Evaluating Transfer Learning Methods on Real-World Data Streams: A Case Study in Financial Fraud Detection

Ricardo Ribeiro Pereira[1,2] (✉), Jacopo Bono[1], Hugo Ferreira[1], Pedro Ribeiro[2], Carlos Soares[2], and Pedro Bizarro[1]

[1] Feedzai, Portugal
[2] University of Porto, Portugal
ricardo.ribeiro@feedzai.com

Abstract. When the available data for a target domain is limited, transfer learning (TL) methods leverage related data-rich source domains to train and evaluate models, before deploying them on the target domain. However, most TL methods assume fixed levels of labeled and unlabeled target data, which contrasts with real-world scenarios where both data and labels arrive progressively over time. As a result, evaluations based on these static assumptions may not reflect how methods perform in practice. To support a more realistic assessment of TL methods in dynamic settings, we propose an evaluation framework that (1) simulates varying data availability over time, (2) creates multiple domains via resampling of a given dataset and (3) introduces inter-domain variability through controlled transformations, e.g., including time-dependent covariate and concept shifts. These capabilities enable the systematic simulation of a large number of variants of the experiments, providing deeper insights into how algorithms may behave when deployed. We demonstrate the usefulness of the proposed framework by performing a case study on a proprietary real-world suite of card payment datasets. To support reproducibility, we also apply the framework on the publicly available Bank Account Fraud (BAF) dataset. By providing a methodology for evaluating TL methods over time and in different data availability conditions, our framework supports a better understanding of model behavior in real-world environments, which enables more informed decisions when deploying models in new domains.

Keywords: Evaluation Framework, Transfer Learning, Fraud Detection

1 Introduction

Machine learning (ML) models often require large volumes of labeled data to achieve strong predictive performance. However, in many real-world applications, obtaining sufficient labeled data can be difficult and costly. Transfer learning (TL) addresses this challenge by leveraging knowledge from one or more source domains to improve performance on a target domain with limited data. Most

TL methods and evaluation protocols assume fixed conditions regarding the availability of labeled and unlabeled data, such as having a large labeled source dataset and only unlabeled target data. However, in many real-world industry settings, these conditions are not permanent, as data from the various domains is progressively collected and labeled over time.

One example of this setting is financial fraud detection. This task involves monitoring streams of financial transactions from different financial institutions, *domains* in the TL terminology, and classifying each transaction as fraudulent or legitimate. New institutions may initially lack historical data, but typically the volume of financial transactions quickly increases over time. However, labeling a transaction as fraudulent often depends on customer complaints and/or manual reviews by analysts, leading to a delay between the moment a transaction is recorded and when it is labeled. This delay can range from several days to a few months, affecting the training and evaluation of ML models. While TL can in principle help mitigate the issues of having insufficient data at the onset, and insufficient labeled data at a later stage, the evolving nature of the data availability itself presents an additional challenge. TL methods are designed for fixed conditions and their performance is expected to change significantly when those conditions are violated. However, they are typically evaluated under those fixed (and favorable) conditions, which would lead to unrealistic expectations concerning their performance in real world settings. The problem therefore remains on how to evaluate TL methods in a way that reflects these dynamic data constraints, such as those encountered in fraud detection.

To address this challenge, we propose an evaluation framework that captures the dynamic nature of data streams in real-world applications. Our framework provides three key capabilities: (1) creating multiple domains from a given dataset through resampling, enabling systematic TL evaluation even when few datasets are available; (2) applying transformations to the data, hence reproducing realistic data shifts over time and across domains, while also introducing controlled variability across experiments; and (3) simulating the gradual arrival of data and labels over time, mimicking the evolving nature of industry environments. These combined features enable our framework to systematically generate a large number of experiments, making it possible to assess TL methods across a wide range of realistic scenarios.

We perform a case study using our framework on a suite of proprietary real-world datasets containing payment events from multiple financial institutions. This case study demonstrates how insights derived from our evaluation framework can inform practical decisions, such as model selection, deployment timing, and the prioritization of data collection efforts. Given the confidential nature of the case study dataset, we perform a similar analysis on the publicly available Bank Account Fraud (BAF) dataset [13], which consists of synthetic examples of account opening applications. The source code that implements the evaluation framework, along with the configurations used for the experiments on the public dataset, are available at https://github.com/feedzai/tred.

The remainder of this paper is structured as follows: Section 2 formalizes our problem setting and compares it with traditional TL setups studied in academia; Section 3 introduces the design of our evaluation framework and its key components; Section 4 describes how we apply the framework in practice, detailing the datasets, experimental setup, and TL methods evaluated; Section 5 presents the results and their practical implications in an industry setting; and Section 6 summarizes our contributions and highlights the broader impact of our work.

2 Background and related work

In this section, we formalize the problem setting and introduce the notation used throughout the paper (Section 2.1). We then review traditional TL paradigms, highlighting their assumptions and differences from our use case (Section 2.2). Finally, we discuss common evaluation strategies for TL and motivate the need for a new framework that better captures real-world data dynamics (Section 2.3).

2.1 Problem definition

We consider the machine learning setting where data is collected from multiple domains over time, with labels becoming available after a delay. This is a common scenario in many real-world applications, such as fraud detection, where instances (e.g., transactions) are initially unlabeled and only later confirmed as fraudulent or legitimate. To formalize this problem, we assume there are m source domains $\mathcal{D}_{S_1}, \ldots, \mathcal{D}_{S_m}$ and a target domain $\mathcal{D}_T$. Each domain $\mathcal{D}_d$ (including the target) is associated with a dataset $D_d = \{(x_i, y_i, t_i^x, t_i^y) \mid i = 1, \ldots, n_d\}$, where $x_i \in \mathcal{X}_d$ is a feature vector, $y_i \in \mathcal{Y}_d$ is the label, t_i^x is the timestamp when x_i is collected, and $t_i^y \geq t_i^x$ is the timestamp when y_i becomes available.[3] At any given time t, D_d can be decomposed into a labeled dataset $D_d^L(t) = \{(x_i, y_i) \mid t_i^y \leq t\}$ which consists of all instances that have already received their labels by time t, and an unlabeled dataset $D_d^U(t) = \{x_i \mid t_i^x \leq t < t_i^y\}$ which consists of instances that have been observed but their labels are still unavailable at time t.

Eventually, at some time t_a, the target domain $\mathcal{D}_T$ is introduced, initially without any data ($D_T^L(t_a) = D_T^U(t_a) = \emptyset$), and target domain data and labels begin to be collected from that point on. Our goal is to leverage $D_{S_1}, \ldots, D_{S_m}$ and D_T to learn a predictive function $f_T : \mathcal{X}_T \to \mathcal{Y}_T$ that approximates $P_T(Y|X)$. Over time, as more data and labels become available, f_T can be updated to improve its approximation of $P_T(y \mid x)$.

2.2 Transfer learning paradigms

Different TL paradigms have been explored, each making different assumptions about the datasets used to train the ML models. All these paradigms assume

[3] To simplify notation, we will sometimes use the letter i to index the entries of the dataset without explicitly stating $i = 1, \ldots, n_d$.

that a large volume of labeled data is available from the source domains. Their main difference relates to the available target domain data at training time.

In Domain Generalization (**DG**) [24, 28], the goal is to use the source domain datasets to learn a predictive function f that generalizes to the target domain without access to any data from $\mathcal{D}_T$. As such, at training time, $D_T^L = D_T^U = \emptyset$. In Unsupervised Domain Adaptation (**UDA**) [26], in addition to the source domain datasets, there is an unlabeled target domain dataset that can be used to adapt the predictive function f_T to the target domain $\mathcal{D}_T$. This means that, at training time, $|D_T^U| > 0$ while $D_T^L = \emptyset$. In Supervised Domain Adaptation (**SDA**) [25], in addition to the source domain datasets, there is both a large unlabeled dataset and a small labeled dataset from the target domain, which are used to adapt the predictive function f_T to the target domain $\mathcal{D}_T$. As such, at training time, $|D_T^U| \gg |D_T^L| > 0$. In Multi-Domain Learning (**MDL**) [27], the goal is to use datasets from multiple domains to learn a single predictive function f that performs well across all observed domains simultaneously. Here, at training time, labeled data is available from all domains, i.e., $\forall d, |D_d^L| \gg 0$.

Each of these paradigms operates under specific assumptions about data availability, but none of them account for the progressive collection of data and possible label delay. In contrast, our problem setting requires a framework that can systematically model the evolving availability of data and labels over time.

2.3 Evaluation of TL methods

Various datasets have been used to evaluate TL methods, under the different paradigms discussed in the previous section. Most TL benchmarks focus on image classification, including datasets such as Office-31 [21], Office-Caltech10 [8], Office-Home [23], DomainNet [19], and PACS [17]. Beyond computer vision, the Amazon Reviews dataset [1] is often used for sentiment analysis.

Another common strategy is to evaluate TL methods across different datasets of the same task. Examples include: digit classification (USPS [12], MNIST [16], SVHN [18]); large-scale image recognition (ImageNet [5], Caltech [9], CiFAR [15]); and semantic segmentation (CityScapes [4], GTA5 [20]).

Additionally, some tools have been developed to facilitate the evaluation of TL methods in specific fields. One example is DomainATM [10], an open-source MATLAB package for domain adaptation in medical data analysis. It provides dataset management functionalities, visualization tools, and a collection of domain adaptation methods with built-in evaluation capabilities.

However, both DomainATM and traditional TL benchmarks assume a static evaluation setting, where data availability conditions remain fixed. This assumption overlooks the temporal dynamics present in real-world applications, such as fraud detection, where data and labels arrive progressively over time. As a result, existing evaluation strategies are insufficient for assessing TL methods in dynamic environments. Addressing this gap requires a framework that systematically models the evolving availability of data and labels, enabling more realistic evaluations that reflect real-world deployment scenarios.

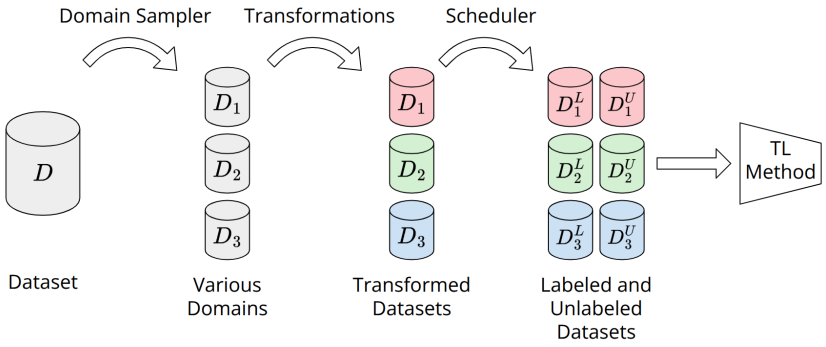

Fig. 1. The evaluation framework is composed of three sequential components: domain sampler, transformations, and scheduler. The number of domains depicted in the diagram is just an example.

3 Method

Our evaluation framework consists of three components (Figure 1). First, the *domain sampler* builds multiple domains from a single dataset, enabling systematic TL evaluation even when few datasets are available. Next, the *transformations* introduce controlled variations to each domain, to reproduce real-world data shifts over time and across domains. Finally, the *scheduler* simulates the progressive arrival of data and labels, enabling the evaluation of TL methods under diverse data availability conditions. We describe each component with more detail in the following subsections.

3.1 Domain Sampler

The *domain sampler* creates domains from a dataset by randomly selecting anchor instances, followed by resampling the events according to the distance to these anchors, as illustrated in Figure 2.

More formally, the *domain sampler* is a stochastic process that receives a dataset $D = \{(x_i, y_i, t_i^x, t_i^y)\}$, a distance function δ, a real number λ, and a positive integer k, and outputs a set of datasets $\{D_1, \ldots, D_k\}$ where $D_d \subseteq D$ for all $d \in \{1, \ldots, k\}$. To extract each D_d, the *domain sampler* first selects an instance x_{anchor}. Then, each instance x_i is assigned a probability of being included in this domain, which decreases exponentially with its distance to x_{anchor},

$$P(x_i | x_{\text{anchor}}) = e^{-\lambda \delta(x_i, x_{\text{anchor}})}.$$

The decay rate of the exponential is controlled by the scaling factor λ, which regulates the expected domain size. Finally, instances are sampled randomly according to their respective probability.

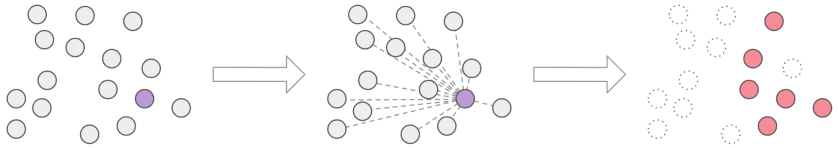

Fig. 2. Toy example of sampling a domain from dataset. First, an anchor instance is selected (purple point). Then, the distances to all other instances are computed. Lastly, the instances are sampled with probability decreasing as distance increases.

3.2 Transformations

The *transformations* apply controlled modifications to the datasets. These transformations are defined by the user to better suit their setting. For example, transformations that may make sense in the image domain would not be suitable for tabular data and vice-versa. Each *transformation* should ideally be parameterized differently for each domain, provoking some level of domain shift. Furthermore, they can be designed to depend on the timestamp of the instance, which effectively simulates data drift over time or seasonalities.

Each *transformation* can be described as a function $\Phi_\theta : (x, y, t^x, t^y) \mapsto (x', y', t'^x, t'^y)$, parameterized by θ. This general formulation allows the instantiation of various types of changes, for example:

- covariate shift (change in $P(X)$): $x' = \phi(x; \theta)$;
- concept shift (change in $P(Y|X)$): $y' = \phi(x, y; \theta)$;
- data drift (change in $P(X)$ over time): $x' = \phi(x, t^x; \theta)$.

If the transformations are parameterized differently for each experiment, the results will express a distribution of each methods' performance on related settings, increasing the robustness of the results. We describe a set of *transformations* for tabular data in detail in Section 4.3 (as well as making them available with our code) and provide a toy example in Figure 3.

3.3 Scheduler

The *scheduler* orchestrates two processes: (1) the progressive arrival of instances and labels over time and (2) the performance estimation over time.

The first process (progressive data arrival) is achieved by discretizing the time range of the target dataset in contiguous periods. At each step, the test period advances, while the training set expands to include all data up to that point. More formally, the *scheduler* receives datasets $D_{S_1}, \ldots, D_{S_m}, D_T$ and a sequence of user-defined timestamps $t_1, \ldots, t_l$ s.t. $\min(t_i^x) \leq t_1 < \ldots < t_l \leq \max(t_i^x)$ for $t_i^x \in D_T$. At each time step t_a for $a = 1, \ldots, l-1$, it decomposes all source and target domain datasets D_d into $D_d^L(t_a)$ and $D_d^U(t_a)$, as described in Section 2.1.

The second process (performance estimation) is achieved by leveraging the data splits that result from the first process to train the TL methods under study and evaluate them on the target domain instances s.t. $t_a \leq t_i^x < t_{a+1}$.[4]

[4] Notice that the label delay is ignored for the purpose of evaluating the methods.

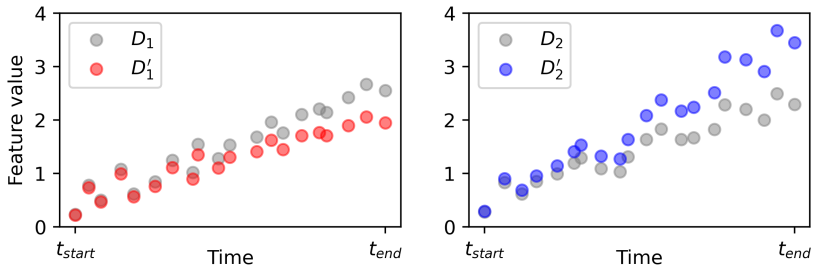

Fig. 3. Toy example showing how applying the same transformation with different parameters affects the same feature of datasets from two domains.

Since this process is repeated for each time step t_a, every model is evaluated multiple times throughout the evolving target dataset. This allows for an analysis of performance trends over time, highlighting how different TL methods adapt to increasing data availability. Figure 4 presents an example of this scheduling.

4 Experimental Setup

In the previous section, we introduce the general architecture of the framework, which is designed to be broadly applicable to many real-world scenarios. In this section, we detail how we apply this framework to our specific use case, including a description of the datasets that we use, the methods that we test and other design decisions that are specific to our experimental setup.[5]

4.1 Datasets

In this section, we provide details about the dataset used in our case study, but due to confidentiality constraints, we can only share general metrics. The Acquirers is a real-world proprietary dataset, containing payment events from 4 different financial institutions (*domains*) over a period of 41 weeks ($\approx$ 9 months). Each domain has $\approx$ 5M events, but their fraud rates (relative frequency of the positive class) vary between $\approx 0.01\%$ and $\approx 0.4\%$. Each instance has 58 features (52 numerical and 6 categorical), the event timestamp and the fraud label.

Because the above dataset is confidential, we also perform experiments on the publicly available Bank Account Fraud (BAF) dataset [13]. BAF is a publicly available synthetic bank account fraud dataset.[6] It contains one million examples of account opening applications, some of which are fraudulent, from February through September. Each instance has 28 features (24 numerical and 4 categorical), the time information and the label.

[5] Implementation details and code are available at https://github.com/feedzai/tred.

[6] In fact, the authors published 6 different variations of this dataset, but we just use the "Base" variant without device_fraud_count and device_os features.

In both experiments, each numerical feature from each domain is standardized to have 0 mean and 1 standard deviation. Also, each categorical feature is label encoded [2], i.e. each category is mapped to an integer starting from 0, to enable the use of embedding layers. Furthermore, to address class imbalance, we oversample the minority class during training by constructing batches with a fixed 10% positive class ratio. For evaluation, the original proportion is used.

4.2 Domain Sampler

The BAF dataset does not contain any explicit separation of domains. As such, in our experiments, we use the *domain sampler* to the create 4 domains: 3 sources and 1 target. Since the domains are randomly sampled, without loss of generality, we always select the first one to be the target. Given that BAF is a tabular dataset, we define a distance function δ to compare rows containing a set of numerical features $\mathcal{N}$ and a set of categorical features $\mathcal{C}$. For numerical features, we compute the squared difference between their standardized values. For categorical features, we use an indicator function that returns 0 if the values are the same, and 1 otherwise. The distance function is then given by

$$\delta(x_i, x_j) = \sum_{f \in \mathcal{N}} \left(\frac{x_{i,f} - x_{j,f}}{\sigma_f} \right)^2 + \sum_{f \in \mathcal{C}} \mathbb{I}[x_{i,f} \neq x_{j,f}] \,,$$

where $x_{i,f}$ is the value of feature f in the feature vector x_i and σ_f is the standard deviation of feature f computed over the dataset from which samples are drawn.

The Acquirers dataset already contains 4 distinct domains, so we decided not to use the domain sampler. However, a user may decide to apply it even when multiple domains are available, to simulate a wider variety of settings.

4.3 Transformations

We define three types of operations that can be applied to features of tabular datasets:

- ϕ_1 rescales numerical features by a time-dependent factor, with scaling parameter $\alpha \in \mathbb{R}^+$,

$$\phi_1(x_{i,j}, t_i^x; \theta) = x_{i,j} \cdot \alpha^{\tau(t_i^x)}, \text{ where } \theta = (\alpha, \tau) \,. \tag{1}$$

- ϕ_2 computes a weighted average between a numerical feature and a certain anchor value β, with $\beta \in \mathbb{R}$ and a mixing coefficient $\gamma \in [0, 1]$,

$$\phi_2(x_{i,j}, t_i^x; \theta) = (1 - \gamma \cdot \tau(t_i^x)) \cdot x_{i,j} + (\gamma \cdot \tau(t_i^x)) \cdot \beta, \text{ where } \theta = (\beta, \gamma, \tau) \,. \tag{2}$$

- ϕ_3 resamples values of a categorical feature, approximating its relative frequencies to some marginal distribution $P(X'_j)$,

$$\phi_3(x_{i,j}, t_i^x; \theta) \sim (1 - \tau(t_i^x))P(X_j) + \tau(t_i^x)P(X'_j), \text{ where } \theta = (\tau, P(X'_j)) \,. \tag{3}$$

Here, τ is a user-defined function that controls the magnitude of the transformation as a function of t_i^x. We use three versions of τ (not in a one-to-one correspondence with the transformations): (1) a constant function equal to 1, simulating fixed changes between domains (e.g., currency changes); (2) a linear function that goes from 0 to 1 over the dataset's time span, simulating gradual drifts (e.g., inflation effects); (3) a sinusoidal function with a configurable period, simulating seasonal patterns (e.g., weekly fluctuations in consumer behavior).

We combine these three types of transformations with the different τ functions to implement various transformations based on domain knowledge relevant to our use case. Each transformation is applied to a subset of features, and we define sensible ranges for the parameters θ to ensure that the resulting transformations are plausible. For each domain in each experiment, we independently sample the transformation parameters from their respective ranges. This approach ensures that the resulting shifts mimic realistic behavior while also introducing controlled variability across experiments, and thus increasing the robustness and generality of our conclusions.

4.4 Scheduler

For each experiment, given a set of source and target datasets with time span $[t_s, t_e)$, we define t_α and t_β as the start times for using source and target domain data respectively in the experiment, and t_γ as the end time of the experiment, such that $t_s \leq t_\alpha < t_\beta < t_\gamma \leq t_e$. The target domain data in the interval $[t_\alpha, t_\beta]$ is ignored to ensure that the first training split contains only source domain data, mimicking real-world deployment scenarios where historical target data is unavailable at launch. We define the time interval between model updates Δ_t, which is also the duration of each test split. Lastly, since neither dataset contains a label timestamp, we define a fixed label delay Δ_l such that $t_i^y = t_i^x + \Delta_l$. Using these parameters, the *scheduler* simulates the progressive arrival of data as described in Section 3.3, generating a sequence of timestamps $t_1, \ldots, t_l$ s.t.

$$t_1 = t_\beta, \quad t_{a+1} = t_a + \Delta_t, \text{ for } a = 1, \ldots, l-1$$

where t_l it the largest timestamp satisfying $t_l \leq t_\gamma$.

For the Acquirers dataset, we use the time unit of one week, with timestamps indexed in the range $[0, 41)$, and set $\Delta_t = 2$ and $\Delta_l = 4$. In each experiment, t_α is randomly selected from $\{0, \ldots, 7\}$ to introduce variability while ensuring the framework leverages the entire data range. Then, t_β is set as $t_\alpha + 16$, ensuring 16 weeks of available source domain data before the target appears, and t_γ is set as $t_\alpha + 34$, resulting in 9 contiguous test periods.

For the BAF dataset, we use the time unit of one month with timestamps indexed in the range $[0, 8)$. We set $t_\alpha = 0$, $t_\beta = 3$, $t_\gamma = 8$ and $\Delta_t = \Delta_l = 1$. The resulting schedule for this dataset is depicted in Figure 4.

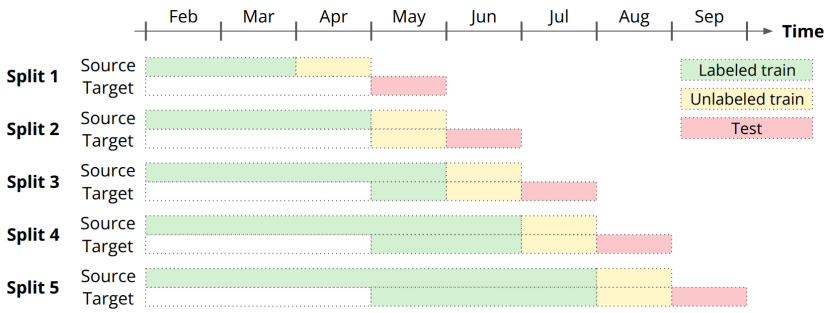

Fig. 4. Schedule of the splits used for the BAF experiments.

4.5 TL Methods

We implemented and tested representative methods from each of the four TL paradigms discussed in Section 2.2: Multi-Task Autoencoder (**MTAE**) [7] for DG; Domain Adaptation Neural Networks (**DANN**) [6] for UDA; Minimax Entropy (**MME**) [22] for SDA; and Multinomial Adversarial Networks (**MAN**) [3] for MDL. We selected these well-established methods, which represent a variety of modeling techniques, not to conduct an exhaustive benchmark but to illustrate how our framework enables the comparison of diverse TL algorithms under evolving data availability conditions. Additionally, we tested three MLP baselines, differing only in their training data: **BL-S** is trained only with labeled source domain data; **BL-T** is trained only with labeled target domain data; and **BL-A** is trained using all labeled data available.

We also tested Kernel Mean Matching (**KMM**) [11] to reweight labeled training data for a LightGBM [14] classifier. The training data is obtained by sampling (with replacement) an equal number of labeled instances from each available domain, which means it may or may not include target domain instances, depending on their availability at that point in time. Due to the computational complexity of solving the optimization problem of the KMM method, the size of the training set was tuned to match the training time of the deep learning methods.

4.6 Evaluation

For all deep learning methods, we use the latest 30% of the labeled training data from each domain as a holdout validation set for early stopping. To ensure a consistent stopping criterion, even when labels are scarce, we measure the average predicted performance across all domains, measured as Recall at 1% False Positive Rate (FPR), which is a standard metric in fraud detection tasks.

For each experiment, we compute paired t-tests for each pair of methods at every data split, to assess the statistical significance of the observed performance differences. Given the substantial number of comparisons, we controlled the False Discovery Rate (FDR) at 1% using the Benjamini-Hochberg procedure, which reduces the risk of identifying spurious effects.

4.7 Pre-training and hyperparameter tuning

Many TL methods use pre-trained state-of-the-art models to initialize the parameters of their deep learning components. In our experiments, we pre-train an MLP-based autoencoder using the first three months of source domain data, using a typical encoder-decoder architecture with reconstruction loss defined per feature type: we use mean squared error for numerical features (after standardization) and cross-entropy loss for categorical features. This self-supervised learning phase enables the networks to learn robust feature representations before applying specific transfer learning methods.

To optimize the autoencoder architecture, we conduct a hyperparameter search over 200 randomly sampled configurations. The search space includes variations in the number and size of hidden layers, the size of the latent space, the learning rate, regularization techniques (dropout, normalization), and the inclusion of skip connections. For method-specific hyperparameters, we primarily followed the values recommended in the respective papers. The details of the search space and best hyperparameters are provided in the code repository.

The encoder block of the best-performing autoencoder, selected based on validation loss, is then used to initialize the feature extractors of the TL methods. For their classifier components, we used a simple architecture with a single hidden layer followed by the output layer.

5 Results

In this section, we first present the results from our case study on the proprietary Acquirers dataset, and then the results on the publicly available BAF dataset. Finally, we discuss the practical implications of these findings and describe how industry practitioners could use them to guide their decision-making process.

5.1 Acquirers dataset case study

We conducted 64 experiments on the Acquirers dataset, following the schedule described in Section 4.4. Figure 5 depicts the results of these experiments, showing the evolution of predictive performance over time for various baselines and TL methods. The x-axis represents the time elapsed since the target domain appeared, while the y-axis depicts the recall percentage at 1% FPR, which is a standard evaluation metric for the fraud detection problem.

There is a clear distinction between methods that leverage labeled target domain data and those that do not. As such, we identify three groups of methods:

- MTAE, DANN and BL-S, which do not use any target domain labels to train, maintain relatively stable performance throughout, but are consistently surpassed by the other methods.
- MAN, BL-A and BL-T, despite requiring target labels before their initial deployment, immediately outperform the other methods, and continue to improve as more data becomes available, with an average gain of approximately 4 percentage points of recall per model update.

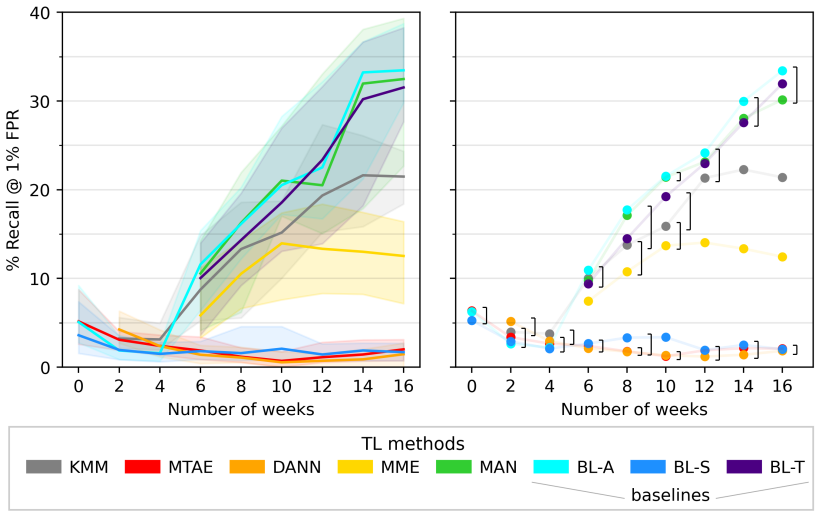

Fig. 5. Predictive performance (recall at 1% FPR) of each method over time on the Acquirers dataset. The left panel shows the median recall per method across experiments (solid lines) and their interquartile ranges (shaded). The right panel presents statistical comparisons at each time step, where each point represents the average recall across experiments, and brackets group methods that are not significantly different, after correcting for multiple comparisons.

– KMM and MME initially follow the trend of the previous group, but plateau earlier. We hypothesize that the sampling we use to run KMM limits the training of LightGBM, while the semi-supervised approach of MME offers diminishing returns as more labels become available.

The statistical tests confirm that, as soon as labeled target domain data becomes available, the methods that leverage it achieve significantly better performance. Furthermore, these tests help to identify the point in time when MME and KMM methods are overtaken by the second group.

5.2 BAF dataset

We conducted 128 independent experiments on the BAF dataset. In each experiment, we sampled four domains from the dataset, applied domain transformations (described in Section 4.3), and followed the schedule depicted in Figure 4 to train and evaluate the methods. Figure 6 depicts the results of these experiments, in the same format of Figure 5.

Similar to the previous experiments, the statistical tests allow us to identify three groups of methods:

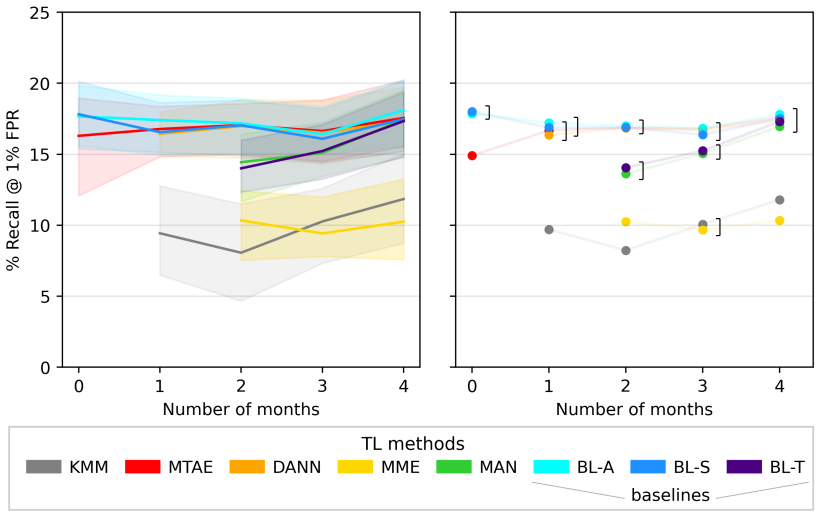

Fig. 6. Predictive performance (recall at 1% FPR) of each method over time on the BAF dataset. The left panel shows the median recall per method across experiments (solid lines) and their interquartile ranges (shaded). The right panel presents statistical comparisons at each time step, where each point represents the average recall across experiments, and brackets group methods that are not significantly different, after correcting for multiple comparisons.

- MTAE, DANN, BL-A and BL-S show similar levels of recall, maintaining a stable distribution of predictive performance over time. This suggests that there is a limited benefit from the additional target domain data and labels.
- MAN and BL-T begin to perform significantly worse than the previous group of methods, but they improve steadily over time (gaining on average 2 percentage points of recall per model update) and eventually reaching the same level of performance. This improvement is not surprising, since both methods use exclusively labeled data from the target domain to train their classifiers.
- KMM and MME are consistently surpassed by the other methods. While MME maintains a relatively stable performance throughout, KMM is improving at the same rate as the previous group.

Furthermore, we observe that the performance of methods such as MTAE and BL-S, which do not use any target data during training, is similar to the performance of BL-T, which follows the traditional ML approach of only using in-domain data to train. This suggests that the source and target domains in the BAF dataset are relatively similar, which means that there is great potential for sharing knowledge across domains. Alternatively, we could adapt the transformation ranges to simulate a setting with more differences between domains.

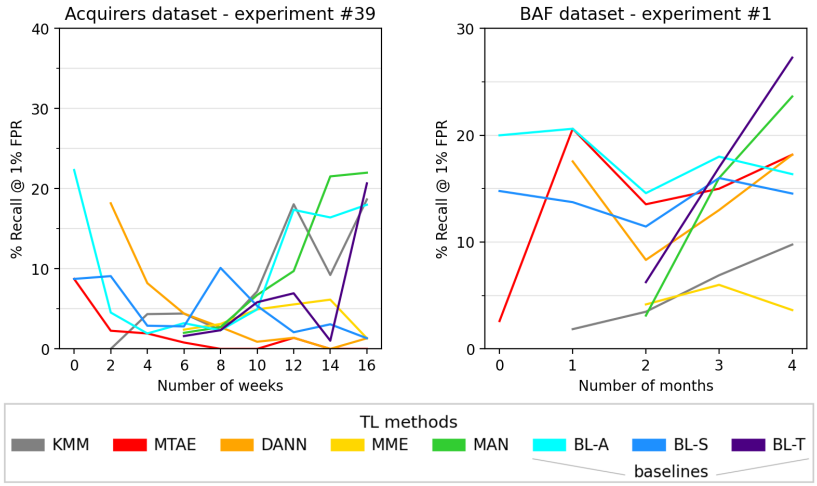

Fig. 7. Predictive performance (recall at 1% FPR) of each method over time. These panels show the results from two specific experiments (one from each dataset) that conflict with the general insights derived from the general analysis.

5.3 Practical implications

Our experimental results highlight how the performance of different TL methods is affected by the evolving data availability conditions. In general, methods that do not require target domain labels maintain stable but potentially limited performance, while methods that leverage target labels tend to noticeably improve as more data becomes available. However, the extent and timing of this improvement vary across settings, meaning that both the effectiveness of these methods and the value of target labels depend on the specific characteristics of the dataset. Recognizing these trends is essential for guiding real-world deployment decisions, helping practitioners select methods that align with their specific constraints and assess the cost-benefit of target labeling efforts.

For example, as seen in our Acquirers case study, practitioners may observe a clear performance gap between methods that leverage labeled target data and those that do not. This gap suggests significant domain drift, making it difficult to share knowledge between domains. In such cases, ML practitioners may decide to conduct a thorough exploratory data analysis to detect potential issues in the data collection pipeline. They may also explore data pre-processing techniques, such as feature normalization, to mitigate domain discrepancies. If the performance gap persists, deploying a DG-based solution initially can be a viable approach, but obtaining labeled target domain data should remain a priority to improve model performance.

Additionally, our results from Acquirers show that MDL methods (such as MAN and BL-A), which optimize performance across multiple domains simultaneously, perform comparably to domain-specific solutions. In those cases, prac-

titioners can benefit from these centralized solutions by minimizing the number of models that need to be developed and maintained. Conversely, as seen in the BAF dataset experiments, practitioners may find that models trained exclusively on source domain data achieve performance levels similar to traditional in-domain models. In those scenarios, DG methods can be confidently deployed for new domains without requiring immediate target domain labels, significantly reducing early labeling efforts and accelerating model deployment timelines.

Finally, we highlight the importance of repeating multiple variants of the experiments to ensure robust conclusions. Figure 7 illustrates two specific cases that deviate from the general trends observed in our main analysis. In the left panel, the results show that methods leveraging labeled target domain data, such as MAN and BL-A, only begin to outperform the other methods 12 weeks after the target domain appears, which is twice as long as observed in Section 5.1. In the right panel, the results show that MAN and BL-T improve steeply over time, eventually surpassing all other methods, whereas the aggregated results in Section 5.2 indicate no significant performance advantage for these methods. These inconsistencies demonstrate the risk of drawing misleading conclusions from isolated experiments, which our evaluation framework helps to mitigate.

Beyond predictive performance, robustness over time is crucial for real-world deployment. Some methods may maintain more stable performance, while others can exhibit a larger variance. In high-risk applications such as fraud detection, consistency may be preferable to occasional peaks in performance. Another important factor when comparing ML methods is computationally efficiency, as highlighted by the KMM method. Evaluating methods in terms of training time or resource usage could provide further insights, and integrating such metrics into the framework is a promising direction for future work. Other practical trade-offs, such as model update complexity or explainability, may also be considered when selecting a TL method for real-world use.

6 Conclusion

In this paper, we introduce an evaluation framework designed to assess transfer learning methods under evolving data availability conditions. Unlike traditional static benchmarks, our framework simulates the progressive arrival of data and labels, allowing for a more realistic and comprehensive evaluation of TL approaches in dynamic settings. Additionally, by generating multiple realistic domain variations from the same dataset and applying controlled transformations, the framework enables systematic testing across diverse and realistic scenarios. We demonstrate the capabilities of our framework through a case study on a proprietary dataset of card payment transactions, and perform an analogous study on the publicly available BAF dataset for reproducibility. Our results illustrate how practitioners can leverage the framework to analyze TL performance trends over time, identify promising methods under varying data availability scenarios, and make informed decisions regarding deployment.

A natural concern with evaluation methodologies is how much the observed results depend on specific design decisions. To address this, we use the framework to systematically conduct a large number of experiments with diverse and realistic variations, and report distributions of performance metrics and statistical significance to highlight trends that are consistent across the different settings. Nonetheless, we do not claim that our results generalize to all possible real-world conditions. Rather, the framework is designed to support this type of studies: by modifying transformation types, parameter ranges, or scheduling strategies, users can tailor experiments to reflect domain-specific assumptions and assess method performance under conditions relevant to their application. Future work may extend this by systematically analyzing sensitivity to each design component or by exploring generalization across broader use cases.

Overall, our framework improves upon traditional evaluations to address practical industry needs, providing a valuable tool for developing robust and adaptable machine learning solutions in dynamic real-world environments.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th annual meeting of the association of computational linguistics (2007)
2. Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., Kasneci, G.: Deep neural networks and tabular data: A survey. IEEE transactions on neural networks and learning systems (2022)
3. Chen, X., Cardie, C.: Multinomial adversarial networks for multi-domain text classification. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 1226–1240 (2018)
4. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213–3223 (2016)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
6. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., Lempitsky, V.: Domain-adversarial training of neural networks. Journal of machine learning research **17**(59), 1–35 (2016)
7. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D.: Domain generalization for object recognition with multi-task autoencoders. In: Proceedings of the IEEE international conference on computer vision. pp. 2551–2559 (2015)
8. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 2066–2073. IEEE (2012)

9. Griffin, G., Holub, A., Perona, P., et al.: Caltech-256 object category dataset. Tech. rep., Technical Report 7694, California Institute of Technology Pasadena (2007)
10. Guan, H., Liu, M.: Domainatm: Domain adaptation toolbox for medical data analysis. NeuroImage **268**, 119863 (2023)
11. Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., Smola, A.: Correcting sample selection bias by unlabeled data. Advances in neural information processing systems **19** (2006)
12. Hull, J.J.: A database for handwritten text recognition research. IEEE Transactions on pattern analysis and machine intelligence **16**(5), 550–554 (1994)
13. Jesus, S., Pombal, J., Alves, D., Cruz, A., Saleiro, P., Ribeiro, R., Gama, J., Bizarro, P.: Turning the tables: Biased, imbalanced, dynamic tabular datasets for ml evaluation. Advances in Neural Information Processing Systems **35**, 33563–33575 (2022)
14. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems **30** (2017)
15. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
17. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: Proceedings of the IEEE international conference on computer vision. pp. 5542–5550 (2017)
18. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y., et al.: Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning. vol. 2011, p. 4. Granada (2011)
19. Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., Wang, B.: Moment matching for multi-source domain adaptation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1406–1415 (2019)
20. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: Computer Vision–ECCV 2016: 14th European Conference on Computer Vision. pp. 102–118. Springer (2016)
21. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Computer vision–ECCV 2010: 11th European conference on Computer Vision. pp. 213–226. Springer (2010)
22. Saito, K., Kim, D., Sclaroff, S., Darrell, T., Saenko, K.: Semi-supervised domain adaptation via minimax entropy. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 8050–8058 (2019)
23. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5018–5027 (2017)
24. Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., Philip, S.Y.: Generalizing to unseen domains: A survey on domain generalization. IEEE transactions on knowledge and data engineering **35**(8), 8052–8072 (2022)
25. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing **312**, 135–153 (2018)
26. Wilson, G., Cook, D.J.: A survey of unsupervised deep domain adaptation. ACM Transactions on Intelligent Systems and Technology (TIST) **11**(5), 1–46 (2020)
27. Yang, Y., Hospedales, T.M.: A unified perspective on multi-domain and multi-task learning. arXiv preprint arXiv:1412.7489 (2014)
28. Zhou, K., Liu, Z., Qiao, Y., Xiang, T., Loy, C.C.: Domain generalization: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(4) (2022)