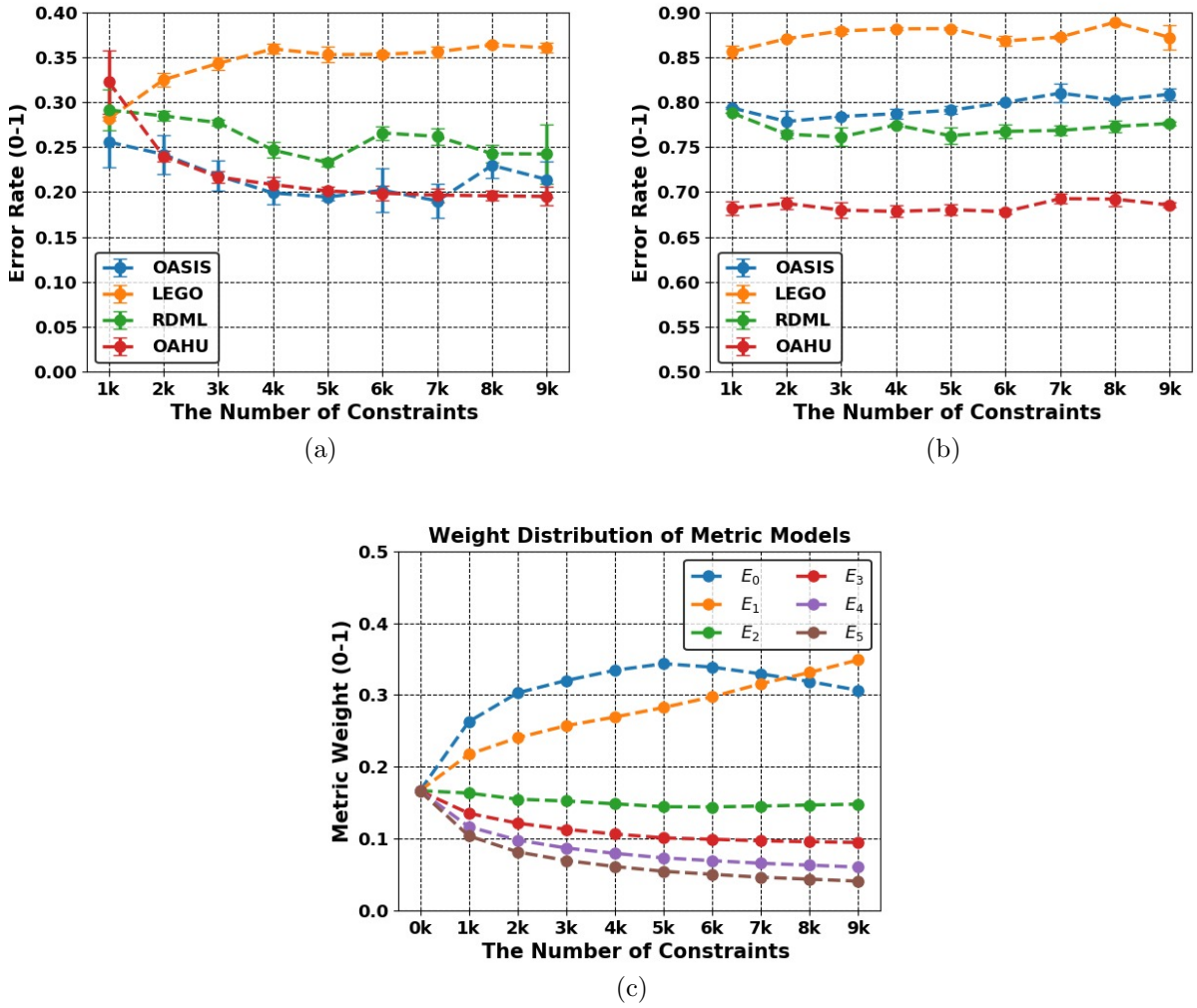


Figure 3.4: (a) / (b) Error rates of competing methods with increasing number of constraints on the FASHION-MNIST / CIFAR-10 dataset. (c) The evolvement of metric weight distribution in OAHU with an increasing number of constraints on FASHION-MNIST dataset.

the development set as input to it. All the other approaches including OAHU adopt the same procedure to generate input constraints. Specifically, $5,000$ pairwise or triplet seed constraints are first randomly sampled from the training data in the development set, and then $5,000$ more constraints are constructed by taking transitive closure over these seeds. Thus, $10,000$ constraints in total are generated, which is consistent with LEGO (Jain et al., 2008b) and OPML (Li et al., 2018b). We set $\beta = 0.99$, $L = 5$, $S_{hidden} = 100$, $s = 0.1$,

$\eta = 0.3$ and $S_{emb} = 50$ as default in OAHU, and find the proper value of τ via 10-fold cross-validation on the development set. In addition, the experiments are repeated 10 times with independent random shuffle processes and the average results are reported to reduce any potential bias due to random partitions.

Evaluation Metric

We accept three evaluation metrics that are widely accepted in previous studies such as LEGO (Jain et al., 2008b) and OPML (Li et al., 2018b):

- *error rate*: the fraction of test examples that are incorrectly classified.
- *macro* F_1[6]: the unweighted mean of F_1 scores across all classes. It does not take class imbalance into account.
- $\mathcal{U}$: constraint utilization rate, i.e., the fraction of input constraints that actually contribute to the learning.

We also perform a statistical significance test (student's t-test) by computing the p-values and report the win/tie/loss statistics based on them.

Analysis

We compare the classification performance of competing methods including OAHU on the three benchmark datasets, and the results are shown in Table 3.2. Because LEGO and RDML attempt to learn a linear Mahalanobis metric from pairwise constraints, they only consider either the similarity or dissimilarity relation at each iteration during the learning process and hence perform poorly on complex image datasets such as EMNIST (47 classes). On the other hand, OASIS outperforms OPML since its bilinear similarity metric does not require

[6]http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

the covariance matrix, i.e., A in $d(\boldsymbol{x_1}, \boldsymbol{x_2}) = \boldsymbol{x_1}^T A \boldsymbol{x_2}$, to be positive and symmetric, leading to a bigger hypothesis space (Chechik et al., 2010b). In general, the overall classification performance of OAHU is best among all competing methods, because it provides not only much lower error rates but also significantly higher F_1 scores compared to baselines.

We also vary the input constraint numbers to study the effect of training set size on the learning of a metric for all competing methods except OPML. It is done by uniformly sampling the desired number of constraints from the $10,000$ generated constraints discussed in Section 3.2.3. Note that manually varying the input triplet number for OPML is impossible, since it relies on a one-pass triplet construction process to internally determine the number of triplets needed for training. Figure 3.4a and Figure 3.4b show the comparison results on FASHION-MNIST and CIFAR-10 dataset respectively. The large error rate variance of both OASIS and RDML indicates that their performance is highly correlated with the quality of input constraints. In addition, all baselines perform unstably and sometimes even worse as more constraints are observed in the sequence. It is because that these methods rely on those "hard" constraints for model updating and incorrectly ignore many failure cases (see Figure 3.2) in the learning process. In contrast, the proposed framework OAHU outperforms all baselines by providing the best classification performance. It also performs most stably among the competing methods with minimal performance variance, which indicates its robustness to the quality of input constraints.

OAHU is better mainly because it not only significantly improves the constraint utilization rate $\mathcal{U}$ by learning from every input constraint, but also automatically adapts its model complexity based on input constraints when necessary (see Figure 3.4c).

3.2.4 Face Verification

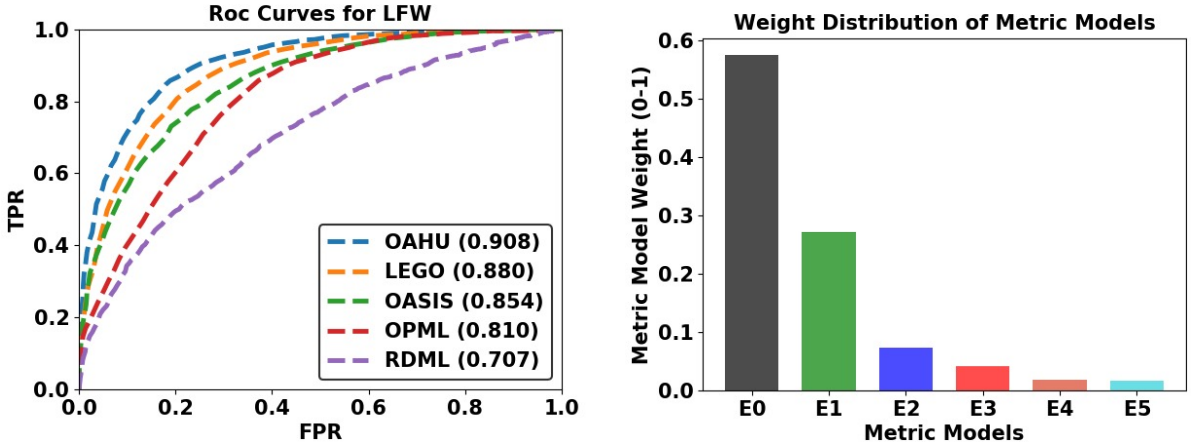
Dataset and Experiment Setup

In the face verification task, we choose the Labeled Faces in the Wild Database (LFW) (Learned-Miller, 2014) as the evaluation dataset. This dataset supports two views. View 1 contains a training and a test set and is used for development purposes. The partitions in View 1 is generated randomly and independently for 10-fold cross-validation. View 2 is taken as a benchmark for comparison, which is 10-fold cross-validation set containing pairwise images. Given a pair of face images, our goal is to determine whether these two images represent the same person. Because OASIS, OPML, and OAHU are triplet-based methods, we cannot directly evaluate their performance on the LFW dataset. Therefore, following OPML (Li et al., 2018b), we adopt the image unrestricted configuration in our experiment for a fair comparison. In this configuration, we directly use the label information, i.e., the actual names of the people in the training data, and formulate as many pairs or triplets as one desires. We use View 1 for hyper-parameter tuning and evaluate the performance of all competing models on each fold of View 2, which contains 300 intra-person and 300 inter-person pairs. The LFW attribute features[7] (73 dimension) provided by Kumar et al. (Kumar et al., 2009) are adopted to represent face images so that the experiment can be easily reproduced. They are "high-level" features describing nameable attributes such as race, age, etc., of a face image. The details of the LFW dataset are described in Table 3.1.

Constraint Generation and Evaluation Metric

For constraint generation, we follow the process described in Section 3.2.3 but add an additional step: if any pair in the generated constraint appears in View 1 or View 2 of the LFW dataset, we simply discard it and re-formulate a new one. This procedure is repeated until

[7]http://www.cs.columbia.edu/CAVE/databases/pubfig/download/lfw_attributes.txt



(a) (b)

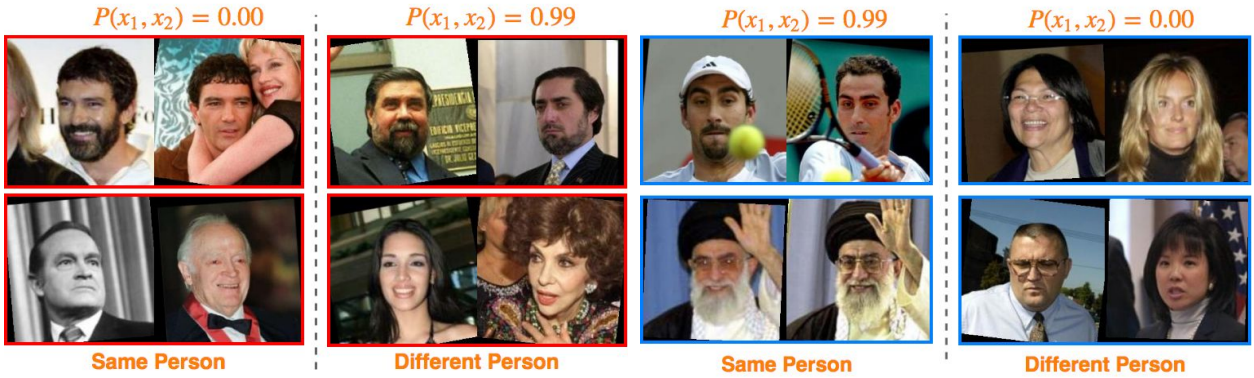
Figure 3.5: (a) ROC curves of competing methods on the LFW dataset. The number in the bracket denotes the AUC score of the corresponding method. (b) The metric weight distribution in OAHU.

the sampled constraint does not contain any pair that overlaps with these two sets. In total, $10,000$ pairs or triplets are generated for training in our experiment.

When it comes to inference, given a test image pair, we first compute the similarity between the two images using the learned metric and then compare it with a user-specified threshold. By varying this threshold, we plot the Receiver Operating Characteristic (ROC) curve and calculate the Area under Curve (AUC) score as the evaluation metric.

Analysis

Figure 3.5a shows the ROC curves and the corresponding AUC scores of all competing methods. Our proposed framework OAHU provides the best AUC score and performs significantly better than the baselines. The superiority of OAHU mainly comes from its capability of automatically adapting its complexity to improve the model expressiveness, which is demonstrated in Figure 3.5b where the metric weights of non-linear components in OAHU are increased. It helps to separate instances that are difficult to distinguish with simple linear models. Figure 3.6 presents some successful and failure example comparisons



(a) Failure Cases (b) Successful Cases

Figure 3.6: The successful and failure face verification test examples with $\mathcal{T} = 0.55$. $P(\boldsymbol{x_1}, \boldsymbol{x_2})$ is expected to be close to 1 if two images belong the same person and 0 otherwise.

in the test set. In this figure, the comparison probability $P(\boldsymbol{x_1}, \boldsymbol{x_2})$ is computed based on Eq. 3.26, and is expected to approach 1 if both images represent the same person and 0 otherwise. Unfortunately, similar to existing solutions, although OAHU performs better than the baselines, it may still fail in cases where sever concept drift, e.g., aging exists.

3.2.5 Image Retrieval

Dataset and Experiment Setup

For image retrieval, we conduct experiments on two real-world benchmark datasets, i.e., CARS-196 and CIFAR-100.

- **CARS-196**[8] **(Krause et al., 2013)**: This dataset contains 16,185 images of 196 classes of cars. The data is divided into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

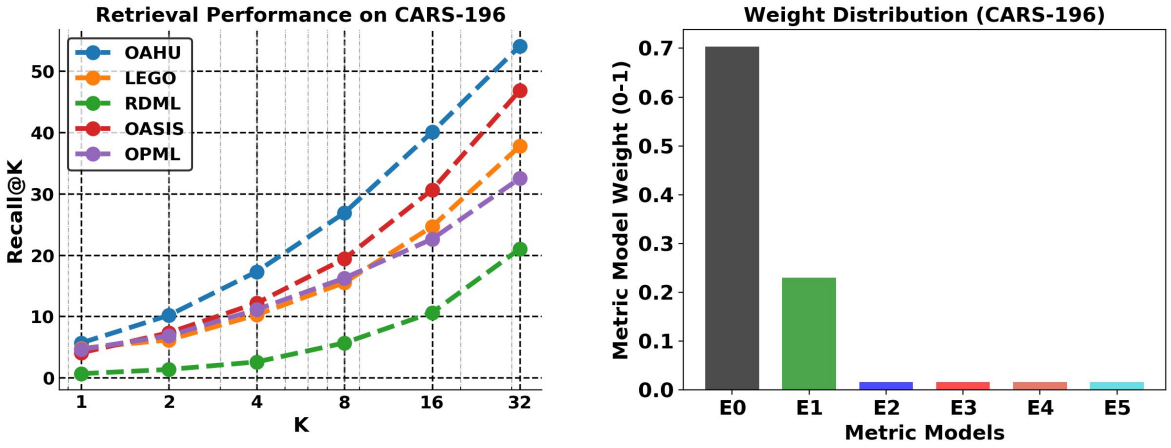
[8]https://ai.stanford.edu/~jkrause/cars/car_dataset.html

- **CIFAR-100**[9] **(Krizhevsky, 2009)**: This dataset consists of $60,000$ 32×32 colour images in 100 classes with 600 images per class. Similar to CARS-196, we randomly split CIFAR-100 into $30,000$ training images and $30,000$ test images by equally dividing images of each class.

A VGG-19 model pre-trained on the ImageNet (Russakovsky et al., 2015) is adopted as the image feature extractor. Specifically, for each image, we feed it into a VGG-19 model and take the output of the last convolutional layer as the input embedding to the competing frameworks. The details of these 2 datasets are described in Table 3.1. The development set of each dataset is formed by including all its training examples. The hyper-parameters of baselines are set based on values suggested by the authors and fine-tuned via 10-fold cross-validation on the development set. All the hyper-parameters except τ in OAHU are set same as Section 3.2.3. The value of τ is found via 10-fold cross-validation on the development set.

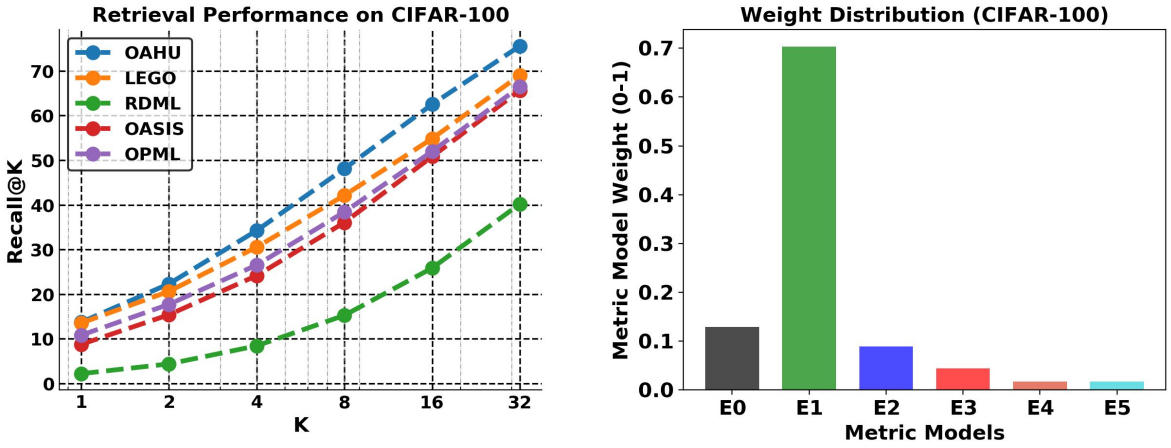
Constraint Generation and Evaluation Metric

We follow the same procedure as described in Section 3.2.3 to generate the training constraints. However, in this experiment, we sample more constraints to better cover the massive amount of classes included in these datasets. Specifically, we generate $50,000$ constraints by taking transitive closure over $25,000$ sampled seeds. The Recall@K metric (Jégou et al., 2011) is adopted to evaluate the model performance. To compute Recall@K, we first retrieve K images in the test set that are most similar to a query image and receive a score of 1 if an image of the same class is among these K images and 0 otherwise. Recall@K simply averages this score over all images.



(a) (b)

Figure 3.7: (a) The Recall@K scores of competing methods on the test set of CARS-196. (b) The corresponding metric weight distribution in OAHU.



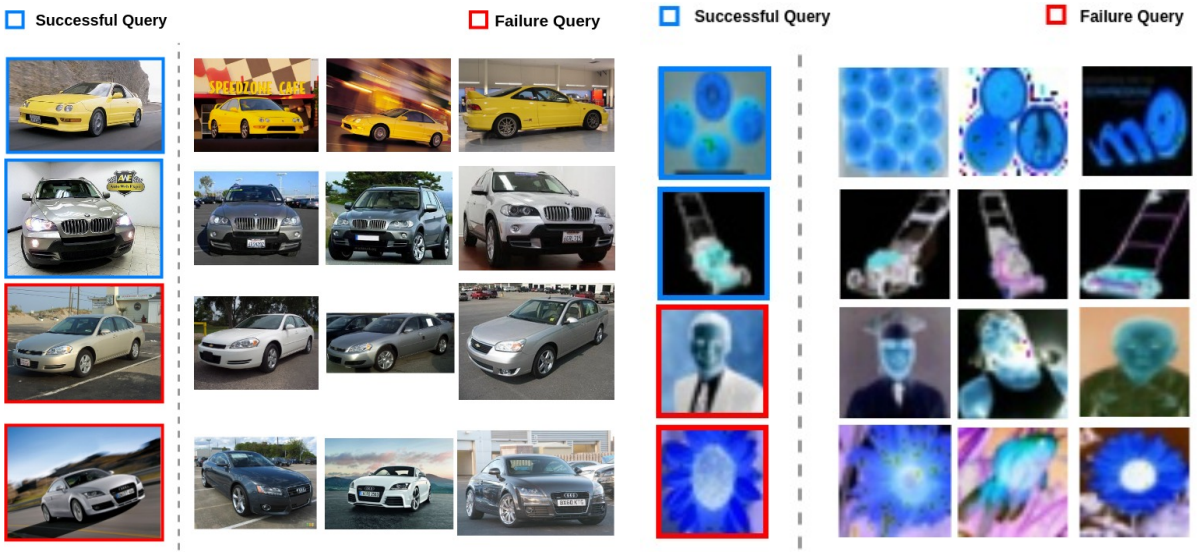
(a) (b)

Figure 3.8: (a) The Recall@K scores of competing methods on the test set of CIFAR-100. (b) The corresponding metric weight distribution in OAHU.

Analysis

Figure 3.7a and Figure 3.8a present the Recall@K scores on the test set of CARS-196 and CIFAR-100 datasets. The proposed framework OAHU provides the highest Recall@K scores

[9]https://www.cs.toronto.edu/~kriz/cifar.html



(a) CARS-196 (b) CIFAR-100

Figure 3.9: The successful and failure query examples on the CARS-196 and CIFAR-100 datasets using the learned embedding. Images in the first column are query images and the rest are the three most similar images. Best viewed on a monitor zoomed-in.

among all the competing methods and significantly outperforms the baselines. Moreover, the performance gap between OAHU and most baselines increase rapidly as more neighbors are considered. In contrast to the baselines that are pure linear models, OAHU improves the model expressiveness by incorporating metrics of various degree of nonlinearity, which is demonstrated by the metric weight distribution of OAHU shown in Figure 3.7b (CARS-196) and Figure 3.8b (CIAR-100). In these figures, E_1, E_2, E_3, E_4 and E_5 are non-linear models with increasing complexities while E_0 is a simple linear model. Some examples of successful and failure queries in both datasets are presented in Figure 3.9. Despite the huge change in the viewpoint, configuration, and illumination, our model can successfully retrieve examples from the same class while most retrieval failures come from the subtle visual difference among images of different classes.

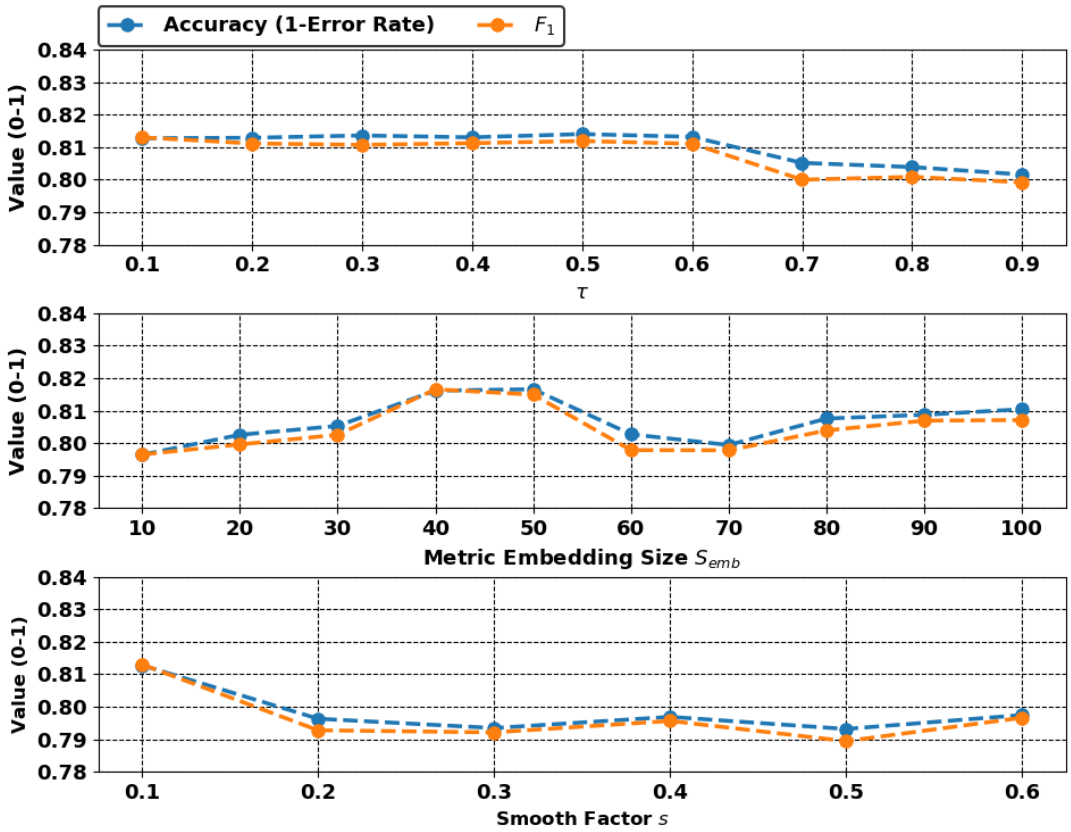


Figure 3.10: Parameter sensitivity of OAHU on FASHION-MNIST dataset as an example.

3.2.6 Sensitivity of Parameters

The three main parameters in OAHU are the control parameter τ, the metric embedding size S_{emb}, and the smooth factor s. We vary these parameters to study their effect on classification performance. Figure 4.8 shows the result of FASHION-MNIST dataset as an example. It is observed that both classification accuracy and F_1 score significantly drop if $\tau > 0.6$ and remains almost same when $\tau \leq 0.6$, which verifies Theorem 1. On the other hand, OAHU gives the best classification performance when $S_{emb} = 50$. Noisy information will be included if we increase the embedding size, which degrades the model performance. Similarly, with a high smooth factor s, the hedge model weight adaptation mechanism is

weakened leading to poor classification performance. Therefore, $\tau = 0.1$, $S_{emb} = 50$ and $s = 0.1$ are suggested as default values in OAHU.

3.3 Discussion

This chapter introduces OAHU that is designed to address the "Online Adaptive Metric Learning" open challenge. It improves the model expressiveness by automatically adapting the model complexity based on input constraints and achieves a full constraint utilization rate. Due to the Adaptive Bound Triplet Loss (ABTL) adopted in OAHU, our framework is more robust to the quality of input constraints compared to baselines. However, similar to existing solutions, OAHU may still fail in cases where severe concept drift exists. We leave the extension of OAHU to such scenarios for future work.

CHAPTER 4

SETCONV: A NEW APPROACH FOR LEARNING FROM IMBALANCED DATA [1]

4.1 Approach

In this chapter, we present the details of our SetConv framework introduced in Chapter 1 and explain how we utilize similarity relations among data to enhance the classification performance on imbalanced data distributions.

4.1.1 Overview

Our goal is to design a classification model that is able to overcome the bias towards the majority classes when learning from imbalanced class distributions. We first consider a binary classification problem for simplicity and then extend the model to the multi-class scenarios. The architecture of our model is illustrated in Fig. 4.1a, which consists of a SetConv layer and a classification layer. During the training process, at each iteration, an *episode* is sampled from the training data, which is comprised of a support set and a query set. The support set is formed by a group of examples where the class imbalance ratio of training data is preserved, and the query set contains only a single example from each class. The SetConv layer customizes the feature extraction process for each example in the episode by estimating a set of weights based on the relation of this example to the minority class and using learned weights to extract its features. A representative is hence produced by SetConv for each class in the support set. We then compare each example in the query set with these class representatives in the classification layer to determine its

[1] This chapter contains material previously published as: Yang Gao, Yi-Fan Li, Yu Lin, Latifur Khan. "SetConv: A New Approach for Learning from Imbalanced Data", In *proceedings of The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1284-1294. 2020. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

label and evaluate the classification loss. This training procedure is referred to as *episodic training*.

The proposed episodic training has several benefits:

- First, with episodic training, the SetConv layer is encouraged to extract discriminative features from data even when the class distribution is highly imbalanced.

- The randomly sampled episodes have a significantly different configuration of support and query sets from iteration to iteration. Therefore, the SetConv layer is required to ignore noisy information in the examples and capture the underlying class concepts that are common among different episodes.

- The episodic training has no dependency on unknown prior knowledge. The only prior knowledge required is the class imbalance ratio that can be easily obtained from training data.

As shown in Fig. 4.1b, a post-training step has to be performed before applying our model for inference. In this step, we randomly sample a large subset S_{post} from the training data and feed it into the SetConv layer to extract a representative for each class. Note that we only forward the data instances through the network in this process and do not update the model. This operation is executable because our model has learned to capture the class concepts that are insensitive to the episode configuration. The inference procedure of our model is presented in Fig. 4.1c. For each query example, we feed it into the SetConv layer to compute the features and compare the resulting embedding with those class representatives obtained in the post-training step. We take the label of the representative that is most similar to the query as the predicted label.

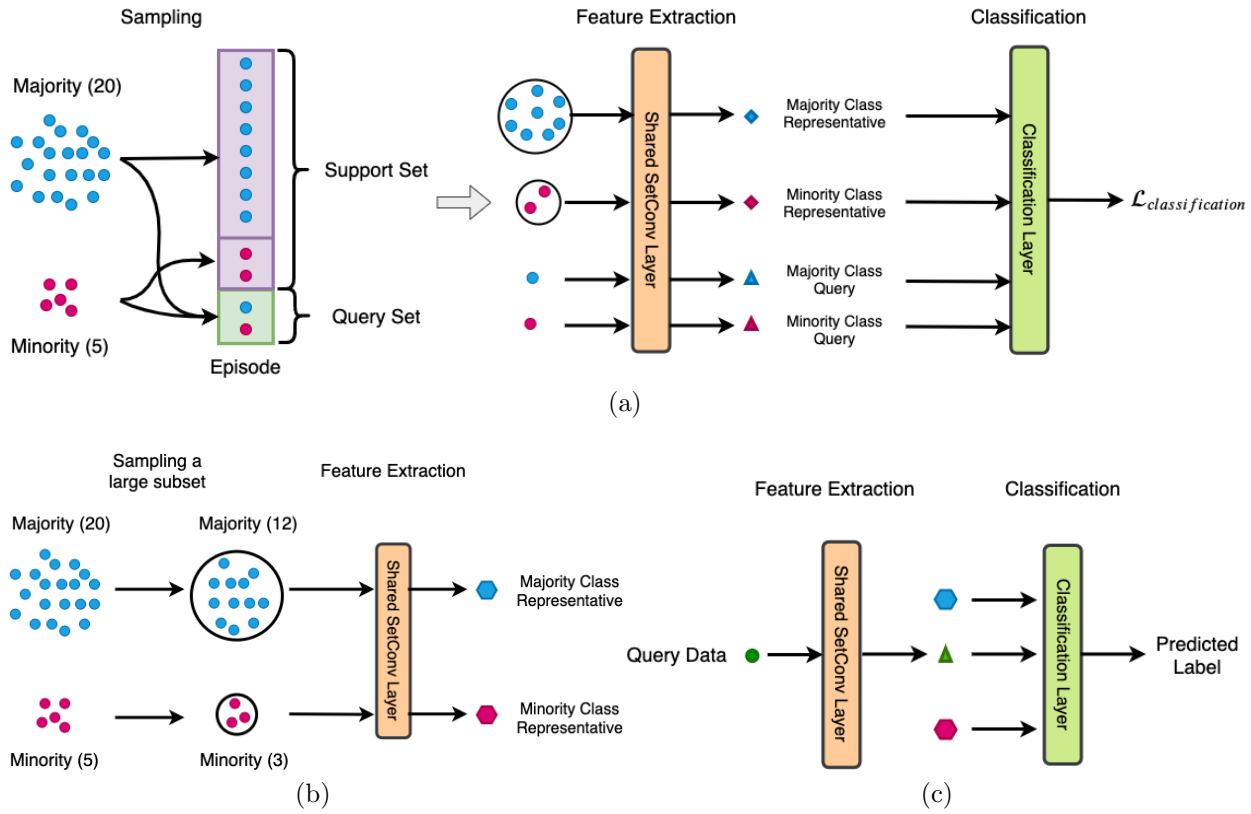


Figure 4.1: Overview of the proposed approach. (a) The training procedure of SetConv. At each iteration, SetConv is fed with an episode to evaluate the classification loss for a model update. Each episode consists of a support set and a query set. The support set is formed by a group of samples where the imbalance ratio is preserved. The query set contains only one sample from each class. (b) The post-training step of SetConv is performed only once after the main training procedure. In this step, we extract a representative for each class from the training data and will later use them for inference. Here we only perform inference using the trained model and do not update it. (c) The inference procedure of SetConv. Each query data is compared with every class representative to determine its label.

4.1.2 SetConv Layer

The minority class instances often require intensive attention of machine learning models, since their rareness may carry important and useful knowledge (He and Garcia, 2009; Sun et al., 2007; Chen and Shyu, 2011). A typical example is incident detection on social media, where we are more interested in tweets related to violence such as shooting than those tweets about entertainment.

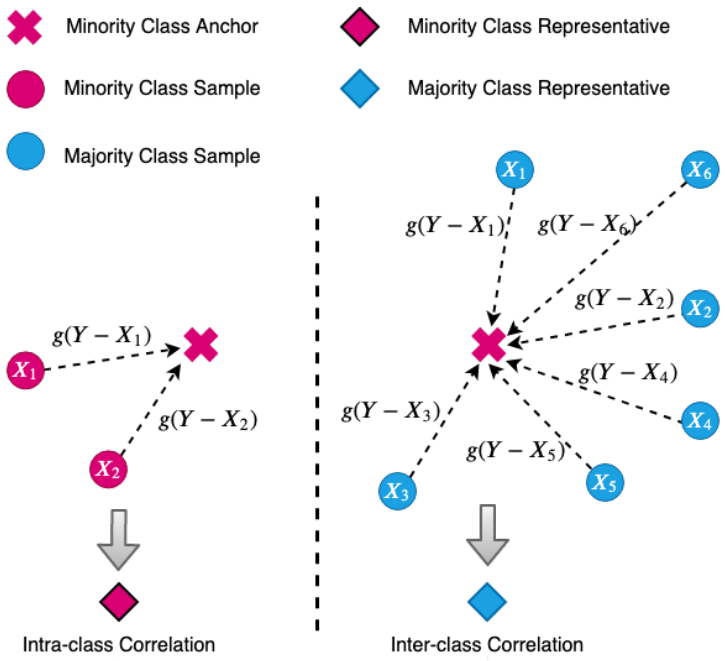


Figure 4.2: Relations between the input samples and a pre-selected minority class anchor are used by SetConv to estimate both intra-class correlations and inter-class correlations.

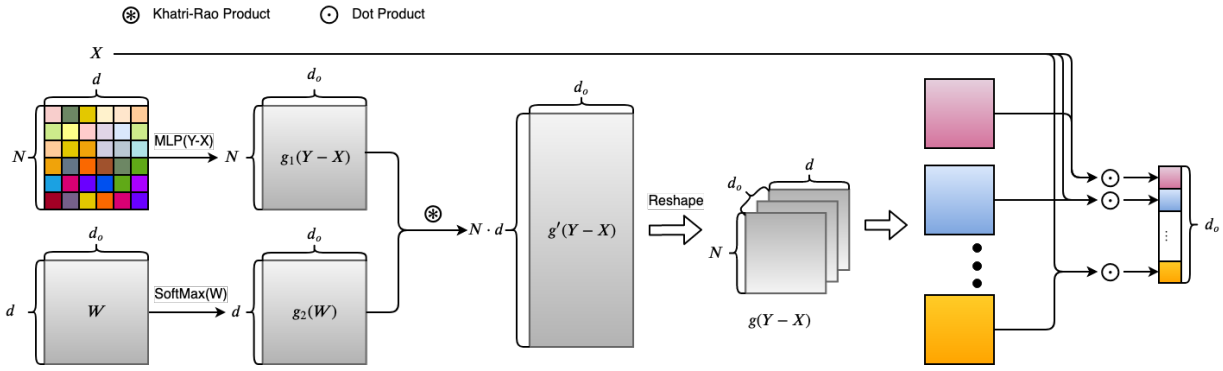


Figure 4.3: The computation graph of the SetConv layer. Here Y is a minority class anchor. $W \in \mathcal{R}^{d \times d_o}$ is a weight matrix to learn that records the correlation between the input and output variables. Specifically, the i_{th} column of $g_2(W)$ gives the weight distribution over input features for the i_{th} output feature. It is indeed a feature-level attention matrix. In addition, we estimate another data-sensitive weight matrix $g_1(Y - X)$ from the input data. The final convolution weight tensor is simply the Khatri-Rao product of $g_1(Y - X)$ and $g_2(W)$.

This prior knowledge encourages us to design the SetConv layer in a way such that the *feature extraction process focuses on the minority class*. Specifically, the SetConv layer estimates its convolution kernel weights based on the relation between a pre-selected minority

class anchor that is freely determined by the user and the input examples. Apparently, as shown in Figure 4.2, if the input examples come from the minority class, the SetConv layer is considering the intra-class correlation. On the other hand, if the input examples belong to the majority class, the SetConv layer is considering the inter-class correlation. In this paper, we choose the *average-pooling* of the minority class examples as the anchor for simplicity, which is also executable due to the limited amount of minority class examples in real-world applications.

Let $\mathcal{E}_t = \{\mathcal{S}_t, \mathcal{Q}_t\}$ denote the episode sampled at iteration t, where $\mathcal{S}_t = \big(X_{maj} \in \mathcal{R}^{N_1 \times d}, X_{min} \in \mathcal{R}^{N_2 \times d}\big)$ is the support set and $\mathcal{Q}_t = \big(q_{maj} \in \mathcal{R}^{1 \times d}, q_{min} \in \mathcal{R}^{1 \times d}\big)$ is the query set. To simplify the illustration, we abstract X_{maj}, X_{min}, q_{maj} and q_{min} as a sample set of size N_1, N_2, 1 and 1 respectively. Mathematically, we represent this sample set as $X \in \mathcal{R}^{N \times d}, N \in \{N_1, N_2, 1\}$.

Recall that the standard discrete convolution is:

$$h[n] = (f \star g)[n] = \sum_{m=-M}^{m=M} f[m]g[n-m] \tag{4.1}$$

where f denotes the feature map and g represents the convolution kernel weights.

In our case, the SetConv operation is defined as:

$$\begin{aligned} h[Y] &= \frac{1}{N} \sum_{i=1}^{N} X_i \cdot g(Y - X_i) \\ &= \frac{1}{N} \Big(X \circ g(Y - X) \Big) \end{aligned} \tag{4.2}$$

Here $Y \in \mathcal{R}^{1 \times d}$, $g(Y - X) \in \mathcal{R}^{N \times d \times d_o}$ and $h[Y] \in \mathcal{R}^{1 \times d_o}$ denote the d-dimensional minority class anchor, the convolution kernel weights with d_o channels and the d_o-dimensional output embedding respectively. $\circ$ represents the channel-wise dot product of tensors, i.e., for every channel $i \in \{1, 2, \ldots, d_o\}$, $h[Y][i] = \frac{1}{N} \sum_j \sum_k X[j,k] \cdot g(Y - X)[j,k,i]$

Since the size of $g(Y-X)$ is proportional to Nd, i.e., the product of the input example size N and the input dimension d, learning $g(Y-X)$ directly for large-scale high-dimensional data is impractical. Therefore, we propose to approximate $g(Y-X)$ via an efficient method to reduce the memory and computational cost. Specifically, we stack samples in X to form a giant dummy example $X' = Concat(X) \in \mathcal{R}^{1\times Nd}$. Eq. 4.2 becomes

$$h[Y] = \frac{1}{N}\Big(X' \cdot g'(Y-X)\Big) \tag{4.3}$$

where $g'(Y-X) \in \mathcal{R}^{Nd\times d_o}$ is the transformed convolution kernel weights. Then, we approximate $g'(Y-X)$ as the *Khatri-Rao* product[2] (Rabanser et al., 2017) of two individual components:

$$\begin{aligned} g'(Y-X) &= g_1(Y-X) \circledast g_2(W) \\ &= \mathrm{MLP}(Y-X;\theta) \circledast \mathrm{SoftMax}(W,0) \end{aligned} \tag{4.4}$$

Here $W \in \mathcal{R}^{d\times d_o}$ is a weight matrix encoding the correlation between input and output variables. $g_2(W)$ is hence a *feature-level attention* matrix and its i_{th} column gives the weight distribution over input features for the i_{th} output feature. On the other hand, $g_1(Y-X)$ estimates weights from input data via a multi-layer perceptron and is hence data-sensitive. It takes the relation between input data and the minority class anchor into account and helps the model customize the feature extraction process for each input example. The detailed computation graph of the SetConv layer is illustrated in Figure 4.3.

4.1.3 Permutation Invariant Property

An important property of the SetConv layer is ***permutation-invariant***, i.e., it is insensitive to the order of input samples. As long as the input samples are the same, no matter in which order they are sent to the model, the SetConv layer always produces the same

[2]https://en.wikipedia.org/wiki/Kronecker_product

feature representation. Mathematically, let π denote an arbitrary permutation matrix, we have $\textsc{SetConv}(\pi X) = \textsc{SetConv}(X)$.

$$
\begin{aligned}
&\textsc{SetConv}(\pi X) \\
&= \frac{1}{N}\Big(Concat(\pi X) \cdot \big[g_1(Y - \pi X) \circledast g_2(W)\big]\Big) \\
&= \frac{1}{N} \cdot \Big(Concat(X)E(\pi) \cdot \big[(\pi \cdot g_1(Y - X)) \circledast g_2(W)\big]\Big) \\
&= \frac{1}{N}\Big(X'E(\pi) \cdot E[\pi]^T \big[g_1(Y - X) \circledast g_2(W)\big]\Big) \\
&= \frac{1}{N}\Big(X' \cdot I \cdot \big[g_1(Y - X) \circledast g_2(W)\big]\Big) \\
&= \textsc{SetConv}(X)
\end{aligned}
\tag{4.5}
$$

Here $Concat$ is the concatenation operation which transforms a N-by-d matrix into a Nd-dimensional row vector. $E(\pi)$ is the expansion operator for the permutation matrix π. For example, considering a 2-by-2 permutation matrix,

$$\pi = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$E(\pi)$ is given by:

$$E(\pi) = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

For a toy example, $Concat(\pi X)$ is computed as below:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a & a \\ b & b \end{bmatrix} = \begin{bmatrix} b & b \\ a & a \end{bmatrix} \rightarrow \begin{bmatrix} b & b & a & a \end{bmatrix}$$

On the other hand, $Concat(X)E(\pi)$ is given by

$$
\begin{aligned}
& Concat(X)E(\pi) \\
&= \begin{bmatrix} a & a & b & b \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} b & b & a & a \end{bmatrix} \\
&= Concat(\pi X)
\end{aligned}
$$

4.1.4 Classification

Assume v^s_{maj}, v^s_{min}, v^q_{maj} and v^q_{min} denote the feature embedding produced by the SetConv layer for X_{maj}, X_{min}, q_{maj} and q_{min} in an episode respectively. The probability that v^q_{maj} or v^q_{min} is predicted as the majority class is:

$$P(c = 0|x) = \frac{\exp(x \odot v^s_{maj})}{\exp(x \odot v^s_{maj}) + \exp(x \odot v^s_{min})} \tag{4.6}$$

where $x \in \{v^q_{maj}, v^q_{min}\}$ and $\odot$ is the dot product operation.

Similarly, the probability that v^q_{maj} or v^q_{min} is predicted as the minority class is:

$$P(c = 1|x) = \frac{\exp(x \odot v^s_{min})}{\exp(x \odot v^s_{maj}) + \exp(x \odot v^s_{min})} \tag{4.7}$$

where $x \in \{v^q_{maj}, v^q_{min}\}$.

The widely-accepted cross-entropy loss is used to estimate the classification loss and the Adam optimizer is adopted for a model update.

4.1.5 Extension to Multi-Class Scenario

Extending SetConv to the imbalanced multi-class classification scenario is straightforward. We simply convert a multi-class classification problem into multiple binary classification

problems. In other words, a binary one-vs-rest classifier is created for each of the N classes, and the smaller one of the two classes is picked as the anchor class. Given a test instance x, the posterior probability of predicting it as class c is given by

$$P(y = c|x) = \frac{\exp(x \odot v^s_{y=c})}{\exp(x \odot v^s_{y\neq c}) + \exp(x \odot v^s_{y=c})} \tag{4.8}$$

we choose the class that maximizes the posterior probability as the predicted label of x, i.e., $\hat{y} = \text{argmax}_c P(y = c|x)$.

4.2 Evaluation

To verify the effectiveness of our method, we evaluate the proposed SetConv framework on two common tasks, including incident detection on social media and sentiment classification.

4.2.1 Benchmark Dataset

Incident Detection on Social Media

In this task, a real-world benchmark dataset *Incident-Related Tweet*[3] (Schulz et al., 2017) (**IRT**) is adopted for evaluation. This dataset contains 22,170 tweets collected from 10 different cities, which allows us to evaluate the competing methods against geographical variations. Another reason to choose this dataset is that it supports both binary and multi-class classifications. In binary classification, each tweet is tagged as either "incident-related" or "not incident-related". On the other hand, in multi-class classification, each tweet is labeled as one of the four categories including "crash", "fire", "shooting", and "not incident-related". Table 4.1 provides the details of this dataset.

[3]http://www.doc.gold.ac.uk/%7Ecguck001/IncidentTweets/

Table 4.1: Class distribution in the IRT dataset.

	Two Classes		Four Classes			
	Yes	No	Crash	Fire	Shooting	No
Boston (USA)	604	2216	347	188	28	2257
Sydney (AUS)	852	1991	587	189	39	2208
Brisbane (AUS)	689	1898	497	164	12	1915
Chicago (USA)	214	1270	129	81	4	1270
Dublin (IRE)	199	2616	131	33	21	2630
London (UK)	552	2444	283	95	29	2475
Memphis (USA)	361	721	23	30	27	721
NYC (USA)	413	1446	129	239	45	1446
SF (USA)	304	1176	161	82	61	1176
Seattle (USA)	800	1404	204	153	139	390

Table 4.2: Class distribution in Amazon Review and SemiEval Datasets.

Dataset	Negative	Positive	IR
Amazon-Books	72039	7389	9.75
Amazon-Electronics	13560	1908	7.11
Amazon-Movies	12896	2066	6.24
SemiEval	39123	7273	5.38

Sentiment Classification

For sentiment classification, we evaluate our approach on two widely-accepted real-world benchmark datasets, i.e., *Amazon Review*[4] (He and McAuley, 2016) and *SemiEval*[5] (Rosenthal et al., 2017). Similar to the settings in MSDA (Li et al., 2019) and SCL-MI (Blitzer et al., 2007), in our experiment, those amazon reviews with rating > 3 is tagged as positive while those with rating < 3 is labeled as negative. The rest reviews are discarded due to their ambiguous polarities. In addition, we choose the 3 biggest categories (i.e., "Books", "Electronics", and "Movies and TV") of the Amazon Review dataset, and uniformly sample reviews from these categories to form a subset with $109,858$ reviews, which is sufficiently large to evaluate the effectiveness of our method. Note that the imbalance ratio of each

[4]http://jmcauley.ucsd.edu/data/amazon/

[5]http://alt.qcri.org/semeval2017/task4/index.php?id=data-and-tools

category in this subset is kept the same as that in the original dataset. Table 4.2 shows the details of both Amazon Review and SemiEval datasets.

4.2.2 Baseline

We compare SetConv with several state-of-the-art imbalanced learning methods.

- **IHT** (Smith et al., 2014) (*under-sampling*) performs under-sampling based on instance hardness to overcome the model bias.
- **WEOB2** (Wang et al., 2015) (*ensemble*) is an under-sampling based ensemble method that maintains both OOB and UOB with adaptive weights for final predictions. It only supports binary classification.
- **KMeans-SMOTE** (Last et al., 2017) (*over-sampling*) combines the k-means clustering algorithm with SMOTE to eliminate both inter-class and intra-class imbalances while at the same time avoiding the generation of noisy samples.
- **IML** (Wang et al., 2018) (*metric learning*) utilizes metric learning to explore the correlations among imbalance data and constructs an effective data space for classification.
- **CS-DMLP** (Díaz-Vico et al., 2018) (*cost-sensitive*) utilizes a cost-sensitive deep MLP to regularize the posterior probability distribution predicted for a given sample.

4.2.3 Evaluation Metric

Five widely-accepted metrics (Wang et al., 2018; Díaz-Vico et al., 2018; Last et al., 2017), i.e., Specificity (*Spec*), Sensitivity (*Sens*), F_1-measure (F_1), Geometric-Mean (*G-Mean*), and the Area Under the receiver operating characteristic Curve (AUC), are adopted to evaluate the model performance in our experiments. Table 4.3 presents the confusion matrix used in the multi-class classification scenario, where the model performance on each minority class is reported. It is because

Table 4.3: Confusion matrix for multi-class classification problem, where c denotes the class to evaluate.

	Predict Label $= c$	Predict Label $\neq c$
True Label $= c$	True Positive (TP)	False Negative (FN)
True Label $\neq c$	False Positive (FP)	True Negative (TN)

- In most imbalance learning problems, the minority classes are usually more important than the majority classes (He and Garcia, 2009; Chen and Shyu, 2011).

- Simply averaging model performance on different classes may cover model defects when class distribution is unbalanced.

Class-specific performance measure:

- $Spec = \frac{TN}{TN+FP}$. Spec measures the model's capability to avoid false positive and finds all negative samples. It provides the accuracy of the majority class.

- $Sens = \frac{TP}{TP+FN}$. Sens measures the model's capability to avoid false-negative and finds all positive samples. It provides the accuracy of the minority class.

Overall performance measure:

- $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$ is the harmonic mean of $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$.

- $G\text{-}Mean = \sqrt{Spec \cdot Sens}$. G-Mean receives a higher value only when both Spec and Sens stay at a higher level. Thus, *G-Mean* can be considered as a trade-off between *Spec* and *Sens*.

- AUC computes the area under the ROC curve. It measures the model's capability to distinguish positive and negative classes.

Higher values on these metrics indicate a model with better performance.

```
'''Please install transformers via pip install transformers'''
from transformers import BertModel, BertTokenizer
import torch

# create model
tokenizer = BertTokenizer.from_pretrained('bert-large-cased')
model = BertModel.from_pretrained('bert-large-cased')

input = 'Here is some text to encode'
tokens = tokenizer.encode(input, add_special_tokens=True, max_length=512)
input_ids = torch.tensor([tokens])
with torch.no_grad():
    last_hidden_state = model(input_ids)[0]
# we take the final hidden state of the special classification token [CLS]
# as the embedding for input sentences
embedding = last_hidden_state[:, 0, :]
```

Figure 4.4: Implementation code used to extract sentence embedding via Bert.

4.2.4 Experiment Setup

For each text data, we pre-process it by extracting a 1024-dimensional feature vector using a pre-trained Bert model[6] (Devlin et al., 2019), and utilize these embeddings in our experiments. There is no ground-truth information leakage in this step since Bert is trained on Wikipedia corpus with no supervision signals. Specifically, The BERT-Large, Cased (Whole Word Masking)[7] model released by the Google Research team is selected and the final hidden state of the special classification token [CLS] is used to represent an input text sequence. The code in Figure 4.4 illustrates this process in detail.

[6]https://github.com/huggingface/transformers

[7]https://github.com/google-research/bert

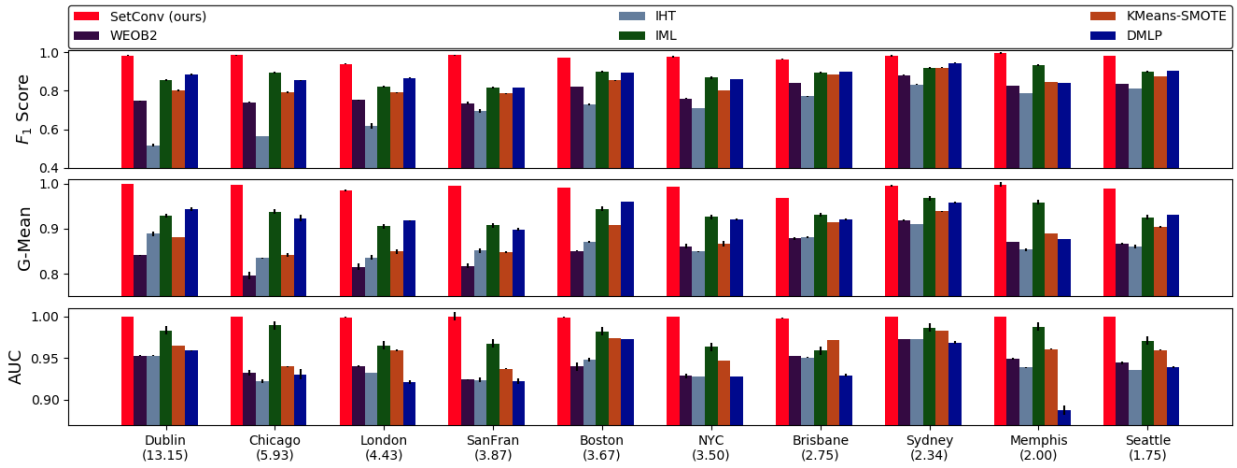


Figure 4.5: Binary classification (incident detection) performance of competing methods on the IRT dataset. The value in the bracket indicates the imbalance ratio (IR).

After data pre-processing, each dataset is uniformly shuffled and is divided into a development set and a test set, where the split ratio is 7:3. Thus, the class distribution of the original dataset is maintained in both development and test sets. The experiments are repeated 10 times and the average classification results are reported to reduce the statistical fluctuation resulting from random divisions.

Our algorithm is implemented via Python 3.7.3 and PyTorch 1.2.0. For all the baselines, we adopt the official code released by corresponding authors and set their default hyper-parameters based on values suggested in the corresponding papers. These hyper-parameters are then fine-tuned via 10-fold cross-validation on the development set. In SetConv, we set the output dimension of the SetConv layer $d_o = 128$, the size of support set $||S_{support}|| = N_1 + N_2 = 64$, the size of post-training subset $||S_{post}|| = 1000$, learning rate $r = 0.01$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

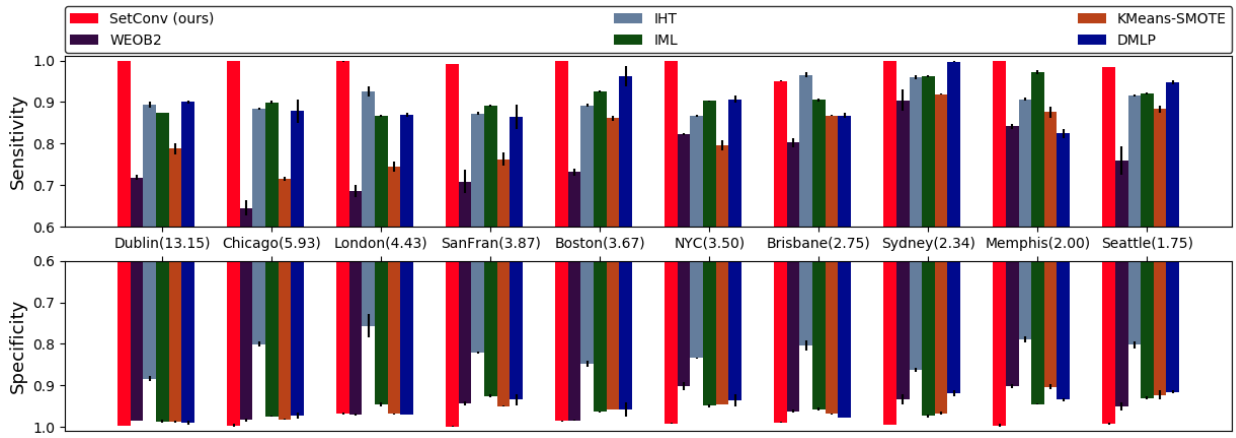


Figure 4.6: The performance diagnosis of competing methods for binary classification. The value in the bracket indicates the imbalance ratio (IR). In contrast to baselines that are biased towards either the majority or minority class, SetConv performs almost equally well on both classes.

4.2.5 Result

Binary Classification

Figure 4.5 and Figure 4.7 present the binary classification results of the incident detection and sentiment classification tasks. Our proposed framework SetConv achieves the best overall classification performance among the competing methods by providing the highest F_1, G-Mean, and AUC scores. In addition, as shown in Figure 4.6, all the baselines are biased towards either the majority class or the minority class. In contrast, our algorithm provides high specificity and sensitivity simultaneously and predicts accurately on both classes. It is also insensitive to geographical variations.

Multi-Class Classification

To verify the effectiveness of the SetConv framework for multi-class classification, we compare it with the baselines in the incident detection task. The classification performance of competing methods on the three minority classes, i.e., "Fire", "Shooting", and "Crash", are reported in Table 4.4. Here we only show the results in New York City (NYC) due to space

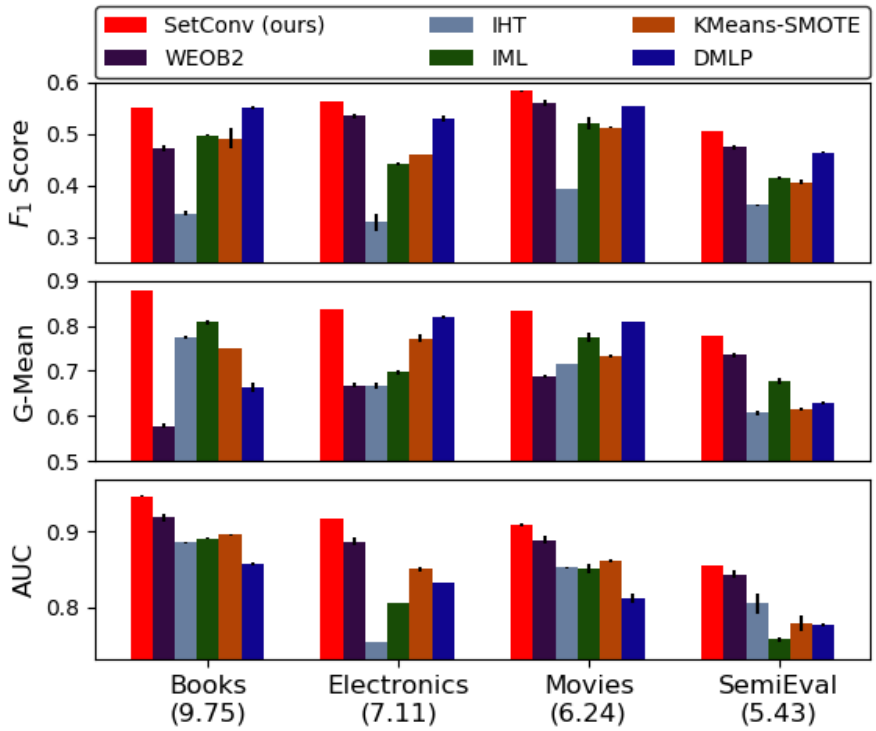


Figure 4.7: Binary sentiment classification performance of competing methods on the Amazon Review and SemiEval datasets. The value in the bracket indicates the imbalance ratio (IR).

limitation. Similar results are observed in other cities. The overall classification performance of our method is best among the competing methods since SetConv provides better F_1, G-Mean, and AUC scores than the baselines in most cases. Moreover, it performs almost equally well on all the three minority classes, which is desired in real-world applications.

4.2.6 Sensitivity Analysis

Our algorithm mainly depends on a single hyper-parameter, i.e., the size of post training subset $||S_{post}||$. Therefore, we perform a sensitivity analysis of SetConv on $||S_{post}||$ and the result is shown in Figure 4.8. Our method performs stably with respect to various values of $||S_{post}||$, which demonstrates that it learns to capture the class concept common to different samples of a class. Thus, as long as $||S_{post}||$ is large enough, varying $||S_{post}||$ has little effect on the model performance of SetConv.

Table 4.4: Multi-class classification performance of competing methods on the IRT-NYC dataset. 0.000 indicates a value of less than 0.0005.

Fire					
	F_1	G-Mean	AUC	Spec	Sens
IHT	0.601±0.000	0.866±0.002	0.947±0.001	0.841±0.002	0.891±0.005
KMeans-SMOTE	0.831±0.005	0.894±0.003	0.967±0.001	0.978±0.001	0.818±0.005
IML	0.889±0.001	0.947±0.002	0.987±0.002	0.978±0.001	0.917±0.002
CS-DMLP	0.931±0.004	0.951±0.006	0.998±0.001	**0.993±0.004**	0.911±0.016
SetConv (ours)	**0.972±0.002**	**0.996±0.000**	**0.999±0.000**	0.992±0.001	**0.999±0.001**
Shooting					
	F_1	G-Mean	AUC	Spec	Sens
IHT	0.333±0.001	0.471±0.002	0.984±0.001	**0.997±0.001**	0.222±0.002
KMeans-SMOTE	0.895±0.002	0.969±0.003	0.962±0.001	0.996±0.002	0.944±0.003
IML	0.688±0.001	0.780±0.002	0.986±0.001	0.996±0.001	0.611±0.002
CS-DMLP	0.822±0.002	0.910±0.029	0.994±0.002	0.995±0.002	0.883±0.006
SetConv (ours)	**0.912±0.012**	**0.998±0.003**	**0.999±0.001**	0.995±0.001	**0.999±0.001**
Crash					
	F_1	G-Mean	AUC	Spec	Sens
IHT	0.306±0.023	0.762±0.019	0.865±0.011	0.755±0.020	0.769±0.019
KMeans-SMOTE	0.633±0.009	0.802±0.016	0.920±0.014	0.955±0.011	0.673±0.019
IML	0.662±0.002	0.937±0.003	0.959±0.001	0.932±0.001	0.942±0.003
CS-DMLP	0.702±0.054	0.917±0.002	0.969±0.013	0.951±0.017	0.885±0.019
SetConv (ours)	**0.931±0.013**	**0.977±0.001**	**0.997±0.001**	**0.992±0.002**	**0.962±0.001**

4.3 Discussion

In this chapter, we describe the proposed permutation-invariant SetConv operation and episodic training strategy in details. By jointly utilizing SetConv and episodic training, our approach is able to extract discriminative features from data and automatically balance the class distribution for the subsequent classifier. Our model performs better than existing solutions because (1) compared to resampling based approaches, e.g., IHT and WEOB2, our method makes full utilization of data and avoids losing important information by removing samples from the majority classes. (2) Compared to cost-sensitive approaches, e.g., CS-DMLP, our method assigns equal weights to both the majority and minority classes via episodic training and eliminates the overhead of finding suitable cost values for different

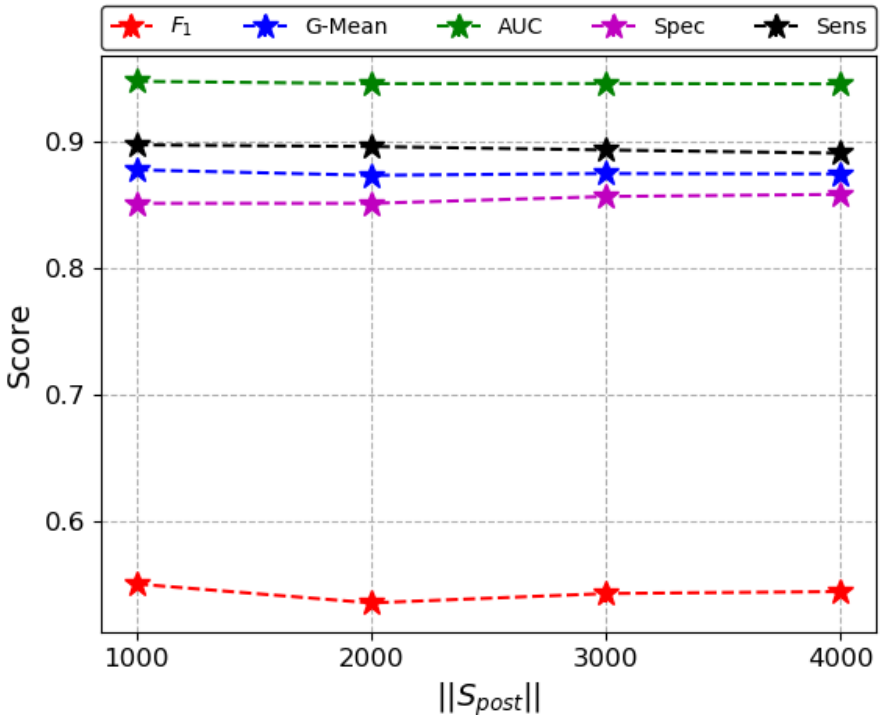


Figure 4.8: Effect of post-training subset size ($||S_{post}||$) on classification performance.

applications. The model is forced to address the class imbalance by learning to extract discriminative features.

Although SetConv is superior to existing solutions in many applications, it may not be suitable for classifying high-dimensional sparse data. These data produce close-to-zero convolution kernel weights, which may limit the model's expressiveness. Incorporating sparse deep learning techniques into SetConv is a potential solution to this problem. We will study it for future work.

CHAPTER 5

SIM: OPEN-WORLD MULTI-TASK STREAM CLASSIFIER WITH INTEGRAL SIMILARITY METRICS [1]

5.1 Approach

In this chapter, we discuss our proposed stream classifier SIM in details and explain how to employ similarity learning to assist classification on open-ended data distributions.

5.1.1 Problem Setting

In stream classification, a small labeled dataset, $D = \{(\boldsymbol{x_i} \in \mathcal{R}^d, y_i \in \mathcal{Y})\}_{i=1}^{m}$ with label set $\mathcal{Y} = \{1, 2, \ldots, c\}$, is usually given to initialize a stream classifier. Let $\mathcal{S} = \{(\boldsymbol{x_t}, y_t)\}_{t=1}^{\infty}$, where $\boldsymbol{x_t} \in \mathcal{R}^d$ and $y_t \in \mathcal{Y}' = \{1, 2, \ldots, c, c+1, \ldots, c'\}$ $(c' > c)$, denote a non-stationary data stream. Our goal is to learn a model f initially with D that maps each incoming instance $\boldsymbol{x_t}$ from $\mathcal{S}$ to a specific label in $\mathcal{Y}'$, i.e., $f(\boldsymbol{x_t}) \rightarrow \mathcal{Y}'$. Particularly, f determines whether the input instance belongs an existing class or an unknown class (also referred as a novel class), and is later automatically adapted to incorporate those instances from unknown classes. Note that the intrinsic cohesion and separation property of data may be invalid in the observed feature space. That is, for any two arbitrary classes $c_m, c_n \in \mathcal{Y}'$ $(c_m \neq c_n)$, if $\{\boldsymbol{x_i}, \boldsymbol{x_j}\} \in c_m$ and $\{\boldsymbol{x_k}\} \in c_n$, it is possible that $||\boldsymbol{x_i} - \boldsymbol{x_k}||_2 < ||\boldsymbol{x_i} - \boldsymbol{x_j}||_2$. As a result, f should actively look for a latent feature space where the cohesion and separation constraint for novel class detection is satisfied. For example, as shown in Figure 5.1, in the observed feature space, instances of a novel class C are close to those of class A or B but are relatively far away from each other. After transforming all instances into another latent feature space,

[1] This chapter contains material previously published as: ©2019 IEEE. Reprinted, with permission, from Yang Gao, Yi-Fan Li, Bo Dong, Yu Lin, and Latifur Khan. "SIM: Open-World Multi-Task Stream Classifier with Integral Similarity Metrics." In *IEEE International Conference on Big Data (Big Data)*, pp. 751-760. December, 2019. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

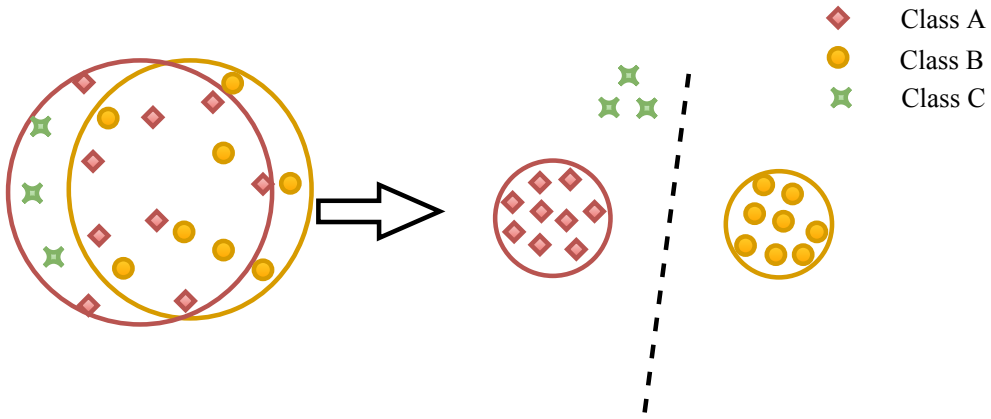


Figure 5.1: Novel class detection without (left) and with (right) metric learning. Here, classes A and B denote two known classes while class C represents a novel class.

Table 5.1: Summary of the notations.

Notation	Description
$\mathcal{S}$	Stream data
$\mathcal{Y}$	Label set of initial training data
$\mathcal{Y}'$	Open set of possible labels in $\mathcal{S}$
f	Mult-Task Open-World classifier (MT-OWC)
$\boldsymbol{x} \in \mathcal{R}^d$	A d-dimensional instance arriving in $\mathcal{S}$
$y \in \mathcal{Y}'$	The associated ground truth class label of $\boldsymbol{x}$
D	Initial training data in warm-up phase
$\hat{y}$	Estimated label of $\boldsymbol{x}$ given by the multi-task open-world classifier f
$\tilde{y}$	Final predicted label of $\boldsymbol{x}$ given by SIM
$\mathcal{B}$	Novel class candidate buffer
$S_{\mathcal{B}}$	The maximum size of $\mathcal{B}$
$S_{\mathcal{D}}$	The maximum storage capacity of each class in $\mathcal{D}$
$\mathcal{T}_{\mathcal{D}}$	Confidence threshold for updating $\mathcal{D}$
$\mathcal{P}(\boldsymbol{x})$	Prediction confidence for $\boldsymbol{x}$ using f
$\mathcal{D}$	Data storage - A buffer that stores at most $S_{\mathcal{D}}$ instances for each class
$\mathcal{D}'$	Subset of the data storage $\mathcal{D}$ for training f
ϕ	The metric embedding function that maps $x \in \mathcal{R}^d$ to $\phi(\boldsymbol{x}) \in \mathcal{R}^{d'}$
γ	Significance level of $\mathcal{L}_{triplet}$
S_{mini}	Mini-batch size for triplet generation and training of f
S_{update}	Minimum number of instances of each class in $\mathcal{D}$ for classifier update
N_{mini}	Number of triplets to generate in each mini-batch.
n_{epoch}	number of epochs to train MT-OWC

instances of classes A, B, and C form their own clusters respectively. Those in the cluster of class C are well separated from instances of class A and B so that they can be detected easily. On the other hand, due to the non-stationary nature of a data stream and the limited

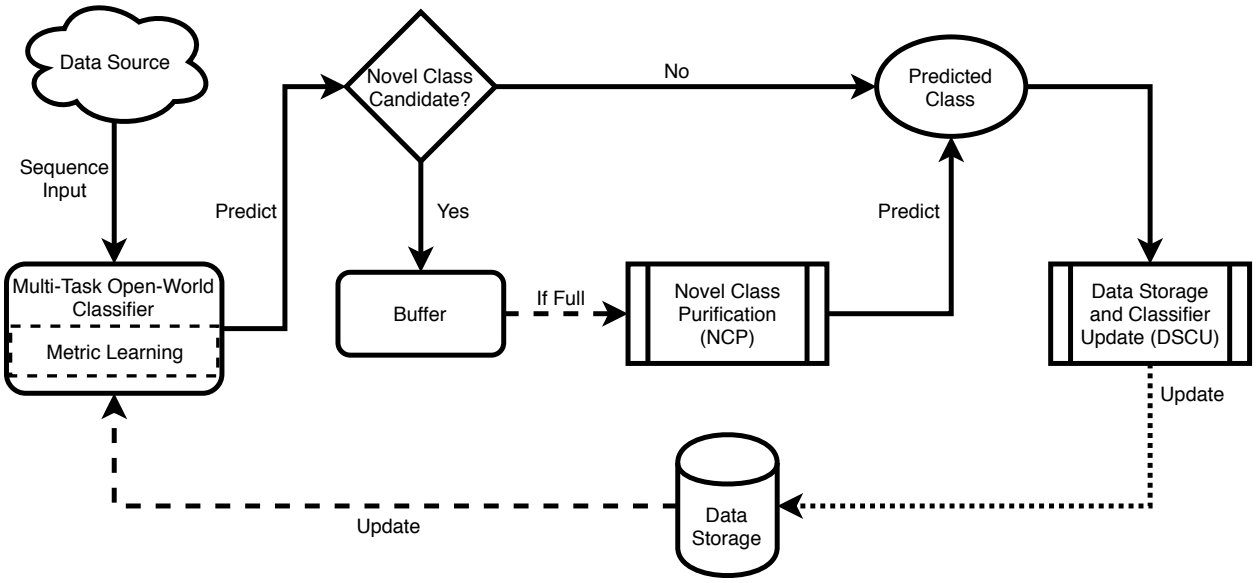


Figure 5.2: Overview of the SIM framework.

computational resources, e.g., storage, f has to effectively detect the emergence of novel classes while requiring a small number of truth-values at a time for a model update and should predict well over long periods of time afterward.

5.1.2 Overview

A typical technique requires sufficient novel class candidates to determine whether these instances belong to a novel class or not. A threshold function over instance density in feature space is utilized to determine the existence of a novel class. If one or more novel classes are detected, the model is updated by training a new classifier based on data including instances from these new classes. In our work, we adopt this mechanism and apply it along with a unique metric learning schema that aids in the identification of potential novel class candidates.

To do it, we introduce a semi-supervised framework SIM with its core components and workflow illustrated in Figure 5.2. An essential component of SIM is a multi-task open-world classifier, which is trained only on data of known classes but is capable of dealing with

many instances belonging to unseen classes during evaluation. A metric learning mechanism is internally employed in this classifier to actively search for a latent feature space where the strong cohesion among instances from the same class and large separation among instances from different classes holds. Thus, a multi-task open-world classifier performs metric learning, novel class detection, and classification tasks simultaneously.

In the beginning, a small set of training data denoted as D is used to initialize a multi-task open-world classifier. For any incoming instance $\boldsymbol{x}$ from the data stream $\mathcal{S}$, the classifier predicts its estimated label $\hat{y}$ as the one that maximizes the likelihood. If $\boldsymbol{x}$ is not a potential novel class candidate, i.e., $\hat{y} \neq -1$, then its final predicted label $\tilde{y}$ is same as its estimated label $\hat{y}$, i.e., $\tilde{y} = \hat{y}$. Otherwise, we temporarily store $\boldsymbol{x}$ in a candidate buffer $\mathcal{B}$, and process it later via the Novel Class Purification (NCP) module.

Once the buffer $\mathcal{B}$ is full, the NCP module is utilized to separate those instances of unknown classes from others that are incorrectly introduced into the buffer due to noise in the stream. These novel class instances are then used to update the data storage $\mathcal{D}$, which is a buffer storing at most $S_{\mathcal{D}}$ instances for each class. Once the update condition is satisfied, a new classifier is trained on the updated $\mathcal{D}$ to incorporate information from new classes. The details of the classification and novel class detection processes are described in Algorithm 2.

5.1.3 Multi-Task Open-World Classifier (MT-OWC)

Since the intrinsic cohesion and separation assumption is often invalid in real-world high-dimensional data streams, to perform novel class detection in this case, we leverage metric learning mechanism to actively search for a latent space where similar instances are close to each other (i.e., the Euclidean distance between them is small) and dissimilar instances are far away from each other (i.e., the Euclidean distance between them is large). Moreover, we fuse metric learning and novel class detection into classification, because a high-quality metric obtained by metric learning is critical to both classification and novel class detection

Algorithm 2 SIM: Stream Classification

Require: $\mathcal{S}$ - Stream data; $S_{\mathcal{B}}$ - The maximum size of the candidate buffer $\mathcal{B}$; $\mathcal{T}_{\mathcal{D}}$ - Confidence threshold for updating the data storage $\mathcal{D}$; D - Initial training data;

Ensure: Label $\tilde{y}$ predicted on $\mathcal{S}$ data.

```
Learn an initial model f from D by solving the optimization problem. (Eq. 5.6)
repeat
    Receive a new instance x.
    Predict estimated label ŷ for x using f according to Eq. 5.8
    if ŷ = −1 then
        Store x in the candidate buffer B.
    else
        Final predicted label ỹ ← ŷ
        Prediction confidence P(x) ← P(y̆ = ŷ|x) (Section 5.1.4: Classification)
    end if
    if size(B)≥ S_B then
        Check for occurrence of any novel class in data using PurifyNovel (Algorithm 3)
        if PurifyNovel returns True then
            if Update-Condition (Section 5.1.6: DSCU) Satisfied then
                Retrain f with D′ (a subset of D) (Section 5.1.6: DSCU).
            end if
        end if
    end if
    if ŷ ≠ −1 and P(x) > T_D then
        Update D using (x, ỹ) (Section 5.1.6: DSCU).
    end if
until S exits
```

tasks. Hence, we propose a novel classifier to perform all these tasks jointly, which is referred to as *Multi-Task Open-World Classifier (MT-OWC)*. The architecture of MT-OWC is illustrated in Figure 5.3. In this classifier, the *Input Layer*, *Hidden Layer* and *Embedding Layer* minimize our proposed loss function to learn a metric function ϕ that projects instances from the observed feature space to the desired latent feature space where the cohesion and separation among instances are supported. Following (Shu et al., 2017a), instead of using softmax as the final output layer, we create a *Classification/Novel Class Detection Layer* that assigns a sigmoid function to each of the K classes, which is able to handle the common cases where the number of novel classes is unknown. For the i^{th} sigmoid function corresponding to class

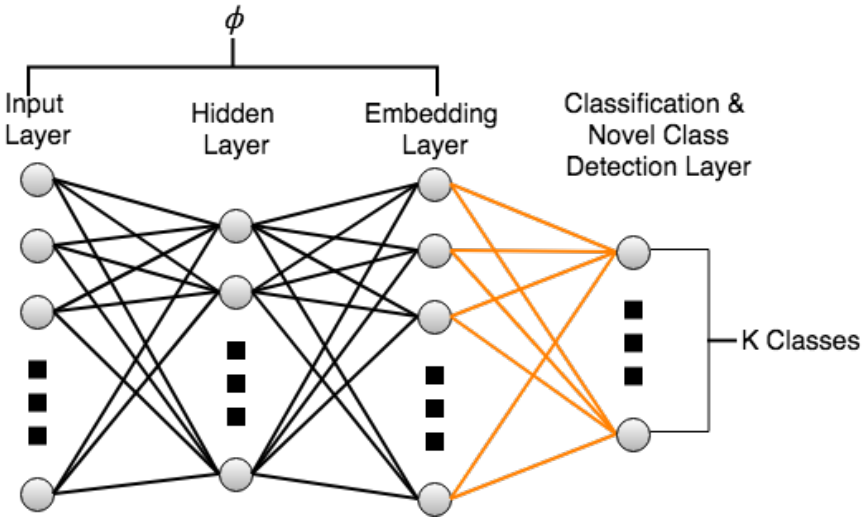


Figure 5.3: The network architecture of the proposed Multi-Task Open-World Classifier.

c_i, MT-OWC takes all examples with label $y = c_i$ as positive examples and the rest with $y \neq c_i$ as negative examples, forming a one-vs-rest binary classification scenario.

We rely on the *triplets* defined in Definition 1 to train the proposed MT-OWC. A triplet allows evaluating both similar and dissimilar relations simultaneously, which could potentially improve the quality of learned metrics and the resulting model performance. For simplicity, we denote a training set containing M triplets as $\mathbb{D}$, the triplet loss introduced by the metric function ϕ as $\mathcal{L}_{triplet}$, and the classification loss introduced by the final layer of MT-OWC as $\mathcal{L}_{class}$, respectively.

Definition 1 (Triplet)**.** *A triplet* $(\boldsymbol{x^a}, \boldsymbol{x^p}, \boldsymbol{x^n})$ *is a group of three instances where* $\boldsymbol{x^a}$ *(anchor) is similar to* $\boldsymbol{x^p}$ *(positive) but is dissimilar to* $\boldsymbol{x^n}$ *(negative).*

The Triplet Loss ($\mathcal{L}_{triplet}$)

Suppose $(\boldsymbol{x_i^a}, \boldsymbol{x_i^p}, \boldsymbol{x_i^n})$ is the i^{th} triplet in $\mathbb{D}$. Any instance $\boldsymbol{x} \in \mathcal{R}^d$ can be embedded into a d'-dimensional Euclidean latent space by the metric function ϕ, i.e., $\phi(\boldsymbol{x}) \in \mathcal{R}^{d'}$. The Euclidean distance between $\boldsymbol{x_i^a}$ and $\boldsymbol{x_i^n}$ in this space is expected to be at least e^{γ} ($\gamma \geq 1$) times the distance between $\boldsymbol{x_i^a}$ and $\boldsymbol{x_i^p}$. That is,

$$\frac{||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^n})||_2}{||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^p})||_2} \geq \mathrm{e}^{\gamma} \tag{5.1}$$

After smoothing, it becomes

$$\frac{||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^n})||_2 + 1}{||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^p})||_2 + 1} \geq \mathrm{e}^{\gamma} \tag{5.2}$$

The triplet loss $\mathcal{L}_{triplet}$ is hence:

$$\begin{aligned}\mathcal{L}_{triplet} =& \frac{1}{M}\sum_{i=1}^{M}\Big[\log(||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^p})||_2 + 1) + \gamma \\ & - \log(||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^n})||_2 + 1)\Big]_+\end{aligned} \tag{5.3}$$

where γ is a user-specified hyper-parameter (referred as the significance level of $\mathcal{L}_{triplet}$), and is used to adjust the margin: $||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^n})||_2 - ||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^p})||_2 \approx (\mathrm{e}^{\gamma} - 1)||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^p})||_2$. This margin gives the minimum separation between $\boldsymbol{x_i^p}$ and $\boldsymbol{x_i^n}$ for each triplet, and is monotonically increasing with γ. To reduce the potential model search space and accelerate the training, the learned embedding is also constrained to reside on a d'-dimensional unit sphere. The metric function ϕ in $\mathcal{L}_{triplet}$ can be any non-linear function. In SIM, we represent ϕ as the *single-hidden* layer neural network shown in Figure 5.3 for simplicity.

The Classification Loss ($\mathcal{L}_{class}$)

Similar to (Shu et al., 2017b), we use the average Binary Classification Error (BCE) of the K classes to measure the classification loss. Formally, it is:

$$\begin{aligned}\mathcal{L}_{class} =& \frac{1}{3KM}\sum_{j=1}^{M}\sum_{i=1}^{K}\sum_{*\in\{a,p,n\}}[-\mathbb{I}(y_{\boldsymbol{x_j^*}} = c_i)\log\mathcal{P}(y_{\boldsymbol{x_j^*}} = c_i) \\ & - \mathbb{I}(y_{\boldsymbol{x_j^*}} \neq c_i)\log(1 - \mathcal{P}(y_{\boldsymbol{x_j^*}} = c_i))]\end{aligned} \tag{5.4}$$

where $\mathcal{P}(y_{\boldsymbol{x_j^*}} = c_i) = \sigma(W_i\phi(\boldsymbol{x_j^*}) + b)$ and W_i is the weight associated with i^{th} class in the Classification/Novel Class Detection layer.

Training of MT-OWC

In contrast to (Shu et al., 2017b), we do not optimize $\mathcal{L}_{class}$ on its own. Instead, MT-OWC jointly learns the tasks of metric learning, classification, and novel class detection by simultaneously optimizing both the classification loss $\mathcal{L}_{class}$ and the triplet loss $\mathcal{L}_{triplet}$. Hence, the knowledge gained in metric learning helps to improve the generalization performance of the classifier in presence of novel classes, and vice versa. This information transfer is critical to stream applications where a limited amount of labeled training data is accessible. The overall multi-task objective function $\mathcal{L}_{overall}$ is:

$$\mathcal{L}_{overall} = \mathcal{L}_{class} + \beta \mathcal{L}_{triplet} \tag{5.5}$$

where $\beta \in [0, 1]$ controls the importance of $\mathcal{L}_{triplet}$ in $\mathcal{L}_{overall}$. The final optimization problem is:

$$\begin{aligned} \underset{\phi, W_1, W_2, \ldots, W_K}{\text{minimize}} \quad & \mathcal{L}_{overall} \\ \text{subject to} \quad & ||\phi(\boldsymbol{x_i^*})||_2 = 1, \forall \boldsymbol{x_i^*} \in \mathbb{D}. \end{aligned} \tag{5.6}$$

Here $\boldsymbol{x_i^*}$ denotes any anchor, positive or negative instance according to the triplet definition.

Triplet Generation and Selection

Another important question that remains to be answered is how do we generate and select triplets from a given data buffer $\mathcal{D}'$ to form a triplet set $\mathbb{D}$ for training? Iterating all possible combinations is not only computationally intractable but also unnecessary because many of the resulting triplets produce a zero triplet loss value and hence make little contribution to model update. Therefore, selecting *hard* triplets is critical to continuously improving the model performance. Here, we use the term "hard" to indicate a positive triplet loss.

Suppose $\mathcal{D}' = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\} \in \mathcal{R}^{d \times C}$, where $C = \{c_1, \ldots, c_k\}$ denotes k different classes. A triplet can be generated by:

Step 1: Select an "anchor" class c_a and a "negative" class c_n ($c_a \neq c_n$) from $\mathcal{D}'$;

Step 2: Randomly sample an anchor instance $\boldsymbol{x_i^a}$ from the class c_a.

Step 3: Given $\boldsymbol{x_i^a}$, we'd like to select a positive instance $\boldsymbol{x_i^p}$ from class c_a such that $||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^p})||_2$ is maximized, and a negative instance $\boldsymbol{x_i^n}$ from class c_n such that $||\phi(\boldsymbol{x_i^a}) - \phi(\boldsymbol{x_i^n})||_2$ is minimized.

Note that the triplets should be generated in an online fashion due to the nature of data streams. To substantially accelerate the search, at each step, we uniformly sample a subset of instances from $\mathcal{D}'$ to form a mini-batch. We then transform the instances within that mini-batch to the embedding space and compute the arg max and arg min among them to generate desired triplets. The expected search space becomes $O(\frac{\mathcal{S}_{mini}^3}{k^3})$, where $\mathcal{S}_{mini}$ represents the mini-batch size. A total of N_{mini} triplets are generated for each mini-batch.

5.1.4 Classification

Let f and $\breve{y}$ denote the multi-task open-world classifier and the preliminary label predicted by f respectively. For every incoming instance $\boldsymbol{x}$ from stream $\mathcal{S}$, the preliminary prediction probability $\mathcal{P}(\breve{y} = c_i|\boldsymbol{x})$ of class c_i is computed by $\mathcal{P}(\breve{y} = c_i|\boldsymbol{x}) = \sigma(W_i\phi(\boldsymbol{x}) + b)$. To decide the final predicted label of $\boldsymbol{x}$, a threshold $\mathcal{T}_{novel}$ for novel class detection has to be determined first. However, manually setting a fixed threshold is improper in our case due to the non-stationary nature of data streams. Therefore, we leverage the concept of one-sided confidence bound in statistics to automatically obtain a suitable $\mathcal{T}_{novel}$.

Since Gaussian distributions introduce the minimum prior constraints to the underlying distribution of $\mathcal{P}(\breve{y} = c_i|\boldsymbol{x})$, we assume $\mathcal{P}(\breve{y} = c_i|\boldsymbol{x})$ follow a Gaussian distribution with unknown mean and unknown variance. A good statistic for the confidence threshold is the average prediction probability of training data, i.e., $\bar{\mathcal{P}}(c_i) = \frac{1}{||\mathcal{D}_i'||}\sum_{\mathcal{D}_i'}\mathcal{P}(\breve{y} = c_i|\boldsymbol{x} \in \mathcal{D}_i')$, where $\mathcal{D}_i' = \{(\boldsymbol{x_j}, y_j = c_i), \forall \boldsymbol{x}_j \in \mathcal{D}'\}$. Note that $\bar{\mathcal{P}}(c_i)$ follows a t distribution with $||D_i'|| - 1$ degrees of freedom. Therefore, the desired $\mathcal{T}_{novel}$ for class c_i is the $100(1-\alpha)\%$ confidence lower bound of $\bar{\mathcal{P}}(c_i)$:

$$\mathcal{T}_{novel}(c_i) = \bar{\mathcal{P}}(c_i) - t_{\alpha,||\mathcal{D}_i'||-1}S_{c_i}/\sqrt{||\mathcal{D}_i'||} \tag{5.7}$$

Here, S_{c_i} is the sample standard deviation of $\{\mathcal{P}(\breve{y} = c_i|\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{D}'_i\}$.

Once the threshold $\mathcal{T}_{novel}$ is determined, the classification process is as usual. Given an instance $\boldsymbol{x}$, for the i^{th} class c_i, we examine if the preliminary prediction probability $\mathcal{P}(\breve{y} = c_i|\boldsymbol{x})$ is less than the corresponding novel class detection threshold $\mathcal{T}_{novel}(c_i)$. If this condition holds for all classes, $\boldsymbol{x}$ is regarded as a candidate from a novel class. As a result, SIM simply rejects $\boldsymbol{x}$, i.e., predicts its estimated label as -1, and temporarily stores $\boldsymbol{x}$ in $\mathcal{B}$. On the other hand, if this condition holds for one or more classes, the estimated label of $\boldsymbol{x}$ is the class that gives the highest preliminary prediction probability. Mathematically, it is:

$$\hat{y} = \begin{cases} -1 & \text{if } \mathcal{P}(\breve{y} = c_i|\boldsymbol{x}) < \mathcal{T}_{novel}(c_i), \\ & \forall c_i \in \mathcal{Y}_{\mathcal{D}'} \\ \underset{c_i \in \mathcal{Y}_{\mathcal{D}'}}{\arg\max}\, \mathcal{P}(\breve{y} = c_i|\boldsymbol{x}) & \text{otherwise} \end{cases} \tag{5.8}$$

where $\hat{y}$ and $\mathcal{Y}_{\mathcal{D}'}$ represent the the estimated label for an instance $\boldsymbol{x}$ and the label set of $\mathcal{D}'$ respectively. If $\hat{y} \neq -1$, the final predicted label $\tilde{y}$ is the same as $\hat{y}$, i.e., $\tilde{y} = \hat{y}$; Otherwise, the prediction of $\tilde{y}$ is left to the novel class purification module.

5.1.5 Novel Class Purification (NCP)

In some cases, a few noisy known class instances in data streams may be incorrectly reported as from novel classes. To reduce the false alarm rate, once the novel class candidate buffer $\mathcal{B}$ is full, we invoke the novel class purification module to separate novel class instances from other candidates in $\mathcal{B}$. The procedure is described below:

Step 1: We project every candidate instance in $\mathcal{B}$ to the latent feature space learned via metric learning and obtain its corresponding embedding $\phi(\boldsymbol{x})$.

Step 2: An unsupervised clustering algorithm, *DBSCAN* (Ester et al., 1996), is applied over the transformed instances to form clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$. DBSCAN is selected because it does not require the number of clusters being specified in advance.

Algorithm 3 PurifyNovel

Require: Candidate Buffer $\mathcal{B}$
Ensure: True/False
$\mathbb{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\} \leftarrow DBSCAN(\mathcal{B})$
$Novel \leftarrow False$
for $\mathcal{C}_i \in \mathbb{C}$ **do**
  Uniformly sample a instance $\boldsymbol{x}_{c_i} \in \mathcal{C}_i$.
  Request truth label y_{c_i} of $\boldsymbol{x}_{c_i}$.
  if y_{c_i} is unknown before **then**
    $Novel \leftarrow True$
  end if
  for $\boldsymbol{x} \in \mathcal{C}_i$ **do**
    Update the data storage $\mathcal{D}$ using $(\boldsymbol{x}, y_{c_i})$ (Section 5.1.6: DSCU).
    $\tilde{y} \leftarrow y_{c_i}$
  end for
end for
return $Novel$

Step 3: For each cluster $\mathcal{C}_i$, we uniformly sample one instance out from $\mathcal{C}_i$ and request its true label. This true label would be the prediction label for all instances within $\mathcal{C}_i$. It is executable because we assume an external source, e.g., image captions, context information, or human experts, can provide the true label for a particular instance, following (Masud et al., 2011a; Al-Khateeb et al., 2012b; Haque et al., 2016a; Mu et al., 2017b; Zhang et al., 2018b). However, we greatly reduce the number of label requests by grouping similar instances together and propagating the label of a group representative to other members within the same group. Algorithm 3 formally describes the workflow of NCP module.

5.1.6 Data Storage and Classifier Update (DSCU)

The data storage $\mathcal{D}$ contains a unique buffer for each of known classes, which stores at most $S_{\mathcal{D}}$ instances of that class. Whenever an unseen class is discovered, a corresponding new buffer is added to $\mathcal{D}$. Suppose $\mathcal{D}_i$ represents the buffer of class c_i. For every $\boldsymbol{x}$ of class c_i sent to update $\mathcal{D}$, if $\mathcal{D}_i$ is not full, $\boldsymbol{x}$ is simply appended to $\mathcal{D}_i$; Otherwise, the "oldest" instance

in $\mathcal{D}_i$ is replaced by $\boldsymbol{x}$. We update the classifier f if and only if both of the following update conditions are satisfied:

- Algorithm 3 returns $True$, i.e., some novel class instances are discovered.
- If m different novel classes are found by Algorithm 3 since last update of f, at least one of the m classes have more than S_{update} instances in $\mathcal{D}$.

Once these conditions are satisfied, a new training dataset $\mathcal{D}'$ is formed by including all classes with more than S_{update} instances in $\mathcal{D}$, and is used to train a new classifier f.

5.1.7 Time and Space Complexity

The computational burden of SIM mainly comes from the training of the multi-task open-world classifier. Assume the time cost of gradient computation for one example is a constant C, then the time complexity of processing a mini-batch is $O(S_{mini}^3 C)$. The overall time complexity of SIM is $O(n_{epoch} S_{mini}^2 C S_{\mathcal{D}} ||\mathcal{Y}'||)$, where $||\mathcal{Y}'||$ represents the number of classes in $\mathcal{Y}'$. On the other hand, the overall space complexity of SIM is $O(S_{\mathcal{B}} + S_{\mathcal{D}}||\mathcal{Y}'|| + B_{\text{space}})$, where B_{space} is the space complexity of the model representing ϕ.

5.1.8 Extendibility to Handle Concept Drift

Another common challenge in stream classification is the so-called concept drift. It occurs in a stream when the underlying concept of data changes over time (Haque et al., 2016a). Extending SIM to address both concept evolution and concept drift is easy. As discussed in Section 5.1.5, in the novel class purification module of SIM, the novel class candidates are grouped into a number of clusters and one instance is uniformly sampled out from each cluster to determine the label of instances within that cluster. Many of these clusters are formed by instances from known classes if concept drift exists in data streams. Therefore, if the representative instance of a cluster belongs to one of the known classes, all instances

Table 5.2: Description of Datasets

Type	Dataset	# features	# classes	# instances
Image	FASHION-MNIST	784	10	70,000
	MNIST	784	10	70,000
	EMNIST	784	47	131,600
	CIFAR-10	1024	10	70,000
Text	New York Times	300	21	70,000
	Guardian	300	10	66,112

within that cluster are marked as potentially appearing due to concept drift. We will evaluate this approach in our future work.

5.2 Evaluation

5.2.1 Datasets

We use four publicly available benchmark real-world image datasets including **FASHION-MNIST**[2] (Xiao et al., 2017b), **MNIST**[3] (LeCun et al., 2010), **EMNIST**[4] (Cohen et al., 2017b) and **CIFAR-10**[5] (Krizhevsky and Hinton, 2009) for evaluation. In particular, the images in CIFAR-10 are converted into gray-scale via OpenCV API, leading to 1024 features per image. Table 5.2 lists the details of these datasets.

For every stream, we build an initial stream classifier based on a small training set with $\lfloor n \cdot r \rfloor$ classes, where n is the total number of classes in the dataset and $r \in [0, 1]$ is a user-specified constant indicating the fraction of classes that are known in the beginning. Hence, the novel class collection contains instances from the $n - \lfloor n \cdot r \rfloor$ remaining classes. For each benchmark dataset, a data stream is simulated by including instances from both

[2]https://github.com/zalandoresearch/fashion-mnist

[3]http://yann.lecun.com/exdb/mnist/

[4]https://www.nist.gov/itl/iad/image-group/emnist-dataset

[5]https://www.cs.toronto.edu/ kriz/cifar.html

the known classes and new classes in the novel class collection. Those instances of unknown classes appear in different periods of the stream uniformly and classes may disappear and/or reappear along the stream. For example, suppose $\{c_1 : \{\boldsymbol{x_1}, \boldsymbol{x_2}\}, c_2 : \{\boldsymbol{x_3}, \boldsymbol{x_4}\}\}$ and $\{c_3 : \{\boldsymbol{x_5}, \boldsymbol{x_6}\}, c_4 : \{\boldsymbol{x_7}, \boldsymbol{x_8}\}\}$ are collections of existing class (c_1, c_2) and novel class (c_3, c_4) candidates respectively at time t. An incoming instance $\boldsymbol{x}$ arriving at time $t+1$ (y is its associated label) is equally likely to be from the existing or novel class set, i.e., $P(y \in \{c_1, c_2\}) = P(y \in \{c_3, c_4\})$. If y is selected as existing class, then $\boldsymbol{x}$ is uniformly sampled from existing class set $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}\}$; otherwise it is uniformly sampled from novel class set $\{\boldsymbol{x_5}, \boldsymbol{x_6}, \boldsymbol{x_7}, \boldsymbol{x_8}\}$.

We also conduct additional experiments on two real-world text streams. One of them is the **New York Times** news stream crawled via the New York Times API[6], which contains news articles in 21 categories published from Jan.1, 2005, to Jan.1, 2018. The biggest three classes, i.e., "Arts", "Business Day", and "Sports", are uniformly sampled to obtain 3000 articles as the initial training dataset. Another stream is the news story stream crawled via the **Guardian** API[7]. It contains news stories in 10 categories published between Jan.1, 2016, and Jan.1, 2018. Similarly, 3000 stories are uniformly sampled from the classes "Film", "Politics", and "Technology" to form the initial training dataset. Note that news articles or stories are crawled in chronological order in both cases, and each news item is fed into a pre-trained Word2Vec model[8] to obtain a 300-dimensional feature vector. These text streams are also adopted by our baselines such as SENC-MaS (Mu et al., 2017b) so that the comparison between existing solutions and our model is fair.

[6]http://developer.nytimes.com/

[7]https://open-platform.theguardian.com/explore/

[8]https://radimrehurek.com/gensim/index.html

Table 5.3: Classification performance of competing stream classifiers over different image streams. ∘/• indicates that **SIM** performs statistically worse/better (0.05 significance level) than the corresponding methods. The number of initial known classes is $\lfloor 47 \cdot r \rfloor$ for EMNIST and is $\lfloor 10 \cdot r \rfloor$ for other datasets.

Dataset	**r=0.3**	**ECSMiner**	**SENC-MaS**	**ECHO-D**	**SIM (Ours)**
CIFAR-10	Accuracy (%)	28.30±0.38 •	44.67±0.01 •	27.68±0.40 •	**55.47±0.66**
	label ratio	2.50±0.05	**0.86±0.02**	1.13±0.05	1.00±0.00
	effectiveness	11.32	51.94	24.50	**55.47**
MNIST	Accuracy (%)	90.27±0.26 •	47.70±0.01 •	86.03±0.41 •	**96.94±0.12**
	label ratio	4.38±0.14	1.67±0.05	2.01±0.03	**1.00±0.00**
	effectiveness	20.61	28.56	42.80	**96.94**
FASHION-MNIST	Accuracy (%)	76.50±0.61 •	44.10±0.01 •	66.57±0.47 •	**88.97±0.46**
	label ratio	3.66±0.09	1.36±0.03	1.63±0.08	**1.00±0.00**
	effectiveness	20.90	32.43	40.84	**88.97**
EMNIST	Accuracy (%)	61.44±0.51 •	53.77±0.01 •	48.33±0.97 •	**81.05±1.26**
	label ratio	2.43±0.04	1.09±0.02	1.11±0.04	**1.00±0.00**
	effectiveness	25.28	49.33	43.54	**81.05**
win/tie/loss		**4/0/0**	**4/0/0**	**4/0/0**	**N/A**
Dataset	**r=0.4**	**ECSMiner**	**SENC-MaS**	**ECHO-D**	**SIM (Ours)**
CIFAR-10	Accuracy (%)	28.13±0.15 •	49.52±0.01 •	27.40±0.18 •	**54.92±0.30**
	label ratio	2.34±0.01	1.02±0.01	1.09±0.01	**1.00±0.00**
	effectiveness	12.02	48.55	25.14	**54.92**
MNIST	Accuracy (%)	88.93±0.18 •	48.62±0.01 •	85.71±0.03 •	**96.32±0.09**
	label ratio	4.13±0.12	1.38±0.04	1.96±0.06	**1.00±0.00**
	effectiveness	21.53	35.23	43.73	**96.32**
FASHION-MNIST	Accuracy (%)	75.94±0.44 •	41.06±0.01 •	63.40±0.52 •	**90.62±0.15**
	label ratio	3.52±0.04	1.01±0.01	1.58±0.01	**1.00±0.00**
	effectiveness	21.57	40.65	40.13	**90.62**
EMNIST	Accuracy (%)	57.48±0.28 •	46.18±1.28 •	46.12±0.48 •	**85.26±0.78**
	label ratio	1.79±0.02	**0.69±0.01**	0.77±0.02	1.00±0.00
	effectiveness	32.11	66.93	59.90	**85.26**
win/tie/loss		**4/0/0**	**4/0/0**	**4/0/0**	**N/A**

5.2.2 Baselines

To examine the effectiveness of our proposed SIM, we compare it with several state-of-the-art stream classifiers:

- **ECSMiner** (Masud et al., 2011a): a fully-supervised framework that maintains an ensemble of K-Means clusters to detect novel classes and make predictions via a k-nearest neighbors classifier.

Table 5.4: Novel class detection performance over image and text streams. - indicates a failure of novel class detection, i.e., $F_{\text{new}} = 0$ and $M_{\text{new}} = 100$. We adopt $\lfloor 47 \cdot r \rfloor$ initial known classes for the EMNIST stream and $\lfloor 10 \cdot r \rfloor$ intial known classes for other streams.

Image Data Stream	**r=0.3**	**ECSMiner**	**SENC-MaS**	**ECHO-D**	**SIM (Ours)**
MNIST	M_{new}	77.90±2.23	95.10±0.01	67.58±2.12	**28.38±0.68**
	F_{new}	1.08±0.23	11.81±0.01	1.21±0.01	**0.04±0.02**
FASHION-MNIST	M_{new}	88.10±0.87	64.93±0.01	-	**61.02±1.71**
	F_{new}	1.60±0.13	48.37±0.01	-	**0.27±0.06**
EMNIST	M_{new}	-	87.90±0.01	-	**21.67±2.38**
	F_{new}	-	18.81±0.01	-	**0.47±0.04**
CIFAR-10	M_{new}	-	96.36±0.01	-	**61.03±1.71**
	F_{new}	-	1.47±0.01	-	**0.28±0.06**
Image Data Stream	**r=0.4**	**ECSMiner**	**SENC-MaS**	**ECHO-D**	**SIM (Ours)**
MNIST	M_{new}	90.27±0.56	85.47±0.01	87.13±0.50	**18.94±0.74**
	F_{new}	0.25±0.17	17.01±0.01	0.36±0.01	**0.19±0.04**
FASHION-MNIST	M_{new}	87.63±1.37	-	-	**56.89±1.08**
	F_{new}	**0.03±0.01**	-	-	0.34±0.01
EMNIST	M_{new}	-	90.88±0.43	-	**14.89±1.54**
	F_{new}	-	15.65±0.27	-	**0.08±0.01**
CIFAR-10	M_{new}	-	98.49±0.01	-	**41.88±0.21**
	F_{new}	-	1.47±0.01	-	**1.36±0.21**
Text Data Stream		**ECSMiner**	**SENC-MaS**	**ECHO-D**	**SIM (Ours)**
New York Times	M_{new}	94.59	94.91	97.88	**50.79**
	F_{new}	2.94	4.18	**0.46**	1.10
Guardian	M_{new}	95.33	96.72	-	**62.14**
	F_{new}	**0.59**	2.95	-	0.89

- **ECHO-D** (Haque et al., 2016a): a semi-supervised framework that improves ECSMiner by maintaining a cluster ensemble for classification.

- **SENC-MaS** (Mu et al., 2017b): a semi-supervised framework that dynamically maintains two low-dimensional matrix sketches to detect novel classes and classify instances of known classes.

- **SIM-NM**: a semi-supervised variant of our SIM framework with no integral similarity metric. In other words, we optimize $\mathcal{L}_{class}$ rather than $\mathcal{L}_{overall}$.

Table 5.5: Classification and novel class detection performance of SIM-NM and SIM over image streams with $r = 0.3$. $\circ$/$\bullet$ indicates that SIM performs statistically worse/better than SIM-NM (0.05 significance level).

Method		**MNIST**	**FASHION-MNIST**	**EMNIST**	**CIFAR-10**
Accuracy (%)	**SIM-NM**	87.72±0.02 $\bullet$	80.92±0.01 $\bullet$	76.96±0.01 $\bullet$	45.53±0.14 $\bullet$
	SIM	**96.94±0.12**	**88.97±0.46**	**81.05±1.26**	**55.47±0.66**
label ratio	**SIM-NM**	1.35±0.01	1.41±0.03	1.03±0.02	**0.95±0.02**
	SIM	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	1.00±0.00
effectiveness	**SIM-NM**	64.98	57.39	74.72	47.93
	SIM	**96.94**	**88.97**	**81.05**	**55.47**
M_{new}	**SIM-NM**	62.36±1.94 $\bullet$	80.58±1.26 $\bullet$	53.50±1.26 $\bullet$	67.62±0.70 $\bullet$
	SIM	**28.38±0.68**	**61.02±1.71**	**21.67±2.38**	**61.03±1.71**
F_{new}	**SIM-NM**	**0.01±0.01**	**0.02±0.01** $\circ$	**0.01±0.01** $\circ$	2.80±0.03 $\bullet$
	SIM	0.04±0.02	0.27±0.06	0.47±0.04	**0.28±0.06**

Table 5.6: Classification performance of competing methods over text streams.

Methods	**New York Times**			**Guardian**		
	Accuracy (%)	*label ratio*	*effectiveness*	Accuracy (%)	*label ratio*	*effectiveness*
ECSMiner	29.56	2.21	13.38	30.17	2.62	11.52
SENC-MaS	39.79	**0.79**	50.37	45.15	1.02	44.26
ECHO-D	43.34	0.80	54.18	49.02	**0.96**	51.06
SIM (Ours)	**57.95**	1.00	**57.95**	**53.74**	1.00	**53.74**

5.2.3 Experiment Setup

The proposed framework SIM is implemented via *Python* 3.6.2 and *Pytorch* 0.4.0 library. The network parameters are randomly initialized based on *Pytorch*'s internal random process. For all baselines except SENC-MaS, we adopt the official implementation provided by corresponding authors. Since there is no fully functional implementation of SENC-MaS provided by the author, we implement it by ourselves based on the description in the published paper (Mu et al., 2017b). The hyper-parameters of these baselines are initially set based on values reported by the authors and then fine-tuned via cross-validation on the initialization dataset. In our method SIM, we set $n = 200$, $S_{\mathcal{B}} = 1000$, $S_{\mathcal{D}} = 200$, $S_{update} = 100$, $\mathcal{T}_{\mathcal{D}} = 0.99$, $\gamma = 1.0$, $S_{mini} = 64$, $N_{mini} = 2000$ and $n_{epoch} = 5$ as default. The initial training dataset size is 1000 per class.

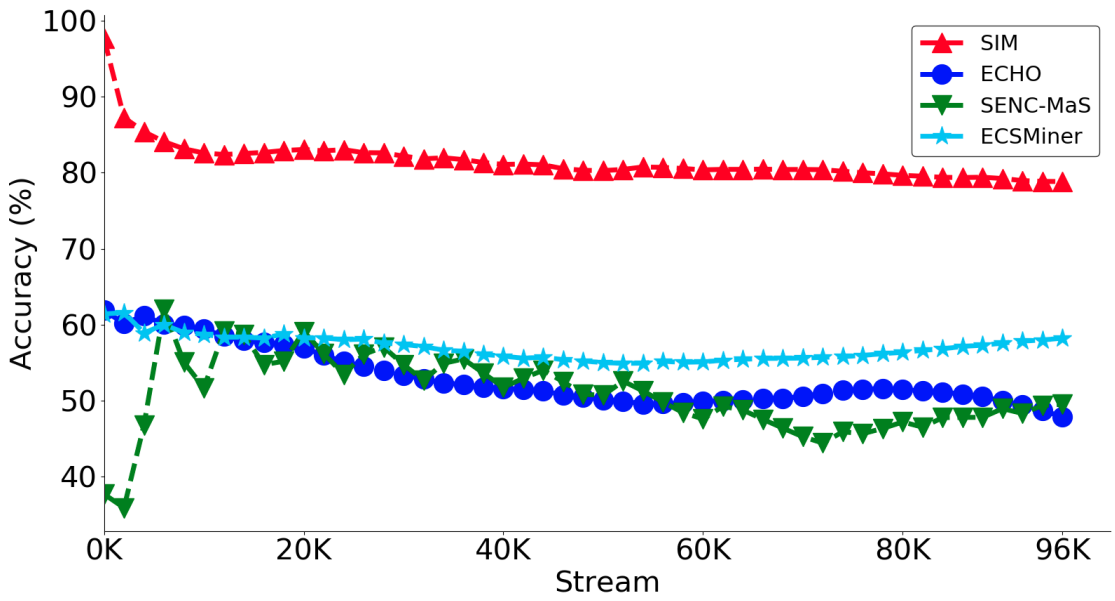


Figure 5.4: Classification accuracy of competing methods over the EMNIST image stream ($r = 0.3$).

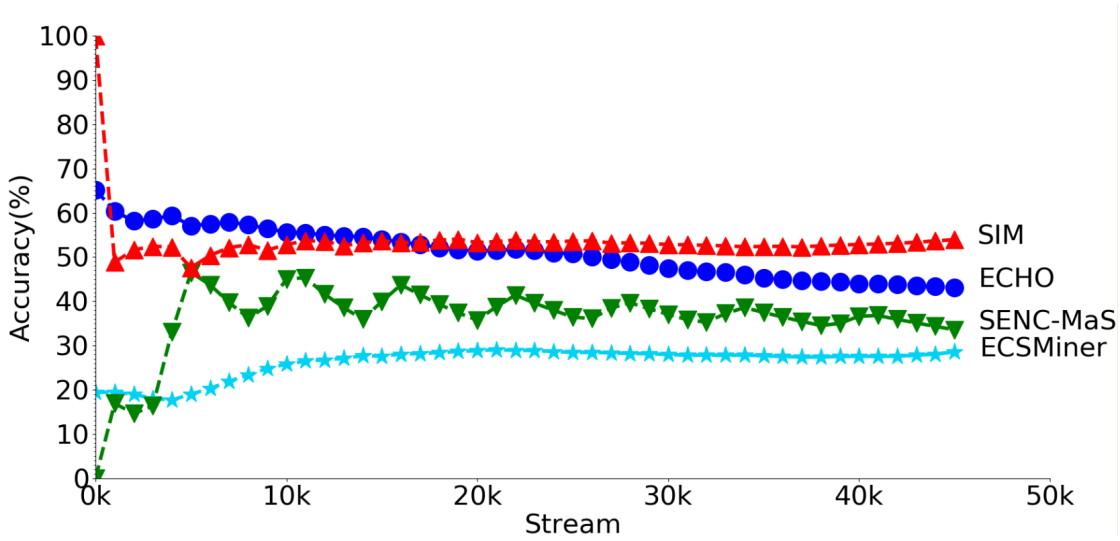


Figure 5.5: Classification accuracy of competing methods over the New York Times text stream.

5.2.4 Evaluation Metrics

Suppose FN = the number of instances belonging to novel classes that are misclassified as existing classes, FP = the number of instances belonging to existing classes that are misclassified as novel classes, F_e = the total number of misclassified existing class instances

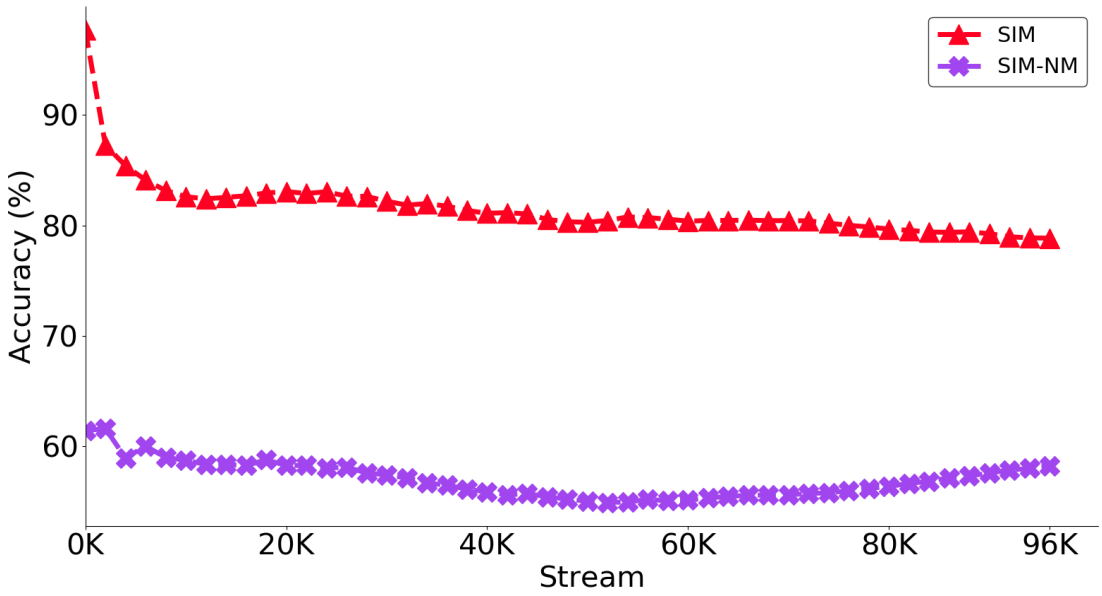


Figure 5.6: Classification accuracy of SIM-NM and SIM over the EMNIST image stream ($r = 0.3$).

(other than FP), N_c = the total number of novel class instances, and N = the total number of instances in the stream. The following widely-accepted metrics (Masud et al., 2010, 2009b; Al-Khateeb et al., 2012b; Masud et al., 2011b) are used for evaluation.

- $Accuracy(\%) = 100 - \frac{(FP+FN+F_e)\times 100}{N}$
- $label\ ratio = \frac{\%\ of\ true\ labels\ requested\ by\ M}{\%\ of\ true\ labels\ requested\ by\ \text{SIM}}$, where M denotes a method in {ECSMiner, SENC-MaS, ECHO, SIM-NM, SIM}.
- $M_{\text{new}} = \frac{FN*100}{N_c}$: fraction of novel class instances misclassified as belonging to existing classes.
- $F_{\text{new}} = \frac{FP*100}{N-N_c}$: fraction of existing class instances misclassified as belonging to novel classes.
- $effectiveness = \frac{Accuracy(\%)}{label\ ratio}$: normalized accuracy, which measures the model performance if the model uses the same amount of ground truth labels as SIM.

In the experiments, we evaluate these metrics in a prequential manner. A model with a lower M_{new}, F_{new} or *label ratio* has a better performance. On the other hand, the higher the *Accuracy(%)* or *effectiveness*, the better the model.

5.2.5 Results

Stream Classification

For both $r = 0.3$ and $r = 0.4$, we simulate different streams and conduct 10 independent experiments on each benchmark image dataset. In addition, we conduct another experiment in the New York Times and Guardian text streams. The classification results on image and text streams are listed in Table 5.3 and Table 5.6 respectively. Note that we do not report variance information for the results in Table 5.6, since articles or stories in text streams have a fixed chronological order. As shown in the results, SENC-MaS underperforms its competitors in most streams due to its linear classifier. ECSMiner provides better performance than ECHO-D because it is optimized for supervised learning and observes far more information than the latter. Most importantly, SIM performs best among the competing methods by providing the highest accuracy and effectiveness in most streams. It indicates that SIM utilizes the training data from streams in a more efficient way compared to baselines. Figure 5.4 and Figure 5.5 illustrates how the classification performance of competing methods evolve along the EMNIST image stream ($r = 0.3$) and the New York Times text stream respectively. In contrast to existing solutions, SIM detects the occurrence of novel classes quickly and performs stably along the streams. Similar results are also observed in other simulated and real-world streams. The good performance of SIM mainly comes from its integral similarity metric learned via multi-task learning, which actively finds a latent feature space suitable for both classification and novel class detection (demonstrated in Section 5.2.5).

Novel Class Detection

The comparison of novel class detection performance between SIM and the baseline methods on each stream is listed in Table 5.4. Apparently, ECSMiner and ECHO-D fail to detect any novel class instance in most image streams. On the other hand, although SENC-MaS detects some novel class instances, it misses most of such instances and produces lots of false alarms. These methods perform poorly because they rely on the intrinsic cohesion and separation assumption that is invalid in complex high-dimensional streams to detect novel class instances. In these streams, instances from known classes are further away than those from novel classes in the observed feature space, making detection difficult or sometimes even impossible. SIM overcomes this issue by actively searching for a latent feature space suitable for both classification and novel class detection, leading to the lowest M_{new} and F_{new} among competing methods.

Metric Impact

An important question that remains to be answered is how the metrics learned in SIM affects its performance? To study it, we introduce a new baseline SIM-NM, which is a variant of SIM with no integral similarity metric. The classification performance of SIM and SIM-NM on the simulated image streams with $r = 0.3$ are reported in Table 5.5. In addition, how the performance of SIM and SIM-NM evolve along the EMNIST image stream ($r = 0.3$) is shown in Figure 5.6. Compared to SIM-NM, SIM provides much higher classification accuracy, significantly lower M_{new}, and lower or same level of F_{new}. Moreover, it detects the occurrence of novel classes much faster than SIM-NM, indicating its better capability of capturing inter-class dissimilarity. Similar results are observed in other streams of different datasets and with different r values. Remind that the only difference between SIM-NM and SIM is the integral similarity metric. Therefore, we are confident to conclude that this

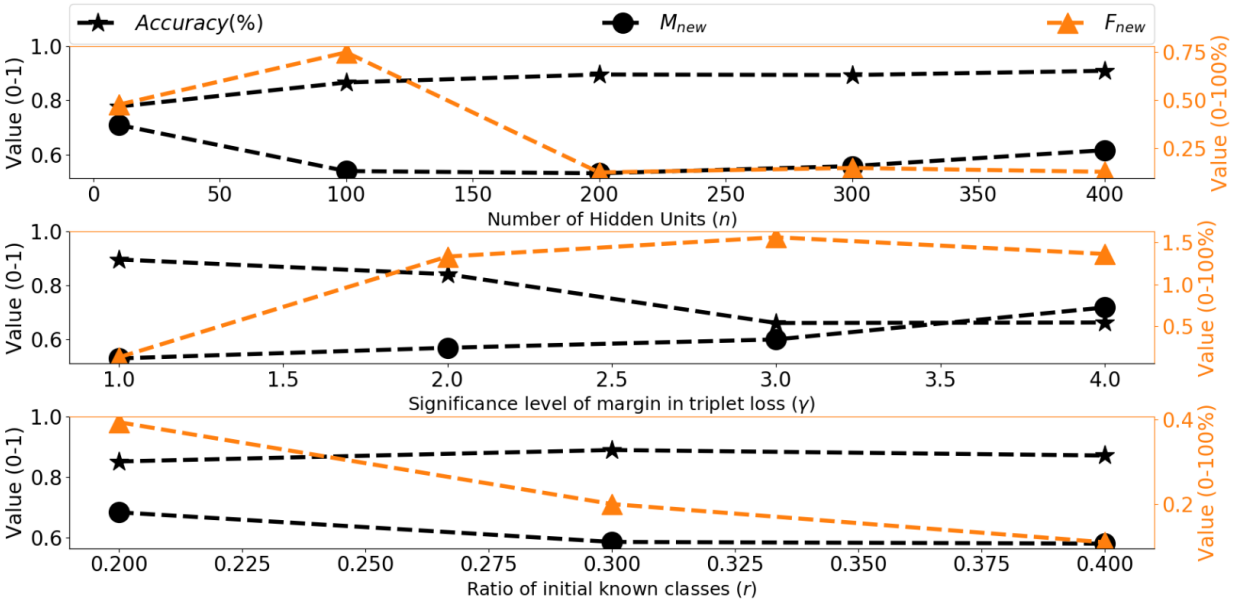


Figure 5.7: Sensitivity analysis of SIM. n, γ, and r are the number of hidden units in the network, the significance level of triplet loss, and the ratio of initially known classes respectively.

similarity metric is critical for improving both the novel class detection and classification performance over data streams.

Sensitivity of Parameters

A hyper-parameter sensitivity analysis is also conducted for SIM and the results are shown in Figure 5.7. The three main hyper-parameters in SIM are n (the number of hidden units in the network), γ (the significance level of the adaptive-bound triplet loss $\mathcal{L}_{triplet}$), and r (the ratio of initially known classes). According to the results, if the network is simple, i.e., n is relatively small, the performance of SIM significantly degrades with much lower accuracy and significantly higher M_{new} as well as F_{new}. On the other hand, a large n produces a complex network that is easily overfitted and declines the model performance. Similarly, with a high γ, SIM tends to excessively push different classes far away from each other and hence overfits on the training data. As a result, we choose to set $n = 200$ and $\gamma = 1.0$ that are neither too big nor too small. Note that SIM is robust to the number of initially known

classes by providing almost equal performance with respect to different r values ($r \geq 0.3$), which is desired for stream classifiers.

5.3 Discussion

In this chapter, we describe the proposed semi-supervised stream classifier SIM in details, which is able to perform classification on open-ended data distribution. Compared to existing solutions, SIM improves both the classification and novel class detection performance by actively searching for a latent feature space via metric learning where the intrinsic cohesion and separation data property holds. Thus, it is more suitable for classifying complex high-dimensional real-world data streams. Note that SIM focuses on addressing the concept evolution problem and ignores the concept drift issue in data streams. Fortunately, extending SIM to handle both concept evolution and concept drift is easy. We leave it for future work.

CHAPTER 6

CONCLUSION AND FUTURE WORK [1] [2] [3]

In this chapter, we will draw a conclusion and discuss our future work. We will start with our online metric learning framework OAHU that addresses the "Online Adaptive Metric Learning" open challenge, and later focus on the SetConv and SIM frameworks that enhance classification performance under the class imbalance and open-world assumptions.

6.1 Online Adaptive Metric Learning

We developed an online metric learning method that learns a ANN-based metric from a stream of triplet constraints. It achieves full constraint utilization, and more importantly, is able to automatically adjust the model complexity to accomodate the input constraints when necessary. Specifically, we first attach an independent metric embedding layer (MEL) to every hidden layer of an Artificial Neural Network (ANN). The output of a hidden layer is hence the input to its corresponding MEL. Each MEL characterizes a latent feature space where similar instances are close to each other while dissimilar instances are far away from each other, and is associated with a metric weight which measures its importance in the

[1] This chapter contains material previously published as: Yang Gao, Yi-Fan Li, Yu Lin, Latifur Khan. "SetConv: A New Approach for Learning from Imbalanced Data", In *proceedings of The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1284-1294. 2020. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

[2] This chapter contains material previously published as: Yang Gao, Yi-Fan Li, Swarup Chandra, Latifur Khan, and Bhavani Thuraisingham. "Towards self-adaptive metric learning on the fly." In *The World Wide Web Conference (WWW)*, pp. 503-513. 2019, https://doi.org/10.1145/3308558.3313503. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

[3] This chapter contains material previously published as: ©2019 IEEE. Reprinted, with permission, from Yang Gao, Yi-Fan Li, Bo Dong, Yu Lin, and Latifur Khan. "SIM: Open-World Multi-Task Stream Classifier with Integral Similarity Metrics." In *IEEE International Conference on Big Data (Big Data)*, pp. 751-760. December, 2019. Lead author, Yang Gao, conducted the majority of the research, including the full writing, the full design, the full implementation, and most of the evaluation.

entire metric model. In other words, our method is indeed an ensemble of metric models with various complexities sharing the low-level knowledge. Then, a novel Adaptive-Bound Triplet Loss (ABTL) is proposed to eliminate the dependency of existing methods on those "hard" constraints, which improves the constraint utilization. Finally, we introduce a Adaptive Hedge Update (AHU) method to optimize the metric models in the ensemble as well as their associated metric weights. The experiments on various applications such as real-world image classification, facial verification and image retrieval demonstrates the effectiveness and efficacy of our proposed method.

Our key contribution in this work is redesigning the common ANN architecture to allow the model adapt its complexity when necessary, and introducing ABTL to improve the constraint utilization rate. The flexibility of model capacity helps to avoid the common under-fitting and over-fitting issues in online metric learning and significantly improves the model performance, which is demonstrated by the superiority of our method compared to the state-of-the-art solutions.

Similar to existing solutions, our method may still fail in cases where severe concept drift exists. For example, in facial verification applications, our method can successfully identify that two images belong to the same person at the beginning, but may fail to do that when this person gets older. Therefore, an important direction of our future research is to extend our work to handle concept drift. Lifelong machine learning requires the model to accumulate the knowledge learned in the past and use the knowledge to help future learning and problem solving with possible adaptations. Combining life-long machine learning with our work is a promising solution to this issue.

6.2 Imbalanced Classification

In this study, we propose a novel set convolution operation SetConv and a new training strategy, episodic training, to assist learning from imbalanced class distributions. To cap-

ture the semantic similarity and dissimilarity relation among data, the SetConv operation directly estimates the convolution kernel weights based on the intra-class and inter-class correlations, and applies the learned kernels to extract discriminative features from data. These features are then compressed into a single class representative used for classification. In this way, SetConv helps the model focus on the latent concept not only common to different samples of the same class but also discriminative to other classes. In episodic training, we assign equal weights to different classes in spite of their relative sizes and do not perform resampling on data. At each iteration during training, the model is fed with an episode formed by a set of samples where the class imbalance ratio is preserved. It is hence encouraged to extract discriminative features even when class distribution is unbalanced. Our proposed method has several advantages including data-sensitive convolution, automatic class balancing and no dependence on unknown prior knowledge. Experiments on incident detection and sentiment classification tasks demonstrate the superiority of our method compared to state-of-the-art baselines.

Unfortunately, our proposed method is not suitable for processing high-dimensional sparse data, as the massive 0s in these data may lead to close-to-zero convolution kernel and limits the expressiveness of our model. Combining sparse deep learning techniques with SetConv is a potential solution to this problem.

6.3 Open-World Classification

In this work, we propose a new stream classifier SIM that is able to make predictions under concept evolution. In particular, we propose a Multi-Task Open-World Classifier (MTOWC) that consists of a feature transformation block and an open-world classifier. The feature transformation block performs metric learning to actively search for a latent feature space in which instances from the same class are closer than those of different classes. The open-world classifier contains a one-vs-rest binary classifier for each of the known classes. It

takes those instances projected to the learned latent feature space as input, and is trained to minimize classification error of the known classes. By incorporating a novel class detection mechanism, the MT-OWC can easily detect those instances from unknown classes. Thus, the metric learning task actually serves as an auxiliary task to both the classification and novel class detection tasks.

Currently, our proposed method focuses on addressing the concept evolution problem in stream classification. Besides concept evolution, another significant challenge in open-world classification is concept drift, which means that the underlying concept of data changes over time. For future work, we will extend SIM to handle both concept evolution and concept drift issues. Fortunately, only a minor modification to the novel class detection mechanism has to be performed to combine existing concept drift solutions with our method.

6.4 Other Future Work

6.4.1 Multi-Label Image Retrieval

In many real-world image search or recommendation applications, a significant amount of images contain different objects and are hence labeled as belonging to multiple categories. For example, as shown in Figure 6.1a and Figure 6.1b, those images of dining rooms usually contain multiple objects including ceiling lamp, dining table, and several dining chairs. A user searching for "dining table" may also be interested in those dining chairs with consistent style to a particular dining table. Hence, these images should be also included in the search results returned to the user. Another example is the outfit recommendation system in e-commerce. If a user has bought a handbag, it is very likely that she will be interested in other outfit components that match the handbag style, e.g., jackets, jeans, and shoes. Figure 6.1c and Figure 6.1d provide two typical examples. Apparently, recommending outfit images to users can significantly improve user experience in this case.

(a) (b)

(c) (d)

Figure 6.1: Examples of multi-label image retrieval.

However, how to define and precisely measure the similarity between these images is still unclear. A typical solution is to measure the similarity between every pair of objects

Figure 6.2: Example of 3D object retrieval.

and obtain the overall similarity score by computing the weighted average of these pairwise similarities. A self-attention module may be incorporated to automatically learn to compute the importance score for each pair of objects. We leave this project for future work.

6.4.2 3D Object Retrieval

The rising prosperity of VR/AR applications has placed more demands on models that are capable of understanding 3D objects. A typical example is the 3D object retrieval, which is widely applicable in modern advertisement and search techniques. Given 2D images with various viewpoints and illumination configurations, we'd like to retrieve its corresponding 3D object from a large-scale object database. Furthermore, it is also possible to retrieve a 3D object that has characteristics of all query images. For example, as shown in Figure 6.2,

given an office chair with a back, a bar chair with a circular base, and a sofa with a seat, we'd like to retrieve 3D chair objects with part or all of the characteristics in query images and rank the results in the order so that the best-matched object appears in front of the ranking list. In other words, the 2D images propose the retrieval requirements to the query. To realize it, the model has to learn a similarity function that matches 2D images with its 3D counterparts. We will work on this project in the future.

REFERENCES

Al-Khateeb, T., M. M. Masud, L. Khan, C. C. Aggarwal, J. Han, and B. M. Thuraisingham (2012a). Stream classification with recurring and novel class detection using class-based ensemble. In M. J. Zaki, A. Siebes, J. X. Yu, B. Goethals, G. I. Webb, and X. Wu (Eds.), *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pp. 31–40. IEEE Computer Society.

Al-Khateeb, T., M. M. Masud, L. Khan, C. C. Aggarwal, J. Han, and B. M. Thuraisingham (2012b). Stream classification with recurring and novel class detection using class-based ensemble. In *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pp. 31–40.

Ali, A., S. M. Shamsuddin, A. L. Ralescu, et al. (2015). Classification with class imbalance problem: a review. *Int. J. Advance Soft Compu. Appl 7*(3), 176–204.

Barua, S., M. M. Islam, X. Yao, and K. Murase (2014). Mwmote-majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Trans. Knowl. Data Eng. 26*(2), 405–425.

Bellet, A., A. Habrard, and M. Sebban (2013). A survey on metric learning for feature vectors and structured data. *CoRR abs/1306.6709*.

Blitzer, J., M. Dredze, and F. Pereira (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In J. A. Carroll, A. van den Bosch, and A. Zaenen (Eds.), *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Cao, P., D. Zhao, and O. R. Zaïane (2013a). An optimized cost-sensitive SVM for imbalanced data learning. In J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu (Eds.), *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II*, Volume 7819 of *Lecture Notes in Computer Science*, pp. 280–292. Springer.

Cao, P., D. Zhao, and O. R. Zaïane (2013b). A pso-based cost-sensitive neural network for imbalanced data classification. In J. Li, L. Cao, C. Wang, K. C. Tan, B. Liu, J. Pei, and V. S. Tseng (Eds.), *Trends and Applications in Knowledge Discovery and Data Mining - PAKDD 2013 International Workshops: DMApps, DANTH, QIMIE, BDM, CDA, CloudSD, Gold Coast, QLD, Australia, April 14-17, 2013, Revised Selected Papers*, Volume 7867 of *Lecture Notes in Computer Science*, pp. 452–463. Springer.

Chawla, N. V., K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer (2002). SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res. 16*, 321–357.

Chechik, G., V. Sharma, U. Shalit, and S. Bengio (2010a). Large scale online learning of image similarity through ranking. *J. Mach. Learn. Res. 11*, 1109–1135.

Chechik, G., V. Sharma, U. Shalit, and S. Bengio (2010b). Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research 11*, 1109–1135.

Chen, C. and M. Shyu (2011). Clustering-based binary-class classification for imbalanced data sets. In *Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2011, 3-5 August 2011, Las Vegas, Nevada, USA*, pp. 384–389. IEEE Systems, Man, and Cybernetics Society.

Chen, T., I. J. Goodfellow, and J. Shlens (2015). Net2net: Accelerating learning via knowledge transfer. *CoRR abs/1511.05641*.

Cohen, G., S. Afshar, J. Tapson, and A. van Schaik (2017a). EMNIST: an extension of MNIST to handwritten letters. *CoRR abs/1702.05373*.

Cohen, G., S. Afshar, J. Tapson, and A. van Schaik (2017b). Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*.

Devlin, J., M. Chang, K. Lee, and K. Toutanova (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics.

Díaz-Vico, D., A. R. Figueiras-Vidal, and J. R. Dorronsoro (2018). Deep mlps for imbalanced classification. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pp. 1–7. IEEE.

Ester, M., H.-P. Kriegel, J. Sander, X. Xu, et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, Volume 96, pp. 226–231.

Freund, Y. and R. E. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci. 55*(1), 119–139.

Galar, M., A. Fernández, E. B. Tartas, H. B. Sola, and F. Herrera (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Syst. Man Cybern. Part C 42*(4), 463–484.

Gao, Y., Y. Li, S. Chandra, L. Khan, and B. M. Thuraisingham (2019). Towards self-adaptive metric learning on the fly. In L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia (Eds.), *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 503–513. ACM.

Gao, Y., Y.-F. Li, Y. Lin, C. Aggarwal, and L. Khan (2020, November). SetConv: A New Approach for Learning from Imbalanced Data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 1284–1294. Association for Computational Linguistics.

Gülçehre, Ç., M. Moczulski, F. Visin, and Y. Bengio (2016). Mollifying networks. *CoRR abs/1608.04980*.

Haque, A., L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal (2016a). Efficient handling of concept drift and concept evolution over stream data. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*, pp. 481–492. IEEE.

Haque, A., L. Khan, M. Baron, B. M. Thuraisingham, and C. C. Aggarwal (2016b). Efficient handling of concept drift and concept evolution over stream data. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pp. 481–492. IEEE Computer Society.

He, H. and E. A. Garcia (2009). Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng. 21*(9), 1263–1284.

He, R. and J. J. McAuley (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, and B. Y. Zhao (Eds.), *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pp. 507–517. ACM.

Hite, K. C., W. S. Ciciora, T. Alison, R. G. Beauregard, et al. (1998, June 30). System and method for delivering targeted advertisements to consumers. US Patent 5,774,170.

Huang, G., Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger (2016). Deep networks with stochastic depth. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pp. 646–661.

Huang, K. and H. Lin (2017). Cost-sensitive label embedding for multi-label classification. *Mach. Learn. 106*(9-10), 1725–1746.

Huo, Z., F. Nie, and H. Huang (2016). Robust and effective metric learning using capped trace norm: Metric learning via capped trace norm. In B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi (Eds.), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1605–1614. ACM.

Jain, P., B. Kulis, I. S. Dhillon, and K. Grauman (2008a). Online metric learning and fast similarity search. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (Eds.), *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 761–768. Curran Associates, Inc.

Jain, P., B. Kulis, I. S. Dhillon, and K. Grauman (2008b). Online metric learning and fast similarity search. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 761–768.

Jégou, H., M. Douze, and C. Schmid (2011). Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell. 33*(1), 117–128.

Jin, R., S. Wang, and Y. Zhou (2009a). Regularized distance metric learning: Theory and algorithm. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta (Eds.), *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pp. 862–870. Curran Associates, Inc.

Jin, R., S. Wang, and Y. Zhou (2009b). Regularized distance metric learning: Theory and algorithm. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pp. 862–870.

Krause, J., M. Stark, J. Deng, and L. Fei-Fei (2013). 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Prog. Artif. Intell. 5*(4), 221–232.

Krawczyk, B., M. Wozniak, and G. Schaefer (2014). Cost-sensitive decision tree ensembles for effective imbalanced classification. *Appl. Soft Comput. 14*, 554–562.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A. and G. Hinton (2009). Learning multiple layers of features from tiny images.

Kumar, N., A. C. Berg, P. N. Belhumeur, and S. K. Nayar (2009). Attribute and simile classifiers for face verification. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pp. 365–372.

Last, F., G. Douzas, and F. Bação (2017). Oversampling for imbalanced learning based on k-means and SMOTE. *CoRR abs/1711.00837*.

Learned-Miller, G. B. H. E. (2014, May). Labeled faces in the wild: Updates and new reporting procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst.

LeCun, Y., C. Cortes, and C. Burges (2010). Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist 2*.

Li, W., Y. Gao, L. Wang, L. Zhou, J. Huo, and Y. Shi (2018a). OPML: A one-pass closed-form solution for online metric learning. *Pattern Recognit. 75*, 302–314.

Li, W., Y. Gao, L. Wang, L. Zhou, J. Huo, and Y. Shi (2018b). OPML: A one-pass closed-form solution for online metric learning. *Pattern Recognition 75*, 302–314.

Li, Y., Y. Gao, G. Ayoade, H. Tao, L. Khan, and B. M. Thuraisingham (2019). Multistream classification for cyber threat data with heterogeneous feature space. In L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, and L. Zia (Eds.), *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 2992–2998. ACM.

Masi, I., Y. Wu, T. Hassner, and P. Natarajan (2018). Deep face recognition: A survey. In *31st SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI 2018, Paraná, Brazil, October 29 - Nov. 1, 2018*, pp. 471–478. IEEE Computer Society.

Masud, M., J. Gao, L. Khan, J. Han, and B. M. Thuraisingham (2011a). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering 23*(6), 859–874.

Masud, M. M., T. Al-Khateeb, L. Khan, C. C. Aggarwal, J. Gao, J. Han, and B. M. Thuraisingham (2011a). Detecting recurring and novel classes in concept-drifting data streams. In D. J. Cook, J. Pei, W. Wang, O. R. Zaïane, and X. Wu (Eds.), *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pp. 1176–1181. IEEE Computer Society.

Masud, M. M., T. Al-Khateeb, L. Khan, C. C. Aggarwal, J. Gao, J. Han, and B. M. Thuraisingham (2011b). Detecting recurring and novel classes in concept-drifting data streams. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pp. 1176–1181.

Masud, M. M., Q. Chen, L. Khan, C. C. Aggarwal, J. Gao, J. Han, and B. M. Thuraisingham (2010). Addressing concept-evolution in concept-drifting data streams. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pp. 929–934.

Masud, M. M., J. Gao, L. Khan, J. Han, and B. M. Thuraisingham (2009a). Integrating novel class detection with classification for concept-drifting data streams. In W. L. Buntine, M. Grobelnik, D. Mladenic, and J. Shawe-Taylor (Eds.), *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part II*, Volume 5782 of *Lecture Notes in Computer Science*, pp. 79–94. Springer.

Masud, M. M., J. Gao, L. Khan, J. Han, and B. M. Thuraisingham (2009b). Integrating novel class detection with classification for concept-drifting data streams. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part II*, pp. 79–94.

Masud, M. M., J. Gao, L. Khan, J. Han, and B. M. Thuraisingham (2011b). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Trans. Knowl. Data Eng. 23*(6), 859–874.

Mu, X., F. Zhu, J. Du, E. Lim, and Z. Zhou (2017a). Streaming classification with emerging new class by class matrix sketching. In S. P. Singh and S. Markovitch (Eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 2373–2379. AAAI Press.

Mu, X., F. Zhu, J. Du, E. Lim, and Z. Zhou (2017b). Streaming classification with emerging new class by class matrix sketching. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 2373–2379.

Rabanser, S., O. Shchur, and S. Günnemann (2017). Introduction to tensor decompositions and their applications in machine learning. *CoRR abs/1711.10781*.

Rosenthal, S., N. Farra, and P. Nakov (2017, August). SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Vancouver, Canada. Association for Computational Linguistics.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*(3), 211–252.

Schulz, A., C. Guckelsberger, and F. Janssen (2017). Semantic abstraction for generalization of tweet classification: An evaluation of incident-related tweets. *Semantic Web 8*(3), 353–372.

Shalev-Shwartz, S., Y. Singer, and A. Y. Ng (2004a). Online and batch learning of pseudo-metrics. In C. E. Brodley (Ed.), *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, Volume 69 of *ACM International Conference Proceeding Series*. ACM.

Shalev-Shwartz, S., Y. Singer, and A. Y. Ng (2004b). Online and batch learning of pseudo-metrics. In *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*.

Shu, L., H. Xu, and B. Liu (2017a). DOC: deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 2911–2916.

Shu, L., H. Xu, and B. Liu (2017b). DOC: deep open classification of text documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pp. 2911–2916.

Smith, M. R., T. R. Martinez, and C. G. Giraud-Carrier (2014). An instance level analysis of data complexity. *Mach. Learn. 95*(2), 225–256.

Sobhani, P., H. Viktor, and S. Matwin (2014). Learning from imbalanced data using ensemble methods and cluster-based undersampling. In A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, and Z. W. Ras (Eds.), *New Frontiers in Mining Complex Patterns - Third International Workshop, NFMCP 2014, Held in Conjunction with ECML-PKDD 2014, Nancy, France, September 19, 2014, Revised Selected Papers*, Volume 8983 of *Lecture Notes in Computer Science*, pp. 69–83. Springer.

Srinivas, S. and R. V. Babu (2016). Learning neural network architectures using backpropagation. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*.

Sun, Y., M. S. Kamel, A. K. C. Wong, and Y. Wang (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition 40*(12), 3358–3378.

Triguero, I., M. Galar, D. Merino, J. Maillo, H. Bustince, and F. Herrera (2016). Evolutionary undersampling for extremely imbalanced big data classification under apache spark. In *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016*, pp. 640–647. IEEE.

Wang, N., X. Zhao, Y. Jiang, and Y. Gao (2018). Iterative metric learning for imbalance data classification. In J. Lang (Ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pp. 2805–2811. ijcai.org.

Wang, S., L. L. Minku, and X. Yao (2015). Resampling-based ensemble methods for online class imbalance learning. *IEEE Trans. Knowl. Data Eng. 27*(5), 1356–1368.

Wu, F., X. Jing, S. Shan, W. Zuo, and J. Yang (2017). Multiset feature learning for highly imbalanced data classification. In S. P. Singh and S. Markovitch (Eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 1583–1589. AAAI Press.

Xiao, H., K. Rasul, and R. Vollgraf (2017a). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xiao, H., K. Rasul, and R. Vollgraf (2017b). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yin, L., Y. Ge, K. Xiao, X. Wang, and X. Quan (2013). Feature selection for high-dimensional imbalanced data. *Neurocomputing 105*, 3–11.

Zhai, A., H. Wu, E. Tzeng, D. H. Park, and C. Rosenberg (2019). Learning a unified embedding for visual search at pinterest. In A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis (Eds.), *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 2412–2420. ACM.

Zhang, Y., P. Pan, Y. Zheng, K. Zhao, Y. Zhang, X. Ren, and R. Jin (2018). Visual search at alibaba. In Y. Guo and F. Farooq (Eds.), *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 993–1001. ACM.

Zhang, Z., Y. Li, Z. Zhang, C. Jin, and M. Gao (2018a). Adaptive matrix sketching and clustering for semisupervised incremental learning. *IEEE Signal Process. Lett. 25*(7), 1069–1073.

Zhang, Z., Y. Li, Z. Zhang, C. Jin, and M. Gao (2018b). Adaptive matrix sketching and clustering for semisupervised incremental learning. *IEEE Signal Process. Lett. 25*(7), 1069–1073.

Zheng, Z., Y. Cai, and Y. Li (2015). Oversampling method for imbalanced classification. *Comput. Informatics 34*(5), 1017–1037.

Zheng, Z., X. Wu, and R. K. Srihari (2004). Feature selection for text categorization on imbalanced data. *SIGKDD Explor. 6*(1), 80–89.

Zhou, W., H. Li, and Q. Tian (2017). Recent advance in content-based image retrieval: A literature survey. *CoRR abs/1706.06064*.

BIOGRAPHICAL SKETCH

Yang Gao received his Bachelor of Science degree in physics from the University of Science and Technology of China in 2012 and his Master of Science degree in physics from The University of Texas at Dallas in 2015. In the same year, he decided to engage in advanced studies to further his knowledge in machine learning and data mining. He was privileged to join The University of Texas at Dallas to pursue his PhD degree in computer science. During his PhD studies, Yang was a member of the Big Data Analytics and Management Lab and worked as a research assistant under the supervision of Prof. Latifur Khan. His research interests are metric learning, representation learning, and transfer learning.

CURRICULUM VITAE

Yang Gao

February, 2021

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: yxg122530@utdallas.edu

Educational History:

BS, Physics, University of Science and Technology of China, 2012
MS, Physics, University of Texas At Dallas, 2015
PhD, Computer Science, University of Texas At Dallas, 2021

Enhancing Classification and Retrieval Performance by Mining Semantic Similarity Relation from Data
PhD Dissertation
Computer Science Department, The University of Texas At Dallas
Advisors: Dr. Latifur Khan

Employment History:

Software Engineer Intern, Facebook, Inc, May 2020 – August 2020
Machine Learning Research Intern, NEC Laboratories America, Inc. May 2019 – August 2019
Research Assistant, The University of Texas at Dallas, September 2017 – present

Technical Skills:

Programming: Python, Java, C/C++, Hack
Deep Learning: PyTorch, TensorFlow
Big Data Analytics: Scipy, Numpy, Pandas, Scikit-Learn, NLTK, Hadoop, Spark, Kafka
Database & Visualization: MySQL, Matplotlib, Kibana