



# Improving neural network's robustness on tabular data with D-layers

Haiyang Xia<sup>1</sup> · Nayyar Zaidi<sup>2</sup> · Yishuo Zhang<sup>2</sup> · Gang Li<sup>2</sup>

Received: 16 January 2023 / Accepted: 12 July 2023 / Published online: 31 August 2023  
© The Author(s) 2023

## Abstract

Artificial neural networks (ANN) are widely used machine learning models. Their widespread use has attracted a lot of interest in their robustness. Many studies show that ANN's performance can be highly vulnerable to input manipulation such as adversarial attacks and covariate drift. Therefore, various techniques that focus on improving ANN's robustness have been proposed in the last few years. However, most of these works have mostly focused on image data. In this paper, we investigate the role of discretization in improving ANN's robustness on tabular datasets. Two custom ANN layers— D1-Layer and D2-Layer (collectively called D-Layers) are proposed. The two layers integrate discretization during the *training phase* to improve ANN's ability to defend against adversarial attacks. Additionally, D2-Layer integrates dynamic discretization during *testing phase* as well, to provide a unified strategy to handle adversarial attacks and covariate drift. The experimental results on 24 publicly available datasets show that our proposed D-Layers add much-needed robustness to ANN for tabular datasets.

**Keywords** Robustness · Covariate drift · Adversarial attack · Tabular data · Discretization

## 1 Introduction

The widespread use of ANN models has attracted a lot of interest in their robustness (Goodfellow et al. 2014; Kurakin et al. 2017). Typically, one measure of ANN's robustness is to see whether it can maintain performance with the changes in input data. These changes can be driven by either malicious or benign intent. An example of malicious intent change is adversarial attacks that manipulate input data to sway the model output towards a desirable outcome (Akhtar and Mian 2018). An example

---

Responsible editor: Charalampos Tsourakakis.

---

Haiyang Xia and Nayyar Zaidi have contributed equally to this work.

---

Extended author information available on the last page of the article

of benign intent change is data variation over time due to covariate drift. With the ever-changing facet of adversarial attack methods and datasets drifting over time, how to add robustness to ANN on tabular datasets remains an open, yet fundamental question. This work is motivated to address the issue of robustness in ANN models by proposing new novel layers in standard ANN architecture.

Traditionally, adversarial attacks constitute imperceptible perturbations on the input image to control ANN's output. Many studies have demonstrated that the perturbed images that fail one model can also fail other models trained on different datasets with different architectures (Goodfellow et al. 2014), highlighting the severity of the problem. In the past few years, plenty of research efforts have been exerted in designing appropriate defence mechanisms for ANN models on image datasets (Kurakin et al. 2017; Shafahi et al. 2019). However, image datasets are not the only datasets susceptible to adversarial attacks. Tabular datasets that are commonly used in various ANN applications domains such as finance and medicine are as vulnerable to adversarial attacks as image datasets (Cartella et al. 2021). Tabular datasets have one trait, i.e., the presence of categorical features that can serve as a natural defence against adversarial attacks, as the adversarial perturbations on categorical features can be easily observed. For instance, in a loan approval scenario, the `level-of-education` is `bachelor`, `master`, and `doctorate` (normally represented as integers like 1, 2, and 3), it is easy for bank managers to find the fraudulent modification when a customer modifies her education level from 2 to 2.5 or from 3 to 4, to obtain a loan. However, numeric features in tabular datasets are as vulnerable to adversarial attacks as pixel values in images (Cartella et al. 2021). Considering the natural defence capability of categorical features, recently, various discretization-based defence methods have been proposed (Buckman et al. 2018).

Covariate drift which informally refers to the situation where testing distribution is different from training distribution can also adversely affect ANN models' performance (Nado et al. 2020). Several studies in domain adaptation and causal inference aim to tackle the covariate drift issue by taking advantage of the information on testing distribution (Magliacane et al. 2017). Discretization can serve as a natural defence against some forms of covariate drifts as well. For example, if the `Salary` feature (at the training time) is discretized with equal frequency discretization into three bins  $\{A, B, C\}$ —even with covariate drift resulting in the monotonic transformation of the testing data—discretization on the transformed `Salary` feature (at the testing time) can result in a similar allocation of the bins.

Given the pivotal role that discretization can play as a defence mechanism against adversarial attacks and covariate drift, there is a need to integrate discretization into ANN's models for increasing their robustness. In this work, we propose two customized new layers for ANN – named `D1-Layer` ('Discretization') and `D2-Layer` ('Dynamic Discretization')—collectively called `D-Layers`, to address this need. The main motivations of these two layers are:

- Existing discretization-based adversarial attack defence methods (Buckman et al. 2018; Zhou et al. 2022) normally discretize data prior to training the model. Despite their effectiveness, if some part of the training data is changed (e.g., as part of adversarial training), the discretization results will be incorrect, as the

discretization boundaries are learned beforehand. Furthermore, any time the model is to be re-trained requires re-discretizing the dataset. The seamless integration of discretization in ANN(during model training) and exploiting its benefits is the main motivation for our proposed *D-Layers*.

- Once the model is trained, there is no way to update the discretization boundaries. However, if the distribution of testing data is changed due to covariate drift or adversarial attack, there is a strong need to update discretization boundaries to accommodate this distribution change. In other words, we need dynamic discretization at the testing phase to resist potential distribution changes caused by covariate drift or adversarial attack. The seamless integration of such dynamic discretization in ANN(during model testing) is the main motivation for our proposed *D2-Layer*.

The main contributions of this paper are:

- We have proposed two new layers for adding robustness to ANN models. Specifically, *D1-Layer* integrates discretization during the training phase to improve ANN's ability to defend against adversarial attacks. Whereas, *D2-Layer* integrates discretization during the training phase, as well as during the testing phase to provide a unified strategy for ANN to handle covariate drift and adversarial attacks.
- We demonstrate that our proposed *D1-Layer* lead to the state-of-the-art (SOTA) defence mechanism against a range of standard attacks on various publicly available tabular datasets.
- We demonstrate that our proposed *D2-Layer* offers an effective unified strategy to address adversarial attacks and covariate drift at the same time.

The rest of this paper is organized as follows. In Sect. 2, we review the related works. In Sect. 3, we present our proposed formulations namely *D-Layers*. Section 4 provides an empirical evaluation of our proposed formulations. In Sect. 5, we conclude the paper with pointers to future works.

## 2 Related work

### 2.1 Adversarial attack methods

To highlight the robustness of ANN models, a large number of adversarial attack models have been proposed in the literature in the past few years. Broadly, these existing adversarial attack models can be divided into white-box attack models and black-box attack models (Kumová and Pilát 2021). Attack models that require access to information of the original ANN model such as parameters, gradient, or structure to conduct attacks are referred to as white-box attack models, otherwise black-box attack models (Huang et al. 2020). Our study focuses on white-box attack models and hence we will mainly review popular white-box attack models. A more comprehensive literature review can be found in (Kong et al. 2021; Huang et al. 2020).

FGSM (Fast Gradient Sign Method) (Goodfellow et al. 2014) is the most classic white-box attack model for both image and tabular data. It creates adversarial samples by adding gradients to the original instance. FGSM is easy to implement but normally has a relatively low success rate, as the adversarial samples created by adding gradient may be insufficient to cross the decision boundary (Shafahi et al. 2019). A direct extension of FGSM is BIM (Basic Iterative Method) which iteratively conduct FGSM multiple times with a small step size to achieve better attack performance (Kurakin et al. 2016). PGD (Projected Gradient Descent) is also a popular white-box attack model built on top of FGSM (Madry et al. 2017). Different from BIM that directly iteratively conducts FGSM from the original sample, PGD initializes the start of the adversarial attack from a random distribution to add variations to the attack to further improve the attack's success rate.

Another popular white-box attack model is DeepFool (Moosavi-Dezfooli et al. 2016). It works by iteratively linearizing the model to generate unperceivable adversarial examples. Compared with other gradient-based white-box attack models, DeepFool is more efficient as it can always generate adversarial examples that are close to the decision boundary. LowProFool is the state-of-the-art white-box attack model on tabular data (Ballet et al. 2019). It induces parameter updates toward the targeted class by utilizing the gradient of adversarial noise. The importance weights of features are evaluated to ensure large perturbations only exist on irrelevant or less important features, such that the generated examples are imperceptible to expert scrutiny.

## 2.2 Adversarial defence methods

Madry et al. (2017) is the most straightforward defence method—it takes advantage of adversarial training to minimize models' adversarial risk to defend against adversarial attacks. Despite its effectiveness, one critical issue of this adversarial-training-based defence model is overfitting to attacks that generate adversarial samples (Kurakin et al. 2017). For example, the models that were adversarially trained to resist FGSM frequently failed to resist L-BFGS and BIM attacks. Thereby, recent studies have started to advocate input discretization as the defence mechanism. Thermometer encoding (Buckman et al. 2018) is one of the most popular discretization-based defence models as it defends against adversarial attacks by discretizing the numeric inputs to  $[0, 1]$  vectors. For example, discretizes 0.23 to  $[0, 0, 1, 1, 1, 1, 1, 1, 1, 1]$ , 0.34 to  $[0, 0, 0, 1, 1, 1, 1, 1, 1, 1]$ , etc. Thermometer's formulation is very similar to one-hot encoding, but it can preserve the order of input after discretization, thus having better performance than one-hot encoding. In the context of deep ANN models implemented via Keras,<sup>1</sup> one can utilize Keras discretization layer.<sup>2</sup> It offers another method to discretize neural network input. It is important to note that unlike other discretization methods which are feature-based

<sup>1</sup> <https://keras.io/>.

<sup>2</sup> [https://keras.io/api/layers/preprocessing\\_layers/categorical/discretization/](https://keras.io/api/layers/preprocessing_layers/categorical/discretization/).

(i.e., different cut-points are learned for different features)—the discretization strategy in this layer learns one set of cut-points for all the features—i.e., data across all features is used to compute the quantiles—which are later used as the cut-points to discretize. Little efforts have been made to investigate the effectiveness of `keras` discretization layer in defending against adversarial attacks. We will explore this direction together by proposing two discretization-inspired algorithms in this work.

D2A3 and D2A3N (Zhou et al. 2022) are state-of-the-art defence models on tabular data. D2A3 defends against adversarial attacks by exploiting both input discretization and adversarial training. In D2A3, the numeric input features are discretized to train a discretized model—this model is then improved by taking advantage of adversarial training. The main limitation of D2A3 is the requirement for accessing input data and changing it from numeric to discrete, which may be impossible in many application scenarios. In D2A3N, the numerical input features are discretized by the cut-points directly learned from the training data—data close to cut-points are considered adversarial samples and are replaced by the median of the bin to defend against adversarial attacks. Although these existing studies demonstrated the effectiveness of input discretization as a defence mechanism against tabular data adversarial attacks, their performance can be further improved by integrating flexible within model cut-points learning strategies as well as dynamic discretization—strategies that we will study in this work. Note, D2A3 and D2A3N are state-of-the-art adversarial defence approaches in the context of deep ANN. Therefore, we will consider these approaches as the baseline when comparing the adversarial defence capability of the proposed methods in this work. The main advantage of our proposed `D-Layers` over D2A3 and its variant is that the defence mechanism does not include adversarial training. Secondly, and importantly, our proposed method integrates discretization in the learning of an ANN model, unlike a pre-discretization strategy of D2A3. We will discuss in the following, that this trait is one of the reasons for the superior performance of `D-Layers`. One limitation of our proposed approach in handling covariate drift is its inability to handle non-monotonic transformations (or drifts). This is because, as we will also discuss below, `D-Layers` are based on equal frequency discretization, and hence assumes that order is preserved during the drift. However, if the order is not preserved, our proposed layers will not be effective. We are working on how to handle non-monotonic drifts as well as concept drift as an extension of this research.

### 2.3 Covariate drift

Covariate drift also known as covariate shift represents a typical model drift scenario that occurs when the distribution of the testing data is different from the training data (Sugiyama et al. 2007). In covariate drift, the distribution change only lies in the input features, whilst the labels of testing data remain the same (Bickel et al. 2009). The case in which the labels of the testing data change as well is called *concept drift* (Gama et al. 2014)—which is outside the scope of this work. Covariate drift can significantly compromise the performance of a well-trained ANN, therefore, a bunch of studies on domain adaptation (Chen et al. 2020) and transfer learning

(Wang et al. 2019) have been conducted to address the covariate drift problem through the alignment of training and testing distributions (Wilson and Cook 2020). For example, (Gretton et al. 2009) proposed a kernel mean matching-based method to match the training and testing distributions by reweighting the training distribution in a reproducing kernel Hilbert space. Li et al. (2020) proposed an important weighting method for addressing the covariate drift by reweighting the residuals of kernel mean matching and non-parametric regression. Zhang et al. (2020) proposed a strategy that learns the weights required to address covariate shifts in only one step. Pathak et al. (2022) proposed a new measurement to measure the distribution mismatch between training and testing data based on the integrated ratio of probabilities of balls at a given radius, and demonstrated its effectiveness in addressing covariate drift in non-parametric regression. The main limitation of these approaches is the dependence on the prior knowledge of testing data that is not always available at the training stage (Nair et al. 2019). Although some recent studies on causal inference have tackled this issue by utilizing the stability of causal graphs (Yu et al. 2020), the proposed models are complicated due to the difficulty of capturing causal relations in the data. This paper will show that simple input discretization can be an effective method to handle some forms of covariate drift, i.e., monotonic covariate drift.

Covariate and concept drift have been widely studied in machine learning, however, most of this work aims to develop models that have a built-in mechanism to handle either concept or covariate drift, e.g., Pfahringer et al. (2007); Bifet and Gavaldà (2009), Oza and Russell (2001). Our work, in this paper, is different from various existing works, as we specifically are interested to address covariate drift in deep ANN models. Therefore, we have not conducted a comparison with the existing concept or covariate drift method in this work, as it does not offer a meaningful comparison. We, however, are interested to do this analysis as part of future works for this research. Note, the baseline for measuring the effectiveness of our proposed method in handling covariate drift is a vanilla deep ANN model.

### 3 Methodology

In this section, we start by formulating the problem of robust ANN, followed by discussing the motivations for using discretization to improve ANN's robustness. Later, we present in detail our proposed D1-Layer and D2-Layer.

#### 3.1 Problem formulation

**Definition 1** (*Adversarial Attack on Tabular Data*) Let  $(\mathbb{X}, \mathbb{Y}) = \{(X^1, Y^1), (X^2, Y^2), \dots, (X^n, Y^n)\}$  be a dataset with  $n$  samples, where  $\mathbb{X}$  is defined by a set of features  $j \in \mathbb{J}$ ,  $\mathbb{Y} = [Y^1, Y^2, \dots, Y^n]$  denotes the corresponding labels. Let  $f : \mathbb{R}^D \rightarrow \mathbb{Y}$  be the trained ANN model. For a given sample  $(X^i, Y^i) \in (\mathbb{X}, \mathbb{Y})$ , the adversarial attack aims to generate an adversarial sample  $X_{\text{adv}}^i = (X^i + r^*)$  such that

$$\begin{aligned} f(X_{\text{adv}}^i) &= Y^t \neq f(X^i) = Y^i \\ \text{s.t. } X_{\text{adv}}^i &\in \mathbb{R}^D \text{ and } r^* = \arg \min_r d(r), \end{aligned} \quad (1)$$

where  $Y^t$  is the target label,  $d(r) = \|r\|_p$  is the perceptibility value that indicates the quantity of the changes in  $X^i$  after adding adversarial perturbation  $r$ .  $r^*$  is perturbation  $r$  that achieves minimum  $d(r)$ .

**Definition 2 (Covariate Drift)** For a model  $f : \mathbb{X} \rightarrow \mathbb{Y}$  covariate drift refers to the case where  $P_{\text{train}}(Y | X) = P_{\text{test}}(Y | X)$ , while  $P_{\text{train}}(X) \neq P_{\text{test}}(X)$ .

Here,  $P_{\text{train}}(X)$  is the distribution of the training data (without labels),  $P_{\text{test}}(X)$  is the distribution of the testing data (without labels),  $P_{\text{train}}(Y | X)$  is the conditional distribution of training data, and  $P_{\text{test}}(Y | X)$  is the conditional distribution of testing data.

Given Definitions 1 and 2, we have the following definition for robust ANN:

**Definition 3 (Robust ANN)** For a model  $f : \mathbb{X} \rightarrow \mathbb{Y}$  trained on the training dataset  $\mathcal{S}_{\text{data-train}} = (\mathbb{X}, \mathbb{Y})$ , suppose its performance on the testing dataset  $\mathcal{S}_{\text{data-test}}$  is  $D\%$ . For a perturbed dataset  $\tilde{\mathcal{S}}_{\text{data-test}}$  (based on Definitions 1 and 2),  $f$  is robust if its performance on  $\tilde{\mathcal{S}}_{\text{data-test}}$  is not less than  $D\% - \delta$ , where  $\delta$  is a user-specified confidence interval.

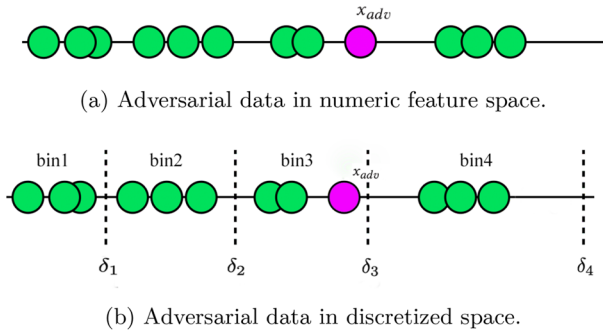
We will make use of this definition to evaluate (and compare) the effectiveness of our proposed formulations.

### 3.2 Rationales

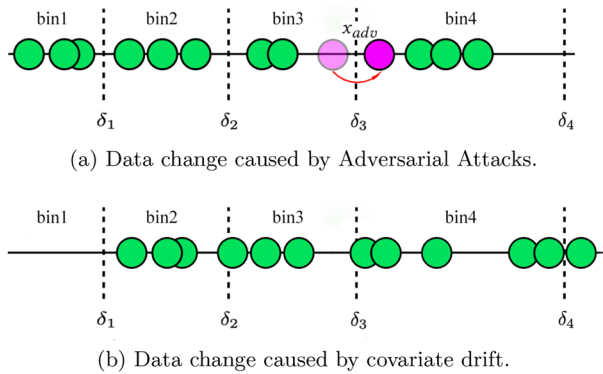
Discretization is performed by sorting the data and separating the numeric features into different bins according to the learned cut-points (also known as discretization boundaries).<sup>3</sup>

Figure 1 demonstrates the rationale of discretization-based adversarial attack defence methods. As shown in Fig. 1a, in the original numeric feature space, there is no way to differentiate adversarial example  $x_{\text{adv}}$  and other data. However, after discretization, the numeric features will be separated into different bins according to specific cut-points (see Fig. 1b). The bin number (e.g., 1, 2, 3, 4) or the median/mean of bin values will be used to train the ANN models. It can be seen that after discretization the adversarial example  $x_{\text{adv}}$  has been scaled back to a value that is expected by ANN (in our example, they are 1, 2, 3, 4). That means, whatever the attacker's intent was, discretization is able to convert adversarial samples back to the values that have a consistent format with training samples. The efficacy of this approach depends on the number of discretized values that cross the bin boundaries. For example, if  $x_{\text{adv}}$  in Fig. 1b moves to the right of  $\delta_3$ —its discretized value will be

<sup>3</sup> The cut-points are obtained based on different strategies, such as MDL, Equal Frequency (EF), etc.



**Fig. 1** Rationale of discretization-based defence models



**Fig. 2** Illustration of limitations of stationary cut-points in case of adversarial attack and covariate drift

incorrect (see Fig. 2a), thereby leading to performance degradation. Furthermore, as shown in Fig. 2b, a small drift of the data on the  $x$ -axis (covariate drift) will result in many data points being assigned to the wrong bins or even invalid bins. Based on this analysis, the following observations can be drawn:

1. Pre-discretizing the data is not an effective defence strategy,<sup>4</sup> as the pre-learned cut-points are learned on original data, and are static.

Every time data is modified, we must re-compute the cut-points and re-train the model (which can be expensive), to make discretization work as a defence strategy. Note, we are assuming that we have access to some adversarial or drifted data at the training time. There is a need for cut-points to be adjusted based on updated data during the training—we will call this *dynamic discretization*.

Our proposed  $D$ -Layers are aimed at incorporating dynamic discretization for adding robustness to the ANN model.

2. Cut-points should be dynamically updated from the data even during the testing time. If the data distribution is changed during the testing time (i.e., covariate

<sup>4</sup> Note, D2A3N adopts the strategy of pre-discretization.



drift), the cut-points should be changed accordingly to accommodate the changes to maintain discretization accuracy. Similar to batch normalization (Ioffe and Szegedy 2015), our proposed `D2-Layer` aims to address this issue by taking advantage of the statistical information of testing data.

### 3.3 D1-Layer

Let us start by formulating our problem. Ideally, we are interested in discretizing an input feature's numeric value, in an ANN model, i.e., a value say 23.5 is transformed into value say 3, based on some cut-points— $\delta_1, \dots, \delta_k$ . Our problem constitutes learning the cut-points in an end-to-end fashion, such that the whole process remains differentiable. For this, we have proposed a novel layer named `D1-Layer`, that does exactly that. The idea of `D1-Layer` is inspired by `VQ-VAE` (Vector Quantized Variational Auto-Encoder) that discretizes the encoder's output via a codebook to improve the quality of image generation (Van Den Oord et al. 2017). In `D1-Layer`, we aim to learn a cut-point space. We denote this space as— $\mathcal{C}$ , also known as the codebook. This codebook will be used to discretize the input features. For example, the simplest way to discretize a data point is by doing a nearest neighbour search in the codebook, i.e., the input data is represented by the index of the nearest codebook vector.

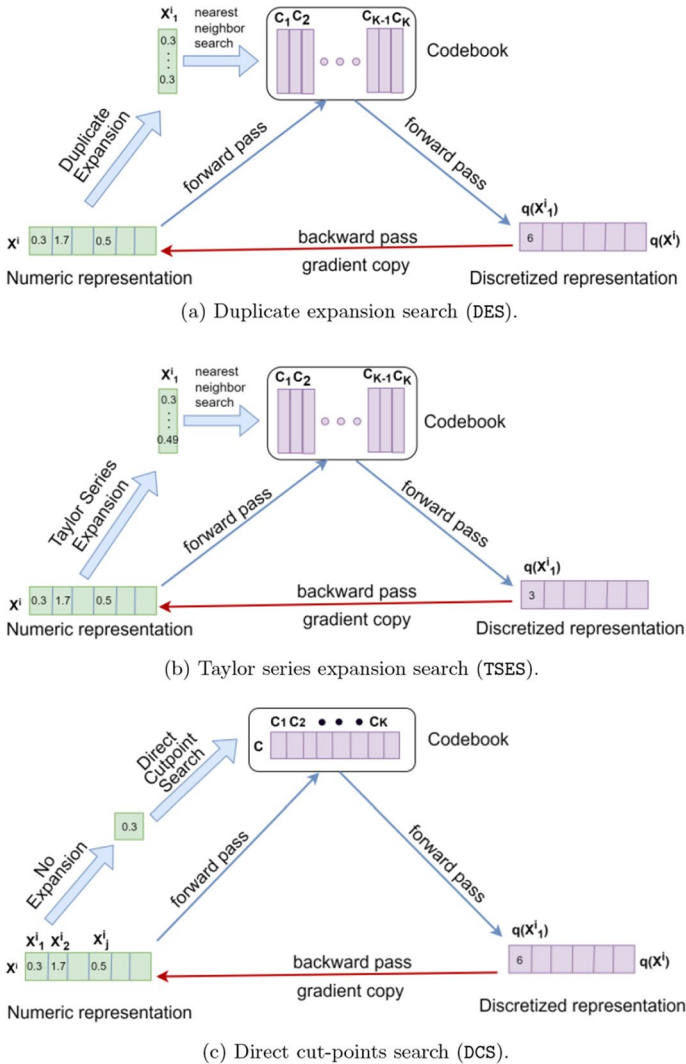
The salient feature of `D1-Layer` is that it actually aims to learn the codebook space which is basically the representation of the cut-points, i.e., the cut-point  $\delta_i$  is actually represented by a  $D$ -dimensional vector. The number of cut-points has to be specified in advance, e.g., if we have  $K$  cut-points, we have  $\mathcal{C} \in \mathbb{R}^{K \times D}$ . An issue that originates from enforcing the dimensions of the cut-point to be  $1 \times D$ , is the dimensionality mismatch between an input data feature (a scale) and the cut-point representation (a vector of size  $D$ ). This renders the comparison between input data feature and codebook vector (or nearest neighbour search) invalid. `D1-Layer` utilizes three strategies to address this dimensionality mismatch.<sup>5</sup> Let us discuss these strategies in the following.

#### 3.3.1 Duplicate expansion search (DES)

The first strategy that `D1-Layer` employs is to duplicate the scalar value  $D$ -times to convert it into a  $D$ -dimensional vector. This is depicted in Fig. 3a. Let  $\mathcal{Z}(\cdot)$  indicate an operator that takes a scalar value as input and returns a vector of size  $D$ . Formally, for the  $j$ -th feature of the data  $i$ , the duplicate expansion search can be defined as:

$$\mathcal{Z}(X_j^i) = \underbrace{[X_j^i, X_j^i, \dots, X_j^i]}_D.$$

<sup>5</sup> All three forms of representation assume that the input features are normalized by a min-max scaling.



**Fig. 3** Illustration of the discretization in D1-Layer. The numeric input in each feature is discretized by the codebook. The gradient of discretized feature representation will be directly copied to the numeric feature representation in the backward pass (straight-through estimator)

### 3.3.2 Taylor series expansion search (TSES)

Considering simple duplication may have less variation on the input representations, D1-Layer also employs a Taylor series expansion of  $\frac{1}{1-x^i_j}$  to expand the scalar value (as shown in Fig. 3b). Formally, the Taylor series expansion search can be defined as:

$$\mathcal{Z}(X_j^i) = [X_j^i, (X_j^i + X_j^{i^2}), \dots, (X_j^i + X_j^{i^2} + \dots + X_j^{i^D})].$$

For simplicity, we ignored the constant 1 in the Taylor series expansion.

For DES and TSES, after aligning the dimensionality of input features and codebook, the nearest neighbor search can be defined as:

$$q(X_j^i) = \operatorname{argmin}_k \| \mathcal{Z}(X_j^i) - C_k \|_2. \tag{2}$$

### 3.3.3 Direct cut-point search (DCS)

Other than expanding our input values to match the size of the cut-point space, one can also reduce the dimensionality of the cut-point space to match the input size. As shown in Fig. 3c, in DCS, we set the dimensionality of the cut-point space to  $1 \times K$ . The discretized value  $q(X_j^i)$  can then be determined as:

$$q(X_j^i) = \operatorname{argmin}_k \| X_j^i - C_k \|_2. \tag{3}$$

### 3.3.4 Learning in D1-Layer

The output of D1-Layer is the discretized data— $q(X_j^i)$ , that is passed through to the next layer for further processing. The forward pass through D1-Layer can be seen as the clustering of the input feature values—the index of each cluster center served as the discretized value.

The main challenge in training D1-Layer is that Eq. (2) (or 3) is non-differentiable due to the presence of the argmin operation. Similar to VQ-VAE, the simple gradient estimator strategy is adopted to address this issue (Bengio et al. 2013). That is, in the backward pass, gradients of the numeric representations are approximated by directly copying the gradient of the discretized representations (see Fig. 3). The loss function of our proposed D1-Layer-based ANN is:

$$L = L_c(Y^i, X^i) + \sum_{j=1}^{|\mathbb{J}|} \| \mathbf{sg}[Z(X_j^i)] - C_k \|_2^2, \tag{4}$$

where  $L_c(Y^i, X^i)$  represents the standard classification loss such as cross-entropy, MSE, etc.  $|\mathbb{J}|$  represents the number of features,  $\sum_{j=1}^{|\mathbb{J}|} \| \mathbf{sg}[q(X_j^i)] - C_k \|_2^2$  represents the codebook learning loss of data  $X^i$ , which directs the codebook embedding  $C_k$  toward the corresponding data value. Note,  $\mathbf{sg}[\cdot]$  is the stop gradient operator that has zero partial derivatives.

We have summarized the learning process of D1-Layer-based ANN in Algorithm 1. In the training phase, D1-Layer first discretizes the input data of each mini-batch—we denote discretized data as  $q(X)_b$  (Algorithm 1, lines 1–12). Note, we provide algorithm for TSES representation. Discretized data  $q(X)_b$  is then used in subsequent layers to train the network with parameters  $\Theta$  and codebook  $C$  (Algorithm 1, lines 13–20). In the testing phase, the learned codebook  $C$  is used to

discretize the input data, which is then fed into the network parameterized by parameter  $\Theta$ , for inference (Algorithm 1, lines 21–30).

---

**Algorithm 1** D1-Layer embedded ANN
 

---

**Input:**  $\mathcal{S}_{\text{data}} = (\mathbb{X}, \mathbb{Y})$ , length-of-embedding-vector  $D$ , number-of-embeddings  $K$ , search-strategy  $\in \{\text{DES}, \text{TSES}, \text{DCS}\}$  (default is TSES)

```

1 /*  $\mathcal{S}_{\text{data}}$  is split into  $\mathcal{S}_{\text{train}}$  and  $\mathcal{S}_{\text{test}}$  */
2 /* Training starts */
3 Initial parameters:  $\theta$  and  $\mathcal{C}$ , of the model  $f: \mathbb{X} \rightarrow \mathbb{Y}$ 
4 for iteration  $t \subset T$  in training do
5   for  $b$ -th mini-batch  $(\mathbb{X}, \mathbb{Y}) \in \mathcal{S}_{\text{train}}$  do
6     for  $i$ -th data in the batch  $(\mathbb{X}_b)$  do
7       /* Taylor Series Expansion */
8       for  $j$ -th feature in the data do
9          $\mathcal{Z}(X_j^i) = [X_j^i, (X_j^i + X_j^{i2}), \dots, (X_j^i + X_j^{i2} + \dots + X_j^{iD})]$ 
10         $q(X_j^i) = \text{argmin}_k \|\mathcal{Z}(X_j^i) - \mathcal{C}_k\|_2$  // Equation 2
11      end
12    end
13    /* Denoting  $q(X)_b$ , the output of discretized batch  $b$  */
14    Forward propagate  $q(X)_b$  through hidden layers
15    Calculate gradients w.r.t  $\theta$  and  $\mathcal{C}$  – denoted as  $\nabla_{\theta}$  and  $\nabla_{\mathcal{C}}$ 
16    /* Update parameters with default Adam optimization */
17     $\theta^{t+1} \leftarrow \theta_s^t + \eta \nabla_{\theta}$ ,  $\mathcal{C}^{t+1} \leftarrow \mathcal{C}_s^t + \eta \nabla_{\mathcal{C}}$ 
18  end
19 end
20 return  $f$  parameterized by  $\theta$  and  $\mathcal{C}$ 
21 /* Testing starts */
22 for  $i$ -th data in  $(\mathbb{X}, \mathbb{Y}) \in \mathcal{S}_{\text{test}}$  do
23   /* Taylor Series Expansion */
24   for  $j$ -th feature in the data do
25      $\mathcal{Z}(X_j^i) = [X_j^i, (X_j^i + X_j^{i2}), \dots, (X_j^i + X_j^{i2} + \dots + X_j^{iD})]$ 
26      $q(X_j^i) = \text{argmin}_k \|\mathcal{Z}(X_j^i) - \mathcal{C}_k\|_2$  // Equation 2
27   end
28   Forward propagate  $q(X^i)$  through hidden layers
29   return  $f(q(X^i))$ 
30 end
```

---

**Algorithm 2** D2-Layer embedded ANN

---

**Input:**  $\mathcal{S}_{\text{data}} = (\mathbb{X}, \mathbb{Y})$ , bin number  $m$

```

1 /*  $\mathcal{S}_{\text{data}}$  is split into  $\mathcal{S}_{\text{train}}$  and  $\mathcal{S}_{\text{test}}$  */
2 /* Training starts */
3 Initial parameters:  $\theta$ , of the model  $f : \mathbb{X} \rightarrow \mathbb{Y}$ 
4 for iteration  $t \subset T$  in training do
5   for  $b$ -th mini-batch  $(\mathbb{X}_b, \mathbb{Y}) \in \mathcal{S}_{\text{train}}$  do
6     /* Learn EF cutpoints on batch  $b$  */
7      $\Phi_{\text{EF}}(\mathbb{X}_b) \sim [\delta_1, \dots, \delta_K]$ 
8     for  $i$ -th data in the batch  $(\mathbb{X}_b)$  do
9       for  $j$ -th feature in the data do
10        |  $q(X_j^i) = \mathbb{O}(\hat{\Phi}(X_j^i))$  // Discretization
11        end
12      end
13      /* Denoting  $q(X)_b$ , the output of discretized batch  $b$  */
14      Forward propagate  $q(X)_b$  through hidden layers
15      Calculate gradients w.r.t  $\theta$  – denoted as  $\nabla_{\theta}$ 
16      /* Update parameters with default Adam optimization */
17       $\theta^{t+1} \leftarrow \theta_s^t + \eta \nabla_{\theta}$ 
18    end
19  end
20 return  $f$  parameterized by  $\theta$ 
21 /* Testing starts */
22 for  $b$ -th mini-batch  $(\mathbb{X}, \mathbb{Y}) \in \mathcal{S}_{\text{test}}$  do
23   /* Re-Learn EF cutpoints on batch  $b$  */
24    $\Phi_{\text{EF}}(\mathbb{X}_b) \sim [\delta_1, \dots, \delta_K]$ 
25   for  $i$ -th data in the batch  $(\mathbb{X}_b)$  do
26     for  $j$ -th feature in the data do
27      |  $q(X_j^i) = \mathbb{O}(\hat{\Phi}(X_j^i))$  // Discretization
28      end
29    end
30    Forward propagate  $q(X^i)$  through hidden layers
31    return  $f(q(X^i))$ 
32  end

```

---

**3.4 D2-Layer**

D1-Layer utilizes an objective function of the form of Eq. (4) to learn a representation of the cut-points. A simpler strategy could be to use the statistical information present in each mini-batch of the data and adjust cut-points accordingly. Our proposed D2-Layer does exactly that. It takes advantage of the statistical information of each mini-batch to dynamically update the cut-points (which can be applied at training as well as testing time).

Let  $\mathcal{B} = [\mathbb{X}_1, \dots, \mathbb{X}_B]$  represent a mini-batch of size  $B$ . D2-Layer sorts the data in each mini-batch and calculates the cut-points using Equal Frequency (EF) discretization. Other forms of discretization can be used, however, we argue that EF discretization has desirable properties that can lead to some robustness in the model.

Let  $\Phi_{\text{EF}}(\mathbb{X})$  represent a discretization function that returns a set of  $K$  cut-points (based on EF discretization), learned on mini-batch  $\mathbb{X}$ :

$$\Phi_{\text{EF}}(\mathbb{X}) \sim [\delta_1, \dots, \delta_K]. \quad (5)$$

We have discretized value  $q(X_j^i) = \mathbb{O}(\hat{\Phi}(X_j^i))$ . Here,  $\hat{\Phi}(\cdot)$  is the function that applies the learned cut-points  $\Phi_{\text{EF}}(\mathbb{X})$  to the data, and  $\mathbb{O}(\cdot)$  is the function that represents the discretized value, e.g., one-hot-encoding, bin-number, etc. The discretized value  $q(X_j^i)$  is then used in subsequent layers of ANN to train the network. Let us discuss some salient features of D2-Layer:

1. The mean and variance of the output of the D2-Layer are guaranteed to be stationary and, therefore, the covariate drift can be largely eliminated (in cases where drift is due to monotonic transformation).
2. The discretization operator used in D2-Layer is not differentiable, hence, the gradient-based attacking for original input  $X^i$  will not be effective. Thus providing a defence against many forms of adversarial attacks.
3. D2-Layer can be deployed at the testing time—i.e., the cut-points can be adjusted based on testing data distribution—making it perfect to address covariate drift even after the model is trained. Note, one can re-train a codebook in D1-Layer at testing time, but this might not be effective, as learning a codebook representation of size  $K \times D$  requires much larger data and hence larger size of the batch. On the contrary, D2-Layer makes use of simpler statistics from the data, which can be obtained from a few test data points.

We have summarized the learning of the D2-Layer-based ANN at Algorithm 2.

In the training phase of D2-Layer, equal frequency discretization is used to learn the cut-points of each training batch (Algorithm 2, lines 1–12). The discretized values resulting from the learned cut-points are used to train the entire network (Algorithm 2, lines 13–20). The selection of feature-specific equal-frequency discretization is critical to the working of D2-Layer's algorithm—i.e., in handling covariate drift, and in warding-off adversarial attack. As we mentioned earlier, Keras discretization layer also learns cut-points but based on the quantiles of the whole input data rather than separately for each feature. We will integrate this quantiles-based discretization strategy in D2-Layer and compare it with other forms of discretizations later in Sect. 4.

In the testing phase, different from D1-Layer and other existing defence methods that use cut-points learned from training data, D2-Layer uses equal frequency discretization to learn new cut-points from each testing batch to ensure the cut-points are suitable for testing data (Algorithm 2, lines 21–32). This dynamic discretization strategy makes sure that D2-Layer can handle distribution drifts during the testing phase.

**Table 1** Statistic information of datasets

Dataset	$n$	$m_n$	$m_c$	Dataset	$n$	$m_n$	$m_c$
<b>covtype</b>	581,012	10	44	<b>sign</b>	12,546	9	3
<b>census-income</b>	299,285	35	5	<b>occupancy</b>	10,129	14	2
<b>skin-segmentation</b>	245,057	4	2	<b>satellite</b>	6435	37	6
<b>localization</b>	164,860	2	3	<b>page-blocks</b>	5473	11	5
<b>accelerometer</b>	153,001	4	0	<b>wall-following</b>	5456	25	4
<b>higgs</b>	98,050	28	0	<b>waveform-5000</b>	5000	21	0
<b>ipums.la.99</b>	88,443	23	38	<b>spambase</b>	4601	58	2
<b>connect-4</b>	67,557	43	3	<b>kr-vs-kp</b>	3196	0	36
<b>adult</b>	48,842	6	8	<b>sick</b>	3772	6	21
<b>letter-recog</b>	20,000	17	26	<b>hypothyroid</b>	3163	6	19
<b>magic</b>	19,020	11	2	<b>cmc</b>	1473	2	7
<b>gassensor</b>	13,790	128	0	<b>german</b>	1000	3	17

## 4 Experiments

In this section, we start by presenting the details of our experimental settings followed by the results and detailed analysis.

### 4.1 Experimental settings

#### 4.1.1 Datasets

We have used 24 classification datasets from UCI machine learning dataset repository.<sup>6</sup> All of these datasets have more than 1000 samples. Of the considered datasets, there are 5 datasets with more than 100,000 samples and are denoted as *Large*, 9 datasets with between 10,000 and 100,000 samples and are denoted as *Medium*, 10 datasets with between 1000 and 10,000 samples and are denoted as *Small*. The statistics information of these datasets is shown in Table 1, where  $n$ ,  $m_n$ , and  $m_c$  represent the number of samples, numeric features, and categorical features individually.

#### 4.1.2 Baseline methods and evaluation metric

In terms of adversarial attacks, three of the most commonly used white-box attack models, namely, FGSM, DeepFool (DPF), and LowProFool (LPF) have been adopted in our experiments. The parameters of these models are set as the values suggested in the respective original papers, e.g., the step size of FGSM is set to 0.1, the maximum iteration of LowProFool and DeepFool is set to 50, the trade-off factor of LowProFool is set to 10.

For defence, the state-of-the-art tabular data adversarial attack defence model D2A3N and Madry are selected as the baselines to test D-Layers embedded ANN's

<sup>6</sup> <https://archive.ics.uci.edu/ml/datasets>.

robustness. The ANN model without any defence method (denoted as `Clean`) is used as the baseline to demonstrate the severity of the robustness problem. The standard evaluation metric `Robust Accuracy` is used to evaluate our proposed `D-Layers`' performance in defending against adversarial attacks. Similar to `Standard Accuracy` that measures the ratio of correct predictions and total data points, `Robust Accuracy` measures model's accuracy under unsettled conditions such as attack and covariate drift (Zhou et al. 2022)—the higher the `Robust Accuracy`, the more robust the model, and vice-versa.

### 4.1.3 Implementations

`D-Layers` and all baselines are implemented with `PyTorch`. `D1-Layer` and `D2-Layer` are integrated into the first layer of an ANN that has 5 hidden layers with `ReLU` activation function and `Softmax` as the output layer. Each of the hidden layers has 100 neurons. The training epochs, batch size, and learning rate is set to 500, 100, and 0.0001 respectively. The number of bins  $K$  is set to 5 for both `D1-Layer` and `D2-Layer`. Embedding dimensions  $-D$ , in `D1-Layer` is set to 10. For the implementation of adversarial attack models, we use the code released in the original papers, which is available on `GitHub`.<sup>7</sup> For `D2A3N`, we implement the `Equal Frequency` discretization-based version without adversarial training for fair comparison (i.e., denoted as `D2A3N-EF` in the original paper). The parameters of `D2A3N` and referred attack models are set to the default values as provided in the paper. All the experiments were conducted on an `i7 - 10750` desktop PC with 16 GB RAM and single `NVIDIA GeForce GTX 1660 Ti` GPU.

#### 4.1.4 Evaluation scenario

To evaluate the effectiveness of our proposed `D-Layers` in improving ANN's robustness, we split the data into *training set* and *testing set*. The testing data is attacked via three attack methods as presented in Sect. 4.1.2 or modified with covariate drift. We will discuss the details of concept drift in the later section. Nonetheless, we call the data *modified testing data*. The proposed `D-Layers` formulation and other baselines are trained with *training data*. The performance of the trained model is evaluated on the *modified testing data*. The two-fold cross-validation is adopted for the train-test split, and the average robust accuracy results over five rounds are reported. The evaluation framework is illustrated in Fig. 4.

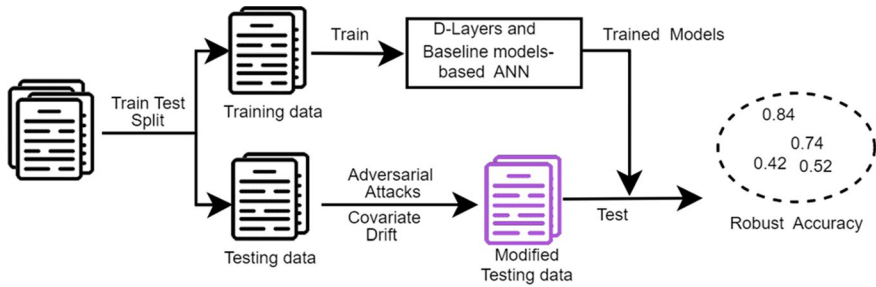
## 4.2 Experimental results

### 4.2.1 Comparison of `D1-Layer` search strategies

Before comparing the defence performance (robust accuracy) against adversarial attacks of our proposed `D-Layers` with baselines, we need to determine

<sup>7</sup> <https://github.com/axa-rev-research/LowProFool>.





**Fig. 4** The evaluation framework of the experiments

the best search strategy for *D1-Layer*. For this, we compared the performance of *D1-Layer* embedded ANN with three different search strategies, i.e., DES, TSES, and DCS. The average robust accuracies are presented in Fig. 5, where the results are broken across all, large, medium and small categories of datasets. Three attack methods of FGSM, LPF and DPF are used. It can be seen that in most cases TSES has higher robust accuracy than DES and DCS in defending all baseline attack methods (especially, in face of LPF attack). The pattern is consistent across *Large* and *Medium* datasets. On *Small* datasets, DCS performs better than other search strategies. For sake of simplicity, in the remainder of this paper, we only present *D1-Layer* results with TSES as representative of the three search techniques. The potential of ANN models is best-achieved with *Large* datasets. This is because on *Medium* and *Small* datasets, they can overfit the data. Our selection of TSES as representative is motivated by its extremely good performance of itself on *Large* collection of datasets.

#### 4.2.2 Defence against adversarial attacks

Let us now compare the performance of our proposed *D-Layers* with other baselines in terms of defending against adversarial attacks. The average robust accuracies of these methods are shown in Fig. 6.

From Fig. 6a, we can see that both *D1-Layer* and *D2-Layer* demonstrate higher robust accuracies than baselines on all datasets. This demonstrates the effectiveness of our proposed *D-Layers* in defending against adversarial attacks. The average robust accuracies of *Clean ANN* on all 24 datasets under FGSM, LPF, and DPF attacks are merely 0.37, 0.24, and 0.26 respectively. These alarming lower robust accuracies demonstrate that white-box adversarial attacks are quite effective in degrading the performance of ANN models. The higher average robust accuracy of *D2A3N* and *Madry* compared to *Clean ANN* demonstrates their effectiveness in defending against adversarial attacks. It is important to note that *D2A3N* is the state-of-the-art defence model. Let us compare the performance of *D-Layers* with *D2A3N* and *Clean ANN* in the following.

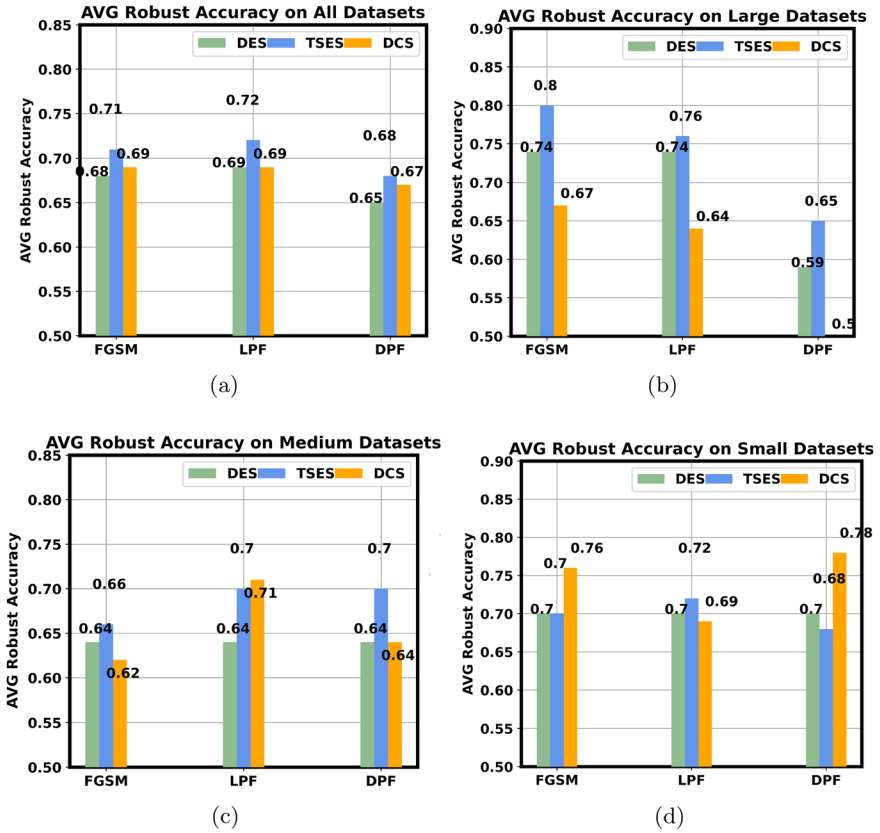


Fig. 5 Robust accuracy comparison of different search strategies for D1-Layer, under adversarial attacks

It is encouraging to see that D2-Layer leads to a performance improvement of 12, 9, and 14% on FGSM, LPF, and DPF attacks respectively over D3A3N. Compared with Clean ANN, the average robust accuracy improvement of D2-Layer on these three attacks reaches 34, 48, and 42% respectively.

It can be seen that D1-Layer achieves the highest average robust accuracy when compared with all other baselines. The average robust accuracy improvement of D1-Layer defence against FGSM, LPF, and DPF attacks compared to D2A3N reaches 18, 17, and 17% respectively; and that robust accuracy improvement compared to Clean ANN reaches 34, 48, and 42% respectively.

From Fig. 6b–d, we can see that D1-Layer wins against all baselines on almost all categories of datasets, the exception is Large with DPF attack. D2-Layer also shows superior performance on almost all categories of datasets, exceptions are Medium with FGSM and LPF attacks. Generally, we can conclude that in most cases D1-Layer and D2-Layer show significant performance improvement than

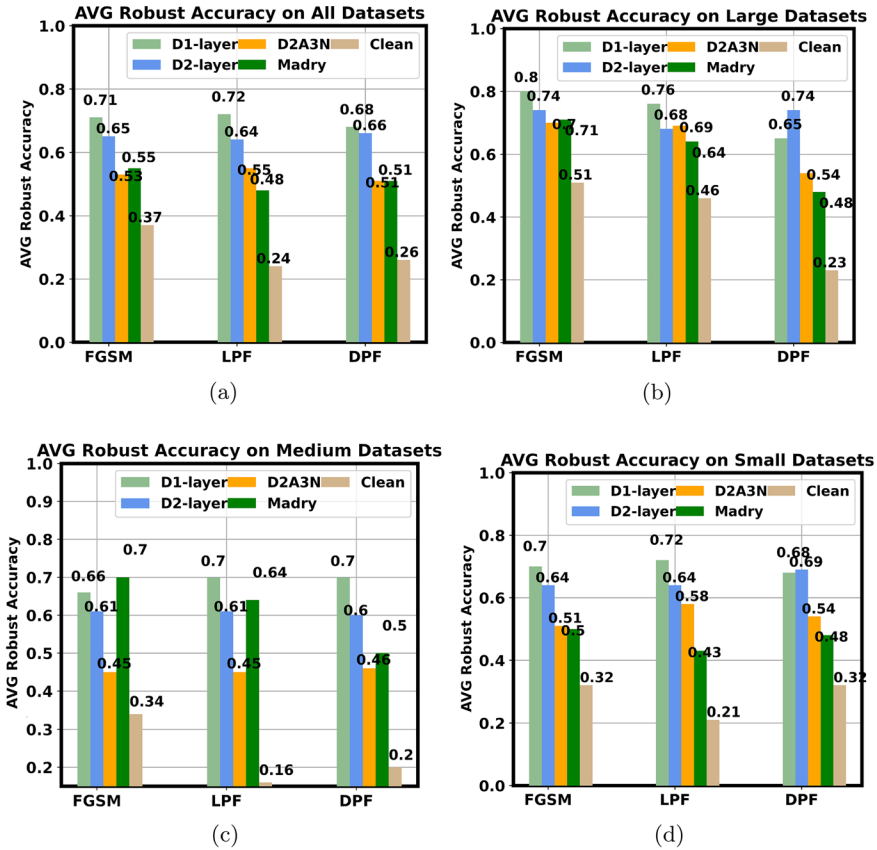


Fig. 6 Robust Accuracy of D1-layer, D2-layer and baselines under adversarial attacks

all other baselines on Large, Medium, and Small datasets. Also, D1-Layer has better performance in defending against adversarial attacks than D2-Layer and of course other baselines.

Let us now demonstrate the effectiveness of our proposed D-Layers' robustness by utilizing the robustness definition from Definition 3. In particular, we summarize the number of times a method's robust accuracy wins against the standard accuracy of an ANN by a certain margin— denoted as  $\delta$ , under LPF attack in Table 2.<sup>8</sup> The results are reported for varying values of  $\delta$ . It can be seen that D1-Layer and D2-Layer outperform all other baselines on all values of  $\delta$ , with D1-Layer (as we found earlier) is more robust than D2-Layer.

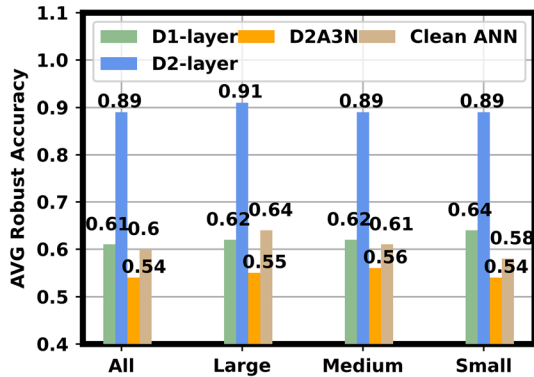
<sup>8</sup> Note, we only present results under LPF attack as a representative attack and also as it is the most powerful form of attack.

**Table 2** Number of wins of D1-Layer, D2-Layer, D2A3N, and Madry with varying the value of  $\delta$

$\delta$	D1-layer wins	D2-layer wins	D2A3N wins	Madry wins
$\delta = 25\%$	<b>17</b>	<b>12</b>	8	10
$\delta = 30\%$	<b>18</b>	<b>13</b>	11	10
$\delta = 35\%$	<b>19</b>	<b>15</b>	12	10

Best results are given in bold

**Fig. 7** Robust accuracy of D1-Layer, D2-Layer, D2A3N and ANN under covariate drift



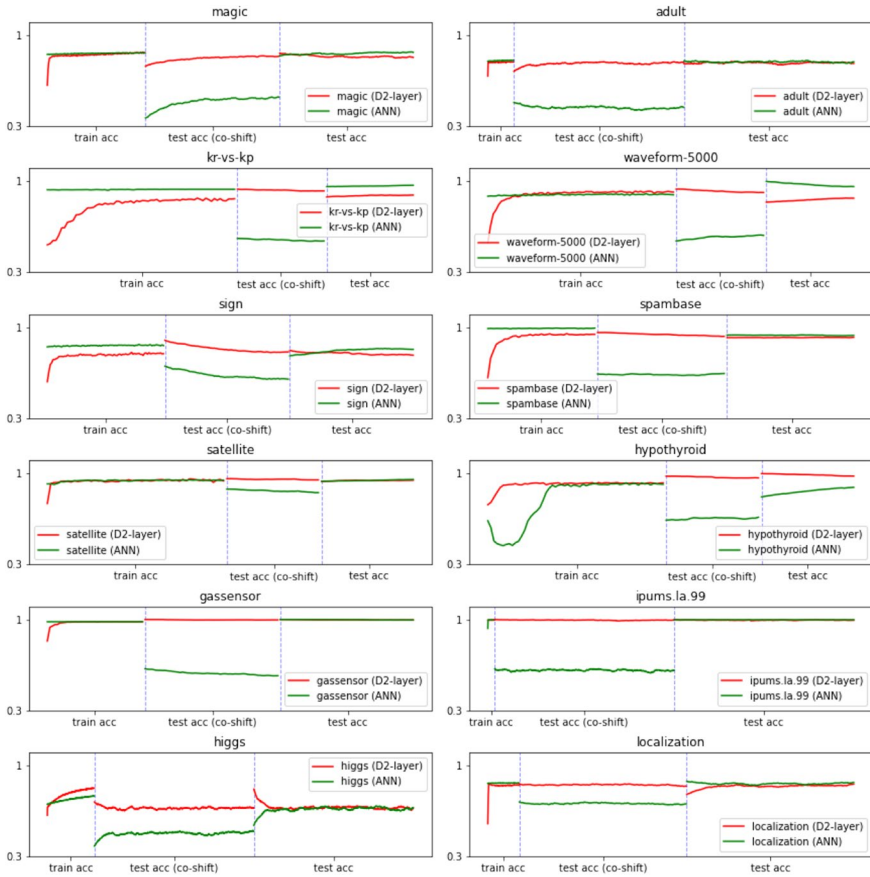
### 4.2.3 Handling covariate drift

The typical way of evaluating the covariate drift handling ability of models is to simulate the drift artificially in the data, then test the models' performance on the drifted data (Nair et al. 2019). For doing this, we followed the following procedures:

- We split each dataset into training set and testing set (as described in Sect. 4.1.4). We will refer to these sets as *training data* and *original test data* respectively, in the following discussions.
- We apply a non-linear transformation to all features ( $X_j^i = \alpha X_j^{i^5} + \beta X_j^i + \gamma$ ) on the testing set. The values of  $\alpha$ ,  $\beta$ , and  $\gamma$  are set to 1, 1, 300 for the transformation. We call this dataset as *drifted test data* in the following.
- The D1-Layer, D2-Layer, D2A3N, and Clean ANN are trained on the training set and tested on the *drifted test data*.

The average robustness of our proposed D2-Layer and baselines under monotonic covariate drift are presented in Fig. 7.<sup>9</sup> We can see that D2-Layer achieves the highest average robust accuracy (0.89) and wins against all baselines on Large, Medium, and Small datasets. The average performance improvement

<sup>9</sup> The detailed performance of D-Layers and baselines in handling monotonic covariate drift on each dataset are provided in Table 3 of the appendix.



**Fig. 8** Illustration of accuracy (with and without covariate drift) on various datasets. Plots show accuracy on the training data (during the training process), followed by accuracy of the trained model on the *drifted testing data*, followed by the accuracy of the trained model on *original testing data*

of D2-Layer compared to Clean ANN is 29% (which is quite impressive). This demonstrates the superiority of D2-Layer in handling monotonic covariate drift. It can be seen that D1-Layer and D2A3N can not address monotonic covariate drift at all.

To further demonstrate the effectiveness of D2-Layer in handling monotonic covariate drift, we visualize the accuracies of clean ANN and D2-Layer with and without drift on various datasets in Fig. 8. In particular, we plot the accuracies on *modified testing data* and *testing data*. For the sake of completeness, we also plot the model's performance during the training as well. From Fig. 8, we can see that during the covariate drift phase, there is a significant performance degradation of clean ANN (green line). However, the performance of the D2-Layer-based ANN model (red line) is maintained, which clearly demonstrates D2-Layer's ability in handling covariate drift. The inclusion of training accuracies in the results reveals that D2-Layer has a different convergence profile as compared to clean ANN.

#### 4.2.4 Selection of discretization strategies in D2-Layer

As we discussed in Sect. 3, D2-Layer can accommodate various discretization strategies. So far, in this work, we have constrained D2-Layer with equal frequency discretization. In this section, we will study the performance of D2-Layer with two other discretization techniques namely—Equal Width discretization (denoted as EW) and Quantile-based discretization technique based on Keras discretization layer (denoted as Quan). Note, Equal Frequency discretization is denoted as EF in the results. We have not tested the performance of D2-Layer with supervised methods such as MDL discretization, because, it is not possible to fix the number of bins with MDL discretization. That is, different batches in the data will lead to different numbers of bins. The inclusion of MDL discretization in D2-Layer has been left as a future work.

The average robust accuracy of D2-Layer with the three discretization methods (namely EF—default option in D2-Layer, EW, and Quan) under adversarial attacks and covariate drift is shown in Fig. 9.<sup>10</sup> We can see that D2-Layer with EF discretization (D2-EF) achieves better performance than that with EW discretization (D2-EW) and quantile-based discretization (D2-Quan).

#### 4.2.5 One strategy for two problems

Based on the experimental results in Sects. 4.2.2 and 4.2.3, we can establish that D2-Layer is efficient in terms of providing a defence against adversarial attacks as well as handling covariate drift. To clearly demonstrate this property, we plot the performance of D2-Layer under covariate drift and adversarial attack simultaneously, on two datasets, in Fig. 10. It can be seen that D2-Layer-based ANN has a consistent performance under the three attack methods and covariate shift. Its performance is consistently maintained within the  $\pm 25\%$  degradation boundaries (shown by orange lines in the figure). In contrast, there is significant performance degradation in the performance of the clean ANN model (green line).

## 5 Conclusions

In this paper, we proposed two ANN layers - D1-Layer and D2-Layer (collectively referred to as D-Layers) to improve the robustness of typical ANN models on tabular datasets. This is an extension of research focusing on the use of discretization in improving ANN's robustness (Zhou et al. 2022, 2023). The two layers are motivated by the need of adding discretization within the training of ANN models and, therefore, learn cut-point for discretizing the input data during the training phase. Furthermore, D2-Layer is motivated by the need for dynamic cut-point adjustment at the testing time. Through empirical evaluations, we demonstrated that

<sup>10</sup> The detailed robust accuracy of D2-Layer with the three discretization strategies on each dataset is provided in Table 4 of the appendix.

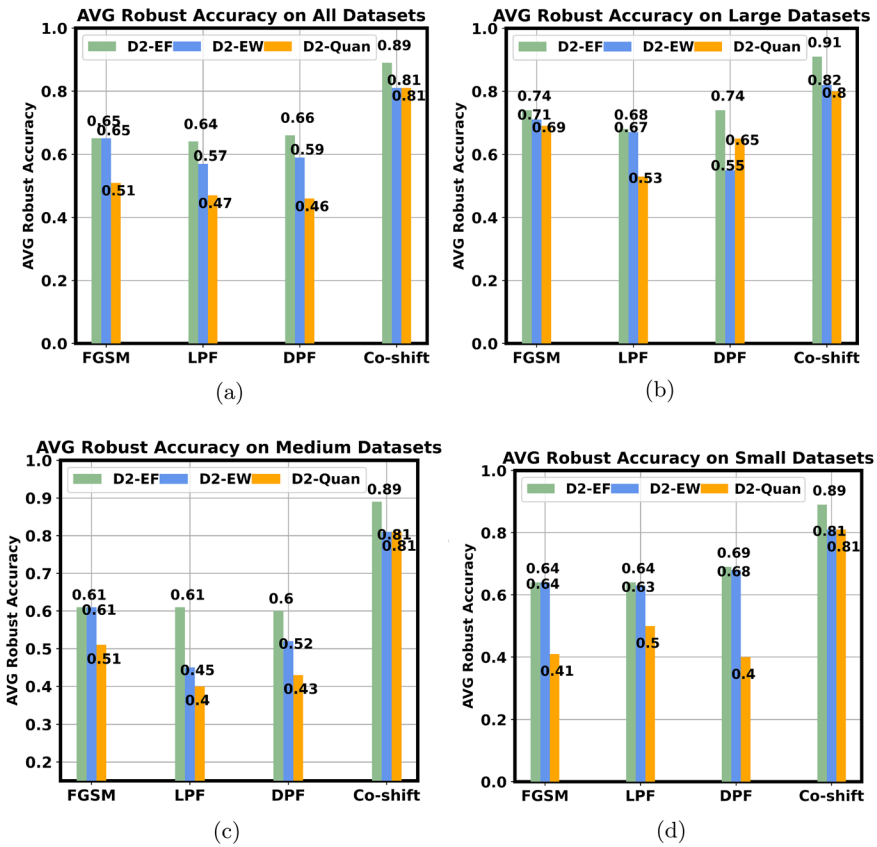


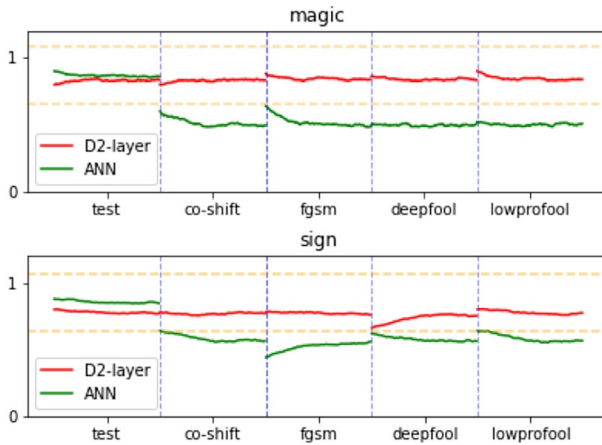
Fig. 9 Robust accuracy of D2-Layer under different discretization strategies (EF, EW and Quan)

D1-Layer and D2-Layer can be easily integrated into existing ANN models and provides an excellent mechanism for defending against adversarial attacks and for addressing some forms of covariate drift. Our experimental results revealed that:

1. D1-Layer leads to state-of-the-art (SOTA) defence performance against major forms of adversarial attacks on various tabular datasets.
2. D2-Layer leads to an effective strategy to address covariate drift and adversarial attacks at the same time.

Our future work entails:

- *Studying the application of D-Layers to the hidden layers of the network:* This will result in obtaining a discrete ANN and can lead to a network that is more robust to attacks and covariate drift. However, it can result in significant performance degradation. How to maintain a good performance while maintaining robustness is a question of great value, and we are currently investigating this.



**Fig. 10** Illustration of the robustness of D2-Layer to adversarial attacks and covariate drift by demonstrating its performance under various forms of attacks as well as covariate drift. Horizontal orange lines depict  $\delta = 25\%$ . The two models are applied on *testing data*, followed by *drifted testing data*, followed by *modified testing data* (due to FGSM, DPF and LPF attacks)

- *Studying the impact of the nature of input data:* That is, how the number of features, the number of categorical/numerical features, data size, etc. influence the performance of D-Layers in defending against adversarial attacks and addressing covariate shifts.
- *Studying the efficacy of D2-Layer for other forms of drift:* Currently, the proposed D2-Layer can only be effective against the monotonic drift in the data. We are currently exploring the effectiveness of D2-Layer against non-monotonic transformations as well as concept drifts.

## Appendix A: Code

The code of D-Layers, as well as the data used in this paper, along with experimental scripts, is available to be used at: <https://github.com/allwenau/DLayers>.

## Appendix B: Detailed results

The detailed results of all the experiments done in this paper can be found in Tables 3, 4, 5, 6.



**Table 3** Robust accuracy comparison of different search strategies for D1-Layer

Datasets	DES			TSES			DCS		
	FGSM	LPF	DPF	FGSM	LPF	DPF	FGSM	LPF	DPF
covtype	0.57	0.58	0.56	0.74	0.65	0.75	0.54	0.52	0.66
census-income	0.97	0.97	0.97	0.96	0.97	0.97	0.95	0.91	0.93
skin-segmentation	0.79	0.79	0.03	0.88	0.81	0.14	0.85	0.84	0.10
localization	0.89	0.89	0.89	0.89	0.89	0.89	0.52	0.39	0.36
accelerometer	0.48	0.48	0.49	0.53	0.48	0.49	0.49	0.53	0.46
higgs	0.54	0.54	0.54	0.51	0.50	0.51	0.53	0.55	0.48
ipums.la.99	0.58	0.58	0.57	0.66	0.92	0.65	0.89	0.95	0.65
connect-4	0.73	0.73	0.73	0.72	0.73	0.70	0.53	0.70	0.82
adult	0.76	0.76	0.76	0.77	0.79	0.79	0.69	0.76	0.66
letter-recog	0.49	0.49	0.49	0.55	0.60	0.50	0.17	0.36	0.33
magic	0.63	0.63	0.63	0.67	0.57	0.75	0.58	0.71	0.78
gassensor	0.52	0.52	0.52	0.72	0.75	0.84	0.78	0.86	0.88
sign	0.57	0.57	0.57	0.39	0.49	0.61	0.38	0.56	0.29
occupancy	0.94	0.94	0.94	0.98	0.95	0.95	1.00	0.89	0.89
satellite	0.57	0.57	0.57	0.84	0.81	0.97	0.67	0.63	0.92
page-blocks	0.91	0.91	0.91	0.92	0.92	0.92	0.91	0.90	0.91
wall-following	0.75	0.75	0.75	0.77	0.70	0.78	0.74	0.70	0.82
spambase	0.48	0.54	0.53	0.56	0.72	0.47	0.44	0.66	0.46
waveform-5000	0.56	0.56	0.56	0.56	0.56	0.56	0.61	0.32	0.60
kr-vs-kp	0.49	0.49	0.49	0.22	0.31	0.15	0.97	0.49	0.90
sick	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
hypothyroid	0.95	0.95	0.94	0.95	0.95	0.94	0.95	0.95	0.94
cmc	0.63	0.63	0.63	0.63	0.63	0.61	0.63	0.63	0.63
german	0.68	0.68	0.68	0.56	0.62	0.44	0.68	0.69	0.63
Average (All)	0.68	0.69	0.65	<b>0.71</b>	<b>0.72</b>	<b>0.68</b>	0.69	0.69	0.67
Average (Large)	0.74	0.74	0.59	<b>0.80</b>	<b>0.76</b>	<b>0.65</b>	0.67	0.64	0.50
Average (Medium)	0.64	0.64	0.64	<b>0.66</b>	<b>0.70</b>	<b>0.70</b>	0.62	0.71	0.64
Average (Small)	0.70	0.70	0.70	<b>0.70</b>	<b>0.72</b>	<b>0.68</b>	<b>0.76</b>	0.69	<b>0.78</b>

The best results across each attack method (for each search strategy) are given in bold

Table 4 Robust accuracy of D1-layer, D2-layer and baselines under adversarial attacks

Datasets	D1-layer			D2-layer			D2A3N			Madry			Clean ANN			ANN	
	FGSM	LPF	DPF	FGSM	LPF	DPF	FGSM	LPF	DPF	FGSM	LPF	DPF	FGSM	LPF	DPF	Std	Acc
<b>covtype</b>	0.74	0.65	0.75	0.46	0.40	0.58	0.57	0.64	0.57	0.58	0.42	0.35	0.37	0.07	0.93		
<b>census-income</b>	0.96	0.97	0.95	0.95	0.87	0.97	0.96	0.97	0.96	0.96	0.96	0.48	0.20	0.05	0.97		
<b>skin-segmentation</b>	0.88	0.81	0.14	0.90	0.81	0.79	0.83	0.90	0.09	0.83	0.80	0.78	0.78	0.28	0.99		
<b>localization</b>	0.89	0.89	0.89	0.87	0.78	0.84	0.58	0.41	0.52	0.64	0.42	0.45	0.49	0.42	0.86		
<b>accelerometer</b>	0.53	0.48	0.49	0.51	0.53	0.50	0.54	0.52	0.55	0.54	0.46	0.51	0.46	0.35	0.64		
<b>higgs</b>	0.51	0.50	0.51	0.44	0.58	0.49	0.48	0.47	0.43	0.53	0.51	0.39	0.34	0.36	0.68		
<b>ipums_la.99</b>	0.66	0.92	0.65	0.98	0.97	0.58	0.57	0.78	0.57	0.92	0.92	0.68	0.12	0.02	1.00		
<b>connect-4</b>	0.72	0.73	0.70	0.52	0.52	0.54	0.27	0.46	0.33	0.38	0.47	0.18	0.15	0.31	0.83		
<b>adult</b>	0.77	0.79	0.79	0.66	0.59	0.59	0.48	0.39	0.33	0.69	0.64	0.29	0.34	0.35	0.80		
<b>letter-recog</b>	0.55	0.60	0.50	0.51	0.07	0.53	0.17	0.01	0.22	0.16	0.01	0.12	0.01	0.03	0.99		
<b>magic</b>	0.67	0.57	0.75	0.49	0.57	0.57	0.27	0.21	0.36	0.39	0.19	0.19	0.12	0.15	0.85		
<b>gassensor</b>	0.72	0.75	0.84	0.62	0.42	0.78	0.59	0.48	0.66	0.33	0.05	0.32	0.00	0.01	0.99		
<b>sign</b>	0.39	0.49	0.61	0.31	0.81	0.45	0.25	0.32	0.28	0.25	0.25	0.23	0.23	0.21	0.82		
<b>occupancy</b>	0.98	0.95	0.95	0.92	0.94	0.83	0.99	0.90	0.98	1.00	1.00	0.69	0.13	0.32	1.00		
<b>satellite</b>	0.84	0.81	0.97	0.75	0.81	0.94	0.75	0.26	0.93	0.73	0.39	0.45	0.08	0.24	0.98		
<b>page-blocks</b>	0.92	0.92	0.92	0.62	0.74	0.61	0.13	0.68	0.27	0.04	0.02	0.37	0.02	0.40	0.98		
<b>wall-following</b>	0.77	0.70	0.78	0.60	0.54	0.75	0.48	0.61	0.64	0.35	0.50	0.45	0.16	0.19	0.98		
<b>waveform-5000</b>	0.56	0.72	0.47	0.54	0.55	0.44	0.34	0.51	0.32	0.29	0.06	0.21	0.06	0.07	0.91		
<b>spambase</b>	0.56	0.56	0.56	0.24	0.18	0.56	0.10	0.48	0.10	0.10	0.17	0.12	0.09	0.56	0.94		
<b>kr-vs-kp</b>	0.22	0.31	0.15	0.51	0.47	0.51	0.23	0.25	0.20	0.74	0.40	0.42	0.03	0.04	0.95		
<b>sick</b>	0.96	0.96	0.96	0.94	0.96	0.96	0.88	0.90	0.77	0.45	0.54	0.30	0.38	0.45	0.92		
<b>hypothyroid</b>	0.95	0.95	0.94	0.93	0.90	0.92	0.95	0.84	0.94	0.95	0.95	0.94	0.55	0.57	0.98		
<b>cmc</b>	0.63	0.63	0.61	0.53	0.59	0.53	0.65	0.67	0.68	0.64	0.60	0.61	0.39	0.36	0.70		

**Table 4** (continued)

Datasets	D1-layer			D2-layer			D2A3N			Madry			Clean ANN			ANN				
	FGSM	LPF	DPF	FGSM	LPF	DPF	FGSM	LPF	DPF	FGSM	LPF	DPF	FGSM	LPF	DPF	FGSM	LPF	DPF	Std	Acc
german	0.56	0.62	0.44	0.69	0.68	0.69	0.61	0.64	0.53	0.69	0.67	0.41	0.50	0.29	0.31	0.71	0.29	0.31	0.71	0.71
Average (All)	<b>0.71</b>	<b>0.72</b>	<b>0.68</b>	<b>0.65</b>	<b>0.64</b>	<b>0.66</b>	0.53	0.55	0.51	0.55	0.48	0.51	0.37	0.24	0.26	0.89	0.24	0.26	0.89	0.89
Average (Large)	<b>0.80</b>	<b>0.76</b>	<b>0.65</b>	<b>0.74</b>	<b>0.68</b>	<b>0.74</b>	0.70	0.69	0.54	0.71	0.64	0.48	0.51	0.46	0.23	0.88	0.46	0.23	0.88	0.88
Average (Medium)	<b>0.66</b>	<b>0.70</b>	<b>0.70</b>	<b>0.61</b>	<b>0.61</b>	<b>0.60</b>	0.45	0.45	0.46	0.70	0.63	0.50	0.34	0.16	0.20	0.88	0.16	0.20	0.88	0.88
Average (Small)	<b>0.70</b>	<b>0.72</b>	<b>0.68</b>	<b>0.64</b>	<b>0.64</b>	<b>0.69</b>	0.51	0.58	0.54	0.50	0.43	0.48	0.32	0.21	0.32	0.91	0.21	0.32	0.91	0.91

The best results across each attack method (for each search strategy) are given in bold  
 The last column of the table presents ANN's accuracy without attacks on each dataset, i.e., standard accuracy of each dataset

**Table 5** Robust accuracy of D1-Layer, D2-Layer, D2A3N and clean ANN under covariate drift

Datasets	D1-Layer	D2-Layer	D2A3N	Clean ANN
covtype	0.43	0.87	0.43	0.57
census-income	0.29	0.90	0.94	0.38
skin-segmentation	0.79	0.97	0.94	0.79
localization	0.81	0.96	0.50	0.81
accelerometer	0.31	0.86	0.21	0.39
higgs	0.72	0.89	0.92	0.72
ipums.la.99	0.70	0.91	0.47	0.70
connect-4	0.70	0.89	0.58	0.65
adult	0.66	0.89	0.73	0.61
letter-recog	0.66	0.89	0.24	0.61
magic	0.65	0.89	0.47	0.60
gassensor	0.65	0.89	0.65	0.60
sign	0.65	0.89	0.47	0.60
occupancy	0.29	0.90	0.50	0.38
satellite	0.65	0.89	0.48	0.60
page-blocks	0.65	0.89	0.93	0.60
wall-following	0.65	0.89	0.72	0.60
spambase	0.65	0.89	0.51	0.60
waveform-5000	0.65	0.89	0.59	0.60
kr-vs-kp	0.65	0.89	0.51	0.60
sick	0.65	0.89	0.06	0.60
hypothyroid	0.65	0.89	0.05	0.61
cmc	0.65	0.89	0.64	0.61
german	0.65	0.89	0.70	0.61
Average (ALL)	0.61	<b>0.89</b>	0.54	0.60
Average (Large)	0.62	<b>0.91</b>	0.55	0.64
Average (Medium)	0.62	<b>0.89</b>	0.56	0.61
Average (Small)	0.64	<b>0.89</b>	0.54	0.58

The best results are given in bold

**Table 6** Robust accuracy of D2-layer with different discretization strategy under adversarial attacks or covariate shift

Datasets	D2-EF				D2-EW				D2-Quan			
	FGSM	LPF	DPF	Co-shift	FGSM	LPF	DPF	Co-shift	FGSM	LPF	DPF	Co-shift
	covtype	0.46	0.40	0.58	0.87	0.37	0.44	0.58	0.75	0.55	0.48	0.55
census-income	0.95	0.87	0.97	0.90	0.87	0.74	0.03	0.82	0.93	0.46	0.87	0.80
skin-segmentation	0.90	0.81	0.79	0.97	0.89	0.82	0.79	0.84	0.83	0.80	0.79	0.84
localization	0.87	0.78	0.84	0.94	0.88	0.82	0.89	0.85	0.64	0.40	0.53	0.84
accelerometer	0.51	0.53	0.50	0.86	0.55	0.51	0.48	0.82	0.50	0.49	0.49	0.82
higgs	0.44	0.58	0.49	0.89	0.47	0.52	0.49	0.80	0.43	0.47	0.44	0.80
ipums_la.99	0.98	0.97	0.58	0.91	0.95	0.93	0.42	0.81	0.90	0.95	0.58	0.81
connect-4	0.52	0.52	0.54	0.89	0.76	0.63	0.74	0.81	0.53	0.48	0.50	0.81
adult	0.66	0.59	0.59	0.89	0.76	0.56	0.62	0.81	0.73	0.45	0.45	0.81
letter-recog	0.51	0.07	0.53	0.89	0.39	0.05	0.48	0.81	0.13	0.01	0.25	0.81
magic	0.49	0.57	0.57	0.89	0.49	0.41	0.59	0.81	0.32	0.33	0.48	0.81
gassensor	0.62	0.42	0.78	0.89	0.30	0.46	0.52	0.81	0.41	0.38	0.63	0.81
sign	0.31	0.81	0.45	0.89	0.41	0.40	0.51	0.81	0.28	0.33	0.49	0.81
occupancy	0.92	0.94	0.83	0.90	0.99	0.05	0.28	0.81	0.88	0.16	0.04	0.81
satellite	0.75	0.81	0.94	0.89	0.69	0.59	0.95	0.81	0.53	0.58	0.94	0.81
page-blocks	0.62	0.74	0.61	0.89	0.41	0.84	0.55	0.81	0.12	0.19	0.07	0.81
wall-following	0.60	0.54	0.75	0.89	0.75	0.72	0.75	0.81	0.53	0.66	0.66	0.81
waveform-5000	0.54	0.55	0.44	0.89	0.42	0.31	0.48	0.81	0.34	0.25	0.36	0.81
spambase	0.24	0.18	0.56	0.89	0.10	0.20	0.56	0.81	0.10	0.15	0.08	0.81
kr-vs-kp	0.51	0.47	0.51	0.89	0.95	0.62	0.67	0.81	0.11	0.18	0.02	0.81
sick	0.94	0.96	0.96	0.89	0.93	0.93	0.78	0.81	0.70	0.85	0.62	0.81
hypothyroid	0.93	0.90	0.92	0.89	0.87	0.76	0.95	0.81	0.42	0.92	0.21	0.81
cmc	0.53	0.59	0.53	0.89	0.64	0.56	0.58	0.81	0.62	0.54	0.58	0.81
german	0.69	0.68	0.69	0.89	0.68	0.72	0.56	0.81	0.63	0.66	0.46	0.81

Table 6 (continued)

Datasets	D2-EF			D2-EW			D2-Quan					
	FGSM	LPF	DPF	Co-shift	FGSM	LPF	DPF	Co-shift	FGSM	LPF	DPF	Co-shift
<b>Average (All)</b>	<b>0.65</b>	<b>0.64</b>	<b>0.66</b>	<b>0.89</b>	0.65	0.57	0.59	0.81	0.51	0.47	0.46	0.81
<b>Average (Large)</b>	<b>0.74</b>	<b>0.68</b>	<b>0.74</b>	<b>0.91</b>	0.71	0.67	0.55	0.82	0.69	0.53	0.65	0.80
<b>Average (Medium)</b>	<b>0.61</b>	<b>0.61</b>	<b>0.60</b>	<b>0.89</b>	0.61	0.45	0.52	0.81	0.51	0.40	0.43	0.81
<b>Average (Small)</b>	<b>0.64</b>	<b>0.64</b>	<b>0.69</b>	<b>0.89</b>	0.64	0.63	0.68	0.81	0.41	0.50	0.40	0.81

The best values across each attack method and covariate shift are given in bold

**Acknowledgements** The authors would like to thank Mark James Carman from Politecnico Di Milano, Italy and Jiahui Zhou from the School of Computer Science, Xi'an ShiYou University, for helpful discussions during the course of this research. The first author of the paper is supported by the Australian Government Research Training Program (AGRTP) Scholarship.

**Funding** Open Access funding enabled and organized by CAUL and its Member Institutions.

## Declarations

**Conflict of interest** Authors do not have any funding information to report. Authors do have any actual, perceived or potential conflict of interests (financial or non-financial) to disclose as well.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Akhtar N, Mian A (2018) Threat of adversarial attacks on deep learning in computer vision: a survey. *IEEE Access* 6:14410–14430
- Ballet V, Renard X, Aigrain J, Laugel T, Frossard P, Detyniecki M (2019) Imperceptible adversarial attacks on tabular data. *arXiv preprint arXiv:1911.03274*
- Bengio Y, Léonard N, Courville A (2013) Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*
- Bickel S, Brückner M, Scheffer T (2009) Discriminative learning under covariate shift. *J Mach Learn Res* 10(9):2137–2155
- Bifet A, Gavaldà R (2009) Adaptive learning from evolving data streams. *Advances in intelligent data analysis VIII*. Springer, Berlin, pp 249–260
- Buckman J, Roy A, Raffel C, Goodfellow I (2018) Thermometer encoding: One hot way to resist adversarial examples. In: *International Conference on Learning Representations*
- Cartella F, Anuniação O, Funabiki Y, Yamaguchi D, Akishita T, Elshocht O (2021) Adversarial attacks for tabular data: application to fraud detection and imbalanced data. In: *CEUR Workshop Proceedings*, vol 2808
- Chen M, Zhao S, Liu H, Cai D (2020) Adversarial-learned loss for domain adaptation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol 34, pp 3521–3528
- Gama J, Žliobaitė I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. *ACM Comput Surv* 46(4):1–37
- Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*
- Gretton A, Smola A, Huang J, Schmittfull M, Borgwardt K, Schölkopf B (2009) Covariate shift by kernel mean matching. *Dataset Shift Mach Learn* 3(4):5
- Huang X, Kroening D, Ruan W, Sharp J, Sun Y, Thamo E, Wu M, Yi X (2020) A survey of safety and trustworthiness of deep neural networks: verification, testing, adversarial attack and defence, and interpretability. *Comput Sci Rev* 37:100270
- Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning*. PMLR
- Kong Z, Xue J, Wang Y, Huang L, Niu Z, Li F (2021) A survey on adversarial attack in the age of artificial intelligence. *Wirel Commun Mobile Comput*. <https://doi.org/10.1155/2021/4907754>

- Kumová V, Pilát M (2021) Beating white-box defenses with black-box attacks. In: 2021 International Joint Conference on Neural Networks (IJCNN). IEEE, pp 1–8
- Kurakin A, Goodfellow IJ, Bengio S (2017) Adversarial machine learning at scale. In: 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings
- Kurakin A, Goodfellow I, Bengio S (2016) Adversarial machine learning at scale. arXiv preprint [arXiv:1611.01236](https://arxiv.org/abs/1611.01236)
- Li F, Lam H, Prusty S (2020) Robust importance weighting for covariate shift. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp 352–362
- Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2017) Towards deep learning models resistant to adversarial attacks. arXiv preprint [arXiv:1706.06083](https://arxiv.org/abs/1706.06083)
- Magliacane S, van Ommen T, Claassen T, Bongers S, Versteeg P, Mooij JM (2017) Domain adaptation by using causal inference to predict invariant conditional distributions. arXiv preprint [arXiv:1707.06422](https://arxiv.org/abs/1707.06422)
- Moosavi-Dezfooli S-M, Fawzi A, Frossard P (2016) Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2574–2582
- Nado Z, Padhy S, Sculley D, D’Amour A, Lakshminarayanan B, Snoek J (2020) Evaluating prediction-time batch normalization for robustness under covariate shift. arXiv preprint [arXiv:2006.10963](https://arxiv.org/abs/2006.10963)
- Nair NG, Satpathy P, Christopher J, et al (2019) Covariate shift: A review and analysis on classifiers. In: 2019 Global Conference for Advancement in Technology (GCAT). IEEE, pp 1–6
- Oza N, Russell S (2001) Online bagging and boosting. Artificial intelligence and statistics. Morgan Kaufmann, San Francisco, pp 105–112
- Pathak R, Ma C, Wainwright M (2022) A new similarity measure for covariate shift with applications to nonparametric regression. In: International Conference on Machine Learning. PMLR, pp 17517–17530
- Pfahringer B, Holmes G, Kirkby R (2007) New options for Hoeffding trees. In: AI 2007: Advances in Artificial Intelligence vol 4830. Springer, pp 90–99
- Shafahi A, Najibi M, Ghiasi A, Xu Z, Dickerson J, Studer C, Davis LS, Taylor G, Goldstein T (2019) Adversarial training for free! In: Advances in Neural Information Processing Systems, vol 32
- Sugiyama M, Krauledat M, Müller K-R (2007) Covariate shift adaptation by importance weighted cross validation. *J Mach Learn Res* 8(5):985–1005
- Van Den Oord A, Vinyals O et al (2017) Neural discrete representation learning. *Advances in neural information processing systems* 30
- Wang B, Qiu M, Wang X, Li Y, Gong Y, Zeng X, Huang J, Zheng B, Cai D, Zhou J (2019) A minimax game for instance based selective transfer learning. In: Proceedings of the 25th ACM SIGKDD
- Wilson G, Cook DJ (2020) A survey of unsupervised deep domain adaptation. *ACM Trans Intell Syst Technol* 11:1–46
- Yu Z, Wang P, Xu J, Xie L, Jin Z, Huang J, He X, Cai D, Hua X-S (2020) Stable learning via causality-based feature rectification. *CoRR*
- Zhang T, Yamane I, Lu N, Sugiyama M (2020) A one-step approach to covariate shift adaptation. In: Asian Conference on Machine Learning. PMLR, pp 65–80
- Zhou J, Zaidi N, Zhang Y, Li G (2022) Discretization inspired defence algorithm against adversarial attacks on tabular data. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp 367–379
- Zhou J, Zaidi N, Zhang Y, Montague P, Kim J, Li G (2023) Leveraging generative models for combating adversarial attacks on tabular datasets. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp 147–158

**Publisher’s Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



## Authors and Affiliations

Haiyang Xia<sup>1</sup> · Nayyar Zaidi<sup>2</sup> · Yishuo Zhang<sup>2</sup> · Gang Li<sup>2</sup>

✉ Nayyar Zaidi  
nayyar.zaidi@deakin.edu.au

Haiyang Xia  
haiyang.xia@anu.edu.au

Yishuo Zhang  
chris.zhang@deakin.edu.au

Gang Li  
gang.li@deakin.edu.au

<sup>1</sup> Research School of Management, Australian National University, Canberra, ACT 2601, Australia

<sup>2</sup> Centre for Cyber Resilience and Trust (CREST), Deakin University, Geelong, VIC 3216, Australia