Adaptive Image Stream Classification via Convolutional Neural Network with Intrinsic Similarity Metrics

Yang Gao
University of Texas at Dallas
Dallas, Texas, USA
yxg122530@utdallas.edu

Zhuoyi Wang
University of Texas at Dallas
Dallas, Texas, USA
Zhuoyi.Wang1@utdallas.edu

Swarup Chandra
University of Texas at Dallas
Dallas, Texas, USA
swarup.chandra@utdallas.edu

Latifur Khan
University of Texas at Dallas
Dallas, Texas, USA
lkhan@utdallas.edu

ABSTRACT
When performing data classification over a stream of continuously occurring instances, a key challenge is to develop an open-world classifier that anticipates instances from an unknown class. Studies addressing this problem, typically called novel class detection, have considered classification methods that reactively adapt to such changes along the stream. Importantly, they rely on the property of cohesion and separation among instances in feature space. Instances belonging to the same class are assumed to be closer to each other (cohesion) than those belonging to different classes (separation). Unfortunately, this assumption may not have large support when dealing with high dimensional data such as images. In this paper, we address this key challenge by proposing a semi-supervised multi-task learning framework called CSIM which aims to intrinsically search for a latent space suitable for detecting labels of instances from both known and unknown classes. Particularly, we utilize a convolution neural network layer that aids in the learning of a latent feature space suitable for novel class detection. We empirically measure the performance of CSIM over multiple real-world image datasets and demonstrate its superiority by comparing its performance with existing semi-supervised methods.

CCS CONCEPTS
• Information systems → Data streams; Data stream mining;
• Computing methodologies → Neural networks;

KEYWORDS
Stream Classification, Novel Class Detection, Metric Learning, Multi-Task Learning.

1 INTRODUCTION
A stream of data typically results from applications such as social networks, online business transactions, news-feeds etc. Recent studies have attempted to address the infinite length challenge by employing a fixed-size sliding window to perform analytics[3, 4, 19]. In particular, the data sources are assumed to be non-stationary whose data distribution changes over time. This property directly affects a trained classifier. Therefore, a reactive mechanism is typically used where a change is first detected and then appropriate actions to adapt the classifier are considered.

In this paper, we focus on another key problem called concept evolution. Here, instances from previously unobserved classes (called novel classes) may occur along the stream. For example, images associated with classes for which the current classifier is not trained may appear along the stream during evaluation. If the classifier fails to account for the emerging classes, its performance would degrade.

Recent studies [11, 16] have leveraged unsupervised clustering mechanisms, such as K-Means, over the observed feature space for detecting instances from novel classes. Here, clusters of instances represent regions in feature space containing instances of the same class label. Any instance that occurs outside the decision boundary
of these clusters is referred as an outlier. Instances from a novel class are detected based on the density of outliers in feature space. Such detection mechanisms rely on the existence of strong cohesion among instances from the same class and large separation among instances from different classes in observed feature space [16]. We refer to this as global class cohesion and separation assumption respectively. However, such a property may not be true in many real-world scenarios. A typical example is a handwritten digit recognition application where images of digit “1” may look very similar to those of digit “7”, as shown in Figure 1a. In such cases, existing approaches fail to detect novel classes (e.g., class “7”) from existing classes (e.g., class “1”). Alternately, Mu et al.[17] proposed a framework which dynamically maintains two kinds of low-dimensional matrix sketches that approximate the original information along the stream. Novel class detection is performed using the encoded information in a low-dimensional space. Yet, this approach may be ineffective since the detection and dimensionality transformation are unrelated processes.

In this paper, we address previous challenges by proposing a framework that can perform label prediction under concept evolution, called CSIM (Convolutional open-world multi-task image Stream classifier with Intrinsic similarity Metrics). The main goal is to transform the observed raw images into a latent feature space such that the classifier loss is minimized from achieving cohesion among instances belonging to the same class and separation of instances belonging to different classes. We achieve this by learning a latent feature space suitable for novel class detection. Particularly, we jointly train three main form of data transformation. First is a set of convolution layers to learn high-level features of images. These features are then transformed into another latent feature space using metric learning mechanisms [2] so that cohesion and separation properties can be distinctly achieved. We then employ novel class detection mechanism within this transformed feature space for data classification. For example, suppose we have three sets of instances, as shown in Figure 1b. Here, \{x_1, x_2, x_3\}, \{x_m, x_n\} and \{x'_1, x'_2, x'_3\} are instances associated with class A, class B and class C respectively. Considering class A and B, in observed feature space. \(x_i\) should be close to either \(x_1\) or \(x_2\), while \(x_m\) should be close to \(x_n\). However, since no assumption is made on the cohesion in \(\{x_1, x_2, x_3\}\) and \(\{x_m, x_n\}\), \(\|x_i - x_j\|_2 \gg \|x_i - x_n\|_2\) is possible. Using CSIM, we aim to transform the instances to an appropriate latent feature space so as to satisfy the closeness constraint for novel class detection, as illustrated in Figure 1b. Here, we first obtain a high-level feature embedding with the aid of convolution layers and then learn a latent feature space from instances of class A and B, while class C is the novel class. In the observed feature space, \(\{x'_1, x'_2, x'_3\}\) are close to instances of class A and are relatively far from each other. After the transformation, instances of each class form a dense cluster and are separated with a large margin, making novel class detection possible.

The contributions of this paper are as follows:

- We present a semi-supervised framework called CSIM that addresses the challenges of classification and concept evolution on high-dimensional real-world image streams.
- We propose a unified multi-task classifier that jointly performs metric learning, stream classification, and novel class detection.
- We empirically evaluate CSIM on real-world datasets, and compare its results with existing state-of-the-art novel class detection systems. We also study the effectiveness of the proposed feature transformation by comparing its performance with other metric learning approaches.

The rest of this paper is organized as follows. In Section 2, we present a brief background on metric learning and stream classification. We then formally define the problem and present its challenges in Section 3, before detailing the proposed solution in Section 4. In Section 5, we present the results of our empirical evaluation and finally conclude in Section 6.

2 RELATED WORKS

2.1 Metric Learning

Distance-based metric learning [2] plays a significant role in pattern recognition. Studies[12, 22] have successfully applied this to address complex classification tasks in the real world. Following the early work of Xing et al.[24], the goal of metric learning is to learn a distance-metric that minimizes the distance between similar examples and maximizes that between dissimilar examples. A distance-metric is usually represented as either an Explicit Metric Function (EMF) or an Implicit Metric Function (IMF).

2.1.1 Explicit Metric Function. The explicit metric function can be viewed as a linear/non-linear embedding function that maps examples in the original feature space into a new transformed feature space \([6, 9, 10, 22]\). A common closed-form linear EMF is the Mahalanobis-like distance \(D_M^2(x, y) = (x - y)^T M(x - y)\) [24], where \(M\) is a positive semi-definite (PSD) matrix satisfying the training constraints. This Mahalanobis-like distance introduces a linear transformation which maps \(x\) to \(x' = Lx\) with \(M = L^TL\). However, the simplicity of linear EMF limits its application on complex tasks. To address this issue, non-linear EMF, which is usually learned by generalizing the Euclidean distance with a non-linear transformation \(\phi\), is proposed. In this case, the distance measure becomes \(d_\phi(x, y) = ||\phi(x) - \phi(y)||_2\).

2.1.2 Implicit Metric Function. In contrast to explicit metric functions, it is inconvenient to obtain an explicit expression of the transformed embedding space for implicit metric functions. Many techniques have been adopted to learn an IMF and the widely accepted method is the kernel approach. For an input feature space \(\mathcal{H}\), a kernel \(\kappa : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{R}^+\) is a positive-definite function that are bivariate measures of similarity based on the inner product between samples embedded in a Hilbert space. Although implicit metric functions work well in some applications like clustering, constructing a kernel matrix is computationally expensive and applying the learned IMF for future predictions is difficult. In this paper, we focus on the non-linear IMF and present a novel approach that learns a high-quality metric via multi-task learning.

2.2 Stream Classification

A novel class at time \(t > 0\) is defined as a class label whose associated instances have never been observed along the stream until
time \( t \). Therefore, a classifier is never trained or updated using instances associated with this class. Studies typically aim to detect such novel class instances and reactively adapt the classifier for better performance. Previous studies in this direction [11, 16] have developed frameworks that leverage an unsupervised mechanism called Q-NSC for novel class detection. It uses the clusters resulting from K-Means to detect outliers, which are then analyzed based on density to detect novel classes. Alternatively, the framework by Mu et al. [17] uses low dimensional matrix sketches [17] by leveraging frequent directions [8] to detect novel class. Furthermore, all these approaches use a user-defined threshold to identify instances from novel classes along the stream. Unlike them, we employ a multi-task learning technique with online threshold evaluation for novel class detection.

3 PRELIMINARIES

In this section, we formally define the problem and list the associated challenges we address in this paper.

3.1 Problem Statement

Given a training dataset \( D = \{(x_t, y_t)\}_{t=1}^m \), where \( x_t \in \mathbb{R}^d \) is a training instance and \( y_t \in \mathcal{Y} = \{1, 2, \ldots, c\} \) is the associated class label, and a non-stationary streaming data \( S = \{(x_t, y_t)\}_{t=1}^n \), where \( x_t \in \mathbb{R}^d \) and \( y_t \in \mathcal{Y}' = \{1, 2, \ldots, c, c+1, \ldots, c'\} \) (\( c' > c \)), the goal is to learn a model \( f \) (initially with \( D \)) such that \( f(x_t) \rightarrow \mathcal{Y}' \). For every incoming instance in the data stream, \( f \) will determine whether it belongs to an unknown (also referred as novel) or an existing class. Note that for any two arbitrary classes \( c_m, c_n \in \mathcal{Y}' \) \( (c_m \neq c_n) \), if \( \{x_t, x_j\} \in c_m \) and \( \{x_k\} \in c_n \), it is possible that \( ||x_t - x_j||_2 < ||x_t - x_k||_2 \). The overall aim of the task is to maintain high classification accuracy along a data stream where instances from novel classes may occur over time.

3.2 Challenge

We assume that a data stream is non-stationary and consider a practical scenario where instances belonging to the same class may be further away from each other than instances from other classes in observed feature space. This introduces three main challenges:

- Model \( f \) has to capture instance similarity and dissimilarity correctly in a latent feature space suitable for class discrimination. It means that model \( f \) should internally learn a similarity metric capable of classifying high-dimensional data patterns with little external information.
- Due to the unbounded length of a data stream, model \( f \) could only be trained with a limited amount of training data at a given time and yet should predict well over long periods of time.
- Since novel classes could appear continuously in a data stream, model \( f \) needs to detect the emergence of novel classes while requiring a small amount of truth-value for the model update when necessary.

4 THE PROPOSED APPROACH

In this section, we describe our proposed framework CSIM for stream classification by first presenting an overview of CSIM and then discussing each component in details.
Algorithm 1 CSIM: Stream Classification

Require: \(\mathcal{S}\): Stream data; \(S_B\): The maximum size of \(\mathcal{B}\); \(T_D\): Confidence threshold for updating \(\mathcal{D}\); \(Y\): Label set of initial training data in warm-up phase; 
Ensure: Label \(\hat{y}\) predicted on \(\mathcal{S}\) data. 
1. Learn an initial model \(f\) from \(\mathcal{D}\) by solving the optimization problem. (Eq. 6) 
2. repeat 
3. Receive a new instance \(x\). 
4. Predict label \(\hat{y}\) for \(x\) using \(f\) according to Eq. 8 
5. if \(\hat{y} = -1\) then 
6. Store \(x\) in the candidate buffer \(\mathcal{B}\). 
else 
7. \(\hat{y} = \hat{y}\); 
9. end if 
10. if size(\(\mathcal{B}\)) \(\geq S_B\) then 
11. Check for occurrence of any novel class in data using DetectNovel (Algorithm 2) 
12. if DetectNovel returns True then 
13. if Update-Condition (Section 4.6) Satisfied then 
14. Retrain \(f\) with \(\mathcal{D}'\) (a subset of \(\mathcal{D}\)) (Section 4.6). 
15. end if 
16. end if 
17. end if 
18. if \(P(x) > T_D\) then 
19. Update \(\mathcal{D}\) using \((x, \hat{y})\) (Section 4.6). 
20. end if 
21. until \(\mathcal{S}\) exits

\(\mathcal{S}\): Stream data 
\(S_B\): The maximum size of \(\mathcal{B}\) 
\(S_d\): The maximum size of each class in \(\mathcal{D}\) 
\(T_D\): Confidence threshold for updating \(\mathcal{D}\) 
\(Y\): Label set of initial training data in warm-up phase 
x: d-dimensional features 
f: Open-world classifier 
\(\hat{y}\): Final predicted label of a data instance 
\(W_i\): Weights associated with class \(c_i\) in 1-vs-rest layer 
\(\hat{S}\): Estimated label of a data instance 
\(S_{update}\): Minimum number of instances of a class in \(\mathcal{D}\) for classifier update

Table 1: Frequently used symbols

| Symbol | Description |
|--------|-------------|
| \(\mathcal{S}\) | Stream data |
| \(S_B\) | Maximum size of \(\mathcal{B}\) |
| \(S_d\) | Maximum size of each class in \(\mathcal{D}\) |
| \(T_D\) | Confidence threshold for updating \(\mathcal{D}\) |
| \(Y\) | Label set of initial training data in warm-up phase |
| \(x\) | d-dimensional features |
| \(f\) | Open-world classifier |
| \(y\) | Data storage |
| \(\hat{y}\) | Final predicted label of a data instance |
| \(W_i\) | Weights associated with class \(c_i\) in 1-vs-rest layer |
| \(\hat{S}\) | Estimated label of a data instance |
| \(S_{update}\) | Minimum number of instances of a class in \(\mathcal{D}\) for classifier update |
| \(n_e\) | Number of epochs |

\(\mathcal{B}\): Novel class candidate buffer 
\(\mathcal{D}\): Data storage 
\(P(x):\) Prediction confidence for \(x\) using \(f\) 
\(\gamma\): Significance level of margin for triplet loss 
\(\mathcal{Y}\): Label set of possible labels in \(\mathcal{S}\) 

4.2.1 Triplet Loss. Let \(\mathcal{D}\) be a given training set that contains \(M\) triplets and \((x^a_i, x^p_i, x^n_i)\) be the \(i^{th}\) triplet in \(\mathcal{D}\). The EMF \(\phi(x)\) embeds an instance \(x\) into a \(d'\)-dimensional Euclidean space. After embedding, we expect the Euclidean distance between \(x^a_i\) and \(x^n_i\) to be at least \(\epsilon\) \((\gamma \geq 1)\) times the distance between \(x^p_i\) and \(x^n_i\). Formally,

\[
\frac{||\phi(x^a_i) - \phi(x^n_i)||_2 + 1}{||\phi(x^p_i) - \phi(x^n_i)||_2 + 1} \geq \epsilon \gamma
\]

where \(1\) is added as a smoothing factor. The resulting triplet loss \(L_{\text{triplet}}\) is

\[
L_{\text{triplet}} = \frac{1}{M} \sum_{i=1}^{M} \left[ \log(||\phi(x^a_i) - \phi(x^n_i)||_2 + 1 + \gamma \right] - \log(||\phi(x^a_i) - \phi(x^n_i)||_2 + 1) \right]
\]

Note that smoothing introduces an implicit constraint that at least one of \(||\phi(x^a_i) - \phi(x^n_i)||_2\) and \(||\phi(x^p_i) - \phi(x^n_i)||_2\) should be much greater than \(1\). Otherwise, the ratio computed by Eq. 1 would be close to \(1\) and the inequality is unsatisfied.

The motivation of introducing \(L_{\text{triplet}}\) is that it pushes different classes further away from each other by introducing a larger instance-sensitive margin, since \(||\phi(x^a_i) - \phi(x^n_i)||_2 - ||\phi(x^p_i) - \phi(x^n_i)||_2 \approx (\epsilon \gamma - 1)||\phi(x^p_i) - \phi(x^n_i)||_2\).

We want to minimize the triplet loss \(L_{\text{triplet}}\) but constrain the learned embedding on a \(d'\)-dimensional unit sphere. So the optimization problem for metric learning is

\[
\min_{\phi} L_{\text{triplet}}
\]

subject to \(||\phi(x^a_i)||_2 = 1, \forall x^a_i, x^p_i, x^n_i \in \mathcal{D}\).

Here \(x^a_i\) denotes any anchor, positive or negative instances in \(\mathcal{D}\), according to the definition. Any non-linear function can be utilized as \(\phi\) in the optimization problem. So, we choose to represent \(\phi\) as a convolutional neural network with a single fully-connected hidden layer of \(n\) units for simplicity.
The choice of triplets used for metric training is critical to the quality of learned metric. However, generating all possible combinations would result in a large number of triplets that are easily satisfied (i.e. fulfill the constraint in Eq. 2) that do not contribute to the training process. This may result in slow convergence. Therefore, it is crucial to select hard triplets that continuously contribute to improving the model. Here, we use the term "hard" to indicate a positive loss.

4.2.2 Triplets Selection. A triplet can be generated by first selecting an "anchor" class $c_a$ and a "negative" class $c_n$ ($c_a \neq c_n$) from $D$ and then choose two different instances $x_i^a$ and $x_i^n$ from $c_a$, and one instance $x_i^p$ from $c_n$. As mentioned above, we want to generate "hard" triplets for fast convergence. This means that, given $x_i^a$, we want to select an $x_i^p$ (hard positive) such that $\arg\min_{x_i^p} ||\phi(x_i^p) - \phi(x_i^a)||_2^2$ and an $x_i^n$ (hard negative) such that $\arg\max_{x_i^n} ||\phi(x_i^n) - \phi(x_i^a)||_2^2$. However, computing the argmin and argmax across $D$ is computationally intractable due to a large search space. Therefore, we aim to generate triplets in an online fashion. We focus on a mini-batch approach consisting of a subset of instances randomly sampled from $D$ at each step. By applying mini-batch gradient descent (MBGD) approach for minimizing $L_{\text{triplet}}$, we transform the instances to the embedding space. Then we compute the argmax and argmin within that mini-batch to generate desired triplets. The motivation behind this decision is to provide hard triplets to the model at any stage during its training to continuously improve the learned embedding.

4.3 Convolutional Open-World Classifier (Conv-OWC)

Due to the important role of a high-quality metric in both classification and novel class detection, we choose to fuse metric learning and novel class detection into classification. Hence, we propose a novel classifier referred as Convolutional Open-World Classifier (Conv-OWC) that performs all these tasks jointly. Figure 3 illustrates the structure of Conv-OWC in CSIM. The Convolutional layer, Max-Pooling layer, Input Layer, Hidden Layer and Embedding Layer learns the metric $\phi$. In contrast to traditional multi-class classifiers that typically use softmax as the final output layer, we build a 1-vs-rest layer (Classification/Novel Detection Layer) containing $K$ sigmoid functions for $K$ classes, following [21]. For $i^{th}$ sigmoid function corresponding to class $c_i$, Conv-OWC takes all examples with label $y = c_i$ as positive examples and the rest with $y \neq c_i$ as negative examples. Let $L_{\text{class}}$ denotes the loss introduced by the 1-vs-rest layer. It is the average Binary Classification Error (BCE) of $K$ sigmoid functions on the training data $D$. Formally, the loss is given by:

$$L_{\text{class}} = \frac{1}{Kn} \sum_{i=1}^{K} \sum_{j=1}^{n} [-\log P(y_j = c_i) \log P(y_j = c_i) - \log (1 - P(y_j = c_i))]$$

(4)

Unlike [21], we do not optimize $L_{\text{class}}$ on its own. Instead, we optimize it with the triplet loss $L_{\text{triplet}}$. Thus the tasks of metric learning, classification and novel class detection are learned jointly in Conv-OWC. The resulting objective function for multi-task optimization, denoted as $L_{\text{overall}}$, is

$$L_{\text{overall}} = \sum_{j=1}^{M} \left\{ \left( \frac{1}{3K} \sum_{i=1}^{K} \sum_{x \in \{a, p, n\}} [-\log P(y_x^* = c_i) \log P(y_x^* = c_i) - \log (1 - P(y_x^* = c_i))] + \frac{\beta}{M} \log (\|\phi(x_i^p) - \phi(x_i^n)\|^2_2 + 1) \right) + \gamma - \log (\|\phi(x_i^n)\|^2_2 + 1) \right\}$$

(5)

where $P(y_x^* = c_i) = \sigma(W_i \phi(x_i^p) + b)$ ($W_i$ is the weight of $i^{th}$ class in 1-vs-rest layer), $\beta$ is a hyper-parameter that controls the importance of $L_{\text{triplet}}$ in $L_{\text{overall}}$ and $M$ is the number of triplets used for training. The overall optimization problem is given by

$\minimize_{\phi, W_1, W_2, \ldots, W_K} L_{\text{overall}}$

subject to $\|\phi(x_i^n)\|^2_2 = 1, \forall x_i^n \in D$. (6)

By optimizing $L_{\text{overall}}$, the knowledge learned via metric learning helps improve the generalization performance of classification and vice versa. This information transfer in $L_{\text{overall}}$ is critical in stream applications where a limited amount of labeled training data is available.

4.4 Classification

Suppose $f$ denotes the convolutional open-world classifier and $\hat{y}$ is the prediction label generated by $f$, for every incoming instance $x$, the prediction probability $P(\hat{y} = c_i | x)$ of class $c_i$ is computed by $P(\hat{y} = c_i | x) = \sigma(W_i \phi(x) + b)$. However, before making a decision on the predicted label of instance $x$, we need to determine the threshold $T_{\text{novel}}$ for novel class detection.

Due to the non-stationary nature of stream, it is inappropriate to manually set a threshold and expect it work well along the stream. This indicates that the threshold for novel class detection should be determined automatically based on current stream property. To obtain a better $T_{\text{novel}}$, we use the idea of one-sided confidence bound in statistics.

Assume the predicted probabilities $P(\hat{y} = c_i | x)$ for all data of each class $c_i$ in a training dataset $D$ follow a Gaussian distribution with unknown mean and unknown variance. A good statistic for confidence threshold is the average prediction probability of
training data, i.e., \( \tilde{P}(c_i) = \frac{1}{||D_i||} \sum_{x \in D_i} P(\hat{y} = c_i | x \in D_i) \), where \( D_i = \{(x_j, y_j = c_i, \forall x_j \in D) \} \). The desired \( T_{novel} \) for class \( c_i \) is the 100(1 - \alpha)% confidence lower bound of \( \tilde{P}(c_i) \) given by

\[
T_{novel}(c_i) = \tilde{P}(c_i) - t_{\alpha, ||D_i||-1}S_{c_i}/\sqrt{||D_i||}
\]

(7)

Here, \( S_{c_i} \) is the sample standard deviation of \( \{ P(\hat{y} = c_i | x), \forall x \in D_i \} \).

Once we have the threshold \( T_{novel} \), classification is trivial. For the \( i^{th} \) sigmoid function, we check if the predicted probability \( P(\hat{y} = c_i | x) \) is less than the NCD threshold \( T_{novel}(c_i) \). If the predicted probabilities of all classes are less than their corresponding thresholds for \( x \), then \( x \) is a candidate from a novel class. As a result, this instance is rejected (predicted as \(-1\)), and is temporarily stored in \( B \). Otherwise, its predicted class is the one with the highest probability. Formally, we have the following.

\[
y = \begin{cases} 
-1 & \text{if } P(\hat{y} = c_i | x) < T_{novel}(c_i), \\
\arg\max_{c_i \in Y_D} P(\hat{y} = c_i | x) & \text{otherwise}
\end{cases}
\]

(8)

Here \( \hat{y} \) is the estimated label for an instance \( x \) and \( Y_D \) is the label set of \( D \). If \( \hat{y} \neq -1 \), the final predicted label \( \hat{y} \) is the same as \( \hat{y} \), i.e., \( \tilde{y} = \hat{y} \). Otherwise, the prediction of \( \tilde{y} \) for those instances with \( \hat{y} = -1 \) is left to the novel class purification module.

### 4.5 Novel Class Purification (NCP)

Unlike many prior stream classifiers [11, 16], we make a more practical assumption that instances from a novel class might be similar to those from known classes in the observed feature space. Moreover, noise in streams may lead to false alarms. Hence, some instances from known classes might be incorrectly reported as coming from a novel class. Once the candidate-buffer \( B \) is full, the novel class purification module is invoked to filter novel class instances out from candidates in \( B \). It is done by following the steps below:

- **Candidates in \( B \) are first transformed to the metric embedding space represented by \( \phi \) and then \( DBSCAN \) [7] is performed on the transformed instances to achieve a set of clusters \( \{C_1, \ldots, C_m\} \).
- For each \( C_i \), we randomly sample out one instance from the cluster to request its true label and this true label would be the prediction label for all instances within this cluster.

Here, \( DBSCAN \) is selected since it is unsupervised and does not have a strong constraint regarding cluster shape like \( K \)-Means. After being transformed into metric embedding space, instances from the same class tend to form a dense cluster. Although clusters of novel classes are separated from those of existing classes with a larger margin in \( CSIM \), they are not sufficiently far away so that global separation assumption could hold. It is due to the lack of novel class information during the training of \( \phi \). Those detection techniques based on the global separation assumption would simply fail in this case. However, the cohesion property of clusters indicates that instances within a cluster are semantically similar to each other. This builds the foundation of our proposed NCP. A formal description of the NCP is shown in Algorithm 2.

#### 4.6 Data Storage and Classifier Update (DSCU)

A data storage \( D \) is actually a storage unit consisting of \( K \) buffers, where \( K \) is number of classes and it stores at most \( S_D \) instances for each class. Let \( D_i \) denote the buffer for class \( c_i \). For every \( (x, \hat{y} = c_i) \) sent to update \( D \), if \( D_i \) is not full, \( x \) is simply added to \( D_i \); Otherwise, the "oldest" instance is replaced by \( x \).

The classifier \( f \) is updated only when both of the following update conditions are satisfied.

- **\( DetectNovel \) returns True.**
- Suppose \( C \) is the set of novel classes detected by \( DetectNovel \) since last update of \( f \), at least one of \( C \) should contain more than \( S_{update} \) instances in \( D \).

If satisfied, all classes with more than \( S_{update} \) instances in \( D \) forms a new training dataset \( D' \), and is then used to retrain the classifier \( f \).
and centered in a fixed-size image. The problem is to identify the
corresponding digit for each image. FASHION-MNIST dataset is
designed as a difficult drop-in replacement for MNIST that shares
class characteristics with it, but it better represents modern computer
vision (CV) tasks. Each example is a $28 \times 28$ gray-scale fashion
image, associated with a label from 10 classes. The EMNIST dataset is
a set of handwritten character digits derived from the NIST Special
Database 19 which contains digits, uppercase, and lowercase hand-
written letters. In the experiment, we select the balanced version of
EMNIST that contains 131,600 characters with 47 balanced classes.
CIFAR-10 is another image dataset containing 60,000 $32 \times 32$
colour images in 10 classes, with 600 images per class. To be con-
istent with other datasets, we convert these images into grayscale
through OpenCV API, resulting in 1024 features. Details of these
datasets are listed in Table 2.

An initial training set with $[n \cdot r]$ known classes is available to
train the model, where $n$ is the total number of classes in the dataset
and $r$ is a user-defined constant between 0 and 1 indicating the ratio
of known classes in each dataset. For our experiments, we generate
two streams, one with $r = 0.3$ and the other with $r = 0.5$. Instances
of leftover classes (i.e., $n - [n \cdot r]$) form the novel class collection.
We simulate a data stream on each benchmark dataset including
instances of both the known classes and new classes in the novel
class collection. Note that those new classes appear in different
periods in this simulated data stream with a uniform distribution.

5.2 Baselines

To examine the quality of metrics learned in CSIM, we compare
the convolutional open-world classifier learned in CSIM with sev-
eral state-of-the-art metric learning algorithms. (1) LMNN (linear,
EMF) [22]: A Mahalanobis distance metric for kNN classification
from labeled examples, trained with the goal that the k-nearest
neighbors always belong to the same class while examples from
different classes are separated by a large margin; (2) HDML (non-
linear, EMF) [18]: A framework applicable to a broad families of
mappings from high-dimensional data to binary codes that preserve
semantic similarity, using a flexible form of triplet ranking loss. The
mapping is represented by a well-designed hash function. In this
experiment, we select the most complex hash function, i.e., multi-
layer neural network, proposed by the author for a fair comparison.

(3) GB-LMNN (non-linear, EMF) [13]: An expansion of LMNN that
substitutes the linear feature mapping with non-linear Gradient
Boosting Trees (GBRT). (4) SKLR (non-linear, IMF) [1]: An implicit
metric which learns a kernel matrix using the log-determinant di-
vergence subject to a set of relative-distance constraints. It is useful
in settings where providing similar and dissimilar constraints is
difficult.

Besides, we also compare CSIM with competing state-of-the-
art stream classifiers. (1) ECSMiner (fully supervised) [16]: an
ensemble framework to detect novel classes using K-Means clus-
tering, with a KNN-based classifier to make predictions; (2) ECHO-
D (semi-supervised) [11]: an improved framework based on EC-
SMiner that maintains an ensemble of clustering-based classifier
models. Each model is trained on different dynamically-determined
partially-labeled chunks of data. It detects novel classes via the
same algorithm as ECSMiner but classifies instances in a differ-
ent way; (3) SENC-MaS (semi-supervised) [17]: a framework that
maintains two low-dimensional sketches of stream data (global and
local sketch) to detect novel classes and make predictions.

5.3 Experiment Setup

We have implemented CSIM using Python 3.6.2, and the con
volutional open-world classifier using the Pytorch 0.4.0 library. All
baseline methods were based on code released by corresponding
authors, except SENC-MaS. Due to unavailability of a fully func-
tional code of SENC-MaS, we use our own implementation based on
the author’s description [17]. Hyper-parameters of these baseline
approaches were set based on values reported by the authors and
fine-tuned via cross-validation either on the validation dataset (met-
ric comparison) or the initialization dataset (stream classification).
In CSIM, we set $n = 200$, $S_2 = 1000$, $S_3 = 200$, $S_{update} = 100$,
$T_2 = 0.99$, $r = 1.0$, $S_{mini} = 64$ and $n_c = 10$ as default. The initial
training dataset size is 1000 per class. In addition, we set the kernel
size $K = 5$ and the stride $S = 1$ for convolutional layer and $K = 2$
and $S = 2$ for max-pooling layer in Conv-OWC.

5.4 Evaluation Metrics

5.4.1 Stream Classification. Let $F_N$ be the total novel class in-
stances misclassified as existing class, $F_P$ be the total existing class
instances misclassified as novel class, $N_c$ be the total novel class
instances in the stream, and $N$ be the total number of instances in
the stream. We use the following metrics to evaluate our approach
and compare it with baseline methods. (a) $\text{Accuracy}\% = \frac{A_{new} + A_{known}}{n}$,
where $A_{new}$ is total number of novel class instances classified cor-
rectly, $A_{known}$ is the number of known class instances identified
correctly, and $m$ is the number of instances in the stream. (b) $\%$ of
labels: $\%$ of true labels requested by the framework for classifier
training and update. (c) $M_{new} = \%$ of novel class instances misclas-
sified as existing class, i.e. $\frac{F_N}{N_c} \times 100$. (d) $F_{new} = \%$ of existing class
instances misclassified as novel class, i.e. $\frac{F_P}{N} \times 100$. Finally, (e) ratio:
$\text{Accuracy}\% = \frac{N}{M}$, where $M$ denotes a method in [ECSMiner,
SENC-MaS, ECHO, CSIM] and $M_{best}$ is the method with the best $\text{Accuracy}\%$
among them.

5.4.2 Metric Learning. Let $N_c$ be the total test instances be-
gong to class $c$, $T_c$ be the total test instances of class $c$ that are
correctly predicted, and $C$ be the set of all classes in the test dataset.
We measure the following evaluation metric. (a) $\text{Accuracy}\% = \frac{\sum_{c \in C} T_c}{N_c}$.
(b) $\%$ of $M = \frac{\text{Accuracy}\%}{\text{Accuracy}\%_{M_{best}}}$, where $M$ denotes a method in (LMNN,
HDML, GB-LMNN, SKLR, CSIM) and $M_{best}$ is the method with the best $\text{Accuracy}\%$
among them.

5.5 Results

5.5.1 Stream Classification. We conduct 10 independent experi-
ments with different simulated streams for both $r = 0.3$ and $r = 0.5$
on each real-world benchmark dataset. However, we only report
the mean and standard deviation of performance on streams with
$r = 0.3$ due to lack of space, though we observed similar result
on $r = 0.5$. Table 3 lists the results on data streams with $r = 0.3$. 

KDD’18 Deep Learning Day, August 2018, London, UK
As mentioned in Section 5.1, the r value indicates the number of classes known in the warm-up phase. For example, with r = 0.3, only 3 classes are known in the initial training data on MNIST. We observe that SENC-MaS performs poorly on most real-world datasets due to its linear classifier. ECSMiner performs better than ECHO-D because the former is fully-supervised and the latter is semi-supervised. However, in both cases of r, CSIM outperforms all the baseline approaches by providing significantly better accuracy while requesting fewer or similar amount of true labels. For example, on EMNIST dataset (r = 0.3, 47 classes), CSIM provides an accuracy of 85.36%, which is 23.92% higher than that provided by the best baseline ECSMiner and reduces the number of ground truth labels requested by 60.07%. We observe similar results for EMNIST stream with r = 0.5. CSIM is much better than all baselines mainly because of the intrinsic similarity metric learned via multi-task learning which improves the performance of both classification and novel class detection. Moreover, the convolutional layer in Conv-OWC aids in detecting edges in images which forms the conceptual representation that helps to improve the quality of learned intrinsic similarity metric.

5.5.2 Novel Class Detection. Table 4 compares novel class detection performance of CSIM with all baseline methods on each dataset. The conceptual representation that helps to improve the quality of classification and novel class detection. Moreover, the convolutional layer in Conv-OWC aids in detecting edges in images which forms the conceptual representation that helps to improve the quality of learned intrinsic similarity metric.
5.5.3 **Stability of CSIM over Data Streams.** Figure 4 shows the classification accuracy of CSIM over the FASHION-MNIST data stream. As shown in the figure, CSIM performs better than baselines and maintains good performance with new classes continuously emerging over time. In particular, CSIM adapts to the occurrence of unknown classes quickly compared to SENC-MaS and produces more accurate predictions. ECHO shows an unstable performance that degrades significantly and ECSMiner results in slightly better performance compared to ECHO since it is fully-supervised. Similar results have been observed on other data streams.

5.5.4 **Metric Learning.** To study the generalization performance of learned metrics in Conv-OWC on unseen data when limited training data is available, a common case on streams, we perform experiments on the two splits (i.e., S1 and S2) over the image datasets. Here, we first randomly shuffle each benchmark dataset and then divide it into training, validation and test sets with the split ratio of 4 : 2 : 4 and 1 : 1 : 8. We denote these splits as S1 and S2 respectively. Here, S2 is more realistic for a data stream. This process is repeated for 10 times to avoid any statistical fluctuation. Both mean and standard deviation of performance are reported in Table 5. As shown in the result, for both ratios, CSIM outperforms all baseline approaches on all benchmark datasets by providing significantly better classification accuracy. For example, on EMNIST dataset that contains 47 classes, CSIM provides a higher accuracy of 73.67% compared to the best baseline GB-LMNN with a margin of 9.62% for the S2 split. The superior performance of CSIM demonstrates its better capability of capturing intra-class similarity and inter-class dissimilarity with a limited amount of labeled training data. Fig. 5 illustrates an example of original and transformed embeddings provided by CSIM on FASHION-MNIST dataset. Hence, compared to other state-of-the-art baselines, our proposed metric learning approach is more suitable for stream applications.

5.5.5 **Sensitivity of Parameters.** The two main parameters in CSIM are the number of hidden units \( n \) in the Conv-OWC, and significance level \( \gamma \) of margin for triplet loss. We vary these parameters to study its sensitivity to classification and novel class detection performance. Figure 6 shows the result on FASHION-MNIST dataset as an example. If \( n \) is relatively small, it indicates a simple network. In this case, the classification and novel class detection performance significantly drops by providing a lower accuracy and higher \( M_{\text{new}} \) and \( F_{\text{new}} \). On the other hand, a larger \( n \) reduces \( M_{\text{new}} \) but provides little improvement on the other metrics and dramatically increases the time and space cost. Similarly, as \( \gamma \) increases, CSIM attempts to push different classes with a margin that is too large, leading to overfitting issues. Therefore, we choose a moderate value of \( n = 200 \) and \( \gamma = 1.0 \) during evaluation.

5.5.6 **Effect of Convolutional Layers.** To study the effect of convolutional layers on both classification and novel class detection performance over data streams, we built two variants of Conv-OWC with 0 (CSIM-0) and 2 (CSIM-2) convolutional layers respectively. Table 6 reports the results on FASHION-MNIST data stream as an example. A significant improvement on classification and novel
class detection performance is observed from CSIM-1 to CSIM-0, which indicates the edges recognized by the convolutional layer actually reduces the difficulty of subsequent metric learning task. However, adding more convolutional layers provides little help for performance improvement but dramatically increases the training time and is hence undesired. It is mainly because of the limited amount of training data along the stream that is insufficient for a bigger network to improve the quality of its conceptual representation. Therefore, our choice of single convolutional layer in CSIM is recommended.

5.5.7 Time and Space Complexity and Limitation. Overall, the execution overhead of CSIM mainly arises from the training and updating procedure, particularly while training the convolutional open-world classifier. Assuming that the time complexity of calculating the gradient of one example is a constant $C$, the time complexity of MBGD within a mini-batch is $O(S^2 + ||Y||)$, where $||Y||$ denotes the number of classes in $Y$. Clearly, the large overhead of CSIM mainly comes from the gradient computing in each mini-batch. In our implementation, we use a GPU for computational acceleration. By using a GTX 1080 Ti 11GB GPU, the average training time for CSIM is 29.33 seconds per epoch and hence total training time is approximately 4.89 minutes. The space complexity of CSIM is $O(S^2 + ||Y|| + B_{space})$, where $B_{space}$ denotes the space complexity of the model used to represent $\phi$.

Although CSIM demonstrates a good performance on many real-world stream application tasks, it has several drawbacks. First, CSIM relies on the quality of true labels. An error in ground truth labels reduces the quality of learned metrics and degrades the classification performance. Second, CSIM requires more computational resources for execution compared to other approaches due to the use of a neural network. We leave the exploration for other non-linear kernel-based approaches, that can replace the neural network, for future work.

6 CONCLUSIONS

In this paper, we propose a novel semi-supervised stream classification framework that utilizes a convolutional open-world classifier with an intrinsic high-quality similarity metric trained via multitask learning. This framework addresses the challenge of novel class detection problem with better performance compared to state-of-the-art baselines. More importantly, we discard the strong global class cohesion and separation assumption in novel class detection and demonstrate a technique to detect instances from multiple new classes using the convolutional open-world classifier. Our empirical evaluation of real-world datasets and streams shows the practical benefit of CSIM as we compare our results with state-of-the-art stream mining systems.

7 ACKNOWLEDGMENTS

We thank the reviewers for their insightful comments. This material is based upon work supported by NSF award number DMS-1737978, AFOSR award number FA9550-14-1-0173, NSA and IBM faculty award (Research).

REFERENCES

[1] Ehsan Amid, Aristides Gionis, and Antti Ukkonen. 2016. Semi-supervised Kernel Metric Learning Using Relative Comparisons. CoRR abs/1612.00886 (2016). arXiv:1612.00886 http://arxiv.org/abs/1612.00886

[2] Aurélien Bellet, Amaury Habrard, and Marc Sebban. 2013. A Survey on Metric Learning for Feature Vectors and Structured Data. CoRR abs/1306.6709 (2013). arXiv:1306.6709 http://arxiv.org/abs/1306.6709

[3] Jürgen Beringer and Eyke Hüllermeier. 2007. Efficient instance-based learning on data streams. Intell. Data Anal. 11, 6 (2007), 627–650. http://content.iospress.com/articles/intelligent-data-analysis/ida00307

[4] Albert Bifet and Ricardo Gavaldà. 2007. Learning from time-changing data with adaptive windowing. In Proceedings of the 2007 SIAM international conference on data mining. SIAM, 443–448.

[5] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. Emnist: an extension of mnist to handwritten letters. arXiv preprint arXiv:1702.05373 (2017).

[6] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. 2007. Information-theoretic metric learning. In Proceedings of the 24th international conference on Machine learning. ACM, 209–216.

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd. Vol. 96. 226–231.

[8] Minh Thai Bui, Eddy Debbaudt, and David P. Woodruff. 2016. Frequent Directions: Simple and Deterministic Matrix Sketching. SIAM J. Comput. 45, 5 (2016), 1762–1792. https://doi.org/10.1137/15M1009718

[9] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. 2005. Neighbourhood components analysis. In Advances in neural information processing systems. 533–540.

[10] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. 2009. Is that you? Metric learning approaches for face identification. In Computer Vision. 2009 IEEE 12th international conference on. IEEE, 498–505.

[11] Zhanasul Haque, Latifur Khan, Michael Baron, Bhavani Thuraisingham, and Charu Aggarwal. 2016. Efficient handling of concept drift and concept evolution over stream data. In Data Engineering (ICDE), 2016 IEEE 32nd International Conference on. IEEE, 481–492.

[12] Junlin Hu, Jowen Lu, and Yap-Peng Tan. 2014. Discriminative deep metric learning for face verification in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 1875–1882.

[13] Dor Kedem, ZhiXiang Eddie Xu, and Kilian Q Weinberger. [n. d.]. Gradient Boosted Large Margin Nearest Neighbors. [n. d.].

[14] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).

[15] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist

[16] Mohammad Mastod, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. 2011. Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Transactions on Knowledge and Data Engineering 23, 6 (2011), 859–874.

[17] Xin Mu, Feida Zhu, Juan Du, Er-Peng Lim, and Zhi-Hua Zhou. 2017. Streaming Classification with Emerging New Class by Class Matrix Sketching. In AAAI. 2373–2379.

[18] Mohammad Norouzi, David J Fleet, and Ruslan R Salakhutdinov. 2012. Hamming Distance Metric Learning Using Relative Comparisons. CoRR abs/1306.6709 http://arxiv.org/abs/1306.6709

[19] Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research 10, Feb (2009), 207–244.

[20] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. 815–823. https://doi.org/10.1109/CVPR.2015.7296682

[21] Junlin Hu, Jowen Lu, and Ying Liu. 2017. 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. 2911–2916. https://aclanthology.info/papers/D17-1314/D17-1314

[22] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. 2003. Distance metric learning with application to clustering with side-information. In Advances in neural information processing systems. 521–528.