Heterogeneous Domain Adaptation for Multistream Classification on Cyber Threat Data

Yi-Fan Li, Yang Gao, Gbadebo Ayoade, Member, IEEE, Latifur Khan, Fellow, IEEE, Anoop Singhal, Senior Member, IEEE and Bhavani Thuraisingham, Fellow, IEEE

Abstract—Under a newly introduced setting of multistream classification, two data streams are involved, which are referred to as source and target streams. The source stream continuously generates data instances from a certain domain with labels, while the target stream does the same task without labels from another domain. Existing approaches assume that domains for both data streams are identical, which is not quite true, since data streams from different sources may contain distinct features. Indeed, they may even have different numbers of features. Furthermore, obtaining labels for every instance in a data stream is often expensive and time-consuming. Therefore, it has become an important topic to explore if classes of labeled instances from other related streams are helpful to predict the classes of unlabeled instances in a different stream. Note that domains of source and target streams may have distinct feature spaces and data distributions. Our objective is to predict class labels of data instances in the target stream by using the classifiers trained by the source stream.

We propose a framework of multistream classification by using projected data from a common latent feature space, which is embedded from both source and target domains. This framework is also crucial for enterprise system defenders to detect cross-platform attacks, such as Advanced Persistent Threats (APTs). Empirical valuation and analysis on both real-world and synthetic datasets are performed to validate the effectiveness of our proposed algorithm, comparing to state-of-the-art techniques. Experimental results show that our approach significantly outperforms other existing approaches.

Index Terms—Attack detection, Multistream classification, Domain adaptation

1 Introduction

Data streams are crucial in the modern connected Internet world, and they have attracted the attention of researchers worldwide. Given important applications of data streams–such as IoT, social networks, and surveillance–mining them properly is becoming a more and more important topic to explore. However, data stream mining is also a challenging task due to its distinctive nature. For example, a data stream is theoretically infinite in length, therefore, its volume is very large. Meanwhile, it is possible that with high velocity of data arrivals, class labels are not available immediately after the arrival of new data instances, which makes it difficult to update the model. These properties have distinguished data stream mining significantly from traditional data mining problems [1], [2].

Normally, one assumption of classification model is that source data with true labels are also representative of target data. However, since data streams are generated by non-stationary processes, it is possible that arbitrary changes may be introduced to data streams. Thus, using a model trained by old source data may introduce bias in prediction [3], and will lead to poor performance in predicting future data labels. Thus, the model needs proper updates to adapt to the current concept.

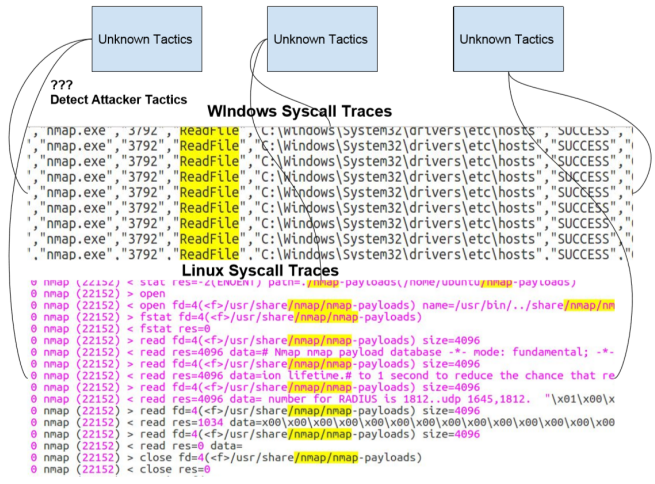


Fig. 1: Sample Trace File showing syscall traces for an attacker scanning session with nmap network scanning tool

When it comes to the setting of multiple streams, this problem becomes even more challenging. The problem setting is described and motivated by a multistream classification (MSC) problem [4]. Two data streams are involved simultaneously under this setting. One is referred to as the source stream, which contains data instances with class labels generated from one non-stationary process with labels. The other is referred to as the target stream, which consists of data generated from another non-stationary process without labels.

- *Y. Li and Y. Gao are with Meta Platforms, Inc. The work was performed at the University of Texas at Dallas.*
  E-mail: {yli, yxg122530}@utdallas.edu
- *G. Ayoade is with Google LLC. The work was performed at the University of Texas at Dallas.*
  E-mail: gbadeboayoade@gmail.com
- *L. Khan and B. Thuraisingham are with the Department of Computer Science, The University of Texas at Dallas.*
  E-mail: {lkhan, bhavani.thuraisingham}@utdallas.edu
- *A. Singhal is with National Institute of Standards and Technology (NIST).*
  E-mail: anoop.singhal@nist.gov

The multistream setting is very common in cybersecurity area, and there are lots of real-world applications under this setting. For example, many organizations are prone to cyber-attacks. These organizations deploy threat monitoring tools that collect network system call events as data streams. In order to detect these attacks, we have to overcome the challenge of linking these low-level traces to attacker tactics and intent. In addition, the scarcity of labeled data introduces more limitations in deploying machine learning techniques in detecting cyber attacks. In Figure 1, a snapshot of a trace file is shown as the scanning session of an attacker using the nmap tool to scan a victim's network. Also, data streams obtained from Linux machines cannot be used to predict attack events on a host with a Windows operating system. This is due to the difference in the way events are recorded on both systems (features may vary from Linux to Windows or vice versa).

Another application in cybersecurity involves extracting malicious products being sold on the dark web forums. Due to the various platforms available for malicious actors to sell illegal goods, such as stolen credit card numbers, social security numbers, malicious software and root-kits, we need a system that can obtain labeled train data from different dark web domains and use them to identify products sold on other dark web domains. Furthermore, these posts in the dark web consist of posts as streams of data. This is because posts are continuously coming and evolving. Therefore, by leveraging multistream classification, we can increase the performance of the product identification technique.

In this paper, a domain adaptation setting is considered within the multistream classification framework. There is a need to address machine learning tasks of building models in one domain by using information available from another related domain. Here, knowledge from the source domain (stream) can be transferred to the target domain (stream) to help get high prediction accuracy. This transfer process is important in many circumstances where labels in a certain domain are limited or too expensive (prevalent in cyber-security as said before) to obtain [5], which fits the need of multistream by nature. Notice that here the feature representations (Number. of features) and distributions may not identical across domains.

In practice, it is assumed that training data are provided as a whole in advance [6]. However, under a data stream setting, all instances may arrive in a sequential manner. Take online spam email detection for example [7]. Normally, a classifier is built using a static training dataset, and it is trained in batches to detect spam emails as accurately as possible. However, the very definition of spam varies from person to person. In this case, the transfer of knowledge to personalize the spam detector for each individual becomes critical. Even for a specific person, the definition of spam could change over time. During a certain time period one may consider an advertisement from Zillow to be spam. When he starts to consider buying a house, Zillow ads are not considered to be spam for him anymore. Such problem can be formulated as a multistream learning task, the objective of which is to build an online prediction model for the target domain by using information available in the source domain with distinct feature space. Compared to the previous example, this task is more challenging, as the concept is evolving in the data streams simultaneously in heterogeneous source and target streams.

To sum up, the problem that we are trying to solve involves two data streams with heterogeneous feature spaces, and this is the aspect which no researchers have been working on in the past.

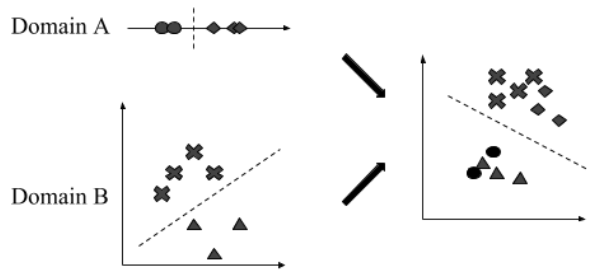


Fig. 2: Feature Space Projection

This paper proposes a framework, called Heterogeneous Domain Adaptation Network (HDAN), to handle the issues described above. The main idea is to find a common feature space for two distinct data streams. This idea is demonstrated in Figure 2. Data in domain A are one-dimensional and in domain B are two-dimensional. Therefore, there are two heterogeneous domains, and the main idea is to find a common feature subspace for these two domains. By applying the proposed projection algorithm, we try to find a latent feature space that the distributions in original and latent feature space are similar. Meanwhile, the structure of data should be preserved, which means distinct classes are still far apart, and the decision boundaries for different categories should still be preserved. Thus the core problem here is to find a feature space that maximizes the similarity between original and projected latent feature space. Main contributions of this paper are as follows:

1) An embedding-based domain adaptation method is introduced in multistream setting, to address the challenges of adapting source and target domains in a non-stationary environment.
2) A concept drift detection method is deployed under the new multistream setting with domain adaptation. Under this setting, true labels in the target stream are not required, while only true labels in the source stream are used.
3) Our approach is empirically evaluated over several benchmark datasets, including security and non-security

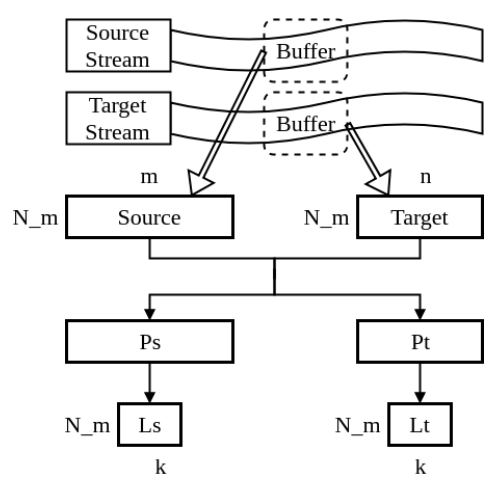


Fig. 3: Feature Projection

datasets. The results are compared with state-of-the-art methods. The performance of our proposed method is significantly better than the other three methods.

2 Related Work

We outline related work under three topics, namely domain adaptation, stream classification, and cyber security analytics.

2.1 Domain Adaptation

One of fundamental assumptions in data mining, known as the "stationary distribution assumption", is that both the training and test data are generated from the same domain and thus represent the same data distribution [8]. Therefore, common techniques normally cannot be directly deployed when training and test data are from different domains. The differences between domains can be normally considered as two aspects: 1. distinct number of features; 2. distinct feature distributions. Several approaches have been proposed to learn a common feature representation for domain adaptation [6], [9]. [10] proposed a linear objective function to project a latent feature space for both source and target domains. This approach focuses on static data only. Recall that stream feature space may evolve over time across streams, which may trigger heterogeneous feature space. [11] uses a co-regularized method to project target domain to source domain in an online manner. However, this method also makes a strong assumption that features in source dataset is a subset of those in target dataset, which in reality, this is not always the case. However, the method works in an incremental manner which may be suitable for stream setting.

Similarly, as we stated in the previous section, it is challenging to derive an online method to reduce the distribution differences between domains. The state-of-the-art methods, including deep learning, mostly address domain adaptation problems for stationary data except [11]. Thus, fitting domain adaptation frameworks to online data streams is yet to explore.

2.2 Stream Classification

Data stream classification is a challenging task due to its inherent properties, such as infinite length and concept drift. The infinite length problem is addressed by dividing the whole stream into fixed-size minibatches or using a gradual forgetting mechanism. Recent approach [12] handles this problem by remembering only the instances within two consecutive concept drifts using a dynamically-sized minibatch. Concept drift detection in multivariate data streams concentrates on tracking any changes in the posterior class distribution, $P(y|x)$. Instead of tracking changes in $P(y|x)$ directly, the approaches proposed in [13] adopt the principle [14] to detect this change indirectly by tracking drift in the error rate of the underlying classifier. However, tracking drift in the error rate requires true labels of test data instances, which are scarce in practice. Recent studies focus on confidence [4] to detect changes in distribution between two different time windows by detecting change points in the process. Apart from working on a single stream, these methods explicitly detect change points.

The problem setting described in this paper is motivated by a Multistream Classification framework [4]. It proposes a solution which focuses on the classification problems when there are asynchronous concept drifts on source and target streams. Since stream classification is a continuous process, data in the target stream is assumed to be generated with very few labels. Yet, there is a strong assumption made for this work: the number of features in both source and target streams is fixed. To overcome those obstacles, our proposed approach relaxes this assumption. features may not be exactly the same between domains, both in number and correspondence. Furthermore, other features in source stream may not have any mapping to features in target stream and vice-versa.

2.3 Cyber Attack

Machine learning algorithms have been deployed to learn attackers' behavioral patterns, either using host-based [15], [16] or network-based [17], [18]. Host-based methods detect attacks by learning sequences of anomalous system calls that characterize attacker behavior [19]. Sequence of packet structure to perform website fingerprinting attacks [20]. Network-based approaches use network data such as packets captured from the network. On the other hand, most network-based systems are deployed at the network edge to detect malicious traffic patterns or payload [21], [22]. However, host-level detection systems are able to detect attacks even with encrypted traffic due to high visibility into the system events [23]. Our method differs from these methods since we learn high level concepts from low level traces, and our method can handle data across different heterogeneous domains.

3 Problem Formulation

3.1 Notations and Problem Statement

In Table 1, frequently used symbols in this paper are listed. There are two continuous streams of data instances generated from source domain D_s and target domain D_t. A data instance is denoted as (x, y), where x is a vector (m-dimensional in source stream and n-dimensional in target stream), and y is the corresponding class label. In the source stream, both x_s and y_s are available, while in the target domain only x_t is available. Therefore, a multistream classification with domain adaptation problem can be described as follows:

Suppose X_s is a set of m-dimensional vectors and Y_s is the corresponding labels in a source stream from a certain domain D_s, whereas X_t is a set of n-dimensional vectors in target stream from another domain D_t. In our problem setting, $m \neq n$. The objective is to construct a classifier M that predicts class labels of $x_t \in X_t$ using X_s, and Y_s.

3.2 Challenges

In the problem setting of multistream classification with domain adaptation, there are two major challenges at the same time. The first challenge is the adaptation of both source and target domain, and it can be represented as $D_s \neq D_t$. As shown in Figure 3, two streams may have different number of instances. However, without loss of generality, buffers (windows) from these streams representing contiguous data points will have the same number of instances N_m. Furthermore, each data point in source stream may have a different number of features compared to those in the target stream. Therefore, source data cannot be directly used as training data to learn the target task, and discovering a latent feature space with k dimensions is the key to handling the feature heterogeneity issue. The second challenge is the asynchronous concept drift in both source and target streams. This phenomenon means that the data pattern evolves, or more formally, the conditional probability distribution changes over time in both streams.

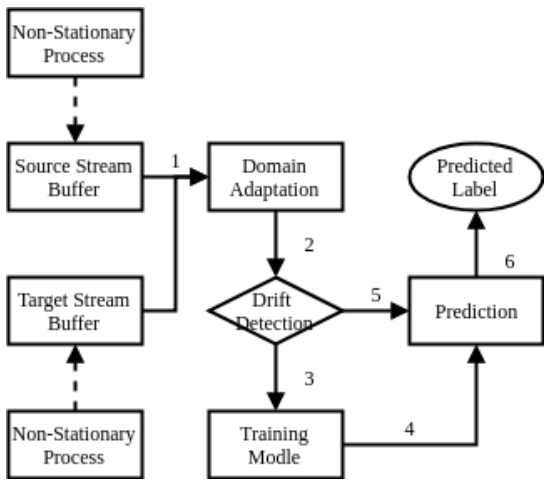


Fig. 4: Multistream Classification with Domain Adaptation

TABLE 1: Notations

Symbol	Meaning
D_s, D_t	Domain of source and target streams
S, T	Source and target stream data
X_s, X_t	Feature space of source stream with dimension m and that of target stream with dimension n
Y_s, Y_t	Label space of source and target stream
x_s, x_t	Data instance of source and target stream
y_s, y_t	True labels of a data instance in source and target stream
h_s, h_t	Encoder function from the source and target to space
g_s, g_t	Decoder function from the source and target to space
L_s, L_t	Projected data from source and target domain to latent space
P_s, P_t	Probability distribution function of source and target data
N_m	Size of sliding window

Here, the problem can be described as $P_s^t(y \mid \mathbf{x}) \neq P_t^t(y \mid \mathbf{x})$ at time t. Furthermore, we assume that source and target data streams have asynchronous concept drifts, which means that the drifts in both streams occur independently.

4 Proposed Approach

In this paper, we propose a Heterogeneous Domain Adaptation Network (HDAN) framework to address the major issues in our problem setting. To achieve this goal, we establish a framework with the following modules:

1) A domain adaptation module that helps find an optimized latent subspace for both source and target streams.
2) A concept drift detection module that detects concept drift in both source and target streams.

Applying these two modules together, once a concept drift is detected, we use data instances from both streams in the most recent window to update the feature mapping, so that the domain adaptation problem can be addressed. The diagram of this algorithm can be found in Figure 1.

Algorithm 1 HDAN Algorithm

Require: Labeled source stream S, Unlabeled target stream T, The size of sliding window N_m. Similarity parameter β.
Ensure: Labels predicted on T.

1: /* Initialization */
2: $B_s, B_t \leftarrow readData(S, T)$
3: /* DA for initial buffer */
4: $h_s, h_t \leftarrow genProjectFunction(B_s, B_t, \beta)$
5: $L_s, L_t \leftarrow genProjectMatrix(B_s, B_t, W_s, W_t)$
6: $M \leftarrow buildModel(L_s, Y_s)$
7: **while** S or T $exists$ **do**
8: $\quad B_s, B_t \leftarrow readData(S, T)$
9: $\quad L_s, L_t \leftarrow genProjectMatrix(B_s, B_t, W_s, W_t)$
10: $\quad$ /* Concept drift detection and correction */
11: $\quad$ Call $ChageDetection$ /* Algorithm 2 */
12: $\quad$ **if** $ChangeDetection = True$ **then**
13: $\quad\quad$ /* Update prediction model */
14: $\quad\quad$ /* DA for stream buffer */
15: $\quad\quad h_s, h_t \leftarrow genProjectFunction(B_s, B_t, \beta)$
16: $\quad\quad L_s, L_t \leftarrow genProjectMatrix(B_s, B_t, W_s, W_t)$
17: $\quad\quad M \leftarrow buildModel(L_s, Y_s)$
18: $\quad$ /* Generate predictions */
19: $\quad \hat{y}_t \leftarrow getPrediction(M, L_t)$

First, a domain adaptation module is triggered to learn projection functions for both source and target data instances (step 1, 2 & line 2-6). Newly arriving instances are transformed to latent feature representation accordingly (line 8-11). Second, the change point detection module detects if there is a significant change in either source or target streams within the sliding windows Third, once a change point is detected, new classifiers are trained based on these two buffers from B_s and B_t (step 3 & line 12-16). Finally, the newly updated classifiers are used to predict the labels of adapted data instances from target stream T (step 4, 5, 6 & line 17-18). Details of our proposed method are as follows.

4.1 Initialization and Domain Adaptation

The general idea of addressing the heterogeneous domain problem is to find domain-agnostic shared-latent space L. Therefore, how to find such space is a challenge, and we need to design a proper framework to find it. Referring to Fig. 6, Stacked Denoising Autoencoder (SDAE) [24], and multi-task learning [25] are fundamentals of our framework. Our proposed framework has a total number of 5 networks: two encoder networks for S and T, two decoder networks for S and T, and a classification network for L. In the following context, networks and structure of our framework will be discussed in details.

4.1.1 Stacked Denoising Autoencoder

The Denoising Autoencoder (DAE) is typically implemented as a one-hidden-layer neural network which is trained to reconstruct a data point $x \in R^d$ from its corrupted version $\tilde{x}$ [24]. It is called "denoising" since corrupted input is used other than original input. This corrupted input $\tilde{x}$ is typically constructed from a conditional distribution $p(\tilde{x}|x)$, where normally multiplicative mask-out noise is applied. It means that with probability q, features of input data may be set to 0, otherwise it is kept unchanged.

Then, the corrupted input $\tilde{x}$ is projected to a latent feature space by encoder h. Here we denote $\mathbf{z} = h(\tilde{x}) \in R^L$ as the

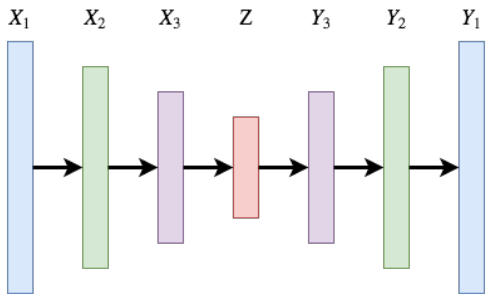


Fig. 5: Structure of a three layer SDAE. Z is the final latent space we generated. Y_i ($i \in \{1, 2, 3\}$) are trained by corresponding X respectively.

L-dimensional representations collected as the output of encoder network h.

After the encoding process, the code $\mathbf{z}$ is decoded in to the network output $\mathbf{y} = g(\mathbf{z}) \in R^d$ as the D-dimensional representation (same dimension as the corrupted input $\tilde{x}$). This decoder connects the latent feature space to the output layer.

As the properties of autoencoder, we want the output of the network, which can be represented as $\mathbf{y} = g(h(\tilde{x}))$, to be as similar as possible to the clean data x. Normally, this process can be formed by using a reconstruction loss function. Given a dataset $D = \{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_q\}$, we optimize the parameters in both encoder and decoder network for p times. Then, the averaged reconstruction loss can be minimized by:

$$\mathcal{L}_{\text{rec}} = \frac{1}{pq} \sum_{i=1}^{p} \sum_{j=1}^{q} l(x, g(h(\tilde{x}_i^j))) \tag{1}$$

where the loss l is the squared loss $||x - y||^2$ since each data instance is a vector composed by real numbers.

The above context describes the training process of a single layer DAE. In our approached, we applied the stacked version of DAE. In order to form this structure, first corrupted inputs $\tilde{x}$ is applied to train a single layer as described above. After getting the parameters θ for the networks, clean data x are applied to the network using trained parameters θ, and encoded $\mathbf{z}$ is used as the input for the next set of single layer DAE. Theoretically, this process can be repeated for as many times as we want, and the structure of SDAE is demonstrated in Fig. 5.

4.1.2 Multi-task Learning

To get a proper joint distribution of source and target domain, five different tasks are designed, which can be seen in Fig. 6. Here **(a)** represents the self-reconstruction loss of D_s and self-reconstruction loss of D_t; **(b)** calculates a circle reconstruction loss that connect D_s and D_t together; **(c)** calculates a circle reconstruction loss that connect D_t and D_s, and; **(d)** is the translation network that translates the label information from D_s to D_t. In the following sections, we will discuss the functionalities of each tasks in details.

Mapping of Source and Target Domain

First, let's discuss the mapping from source domain to the latent space. All $\mathbf{z} \in D_l$ are required to preserve the core information of source domain D_s. To enforce this constraint, here we define the encoder as $h_s : D_s \rightarrow D_l$, and the decoder as $g_s : D_l \rightarrow D_s$. This encoder-decoder structure, as we discussed in Sec. 4.1.1, maps the input data in the source domain to latent space, and take features in latent space back to the source domain. Applying the SDAE, it can be assumed that $\mathbf{z}$ contains core information of source domain and discards the noise. Then the decoder should be able to reconstruct each data instance back from its representation $\mathbf{z}$ in the latent feature space. Therefore, the loss function can be formulated as:

$$\mathcal{L}_{\text{rec\_s}} = \frac{1}{pq} \sum_{i=1}^{p} \sum_{j=1}^{q} l(x, g_s(h_s(x_i^j))) \tag{2}$$

Similarily, for target domain D_t, we are applying the same loss function as Eq. 2, except that we are using data from target domain instead of source domain, and the function can be formulated as:

$$\mathcal{L}_{\text{rec\_t}} = \frac{1}{pq} \sum_{i=1}^{p} \sum_{j=1}^{q} l(x, g_t(h_t(x_i^j))) \tag{3}$$

Cycle Reconstruction Loss

According to the natural of our problem setting, there is no correspondences between the data in source domain and target domain. Therefore, there is a need to ensure the semantic similar for data in both domains. For example, data in D_s are projected into close vicinity of D_t in the latent space. Therefore, a cycle-consistency loss is designed. The basic idea is that using the encoder-decoder structure of D_t to the latent representation generated from D_s, it is still possible to reconstruct the source data instances back by g_s. In other words, the entire cycle should be equivalent to an identity mapping. Given these assumptions, we can construct the cycle reconstruction loss as:

$$\mathcal{L}_{\text{cyc\_s}} = \frac{1}{p} \sum_{i=1}^{p} l(x_i, g_s(h_t(g_t(h_s(x_i))))) \tag{4}$$

$$\mathcal{L}_{\text{cyc\_t}} = \frac{1}{p} \sum_{i=1}^{p} l(x_i, g_t(h_s(g_s(h_t(x_i))))) \tag{5}$$

Translation Loss

To further constrain the translations to maintain the same semantics, and allow encoder-decoder structure to be trained with supervision on target domain data instances, we also define a classification loss from the source to the target translations based on its original label in the source domain:

$$\mathcal{L}_{\text{trans}} = \frac{1}{p} \sum_{i=1}^{p} l(y_i, h_t(g_t(h_s(x_i)))) \tag{6}$$

Total Loss

Finally, by combining these individual loss all together, we have our general loss to be:

$$\mathcal{L} = \mathcal{L}_{\text{trans}} + \mathcal{L}_{\text{cyc\_s}} + \mathcal{L}_{\text{cyc\_t}} + \mathcal{L}_{\text{rec\_s}} + \mathcal{L}_{\text{rec\_t}} \tag{7}$$

4.2 Classification Module

We start to train a classifier with a small set of data instances from both S and T, which are referred to as the initialization data. The new data representation is obtained by using initialization data from B_s and B_t as follows:

$$\begin{cases} L_s^{(i)} = h_s(B_s) & B_s^{(i)} \in B_s \\ L_t^{(j)} = h_t(B_t) & B_t^{(j)} \in B_t \end{cases} \tag{8}$$

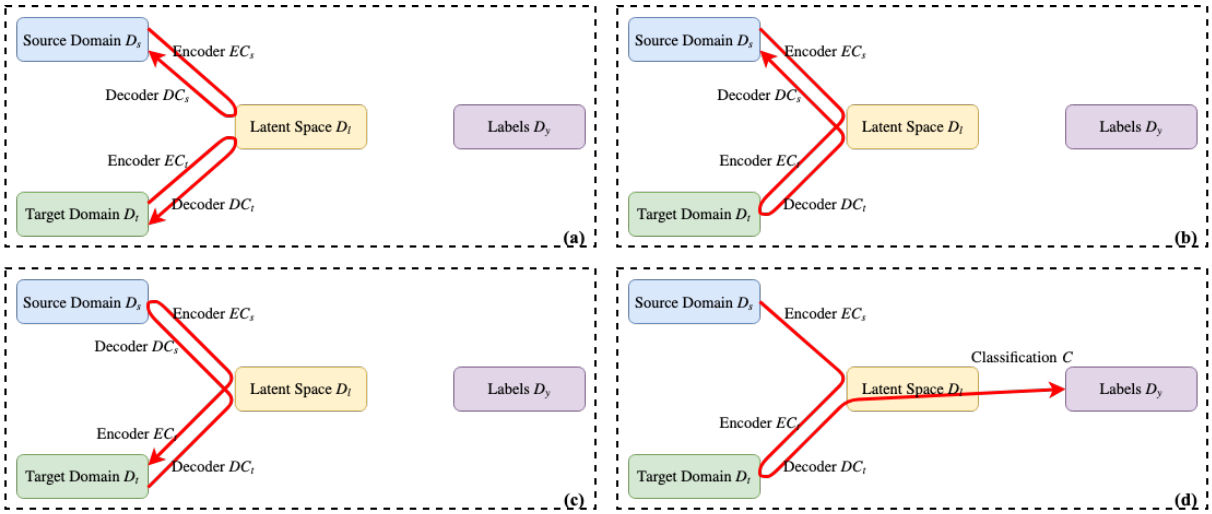


Fig. 6: Different tasks in proposed network

Algorithm 2 ChangeDetection: Change Detection

Require: Source instances $B_s = \{B_s\}^{i=1}$, target instances $B_t = \{B_t\}^{j=1}$, new instance x, initial mean discrepancy $Disc$, the change parameter τ.

Ensure: $True$ if drift is detected, else $False$.

1: /* Instance is from source stream */
2: **if** $x \in S$ **then**
3: $B_s^{(i)}, L_s^{(i)} \leftarrow B_s^{(i+1)}, L_s^{(i+1)}, i = 1, ..., N_m$.
4: $B_s^{(N_m+1)}, L_s^{(N_m+1)} \leftarrow x, h_s(x)$.
5: $\mu_S = \frac{1}{N_m} \sum_{i=1}^{N_m} g_s(L_s^{(i)})$.
6: Go to line 11.
7: /* Instance is from target stream */
8: $B_t^{(N_m+1)}, L_t^{(N_m+1)} \leftarrow x, h_t(x)$.
9: $B_t^{(j)}, L_t^{(j)} \leftarrow B_t^{(j+1)}, L_t^{(j+1)}, j = 1, ..., N_m$.
10: $\mu_T = \frac{1}{N_m} \sum_{j=1}^{N_m} h_t(L_t^{(j)})$.
11: /* Calculate the mean discrepancy $Disc_t$ at time t: */ $Disc_t = \|\mu_s^t - \mu_t^t\|^2$.
12: $s = ln \frac{Disc_t}{Disc_0}$.
13: **Return** $s > -ln(\tau)$.

As new instances arrive in S or T, the classifier is updated if there is a drift to ensure that it represents the current concepts. A new classification model is trained using data in B_s and B_t at that time. Concept drift detection and the updating method used by HDAN will be discussed in the next subsection. HDAN predicts the class label of an incoming target instance from the target stream after projecting the instance into the new data format.

4.3 Change Detection Module (CDM)

Since the data stream in real world applications is dynamically changing all the time, we need to consider

Previous work [4] of multistream classification uses prediction confidence to detect changes in distribution between two different time windows. Due to possible asynchronous concept drifts between source and target streams, an ensemble of classifiers is maintained and updated if a concept drift is detected in either of these streams of data. Therefore, complex ensemble algorithms may lead to very slow execution.

We adopt the Maximum Mean Discrepancy (MMD) as the distance measure to compare different distributions. The distance between two distributions can be computed between the sample means of the two domains in the k-dimensional embeddings:

$$Disc(L_s, L_t) = \left\| \frac{1}{N_m} \sum_{i=1}^{N_m} g_s(L_s^{(i)}) - \frac{1}{N_m} \sum_{j=1}^{N_m} g_t(L_t^{(j)}) \right\|^2 \quad (9)$$

where L_s, L_t are the set of projected data points in the source and target windows.

We invoke a simple but efficient change detection method by monitoring significant changes in B_s and B_t. Since data continuously arrives in the source and target streams, the MMD model in Equation needs to be updated also by updating the mean of adapted data points in the two windows respectively. The online updating process is given by Eq. 8.

Here, we let $Disc_0$ to initialize the mean discrepancy. $Disc_t$ at time t is updated online as new instances arrive in S or T. The difference between the distributions is determined by the log ration between MMD. Therefore, a concept drift point is detected if it is more than a user-defined threshold τ, as follows.

$$S = ln \frac{Disc_t(L_s^t, L_t^t)}{Disc_0(L_s^0, L_t^0)} > \tau \quad (10)$$

Algorithm 2 demonstrates an online updating algorithm for change point detection. If a new instance arrived in the source stream, we update the window L_s by eliminating the oldest instance and storing the newest instance (line 1-6). Then, new arrived data are projected into the new subspace, and the mean of source instances μ_s is updated as well in current window. Otherwise, we update the target window L_t and target mean μ_t instead (line 7-10). At last, the mean discrepancy of two streams and a change sore are calculated (line 11-13).

5 Experiment

5.1 Baseline Methods

To evaluate the effectiveness of our proposed our method HDAN, which uses embedding based methods for domain adaptation, we compare it with several state-of-the-art domain adaptation

TABLE 2: Datasets

Data type	Datasets	# features	# instances
APT detection	Win	28	10587
	Linux	105	10801
Dark web	BlackHat	38	32414
	Nulled	38	9930
	Darkode	67	100,000
	Hack	67	100,000
Digits	USPS	256	10,000
	MNIST	780	60,000
Amazon review	Video	100	5,000
	Electronics	100	30,000
	DVD	200	30,000
	Books	200	30,000
	Music	300	30,000
Synthetic data	Syn01	100	10,000
	Syn02	200	20,000

or online learning frameworks. Also, we applied two different variations of our method, which are HDAN-SVM and HDAN-LR. These two methods are different in a way that classifiers applied are SVM and Logistic Regression. The following paragraphs describe details of baseline methods.

OTL [11]. Online Transfer Learning (OTL) uses a co-regularized method to project target domain data into the source domain. This method is applied in a supervised manner, which means that data in source domain are offline, while data in target domain are online. Furthermore, there is a strong assumption that features in source stream are a subset of those in target stream.

HeMap [10]. Heterogeneous Mapping (HeMap) projects data in two domains with correspondence onto a common latent space. This method is designed as a batch training, which means both source and target data are offline. Also, this method requires class labels in the target dataset.

MSDA [26]. Multi-Stream Domain Adaptation (MSDA) creates projections using a linear transformation function from source and target domains to a common latent embedding space. This approach follows unsupervised training process regarding the target stream and incorporates drift detection mechanism to adapt changes on data stream.

Two classifiers, SVM and LR, are also applied in this method as variations, which are referred as HeMap-S/HeMap-L and MSDA-S/MSDA-L respectively.

5.2 Datasets

We use both synthetic and real-world datasets to evaluate our methods. Specifically, we incorperate security related datasets as well as non security related datasets here to validate our proposed framework. As Table 2 shows, we collect Windows and Linux system calls by ourselves (first two datasets). Then, the following nine datasets are publicly available, and the last two synthetic datasets are generated by MOA [27].

Win and **Linux** This dataset consists of system-call events traces from windows host machine and linux host machines. Each event consist of event types like (e.g., open, read, select), process name and process arguments. We generated various attack instances which consist of attacks such as malicous exlfiltration of data, password theft, setting up a malicious port listening service for command and control, and using **nmap** to scan the system and services on a victim's network. For benign instances, we generated data which include web browsing activities and remote login using ssh.

We collected the system call traces using sysdig trace tool for linux and procmon tool for windows. In total, we collected 10,587 and 10,801 trace instances. We build histograms from these system calls using N-Gram. With N-Gram, we extract features from contiguous sequences of system calls. We extracted 28 uniques features for Windows system and 105 features for linux systems. Our goal is to classify whether a trace in the target stream is an attack.

BlackHat, Nulled, Darkode and **Hack** [28] are datasets extracted from dark web forums where illegal items are listed and sold . Note that all forums here are English forums. The Hack forum contains a mixture of cyber-security and computer gaming blackhat and noncybercrime products; Nulled forum contains data stealing tools and services; Darkode focused on cybercriminal wares, including exploit kits, spam services, ransomware programs, and stealthy botnets; BlackHat focuses on blackhat search engine optimization (SEO) techniques. The forum started in October, 2005 and is still active, although it has changed in character over the past decade. Our goal is to extract words that are products on the dark web forums. Since the forums contains sentences like, "*I want to sell ransomeware tool for $50*". We want to extract which of the words in this sentence are products. To accomplish this, we extracted features for each word and labeled it as a product or not. To satisfy the multistream setting with different domains, for BlackHat and Nulled, we extract the features using Penn Part of Speech Tags [29]. While, for Darkode and Hack datasets, features are extracted following the Stanford pos- tagger system [30]. Since forum posts are posted within a long time period, these streams satisfies the concept drift assumption.

USPS [31] and **MNIST** [32] contain gray-scale images of hand-written digits collected from different sources. In order to satisfy the concept drift assumption in this paper, we shift the concept of positive and negative classes in the middle of datasets. That is, in the first half of both USPS and MNIST, labels 0-4 represent "-" and labels 5-9 represent "", while in the second half class labels 3-7 mean "+" and the rest class labels mean "-". Therefore, the concept drift assumption of data stream is satisfied in this setting.

Music, **DVD**, **Video**, **Books** and **Electronics** [33] are text datasets generated by Amazon reviews based on product categories. Features are extracted from raw review text by implementing word2vec model proposed by Google [34]. We label scores with ratings greater than 3 is defined as"+", and those smaller than 3 are defined as "-". Scores that equals to 3 is discarded due to ambiguous polarity. Meanwhile, weighted average sentence embedding [35] is applied to represent word vectors, which provide more weight to uncommon words. Since data in this stream covers more than one year, we assume that there are concept drift along the change of time by nature in these data streams.

Syn01, **Syn02** [27] are synthetic datasets that are generated by MOA framework. These datasets are generated in a way that the number of features in Syn01 is 100 while that in Syn02 is 200, so that our domain adaptation assumption can be satisfied. Meanwhile, we artificially add abrupt concept drift at the half position in these data streams.

5.3 Experiment Setup

Our HDAN approach involves multiple parameters. We use $N_m = 400$ as our default setting in the experiment. Meanwhile, $k = 6$, and $\tau = 0.1$ are selected to conduct our initial experiment

TABLE 3: Comparison of performance on cyber security datasets (Error %)

Dataset	OTL	HeMap-S	HeMap-L	MSDA-S	MSDA-L	HDAN-S	HDAN-L
Win → Linux	28.53 ± 0.88	30.55 ± 0.86	29.08 ± 0.58	23.73 ± 0.62	21.33 ± 1.62	19.91 ± 0.47	19.59 ± 0.55
BlackHat → Darkode	22.43 ± 0.85	24.70 ± 1.00	25.36± 1.00	22.12 ± 1.29	20.82 ± 0.64	19.37 ± 0.70	18.73 ± 0.74
Nulled → Hack	28.86 ± 1.54	26.78 ± 0.95	29.26 ± 0.50	25.47 ± 1.06	24.75 ± 1.36	21.37 ± 0.89	21.50 ± 0.68

TABLE 4: Comparison of performance on non cyber security dataset (Error %)

Data Type	Dataset	OTL	HeMap-S	HeMap-L	MSDA-S	MSDA-L	HDAN-S	HDAN-L
Image	USPS → MNIST	32.31 ±0.94	38.19 ±0.75	39.04 ±0.68	28.72 ±0.56	29.96 ±0.64	23.19 ± 1.09	23.44 ± 0.52
Text	Music → DVD	33.54 ±0.69	36.06 ±0.51	35.85 ±0.77	32.91 ±0.49	31.64 ±0.57	26.81 ± 0.60	27.36 ± 0.63
	Video → Music	38.01 ±1.03	40.41 ±0.65	39.72 ±0.85	31.76 ±0.72	32.02 ±0.75	25.14 ± 0.76	24.63 ± 0.52
	Video → DVD	35.88 ±0.88	40.28 ±0.52	40.09 ±0.61	32.35 ±0.49	33.84 ±0.65	25.33 ± 0.61	25.64 ± 0.75
	Electronics → Books	31.19 ±0.98	32.94 ±0.77	32.60 ±0.89	28.93 ±0.64	29.25 ±0.81	21.53 ± 0.57	23.27 ± 0.49
	Electronics → Music	32.20 ±1.24	33.72 ±0.65	33.56 ±0.48	28.42 ±0.73	28.87 ±0.92	21.30 ± 0.67	22.20 ± 0.81
Synthetic	Syn01 → Syn02	37.53 ±1.37	41.83 ±0.83	42.12 ±1.06	33.50 ±0.97	35.47 ±0.92	26.48 ± 1.13	28.11 ± 0.86

here. The dropout rate is $p = 0.5$ and the number of layers in autoencoder is 3. Additionally, we apply he ADAM optimizer with learning rate of 0.0001, betas 0.5 and 0.999. The sensitivity of parameters will be discussed later in this section.

5.4 Result Analysis

Figure 7 shows the distribution of features in the latent space. For a single window (N_m = 400), we plot the transformed source and target window together in one figure. Also, t-SNE is applied here to reduce the feature space to 2D. This figure is actually a result corresponding to Figure 2. From this figure, we can see that binary classes from both source and target stream are well separated in two part of the latent space. This good separation makes it possible to find a hyper-plane in the space between two classes, which gives us confident and evidence to apply classifiers in this feature space.

Table 4 shows the average prediction error on the target stream T: $\frac{A_{wrong}}{m}$, where A_{wrong} is the total number of instances identified incorrectly, and m is the number of instances in the target stream. For security related datasets, for instance, the APT attack detection datasets, our method has 19.59% error rate, which outperforms all other benchmark algorithms. It has a better performance as the projected latent feature space. Also, our proposed framework has the best performance on identifying items on dark web forum datasets. Regarding non security datasets, which are digits recognition and Amazon review categorizations, we can tell

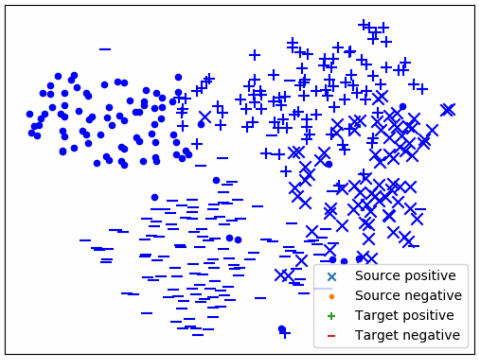


Fig. 7: Latent Feature Embedding USPS→MNIST

that HDAN-S outperforms competing methods on almost every datasets. Also, due to the fact that digits datasets are indeed very similar (all hand-writing digits), which means that the source and target domain are pretty similar in this case, our algorithm shows best performance not only comparing to other benchmark methods, but also among all non security datastes. Furthermore, we can observe that our method works better when adapting from small feature space to larger feature space.

For example, the error rate of OTL, HeMap-S and MSDA-S on USPS → MNIST are 32.31%, 38.19% and 28.72% respectively. We can see that the error rate of our proposed method is 23.19%, which has better performance by a significant margin compared to baseline methods. The reason is that for OTL, the algorithm has a strong assumption that features in the source data are a subset of those in the target stream, which is not quite the case here. Text dataset used in this paper, such as reviews for Video and DVD, would have overlapping features in terms of data distributions. However, not all features in Video would exist in the DVD dataset in our settings. When it comes to HeMap, it also has its own shortcomings. HeMap is an offline method, which makes it not able to adapt the concept drift assumption in our problem. In this dataset specifically, the concept in the second half of data shifts from the first half of data (described in datasets section). Since we applied periodic updates every 4000 instances for the HeMap method, there is a delay on updating the model comparing our proposed HDAN-S approach. When it comes to MSDA-S, it is essentially trying to find a linear projection between the source/target domains and the latent domain. On the other hand, the denoising autoencoder structured used in HDAN-S is much more complex than a linear projection function and can help extract nonlinearities between feature spaces. Therefore, representations learned in the latent space by HDAN can be better separated by classifiers regarding the positive and negative instances.

In synthetic datasets, we observe similar phenomenon. Since the MOA framework can help generate concept drifts in the data stream, the two variations of HeMap don't perform well on this dataset. On the other hand, OTL seems to track the concept drift along the stream, however, since the dataset doesn't quite satisfy its assumption, the overall performance of HDAN-S beats other benchmarks by a significant margin.

5.5 Sensitivity Analysis

The results of our proposed method are further analyzed by tuning defined parameters N_m, n, k, and τ. All experiments are conducted on USPS → MNIST dataset. In this section, we vary

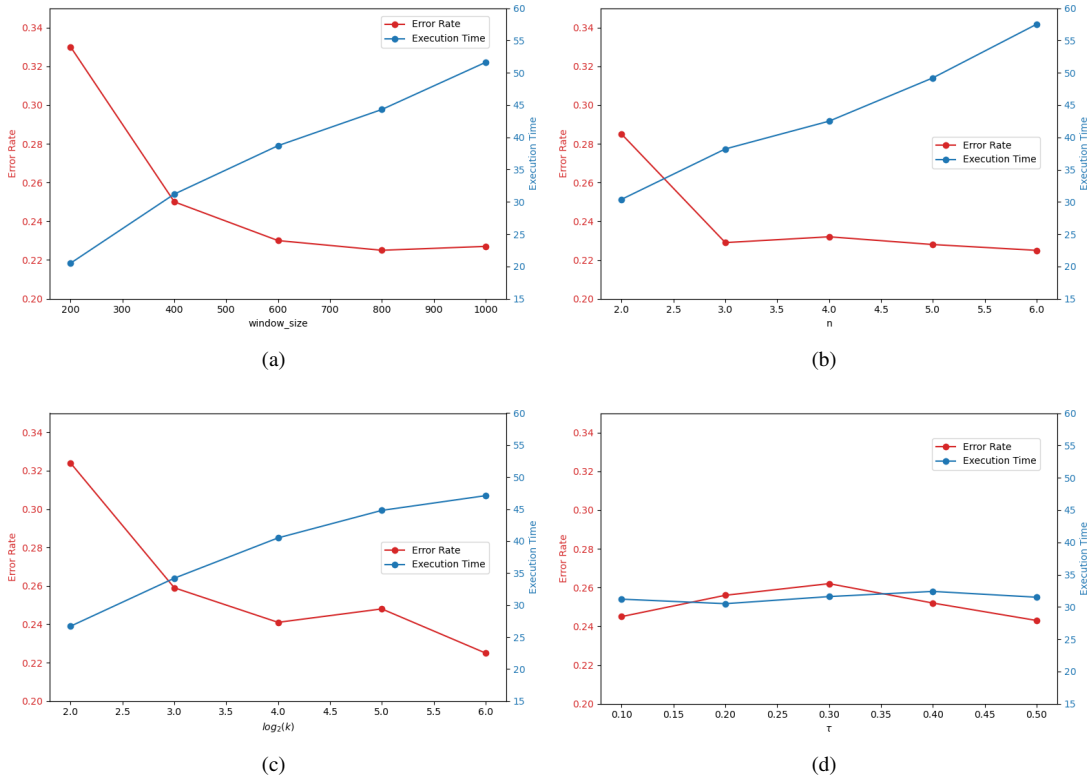


Fig. 8: Sensitivity Experiment Results of HDAN on USPS→MNIST

N_m by setting it to {200, 400, 600, 800, 1000}, n to {2, 3, 4, 5, 6}, k to {4, 8, 16, 32, 64}, and τ to {0.1, 0.2, 0.3, 0.4, 0.5} respectively.

First, the parameter that controls window size (N_m) is set to different values, and results are shown regarding how they affect HDAN approach in Figure 8. We can see that the average error decreases along with window size increases, while the execution time increases with increasing window size. This is close to our expectation, as we state that the execution time of HDAN depends on N_m. Based on this observation, we should choose a medium value so that both error rate and execution time could be balanced.

Second, n determines the number of layers in the denoising autoencoder structure. Essentially, this parameter determines the level of non-linearity that can be extracted from both source and target domain to the latent feature space. From Figure 8, we can tell that the error rate decreases significantly when n increases from 2 to 3. If $n > 3$, the decreasing trend of error rate is becoming not significant. Also, we can see from the figure that the execution time regarding different n remains similar in the experiment. This experiment tells us that 3-layers of encoder-decoder structure can provide the model enough capability of learning a good representation in the latent space. Thus, we choose $n = 3$ as our recommended parameter here.

Third, considering the parameter that defines the dimension of latent feature space k. This parameter influence the model in both embedding and classification. In other words, this parameter determines how large the feature space is to apply our classifiers. From Figure 8, we can see that the performance of the model has improved marginally while the execution time increases dramatically as k increases. Thus, we need to choose as small k as possible, meanwhile we need to make sure that performance doesn't degrade. Thus, we choose $k = 16$ for our approach.

At last, we can see that for τ, which controls the threshold for concept drift detection. In this case, the performance of model doesn't quite change when τ varies. Therefore, based on performance of execution time, we need to choose a smaller τ. Hence, we select $\tau = 0.1$ in our model.

6 Conclusions

In this paper, a multistream classification framework that incorporates domain adaptation techniques is proposed. This framework has the ability to address a wide range of problems, more specifically, detecting attacks, recognizing digits, and etc. Two major challenges, namely heterogeneous domain and concept drift, are addressed simultaneously in two data streams. Our solution involves a stacked denoising autoencoder based approach for domain adaptation, and an online update mechanism using average mean discrepancy for concept drift correction. More specifically, the key idea of the approach is to find a common latent space for both source and target streams, which preserves the structure of data and maximizes the similarities between source and target data. Additionally, the prediction model is updated if a concept drift is detected, which happens when a likelihood ratio is greater than the user-defined threshold τ. Extensive experiments with both real-world and synthetic data show that our approach has significantly better performance in terms of error rate on various datasets, compared to existing state-of-the-art solutions.

Disclaimer

procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the identified products are necessarily the best available for the purpose.

REFERENCES

[1] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Learning with feature evolvable streams," in *Advances in Neural Information Processing Systems*, 2017, pp. 1417–1427.

[2] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream." in *AAAI*, 2016, pp. 1652–1658.

[3] M. Herlihy, "A methodology for implementing highly concurrent data objects," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 15, no. 5, pp. 745–770, 1993.

[4] S. Chandra, A. Haque, L. Khan, and C. Aggarwal, "An adaptive framework for multistream classification," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 1181–1190.

[5] A. Arnold, R. Nallapati, and W. W. Cohen, "A comparative study of methods for transductive transfer learning," in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE, 2007, pp. 77–82.

[6] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[7] Y.-R. Chen and H.-H. Chen, "Opinion spam detection in web forum: a real case study," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 173–183.

[8] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 114.

[9] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1-2, pp. 151–175, 2010.

[10] X. Shi, Q. Liu, W. Fan, S. Y. Philip, and R. Zhu, "Transfer learning on heterogenous feature spaces via spectral transformation," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1049–1054.

[11] P. Zhao and S. C. Hoi, "Otl: A framework of online transfer learning," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 1231–1238.

[12] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*, 2007, pp. 443–448.

[13] A. Haque, H. Tao, S. Chandra, J. Liu, and L. Khan, "A framework for multistream regression with direct density ratio estimation," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7*, 2018.

[14] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*, 2004, pp. 286–295.

[15] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[16] J. B. Cabrera, L. Lewis, and R. K. Mehra, "Detection and classification of intrusions and faults using sequences of system calls," *Acm sigmod record*, vol. 30, no. 4, pp. 25–34, 2001.

[17] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of data mining in computer security*. Springer, 2002, pp. 77–101.

[18] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *null*. IEEE, 2003, p. 14.

[19] X. Shu, D. Yao, and N. Ramakrishnan, "Unearthing stealthy program attacks buried in extremely long execution paths," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 401–413.

[20] K. Al-Naami, S. Chandra, A. Mustafa, L. Khan, Z. Lin, K. Hamlen, and B. Thuraisingham, "Adaptive encrypted traffic fingerprinting with bi-directional dependence," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 2016, pp. 177–188.

[21] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Ieee communications surveys & tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[22] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[23] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross, "Immune system approaches to intrusion detection–a review," *Natural computing*, vol. 6, no. 4, pp. 413–466, 2007.

[24] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv preprint arXiv:1206.4683*, 2012.

[25] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4500–4509.

[26] Y.-F. Li, Y. Gao, G. Ayoade, H. Tao, L. Khan, and B. Thuraisingham, "Multistream classification for cyber threat data with heterogeneous feature space," in *The World Wide Web Conference*, 2019, pp. 2992–2998.

[27] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[28] R. S. Portnoff, S. Afroz, G. Durrett, J. K. Kummerfeld, T. Berg-Kirkpatrick, D. McCoy, K. Levchenko, and V. Paxson, "Tools for automated analysis of cybercriminal markets," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 657–666.

[29] B. Santorini, "Part-of-speech tagging guidelines for the penn treebank project (3rd revision)," *Technical Reports (CIS)*, p. 570, 1990.

[30] M.-C. De Marneffe and C. D. Manning, "The stanford typed dependencies representation," in *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*. Association for Computational Linguistics, 2008, pp. 1–8.

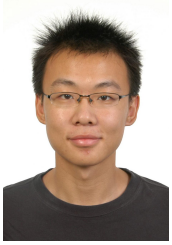
[31] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification," in *Proceedings of the 45th annual meeting of the association of computational linguistics*, 2007, pp. 440–447.

[34] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[35] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.



Yi-Fan Li Yi-Fan Li is currently working toward the PhD degree in the CS department at University of Texas at Dallas. His research interests include stream data mining, deep learning and transfer learning. His current project are about adapting heterogeneous domain knowledge over non-stationary data streams under both CV and NLP settings. He is a member of IEEE.

Yang Gao Yang is currently a Ph.D student, and a member of the Big Data Analytics and Management Lab at University of Texas at Dallas. His research interest are in the area of stream mining, deep learning and deep reinforcement learning, especially the learning and inference problems in non-stationary settings. His recent projects include improving bias correction mechanisms over drifting data streams and designing self-adaptive deep metric learning models.

Gbadebo Ayoade Gbadebo Ayoade is a PhD graduate form the University of Texas at Dallas. His research interest ranges from big data systems, cyber security and applied machine learning particularly in the areas of advanced persistent attack detection. He is currently a Software Engineer at Google.

Latifur Khan Dr. Latifur Khan is currently a full Professor (tenured) in the Computer Science department at the University of Texas at Dallas, USA where he has been teaching and conducting research since September 2000. He received his Ph.D. degree in Computer Science from the University of Southern California (USC) in August of 2000.

Dr. Khan is an ACM Distinguished Scientist and received Fellow of SIRI (Society of Information Reuse and Integration) award in Aug, 2018. He has received prestigious awards including the IEEE Technical Achievement Award for Intelligence and Security Informatics and IBM Faculty Award (research) 2016.

Dr. Latifur Khan has published over 250 papers in premier journals such as VLDB, Journal of Web Semantics, IEEE TDKE, IEEE TDSC, IEEE TSMC, and AI Research and in prestigious conferences such as AAAI, IJCAI, CIKM, ICDE, ACM GIS, IEEE ICDM, IEEE BigData, ECML/PKDD, PAKDD, ACM Multimedia, ACM WWW, ICWC, ACM SACMAT, IEEE ICSC, IEEE Cloud and INFOCOM. He has been invited to give keynotes and invited talks at a number of conferences hosted by IEEE and ACM. In addition, he has conducted tutorial sessions in prominent conferences such as SIGKDD 2017, 2016, IJCAI 2017, AAAI 2017, SDM 2017, PAKDD 2011 2012, DASFAA 2012, ACM WWW 2005, MIS2005, and DASFAA 2007.

Currently, Dr. Khan's research area focuses on big data management and analytics, data mining and its application over cyber security, complex data management including geo-spatial data and multimedia data. His research has been supported by grants from NSF, the Air Force Office of Scientific Research (AFOSR), DOE, NSA, IBM and HPE.

Thuraisingham Bhavani Dr. Bhavani Thuraisingham is the Louis A. Beecherl, Jr. Distinguished Professor of Computer Science and the Executive Director of the Cyber Security Research and Education Institute (CSI) at The University of Texas at Dallas. She is an elected Fellow of IEEE, the AAAS, the British Computer Society, and the SPDS (Society for Design and Process Science). She received several prestigious award including IEEE Computer Society's 1997 Technical Achievement Award for "outstanding and innovative contributions to secure data management", the 2010 ACM SIGSAC (Association for Computing Machinery, Special Interest Group on Security, Audit and Control) Outstanding Contributions Award for "seminal research contributions and leadership in data and applications security for over 25 years" and the SDPS Transformative Achievement Gold Medal for her contributions to interdisciplinary research. She has unique experience working in the commercial industry (Honeywell), federal research laboratory (MITRE), US government (NSF) and academia and her 34+ year career includes research and development, technology transfer, product development, program management, and consulting for the federal government. Her work has resulted in 100+ journal articles, 200+ conference papers, 100+ keynote and featured addresses, eight US patents (three pending) and fifteen books (two pending). She received the prestigious earned higher doctorate degree (DEng) from the University of Bristol England in 2011 for her published work in secure data management since her PhD.

Anoop Singhal Dr. Anoop Singhal, is currently a Senior Computer Scientist and a Program Manager in the Computer Security Division at the National Institute of Standards and Technology (NIST) in Gaithersburg, MD. He has several years of research experience at NIST, George Mason University and ATT Bell Labs. His research interests are in system security, active cyber defense, network forensics, cloud computing security and machine learning systems. He is a member of ACM, senior member of the IEEE and he has co-authored over 50 technical papers in leading conferences and journals. He has taught several graduate level courses in Computer Science as an adjunct faculty and given talks at RSA, IEEE and ACM conferences. He has two patents in the area of attack graphs and he has also co-edited a book on Secure Cloud Computing and Network Security Metrics.