Data Composition for Continual Learning in Application of Cyberattack Detection

Jiayi Lian[1], Xueying Liu[1], Kevin Choi[2], Balaji Veeramani[2], Sathvik Murli[2], Alison Hu[2],
Laura Freeman[1], Edward Bowen[2], and Xinwei Deng[1]

[1] Department of Statistics, Virginia Tech, VA, USA
{lianjiayi, xliu96, laura.freeman, xdeng}@vt.edu
[2] AI Center of Excellence, Deloitte & Touche, CA, USA
{kevchoi, bveeramani, smurli, aehu, edbowen}@deloitte.com

Abstract. Continual learning (CL) focuses on enabling machine learning algorithms to learn from a series of tasks without forgetting previously acquired knowledge. The use of continual learning has not been widely explored in cybersecurity and network safety applications, partially due to the lack of proper datasets. Besides, the benchmark datasets used in CL methods are often relatively restrictive in terms of data distribution shift among the tasks. In this work, we present a CL benchmark framework to construct datasets for CL in cybersecurity applications. For the cybersecurity applications, the proposed framework can generate datasets for CL under distribution shifts in data inputs (e.g., features of internet traffic flow), distribution shifts in data output (e.g., intrusion types), and distribution shifts in both data inputs and outputs, respectively. Moreover, we propose several distance-based and model-based metrics to meticulously quantify the magnitude of distribution shift between datasets of the tasks. We elaborate the construction of benchmark datasets and evaluate the quality of the constructed datasets by applying several existing CL methods and investigating their performance.

Keywords: Cybersecurity · Continual learning · Distribution shift · Network safety.

1 Introduction

1.1 Background and Motivation

Conventional Machine Learning (ML) methods generally perform well on an individual task for which they were trained on the associated dataset. However, when a new task is introduced, the model often needs to be retrained to learn from the new associated dataset. This retraining process often leads to the unintended forgetting of previously learned knowledge, referred to as 'catastrophic forgetting,' a concept derived from human cognition studies [10]. To prevent or mitigate 'catastrophic' forgetting, the so-called continual learning (CL) [7] was developed to learn from a series of tasks without forgetting previously acquired knowledge. CL methods have been extensively developed in a wide range of applications, such as image instances and semantic segmentation, document analysis, and sentiment analysis [14, 16, 19, 24, 35].

However, CL methods have not been widely used in the cybersecurity and network safety applications, partially due to lacking proper benchmark datasets. In the era of data science and AI, cybersecurity has emerged as a vital concern for public safety. Over the past decade, there has been a rise in social engineering attacks [33] including phishing and spear-phishing, in addition to malware and ransomware [3]. To more efficiently address these safety threats, practitioners started to notice the power of AI algorithms and apply them to detect cyber intrusion [1,26]. However, these efforts still remain on training AI model on a specific task (i.e., dataset) instead of updating the AI model to adapt to a series of dissimilar tasks. Only several early-stage methodologies, such as fine-tuning, learning without forgetting [24], and direct data replay, have been employed for cybersecurity CL tasks [28].

For the typical CL, the benchmark datasets involve data distribution shift among the tasks. Here we denote X to be the input and Y to be the output for a dataset. Existing works commonly construct the benchmark by gathering a set of classes from contextually similar datasets and assigning these classes and their samples to the tasks. However, such a conventional procedure only considers (X,Y) distribution shifts in the datasets of tasks, overlooking other possibilities of distribution shifts. It is also not clear on the magnitude of distribution shift among the tasks, which is crucial in assessing performance of CL methods with respect to the data quality of the benchmark. Therefore, the conventional procedure of benchmark construction lacks completeness and comprehensiveness in validating CL algorithms.

1.2 Challenges and Contributions

In the cybersecurity and network safety applications, the distribution shift among the intrusion detection tasks often presents greater complexity. Several intrusion types may undergo updates that significantly alter their input patterns, reflecting the distribution shift on $X|Y$, while new intrusions with distinct patterns can signify the occurrence of (X,Y) distribution shifts. It implies that both distribution shifts on $X|Y$ and (X,Y) are common in datasets for cyber intrusion detection tasks. But most publicly available network intrusion datasets are derived from simulated experiments. The real-world cyber practices are much more complex than the scenarios from the simulation datasets [2, 17, 32]. Therefore, it is necessary to develop a systematic framework for constructing benchmark datasets for CL, with the aim of thoroughly investigating the potential of CL methods in cybersecurity applications.

The proposed framework of constructing benchmark datasets for CL methods in cybersecurity applications has the following contributions. First, since the real-world cyber intrusion datasets are typically unavailable due to security or privacy concerns, the proposed framework aims to construct the benchmark datasets for CL such that they can bridge the gap between simulated cyber intrusion data and real-world cyber intrusion data. Second, the proposed framework is capable of generating distribution shifts that closely mirror the cyber intrusion activities in practice, as well as any specifically designed distribution shifts. Specifically, we employ clustering techniques to establish a series of $X|Y$ distributions for a given class. Next, we define the marginal distribution of response Y as a multinomial distribution. By manipulating the proportions of response labels, the marginal distribution shift on Y can also be constructed. Consequently, we can construct the benchmark datasets with a wide range of distribution shifts on $X|Y$, Y, and (X,Y). It can greatly enrich the scenarios encompassed by simulated cyber intrusions. Moreover, with the knowledge of cybersecurity professionals, we can generate benchmark settings that more accurately reflect real-world cyber intrusion activities. Third, the proposed framework also develops several metrics to quantify the magnitude of distribution shifts among the tasks, which is important for accurate evaluation of the performance of CL methods. We develop both distance-based and model-based metrics to measure the distribution shift discrepancy between datasets in different tasks of CL. In our numerical experiments, we use the Communications Security Establishment and the Canadian Institute for Cybersecurity Intrusion Detection System (CSE-CIC-IDS) 2018 cyber intrusion data [31] as the source to construct several CL benchmark datasets, each is designed to have a specific scenario of distribution shift. The proposed framework can be also applied to other novel Cybersecurity datasets, such as the dataset of advancing DDoS and spoofing attack in IoV CAN bus (CICIoV2024) [27] and DDoS attack dataset (CICEV2023) against EV authentication in charging infrastructure [18]. The effectiveness of the proposed framework of constructing benchmark datasets for CL is evaluated by investigating four widely-used CL models.

2 The Proposed Benchmark Construction

We first denote several notations. Let X be the input features and Y be the output response. The $X|Y$ denotes the input vector conditioning on the output response. Suppose the source dataset (or Datasets) D_s contains L types of attacks in total, with $C_s = \{c_1, c_2, \ldots, c_L\}$ representing the attack label set. Consider a benchmark setting with N tasks constructed from D_s, let $D_1, D_2, \ldots, D_N$ denote the datasets for the corresponding tasks $T_1, T_2, \ldots, T_N$, where $D_i = (X_i, \tilde{y}_i), i = 1, 2, \ldots, N$, $X_i = (\tilde{x}_1^{(i)}, \tilde{x}_2^{(i)}, \ldots, \tilde{x}_n^{(i)})^T$ is a $n \times p$ covariate matrix, and $\tilde{y}_i$ is n-dimensional response vector. Here, n is the number of observations (assuming the sizes of the datasets are consistent), p is the number of covariates, and $\tilde{x}_k^{(i)}$ is the kth sample in D_i, $k = 1, 2, \ldots, n$. Additionally, let $X_1, X_2, \ldots, X_N$ be the input feature populations corresponding to $D_1, D_2, \ldots, D_N$.

2.1 CL Benchmark Datasets Generation

This subsection focuses on constructing benchmark datasets with $X|Y$ and (X,Y) distribution shifts for CL, as these two types of distribution shifts are prevalent in cyber intrusion activities. The key idea of the proposed construction method is to generate the $X|Y$ distribution shifts and then incorporate the marginal distribution shift on Y with the $X|Y$ distribution shifts.

Specifically, we adopt the K-means clustering [25] to group the input X given a specific Y, which produces $X|Y$ distributions to construct $X|Y$ distribution shifts. To maximize the distribution shifts among the clustered groups, we consider employing three goodness of clustering scores. Specifically, we analyze the changing patterns of the Silhouette score [29], Calinski Harabasz score [4], and Davies Bouldin score [6] with respect to the number of clusters to determine the optimal number of clusters for a given attack label Y. The optimal number of clusters is typically indicated by the maximum of the first two scores and the minimum of the Davies Bouldin score. However, practitioners can select a number of clusters to align with their desired scenario or magnitude of distribution shift. Let $\tilde{N}_c = (N_1^{(c)}, N_2^{(c)}, \ldots, N_L^{(c)})^T$ indicate the optimal numbers of clusters for the L attack labels. Given $Y = c_l, l = 1, 2, ..., L$, there will be $N_l^{(c)}$ different $X|Y$ distributions, $(X = X_1^*|Y = c_l), (X = X_2^*|Y = c_l), \ldots, (X = X_{N_l^{(c)}}^*|Y = c_l)$, where $X_t^*, t = 1, 2, \ldots, N_l^{(c)}$ denotes the different input feature populations for $Y = c_l$. $X|Y$ distribution shifts naturally exist among these clusters.

It is important to note that the definition of Y distribution shift in current CL research is not well established. Differences in the label set across tasks are only considered evidence of distribution shifts for the entire (X,Y) distribution. In this work, we define the marginal Y distribution as a multinomial distribution, which takes the labels as its values and is parameterized by the labels' corresponding probabilities. The support set is assumed to contain the possible labels. The marginal Y distribution shift is defined as the change in probabilities for the labels. As the number of attack types L is fixed for D_s, the marginal Y distribution shift can be achieved by adjusting the percentages of the labels in the tasks. Let $\tilde{\omega}_i = (\omega_{i,1}, \omega_{i,2}, \ldots, \omega_{i,L})^T$ represent the vector of the percentages of the attack types for dataset D_i, where $i = 1, 2, \ldots, N$. The condition $0 \leq \sum_{l=1}^{L} \omega_{i,l} \leq 1$ holds, with the percentage of normal (i.e., benign) samples being $1 - \sum_{l=1}^{L} \omega_{i,l}$. A marginal Y distribution shift between $T_i, T_j, i \neq j$ is considered when $\tilde{\omega}_i \neq \tilde{\omega}_j$. According to the Bayes' rule $f(X,Y) = f(X|Y) \times f(Y)$, the (X,Y) distribution shift generation can be controlled by simultaneously managing the $X|Y$ and Y distribution shift generations.

To construct a benchmark setting with $X|Y$ distribution shift, the percentages of attacks for $D_1, D_2, \ldots, D_N$ are kept constant while the generated clusters for a given attack label are distributed into different tasks. The $X|Y$ distribution shift can also be realized by varying the percentages of clusters across tasks. For example, suppose the set of attacks for a benchmark setting contains only c_1. For c_1, there are $N_1^{(c)} = 4$ generated $X|Y$ distributions (clusters). Let $\tilde{\omega}_1 = \tilde{\omega}_2 = (30\%, 0, 0, \ldots, 0)$ to fix the percentages of the attack types. Then, to generate the $X|Y$ distribution shift between D_1 and D_2, one can assign the samples of the first and second clusters to T_1 while putting the samples of the third and fourth clusters to T_2. To achieve $X|Y$ distribution shift, we can also generate $(40\%, 20\%, 30\%, 10\%) \times 30\% \times n$ samples from the four clusters for T_1 and $(5\%, 5\%, 30\%, 60\%) \times 30\% \times n$ samples from the four clusters for T_2. To more explicitly construct (X,Y) distribution shift, the benchmark construction can begin with creating $X|Y$ distribution shifts among tasks, then adjust the percentages of labels for tasks, $\tilde{\omega}_i$'s, to develop various (X,Y) distribution shift scenarios.

2.2 Metrics for Evaluation of Distribution Shift

This section proposes several approaches for assessing the magnitude of distribution shift: Cosine Similarity [22], which represents the overall point-wise linear relationship for input X between two different datasets while correlation coefficient evaluates the linear relationship between populations; Wasserstein Distance [30], which measures the distribution discrepancy for input X between two datasets; and two measures depending on a variational autoencoder (VAE)-based distribution (i.e., sampler) emulator [15], which evaluate input X and $Y|X$ distribution shifts, respectively. The inclusion of the measurement for $Y|X$ distribution shift is for two reasons. Firstly, when developing a model to predict Y based on X, the primary concern is how to build a model to estimate $E(Y|X)$. Secondly, combining the measurements for input X distribution shift and $Y|X$ distribution shift can provide a more comprehensive perspective for investigating (X,Y) distribution shift.

The Cosine Similarity $d^{cs}_{i,j}$ for the inputs X_i and X_j is formulated as

$$d^{cs}_{i,j} = \frac{Coss(X_i, X_j)}{\sqrt{Coss(X_i, X_i) Coss(X_j, X_j)}},$$

where $Coss(X_i, X_j) = \frac{1}{n^2} \sum \frac{\tilde{x}^{(i)}_k \cdot \tilde{x}^{(j)}_{k'}}{\|\tilde{x}^{(i)}_k\| \|\tilde{x}^{(j)}_{k'}\|}$, $Coss(X_i, X_i) = \frac{1}{n(n-1)} \sum_{all\ k \neq k'} \frac{\tilde{x}^{(i)}_k \cdot \tilde{x}^{(i)}_{k'}}{\|\tilde{x}^{(i)}_k\| \|\tilde{x}^{(i)}_{k'}\|}$, and $k, k' \in \{1, 2, \ldots, n\}$.

Suppose $\mu \in P$ and $\nu \in Q$ are distribution measures, where P is a set of distribution measurements for X_i and Q is a set of distribution measurements for X_j. Define $\pi \in J(P,Q)$ as a joint distribution measure, where $J(P,Q)$ is the set of joint distribution measurements for (P,Q). Wasserstein Distance $d^{wd}_{i,j}$ for the populations X_i and X_j is formulated as

$$d^{wd}_{i,j} = \left[Inf_{\pi \in J(P,Q)} \int \|x_i - x_j\|^{\gamma} d\pi(x_i, x_j) \right]^{1/\gamma},$$

where $x_i \in X_i$, $x_j \in X_j$, and $\gamma > 1$. It is difficult to directly calculate the Wasserstein distance between the populations as their true marginal distributions are unknown. Thus, the Wasserstein Distances is estimated based on the datasets for the two populations, X_i and X_j:

$$\hat{d}^{wd}_{i,j} = wass(X_i, X_j) = \left[\frac{1}{n} \sum \|\tilde{x}^{(i)}_k - \tilde{x}^{(j)}_{k'}\|^{\gamma} \pi^*(\tilde{x}^{(i)}_k, \tilde{x}^{(j)}_{k'}) \right]^{1/\gamma},$$

where $\pi^*(\cdot)$ is the optimal transportation plan with the overall smallest cost to transfer all the points of X_i to all the points of X_j, $\|\tilde{x}^{(i)}_k - \tilde{x}^{(j)}_{k'}\|$ is the cost for moving the point $\tilde{x}^{(i)}_k$ to the point $\tilde{x}^{(j)}_{k'}$.

As depicted in Figure 1, the **Encoder** takes an input X and produces two layers: one that serves as the mean vector $\tilde{\mu}$ and the other as the standard deviation vector $\tilde{\sigma}$. These are used to sample the latent vector z, which follows a multivariate distribution with mean $\tilde{\mu}$ and a covariance function $Diag(\tilde{\sigma})$, a diagonal matrix with $\tilde{\sigma}$ as its diagonal elements. The latent vector is then input to both the **Decoder** to generate X_{recon} and the **Classifier** to obtain the predicted response y_{pred} as well as the predicted probabilities for the possible labels. The expected output of a standard VAE is a reconstructed input X_{recon} that closely resembles the original input X. In this model, we also require the distribution of X_{recon} to be close to that of X, suggesting that we can sample through the VAE structure using X_{recon} as X_e. In addition, the **Classifier** is expected to predict the label of X as accurately as possible. To fulfill these proposed functionalities, we design a total loss function L_{total} comprising a Cross Entropy loss for the classifier L_{cl}, a

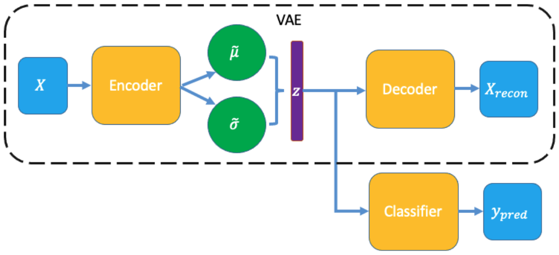


Fig. 1: The structure of the model for emulating the true distribution of a dataset. The part in the dash line is the VAE part. The latent vector z produced by the Encoder is used for the classification purpose as well as the reconstruction for input X.

reconstruction loss L_{recon}, a Kullback-Leibler (KL) divergence loss, and a distribution discrepancy loss L_{dd} for the VAE:

$$\begin{aligned} L_{total} =& \lambda_1 L_{cl}(y_{pred}, y) + \lambda_2 L_{recon}(X, X_{recon}) \\ &+ \lambda_3 KL(N(\tilde{\mu}, Diag(\tilde{\sigma}))||N(\tilde{0}, I)) + \lambda_4 L_{dd}(X, X_{recon}), \end{aligned}$$

where the reconstruction loss is the mean square error between X and X_{recon}. The KL divergence loss quantifies the distribution divergence between the posterior distribution $(N(\tilde{\mu}, Diag(\tilde{\sigma})))$ and prior distribution $(N(\tilde{0}, I))$ for latent vector z, where I represents the identity matrix. It assists in shaping the latent space distribution to be more like a standard multivariate normal distribution. We use the Wasserstein Distance to calculate L_{dd}, a loss aiming to gauge the distribution discrepancy between the distributions of X and X_{recon}. To evaluate distribution discrepancy between T_i and T_j, we train two separate models M_i^* and M_j^* on D_i and D_j, respectively, with the proposed structure. Assume that the training process is ideal to yield true data samplers for both T_i and T_j. Then one can generate emulated samples, $(X_{e,i}, \tilde{y}_{e,i}) = M_i^*(X_j)$ and $(X_{e,j}, \tilde{y}_{e,j}) = M_j^*(X_i)$, that are capable of simulating the true samples of T_i and T_j, respectively. The proposed model-based Wasserstein Distance $d_{i,j}^{wd^*}$ for input X between the distributions of D_i and D_j is constructed as follows:

$$d_{i,j}^{wd^*} = \frac{1}{2}(wass(X_{e,i}, X_j) + wass(X_{e,j}, X_i)).$$

Similarly, the $Y|X$ discrepancy $d_{i,j}^{ce}$ between D_i and D_j is evaluated by

$$d_{i,j}^{ce} = \frac{1}{2}(L_{cl}(\tilde{y}_{e,i}, \tilde{y}_j) + L_{cl}(\tilde{y}_{e,j}, \tilde{y}_i)).$$

Once we have collected the mutual distribution divergences for the dataset pairs in the benchmark setting, we can create a task-by-task discrepancy map.

3 Experiments

In this section, we get into details of how to use the developed framework to construct benchmark settings according to specific evaluation purposes. Three CL benchmark scenarios are established to present various levels of magnitude in distribution shift. Particularly, one of the scenarios is built for investigating the impact of $X|Y$ distribution shift while the rest are developed for examining the impact of (X,Y) distribution shift. Next, four CL methods, Joint Learning (JL) [34], Fine-tuning (FT) [28], Elastic Weight Consolidation (EWC) [19], and Learning without Forgetting (LWF) [24], are evaluated through the CL scenarios constructed.

3.1 Constructed Benchmark Settings

In this study, we utilize the CSE-CIC-IDS2018 dataset [31] as our primary source to illustrate the construction of CL benchmark settings using publicly available cyber intrusion datasets. Excluding 'Benign' samples, the response variable Y encompasses 14 types of attacks, whereas the input features X are statistics of internet traffic flow, derived from a traffic flow generator and analyzer, CICFlowMeter [23]. The attack labels and their corresponding optimal numbers of clusters are presented in Table 1.

Table 1: A summary of the attack labels with their corresponding optimal number of clusters. $N^{(c)}$ is referred to as the optimal number of clusters.

Attack Label	$N^{(c)}$
Botnet	4
Brute-Force Website	5
Brute-Force Cross-Site Scripting (XSS)	5
Brute-Force File Transfer Protocol (FTP)	5
Brute-Force Secure Shell (SSH)	2
DoS Slow Hypertext Transfer Protocol (SlowHttp)	8
DoS HTTP Unbearable Load King (Hulk)	6
DoS Low Orbit Ion Cannon Http (LoicHttp)	4
DoS GoldenEye	5
DoS SlowLoris	6
DoS High Orbit Ion Cannon (Hoic)	2
DoS Low Orbit Ion Cannon User Datagram Protocol (LoicUdp)	2
Infiltration	8
Structured Query Language (SQL) Injection	6

Once the optimal number of clusters has been determined, the tasks within a benchmark setting can be developed by deliberately choosing the attack labels, their corresponding clusters and proportions, as well as the appropriate proportions for the clusters. For real-world cyber events, typically, victims of cyber intrusions fall into two categories: target objects and non-target objects. Non-target objects are randomly attacked, while target objects are intentionally attacked, often with greater effort by hackers. This implies that non-target objects, being less frequently attacked, offer less evidence and patterns of cyber intrusion. Conversely, target objects are attacked more often during intrusion activities. A non-target object may be subjected to various types of attacks, while hackers tend to employ a specific attack or a set of similar attacks for a single attack event, rather than utilizing a wide range of attack types as well. This paper focuses on the scenarios showing attack patterns for target objects.

We construct three benchmark settings for target systems to illustrate the framework. Each task contains 30% attack samples while with the remainder being benign samples. Besides, attack types are randomly distributed among each task.

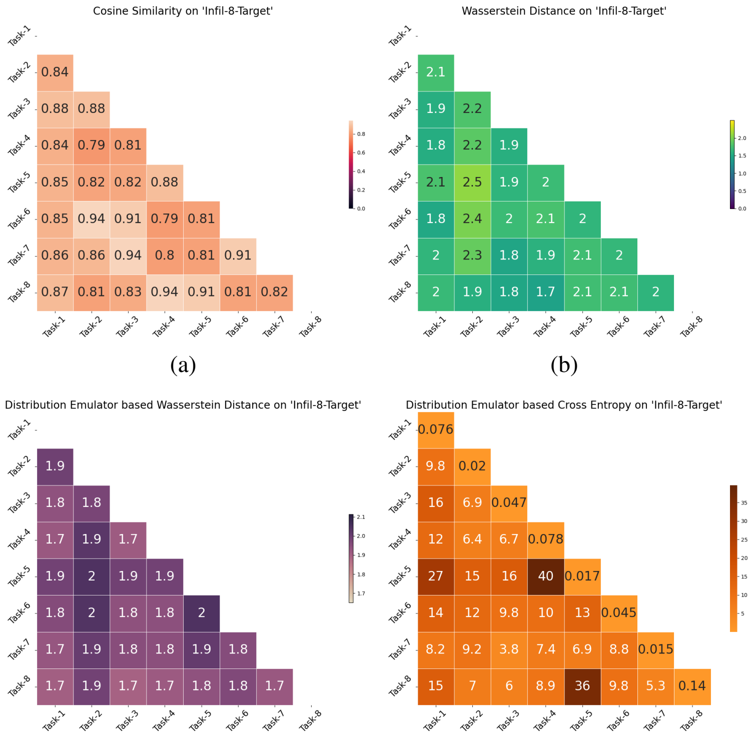
S_1: The first benchmark setting (named as **Infil-8-Target**) simulates a sequence of eight Infiltration attack events of a target system. This setting is designed to exclusively present distribution shifts in $X|Y$. The attack samples for each task are drawn solely from one cluster out of the eight clusters identified for Infiltration attacks.

S_2: The second benchmark setting (named as **DoS-10-Target**) simulates a series of 10 attack events that solely involve four types of DoS attacks for a target system. These four DoS attack types are DoS SlowHttp, DoS Hulk, DoS GoldenEye, and DoS SlowLoris, which are similar to each other. Each task contains two DoS attacks, meanwhile, for each attack, approximately one-third to half of the clusters are selected. The percentages assigned to the attacks and clusters are arbitrary.

S_3: The third benchmark setting (named as **All-20-Target**) is explicitly designed for CL studies, aiming to analyze the performance of CL methods when several attack types reappear in later tasks, yet the techniques employed for these attacks significantly differ among the tasks. The setting is designed to include 20 distinct attack events, involving various types of attacks. Each task is specific to a single type of attack.

3.2 Experiment Results

For **DoS-10-Target**, although the overall level of two versions of Wasserstein Distance are considerably high, the values of Cosine Similarity are greater than 0.9 and there is no cell having the cross entropy greater than 10. It indicates that linear relationship between the similar types of attacks are quite concrete even the corresponding distributions are distinct. The distribution discrepancy are also being reduced by resampling the clusters into the later tasks as what we planned. For example, as T_9 shares the second and fourth clusters of DoS SlowHttp with T_4 and T_6, the cross entropy measures are only 1 and 0.59 for the pairs of (T_4, T_9) and (T_6, T_9), respectively. For **All-20-Target**, as designed, the setting contains sufficient input X distribution discrepancy as the average of Cosine Similarity is smaller than the corresponding average in the distribution discrepancy maps of **DoS-10-Target**. Conversely, the averages of Wasserstein Distance and distribution emulator based Wasserstein Distance are greater than their respective averages in the distribution discrepancy maps of **DoS-10-Target**. It also has substantial $Y|X$ distribution



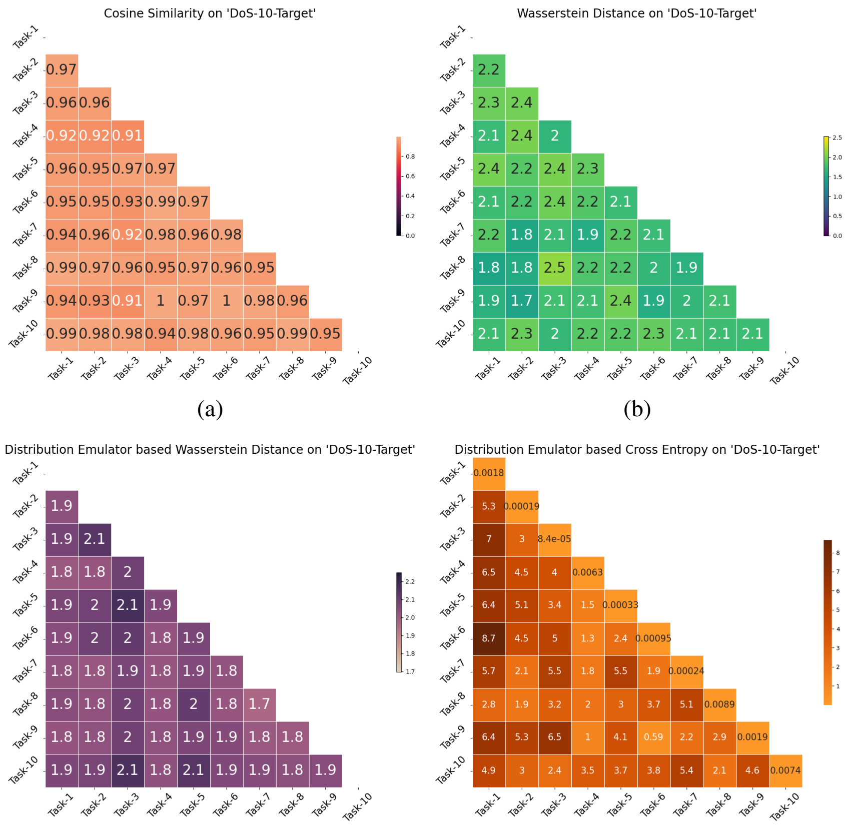
(a) (b)

(c) (d)

Fig. 2: Distribution discrepancy maps for **Infil-8-Target**: (a) Cosine Similarity, (b) Wasserstein Distance, (c) Distribution Emulator based Wasserstein Distance, and (d) Distribution Emulator based Cross Entropy. x-axis presents the current tasks while y-axis presents the available tasks respectively.

shifts as the approximately 80% cells of the cross entropy is greater than 6. Although the clusters are not repeatedly used, it involves several similar types of attacks, which reduces the distribution discrepancy. It can be demonstrated by that the percentage of cells with the cross entropy less than 6 is higher for **All-20-Target** (about 20%) than for **Infil-8-Target** (about 9%).

The performances of four CL benchmark methods, JL, FT, EWC, and LWF, are displayed in Figures 5, 6, and 7 for **Infil-8-Target**, **DoS-10-Target**, and **All 20 Target**, respectively. It can be seen that FT and EWC do not have capacity of CL as they can only perform well on the current tasks for **Infil-8-Target**. However, the two algorithms do show some ability of CL for **DoS-10-Target** and **All-20-Target**. It indicates that these two algorithms do not work when the distribution discrepancy is significant among the tasks. Besides, there is evidence that EWC tends to remain in the optimal parameter space of T_1 although it is able to handle the distribution discrepancy to some extent. However, it has hindered the performance of later tasks, especially when the number of

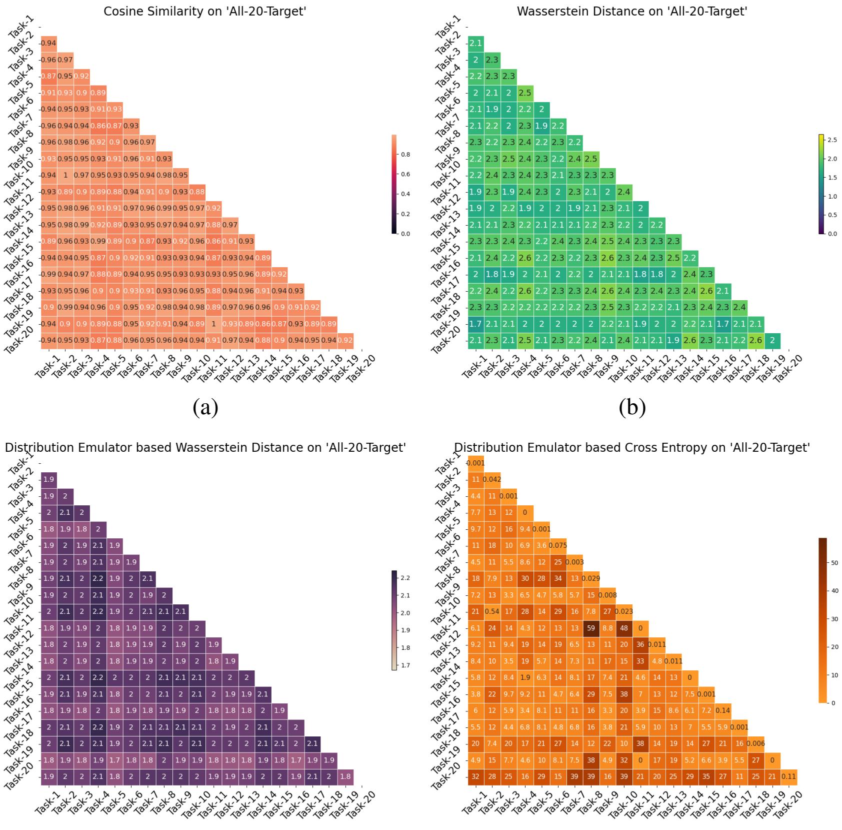


(a) (b)

(c) (d)

Fig. 3: Distribution discrepancy maps for **DoS-10-Target**.

tasks N is high. The performance of JL on the earlier tasks are likely to deplete as there are less and less samples for these tasks in the training dataset $D_{train,k}$. This degradation can be mitigated by reviewing the previous knowledge. For instance, the performance of JL on T_3 is improved from 0.89 to 0.92 when T_7 is available and M_7 is trained for **Infil-8-Target**. It is due to that T_7 are close to T_3 in terms of input space X as $d^{cs}_{3,7} = 0.94$, $\hat{d}^{wd}_{3,7} = 1.7$, and $d^{wd^*}_{3,7} = 1.7$. LWF can only work for the tasks within a certain period from the current tasks for **DoS-10-Target**. However, when dealing with **Infil-8-Target**, it appears that the knowledge of T_1 seems to be difficult to transfer by LWF. The overall performance for **All-20-Target** is subpar (the average accuracy is only about 0.7), but the **All-20-Target**'s overall input X and $Y|X$ distribution discrepancy is smaller than **Infil-8-Target**.


Cosine Similarity on 'All-20-Target'

Wasserstein Distance on 'All-20-Target'

(a)

(b)

Distribution Emulator based Wasserstein Distance on 'All-20-Target'

Distribution Emulator based Cross Entropy on 'All-20-Target'

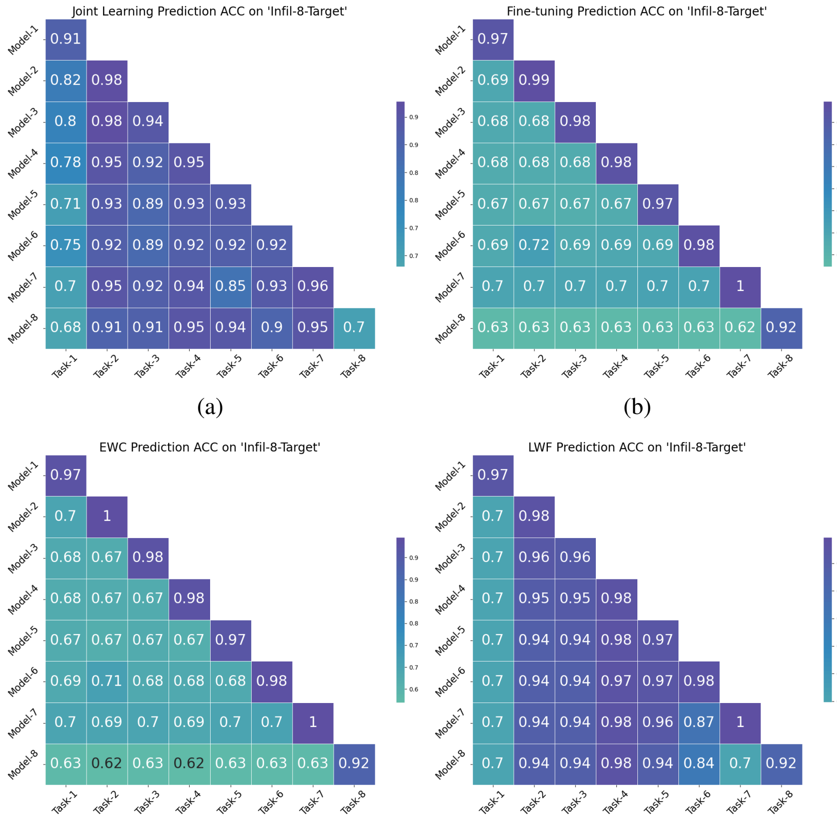
(c)

(d)

Fig. 4: Distribution discrepancy maps for **All-20-Target**.

3.3 Summary of Findings

Our investigation of the distribution discrepancy maps produced several significant findings. We found that clustering is capable of generating substantial shifts in the $X|Y$ distribution. Distributing similar attack types across various tasks can decrease the $Y|X$ distribution discrepancy and improve the linear similarity among tasks. Furthermore, resampling clusters to different datasets can lead to further reduction in distribution discrepancies among tasks within a Continual Learning (CL) benchmark scenario. Additionally, we unearthed intriguing insights, by investigating and analyzing the distribution discrepancy maps and the model-by-task performance maps together. Specifically, we found that EWC and FT models struggle to retain previous knowledge when facing significant distribution shifts, while JL is effective if the sample size for each task in $D_{train,k}$ is sufficient. LWF model shows a peculiar behavior, necessitating further experimentation to understand the underlying causes. These results collectively demonstrate that merging a distribution discrepancy map with a model-by-task performance map



(a) (b)

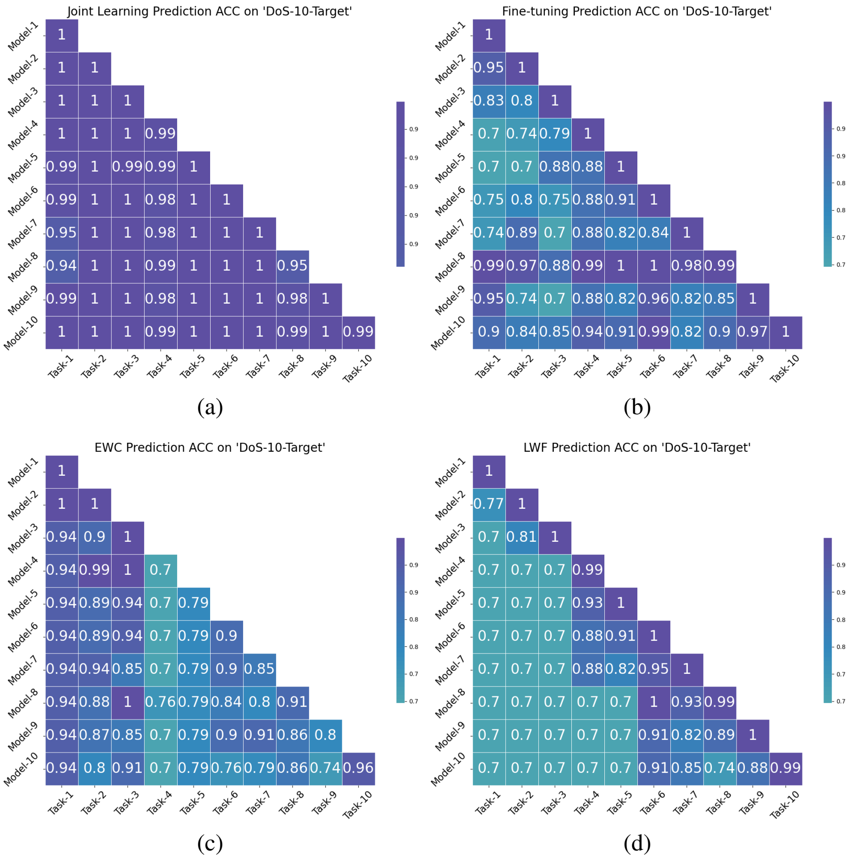
(c) (d)

Fig. 5: Prediction Accuracy of four CL method on **Infil-8-Target**: (a) Joint Learning, (b) Fine-tuning, (c) EWC, and (d) LWF. *x*-axis presents the current tasks while *y*-axis presents the series of built CL models.

offers valuable insights into the functionality of CL methods, enabling a more thorough evaluation of these approaches.

4 Related Work

There are various efforts for constructing CL benchmark settings. For instance, [9] introduced two CL benchmark scenarios using Modified National Institute of Standards and Technology database (MNIST) [8], distributing either five or two labels into a task, resulting in two and five tasks, respectively. Similarly, [13] manipulated permuted MNIST pixels in different ways to develop and examine the catastrophic forgetting phenomenon of neural networks in old and new tasks. Moreover, [36] employed the



(a)

(b)

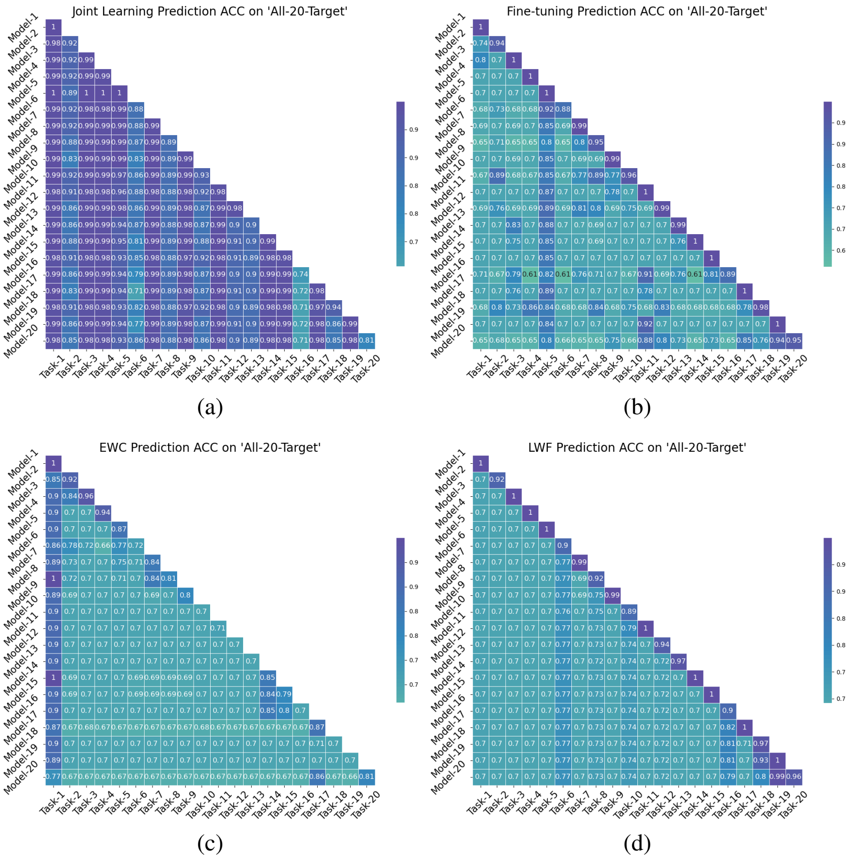
(c)

(d)

Fig. 6: Prediction accuracy maps of four CL method on **DoS-10-Target**: (a) Joint Learning, (b) Fine-tuning, (c) EWC, and (d) LWF. *x*-axis presents the current tasks while *y*-axis presents the series of built CL models.

Canadian Institute For Advanced Research (CIFAR) image collection datasets, CIFAR-10 and CIFAR-100 [20], to construct CL benchmark settings.

For distribution discrepancy quantification, commonly, the Kullback-Leibler (KL) Divergence and Jensen-Shannon Divergence (JSD) are used to measure divergence between two probability distributions, particularly the parametric distributions [11, 21]. Bhattacharyya Distance is used as a training metric for machine learning algorithms [5]. We would like to remark that several metrics for machine learning modeling are measures indicating distribution discrepancy, such as the metrics used in this literature, Cross Entropy [12] and Wasserstein Distance [30].



(a)

(b)

(c)

(d)

Fig. 7: Prediction accuracy maps of four CL method on **All-20-Target**: (a) Joint Learning, (b) Fine-tuning, (c) EWC, and (d) LWF. x-axis presents the current tasks while y-axis presents the series of built CL models.

5 Conclusion and Discussion

In this work, we have developed a systematic framework to generate scenarios of datasets for CL in cybersecurity applications. The framework focuses on how to identify $X|Y$ distributions through clustering methods and then how to design CL scenarios more explicitly with the identified $X|Y$ distributions and the percentages of Y labels, which are capable of generating a wide range of distribution shifts on $X|Y$, Y and (X,Y). In addition, we have established several metrics to track task-by-task distribution discrepancy to evaluate the magnitude and details of distribution shift in CL scenarios of datasets, which can provide practitioners a clearer understanding of the challenges presented by the CL setting.

In future work, one can consider the estimate of the overall magnitude of distribution shift through a surrogate model (e.g., Gaussian process) with the number of labels for Y, the corresponding generated number of $X|Y$ distributions, the amount of distribution discrepancy between the labels and $X|Y$ distributions, and the number of tasks N as inputs. In addition, the variance of metrics for X distribution can be improved by extracting feature importance to remove the redundant features, as the redundant features are likely to provide less information and variance.

References

1. Alqahtani, H., Sarker, I.H., Kalim, A., Minhaz Hossain, S.M., Ikhlaq, S., Hossain, S.: Cyber intrusion detection using machine learning classification techniques. In: Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, March 26–27, 2020, Revised Selected Papers 1. pp. 121–131. Springer (2020)
2. Bay, S.D., Kibler, D., Pazzani, M.J., Smyth, P.: The uci kdd archive of large data sets for data mining research and experimentation. ACM SIGKDD explorations newsletter **2**(2), 81–85 (2000)
3. Beaman, C., Barkworth, A., Akande, T.D., Hakak, S., Khan, M.K.: Ransomware: Recent advances, analysis, challenges and future research directions. Computers & Security (2021). https://doi.org/10.1016/j.cose.2021.102490
4. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics-theory and Methods **3**(1), 1–27 (1974)
5. Choi, E., Lee, C.: Feature extraction based on the bhattacharyya distance. Pattern Recognition **36**(8), 1703–1709 (2003)
6. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE transactions on pattern analysis and machine intelligence (2), 224–227 (1979)
7. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: Continual learning: A comparative study on how to defy forgetting in classification tasks. arXiv preprint arXiv:1909.08383 **2**(6), 2 (2019)
8. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine **29**(6), 141–142 (2012)
9. Farajtabar, M., Azizan, N., Mott, A., Li, A.: Orthogonal gradient descent for continual learning. In: International Conference on Artificial Intelligence and Statistics. pp. 3762–3773. PMLR (2020)
10. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
11. Fuglede, B., Topsoe, F.: Jensen-shannon divergence and hilbert space embedding. In: International symposium on Information theory, 2004. ISIT 2004. Proceedings. p. 31. IEEE (2004)
12. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
13. Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y.: An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211 (2013)
14. Gupta, P., Chaudhary, Y., Runkler, T., Schuetze, H.: Neural topic modeling with continual lifelong learning. In: International Conference on Machine Learning. pp. 3907–3917. PMLR (2020)
15. Islam, Z., Abdel-Aty, M., Cai, Q., Yuan, J.: Crash data augmentation using variational autoencoder. Accident Analysis & Prevention **151**, 105950 (2021)

16. Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. arXiv preprint arXiv:1711.10563 (2017)
17. Kent, A.D.: Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory (2015). `https://doi.org/10.17021/1179829`
18. Kim, Y., Hakak, S., Ghorbani, A.: Ddos attack dataset (cicev2023) against ev authentication in charging infrastructure. In: 2023 20th Annual International Conference on Privacy, Security and Trust (PST). pp. 1–9. IEEE (2023)
19. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
20. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
21. Kullback, S., Leibler, R.A.: On information and sufficiency. The annals of mathematical statistics **22**(1), 79–86 (1951)
22. Lahitani, A.R., Permanasari, A.E., Setiawan, N.A.: Cosine similarity to determine similarity measure: Study case in online essay assessment. In: 2016 4th International Conference on Cyber and IT Service Management. pp. 1–6. IEEE (2016)
23. Lashkari, A.H., Draper-Gil, G., Mamun, M.S.I., Ghorbani, A.A., et al.: Characterization of tor traffic using time based features. In: ICISSp. pp. 253–262 (2017)
24. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
25. MacQueen, J.: Classification and analysis of multivariate observations. In: 5th Berkeley Symp. Math. Statist. Probability. pp. 281–297. University of California Los Angeles LA USA (1967)
26. Mohammadi, S., Mirvaziri, H., Ghazizadeh-Ahsaee, M., Karimipour, H.: Cyber intrusion detection by combined feature selection algorithm. Journal of information security and applications **44**, 80–88 (2019)
27. Neto, E.C.P., Taslimasa, H., Dadkhah, S., Iqbal, S., Xiong, P., Rahman, T., Ghorbani, A.A.: Ciciov2024: Advancing realistic ids approaches against dos and spoofing attack in iov can bus. Internet of Things **26**, 101209 (2024)
28. Prasath, S., Sethi, K., Mohanty, D., Bera, P., Samantaray, S.R.: Analysis of continual learning models for intrusion detection system. IEEE Access **10**, 121444–121464 (2022)
29. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20**, 53–65 (1987)
30. Rüschendorf, L.: The wasserstein distance and approximation theorems. Probability Theory and Related Fields **70**(1), 117–129 (1985)
31. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp **1**, 108–116 (2018)
32. Turcotte, M.J.M., Kent, A.D., Hash, C.: Unified Host and Network Data Set, chap. chapter 1, pp. 1–22. World Scientific (nov 2018). `https://doi.org/10.1142/9781786345646_001`
33. Venkatesha, S., Reddy, K., Chandavarkar, B.: Social engineering attacks during the covid-19 pandemic. SN Computer Science **2**(78) (2021). `https://doi.org/10.1007/s42979-020-00443-1`
34. Wang, Z., Ni, Y., Jing, B., Wang, D., Zhang, H., Xing, E.: Dnb: A joint learning framework for deep bayesian nonparametric clustering. IEEE Transactions on Neural Networks and Learning Systems **33**(12), 7610–7620 (2021)
35. Xiao, H., Huang, M., Zhu, X.: Transg: A generative model for knowledge graph embedding. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2316–2325 (2016)
36. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning. pp. 3987–3995. PMLR (2017)