# Dual-Component Deep Domain Adaptation: A New Approach for Cross Project Software Vulnerability Detection

Van Nguyen[1(✉)], Trung Le[1(✉)], Olivier de Vel[2(✉)], Paul Montague[2(✉)], John Grundy[1(✉)], and Dinh Phung[1(✉)]

[1] Monash University, Clayton, Australia
{van.nk,trunglm,john.grundy,dinh.phung}@monash.edu
[2] Defence Science and Technology Group, Canberra, Australia
{Olivier.DeVel,Paul.Montague}@dst.defence.gov.au

**Abstract.** Owing to the ubiquity of computer software, software vulnerability detection (SVD) has become an important problem in the software industry and computer security. One of the most crucial issues in SVD is coping with the scarcity of labeled vulnerabilities in projects that require the laborious manual labeling of code by software security experts. One possible solution is to employ deep domain adaptation (DA) which has recently witnessed enormous success in transferring learning from structural labeled to unlabeled data sources. Generative adversarial network (GAN) is a technique that attempts to bridge the gap between source and target data in the joint space and emerges as a building block to develop deep DA approaches with state-of-the-art performance. However, deep DA approaches using the GAN principle to close the gap are subject to the mode collapsing problem that negatively impacts the predictive performance. Our aim in this paper is to propose Dual Generator-Discriminator Deep Code Domain Adaptation Network (Dual-GD-DDAN) for tackling the problem of transfer learning from labeled to unlabeled software projects in SVD to resolve the mode collapsing problem faced in previous approaches. The experimental results on real-world software projects show that our method outperforms state-of-the-art baselines by a wide margin.

**Keywords:** Domain adaptation · Cyber security · Software vulnerability detection · Machine learning · Deep learning

## 1 Introduction

In the software industry, software vulnerabilities relate to specific flaws or oversights in software programs which allow attackers to expose or alter sensitive information, disrupt or destroy a system, or take control of a program or computer system. The software vulnerability detection problem has become an important issue in the software industry and in the field of computer security. Computer software development employs of a vast variety of technologies and different software development methodologies, and much computer software contains vulnerabilities.


H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12084, pp. 699–711, 2020.
https://doi.org/10.1007/978-3-030-47426-3_54

This has necessitated the development of automated advanced techniques and tools that can efficiently and effectively detect software vulnerabilities with a minimal level of human intervention. To respond to this demand, many vulnerability detection systems and methods, ranging from open source to commercial tools, and from manual to automatic methods have been proposed and implemented. Most of the previous works in software vulnerability detection (SVD) [1,8] have been developed based on handcrafted features which are manually chosen by knowledgeable domain experts who may have outdated experience and underlying biases. In many situations, handcrafted features normally do not generalize well. For example, features that work well in a certain software project may not perform well in other projects. To alleviate the dependency on handcrafted features, the use of automatic features in SVD has been studied recently [11–13]. These works have shown the advantages of automatic features over handcrafted features in the context of software vulnerability detection.

However, most of these approaches lead to another crucial issue in SVD research, namely the scarcity of labeled projects. Labelled vulnerable code is needed to train these models, and the process of labeling vulnerable source code is very tedious, time-consuming, error-prone, and challenging even for domain experts. This has led to few labeled projects compared with the vast volume of unlabeled ones. A viable solution is to apply transfer learning or domain adaptation which aims to devise automated methods that make it possible to transfer a learned model from the source domain with labels to the target domains without labels. Studies in domain adaptation can be broadly categorized into two themes: shallow [6] and deep domain adaptations [3,14,18]. These recent studies have shown the advantages of deep over shallow domain adaptation (i.e., higher predictive performance and capacity to tackle structural data). Deep domain adaptation encourages the learning of new representations for both source and target data in order to minimize the divergence between them [3,14,18]. The general idea is to map source and target data to a joint feature space via a generator, where the discrepancy between the source and target distributions is reduced. Notably, the work of [3,18] employed generative adversarial networks (GANs) [4] to close the gap between source and target data in the joint space. However, most of aforementioned works mainly focus on transfer learning in the computer vision domain. The work of [16] is the first work which applies deep domain adaptation to SVD with promising predictive performance on real-world source code projects. The underlying idea is to employ the GAN to close the gap between the source and target domains in the joint space and enforce the clustering assumption [2] to utilize the information carried in the unlabeled target samples in a semi-supervised context.

GANs are known to be affected by the mode collapsing problem [5,7,10,17]. In particular, the study in [17] recently studied the mode collapsing problem and further classified this into the missing mode problem i.e., the generated samples miss some modes in the true data, and the boundary distortion problem i.e., the generated samples can only partly recover some modes in the true data. It is certain that deep domain adaptation approaches that use the GAN principle will inherently encounter both the missing mode and boundary distortion problems. Last but not least, deep domain adaptation approaches using the GAN principle also face the data distortion problem. The representations of source and target examples in the joint feature space degenerate to

very small regions that cannot preserve the manifold/clustering structure in the original space.

Our aim in this paper is to address not only deep domain adaptation mode collapsing problems but also boundary distortion problems when employing the GAN as a principle in order to close the gap between source and target data in the joint feature space. Our two approaches are: i) apply manifold regularization for enabling the preservation of manifold/clustering structures in the joint feature space, hence avoiding the degeneration of source and target data in this space; and ii) invoke dual discriminators in an elegant way to reduce the negative impacts of the missing mode and boundary distortion problems in deep domain adaptation using the GAN principle as mentioned before. We name our mechanism when applied to SVD as *Dual Generator-Discriminator Deep Code Domain Adaptation Network* (Dual-GD-DDAN). We empirically demonstrate that our Dual-GD-DDAN can overcome the missing mode and boundary distortion problems which is likely to happen as in Deep Code Domain Adaptation (DDAN) [16] in which the GAN was solely applied to close the gap between the source and target domains in the joint space (see the discussion in Sects. 2.3 and 3.3, and the visualization in Fig. 3). In addition, we incorporate the relevant approaches – minimizing the conditional entropy and manifold regularization with spectral graph – proposed in [16] to enforce the clustering assumption [2] and arrive at a new model named *Dual Generator-Discriminator Semi-supervised Deep Code Domain Adaptation Network* (Dual-GD-SDDAN). We further demonstrate that our Dual-GD-SDDAN can overcome the mode collapsing problem better than SCDAN in [16], hence obtaining better predictive performance.

We conducted experiments using the data sets collected by [13], that consist of five real-world software projects: FFmpeg, LibTIFF, LibPNG, VLC and Pidgin to compare our proposed Dual-GD-DDAN and Dual-GD-SDDAN with the baselines. The baselines consider to include VULD (i.e., the model proposed in [12] without domain adaptation), MMD, DIRT-T, DDAN and SCDAN as mentioned [16] and D2GAN [15] (a variant of the GAN using dual-discriminator to reduce the mode collapse for which we apply this mechanism in the joint feature space). Our experimental results show that our proposed methods are able to overcome the negative impact of the missing mode and boundary distortion problems inherent in deep domain adaptation approaches when solely using the GAN principle as in DDAN and SCDAN [16]. In addition, our method outperforms the rival baselines in terms of predictive performance by a wide margin.

## 2 Deep Code Domain Adaptation with GAN

### 2.1 Problem Statement

A source domain data set $S = \{(x_1^S, y_1), \ldots, (x_{N_S}^S, y_{N_S})\}$ where $y_i \in \{-1, 1\}$ (i.e., 1: vulnerable code and $-1$: non-vulnerable code) and $x_i^S = [x_{i1}^S, \ldots, x_{iL}^S]$ is a sequence of $L$ embedding vectors, and the target domain data set $T = \{x_1^T, \ldots, x_{N_T}^T\}$ where $x_i^T = [x_{i1}^T, \ldots, x_{iL}^T]$ is also a sequence of $L$ embedding vectors. We wish to bridge the gap between the source and target domains in the joint feature space. This allows us to transfer a classifier trained on the source domain to predict well on the target domain.

### 2.2 Deep Code Domain Adaptation with a Bidirectional RNN

To handle sequential data in the context of domain adaptation of software vulnerability detection, the work of [16] proposed an architecture referred to as the Code Domain Adaptation Network (CD*A*N). This network architecture recruits a Bidirectional RNN to process the sequential input from both source and target domains (i.e., $x_i^S = [x_{i1}^S, \ldots, x_{iL}^S]$ and $x_i^T = [x_{i1}^T, \ldots, x_{iL}^T]$). A fully connected layer is then employed to connect the output layer of the Bidirectional RNN with the joint feature layer while bridging the gap between the source and target domains. Furthermore, inspired by the Deep Domain Adaptation approach [3], the authors employ the source classifier $C$ to classify the source samples, the domain discriminator $D$ to distinguish the source and target samples and propose Deep Code Domain Adaptation (DDAN) whose objective function is as follows:

$$\mathcal{J}(G, D, C) = \frac{1}{N_S} \sum_{i=1}^{N_S} \ell(C(G(x_i^S)), y_i) + \lambda\left(\frac{1}{N_S} \sum_{i=1}^{N_S} \log D(G(x_i^S)) + \frac{1}{N_T} \sum_{i=1}^{N_T} \log\left[1 - D(G(x_i^T))\right]\right)$$

### 2.3 The Shortcomings of DDAN

We observe that DDAN suffers from several shortcomings. First, the *data distortion* problem (i.e., the source and target data in the joint space might collapse into small regions) may occur since there is no mechanism in DDAN to circumvent this. Second, since DDAN is based on the GAN approach, DDAN might suffer from the mode collapsing problem [5,17]. In particular, [17] has recently studied the mode collapsing problem of GANs and discovered that they are also subject to i) the *missing mode* problem (i.e., in the joint space, either the target data misses some modes in the source data or vice versa) and ii) the *boundary distortion* problem (i.e., in the joint space either the target data partly covers the source data or vice versa), which makes the target distribution significantly diverge from the source distribution. As shown in Fig. 1, both the missing mode and boundary distortion problems simultaneously happen since the target distribution misses source mode 2, while the source distribution can only partly cover the target mode 2 in the target distribution and the target distribution can only partly cover the source mode 1 in the source distribution.

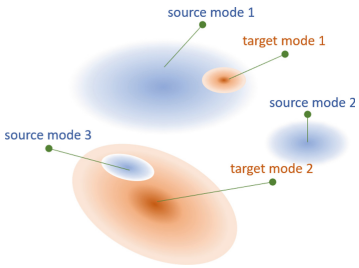


**Fig. 1.** An illustration of the missing mode and boundary distortion problems of DDAN. In the joint space, the target distribution misses source mode 2, while the source distribution can only partly cover the target mode 2 in the target distribution and the target distribution can only partly cover the source mode 1 in the source distribution.

# 3 Dual Generator-Discriminator Deep Code Domain Adaptation

## 3.1 Key Idea of Our Approach

We employ two discriminators (namely, $D_S$ and $D_T$) to classify the source and target examples and vice versa and two separate generators (namely, $G_S$ and $G_T$) to map the source and target examples to the joint space respectively. In particular, $D_S$ produces high values on the source examples in the joint space (i.e., $G_S(x^S)$) and low values on the target examples in the joint space (i.e., $G_T(x^T)$), while $D_T$ produces high values on the target examples in the joint space (i.e., $G_T(x^T)$) and low values on the source examples (i.e., $G_S(x^S)$). The generator $G_S$ is trained to push $G_S\left(x^S\right)$ to the high value region of $D_T$ and the generator $G_T$ is trained to push $G_T(x^T)$ to the high value region of $D_S$. Eventually, both $D_S(G_S(x^S))$ and $D_S(G_T(x^T))$ are possibly high and both $D_T(G_S(x^S))$ and $D_T(G_T(x^T))$ are possibly high. This helps to mitigate the issues of missing mode and boundary distortion since as in Fig. 1, if the target mode 1 can only partly cover the source mode 1, then $D_T$ cannot receive large values from source mode 1. Another important aspect of our approach is to maintain the cluster/manifold structure of source and target data in the joint space via the manifold regularization to avoid the data distortion problem.

## 3.2 Dual Generator-Discriminator Deep Code Domain Adaptation Network

To address the two inherent problems in the DDAN mentioned in Sect. 2.3, we employ two different generators $G_S$ and $G_T$ to map source and target domain examples to the joint space and two discriminators $D_S$ and $D_T$ to distinguish source examples against target examples and vice versa together with the source classifier $C$ which is used to classify the source examples with labels as shown in Fig. 2. We name our proposed model as Dual Generator-Discriminator Deep Code Domain Adaptation Network (Dual-GD-DDAN).

*Updating the Discriminators.* The two discriminators $D_S$ and $D_T$ are trained to distinguish the source examples against the target examples and vice versa as follows:

$$\min_{D_S}\left(\frac{(1+\theta)}{N_S}\sum_{i=1}^{N_S}[-\log D_S(G_S(x_i^S))]+\frac{1}{N_T}\sum_{i=1}^{N_T}[-\log[1-D_S(G_T(x_i^T))]]\right) \tag{1}$$

$$\min_{D_T}\left(\frac{1}{N_S}\sum_{i=1}^{N_S}[-\log[1-D_T(G_S(x_i^S))]]+\frac{(1+\theta)}{N_T}\sum_{i=1}^{N_T}[-\log D_T(G_T(x_i^T))]\right) \tag{2}$$

where $\theta > 0$. Note that a high value of $\theta$ encourages $D_S$ and $D_T$ place higher values on $G_S\left(x^S\right)$ and $G_T\left(x^T\right)$ respectively.

*Updating the Source Classifier.* The source classifier is employed to classify the source examples with labels as: $\min_C \frac{1}{N_S}\sum_{i=1}^{N_S}\ell(C(G_S(x_i^S)),y_i)$, where $\ell$ specifies the cross-entropy loss function for the binary classification (e.g., using cross-entropy).

*Updating the Generators.* The two generators $G_S$ and $G_T$ are trained to i) maintain the manifold/cluster structures of source and target data in their original spaces to avoid the data distortion problem and ii) move the target samples toward the source samples in the joint space and resolve the missing mode and boundary distortion problems in the joint space.

To maintain the manifold/cluster structures of source and target data in their original spaces, we propose minimizing the manifold regularization term as: $\min_G \mathcal{M}(G_S, G_T)$ where $\mathcal{M}(G_S, G_T)$ is formulated as:

$$\mathcal{M}(G_S, G_T) = \sum_{i,j=1}^{N_S} \mu_{ij} ||G_S(x_i^S) - G_S(x_j^S)||^2 + \sum_{i,j=1}^{N_T} \mu_{ij} ||G_T(x_i^T) - G_T(x_j^T)||^2$$

in which the weights are defined as $\mu_{ij} = \exp\{-||h(x_i) - h(x_j)||^2/(2\sigma^2)\}$ with $h(x) = \text{concat}(\overleftarrow{h_L}(x), \overrightarrow{h_L}(x))$ where $\overrightarrow{h_L}(x)$ and $\overleftarrow{h_L}(x)$ are the last hidden states of the bidirectional RNN with input $x$.

To move the target samples toward the source samples and resolve the missing mode and boundary distortion problems in the joint space, we propose minimizing the following objective function: $\min_D \mathcal{K}(G_S, G_T)$ where $\mathcal{K}(G_S, G_T)$ is defined as:

$$\mathcal{K}(G_S, G_T) = \frac{1}{N_S} \sum_{i=1}^{N_S} [-\log D_T(G_S(x_i^S))] + \frac{1}{N_T} \sum_{i=1}^{N_T} [-\log D_S(G_T(x_i^T))] \tag{3}$$

Moreover, the source generator $G_S$ has to work out the representation that is suitable for the source classifier, hence we need to minimize the following objective function:

$$\min_{G_S} \frac{1}{N_S} \sum_{i=1}^{N_S} \ell(C(G_S(x_i^S)), y_i)$$

Finally, to update $G_S$ and $G_T$, we need to minimize the following objective function:

$$\frac{1}{N_S} \sum_{i=1}^{N_S} \ell(C(G_S(x_i^S)), y_i) + \alpha \mathcal{M}(G_S, G_T) + \beta \mathcal{K}(G_S, G_T)$$

where $\alpha, \beta > 0$ are two non-negative parameters.

### 3.3 The Rationale for Our Dual Generator-Discriminator Deep Code Domain Adaptation Network Approach

Below we explain why our proposed Dual-GD-DDAN is able to resolve the two critical problems that occur with the DDAN approach. First, if $x_i^S$ and $x_j^S$ are proximal to each other and are located in the same cluster, then their representations $h(x_i^S)$ and $h(x_j^S)$ are close and hence, the weight $\mu_{ij}$ is large. This implies $G_S(x_i^S)$ and $G_S(x_j^S)$ are encouraged to be close in the joint space because we are minimizing $\mu_{ij} ||G_S(x_i^S) - G_S(x_j^S)||^2$. This increases the chance of the two representations residing in the same cluster in the joint space. Therefore, Dual-GD-DDAN is able to preserve the clustering structure of

the source data in the joint space. By using the same argument, we reach the same conclusion for the target domain.

Second, following Eqs. (1, 2), the discriminator $D_S$ is trained to encourage large values for the source modes (i.e., $G_S(x^S)$), while the discriminator $D_T$ is trained to produce large values for the target modes (i.e., $G_T(x^T)$). Moreover, as in Eq. (3), $G_s$ is trained to move the source domain examples $x^S$ to the high-valued region of $D_T$ (i.e., the target modes or $G_T(x^T)$) and $G_T$ is trained to move the target examples $x^T$ to the high-valued region of $D_S$ (i.e., the source modes or $G_S(x^S)$). As a consequence, eventually, the source modes (i.e., $G_S(x^S)$) and target modes (i.e., $G_T(x^T)$) overlap, while $D_S$ and $D_T$ place large values on both source (i.e., $G_S(x^S)$) and target (i.e., $G_T(x^T)$) modes. The mode missing problem is less likely to happen since, as shown in Fig. 1, if the target data misses source mode 2, then $D_T$ cannot receive large values from source mode 2. Similarly, the boundary distortion problem is also less likely to happen since as in Fig. 1, if the target mode 1 can only partly cover the source mode 1, then $D_T$ cannot receive large values from source mode 1. Therefore, Dual-GD-DDAN allows us to reduce the impact of the missing mode and boundary distortion problems, hence making the target distribution more identical to the source distribution in the joint space.

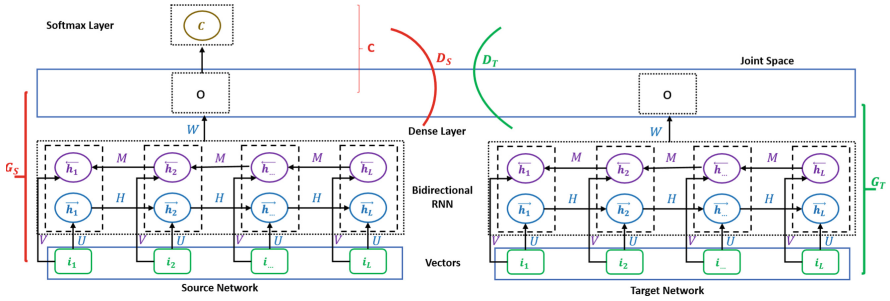


**Fig. 2.** The architecture of our Dual-GD-DDAN. The generators $G_S$ and $G_T$ take the sequential code tokens of the source domain and target domain in vectorial form respectively and map this sequence to the joint layer (i.e., the joint space). The vector representation of each statement $x$ in source code is denoted by $\mathbf{i}$. The discriminators $D_S$ and $D_T$ are invoked to discriminate the source and target data. The source classifier $C$ is trained on the source domain with labels. We note that the source and target networks do not share parameters and are not identical.

### 3.4 Dual Generator-Discriminator Semi-supervised Deep Code Domain Adaptation Network

Our proposed model can be incorporated with minimizing the conditional entropy and using the spectral graph to inspire the smoothness to enforce the clustering assumption [2] proposed in [16] to form Dual Generator-Discriminator Semi-supervised Deep Code Domain Adaptation Network (Dual-GD-SDDAN). Please read our Supplementary Material for more technical details, available at https://app.box.com/s/aijcavbcp.

# 4 Experiments

In this section, firstly, we compare our proposed Dual-GD-DDAN with VulDeePecker without domain adaptation, MMD, D2GAN, DIRT-T and DDAN using the architecture CDAN proposed in [16]. Secondly, we do Boundary Distortion Analysis to further demonstrate the efficiency of our proposed Dual-GD-DDAN in alleviating the boundary distortion problem caused by using the GAN principle. Finally, we compare our Dual-GD-SDDAN and SCDAN introduced in [16].

## 4.1 Experimental Setup

**Experimental Data Set.** We use the real-world data sets collected by [13], which contain the source code of vulnerable and non-vulnerable functions obtained from five real-world software projects, namely FFmpeg (#vul-funcs: 187, #non-vul-funcs: 5,427), LibTIFF (#vul-funcs: 81, #non-vul-funcs: 695), LibPNG (#vul-funcs: 43, #non-vul-funcs: 551), VLC (#vul-funcs: 25, #non-vul-funcs: 5,548) and Pidgin (#vul-funcs: 42, #non-vul-funcs: 8,268) where #vul-funcs and #non-vul-funcs is the number of vulnerable and non-vulnerable functions respectively. The data sets contain both multimedia (FFmpeg, VLC, Pidgin) and image (LibPNG, LibTIFF) application categories. In our experiment, data sets from the multimedia category were used as the source domain whilst data sets from the image category were used as the target domain (see Table 1).

**Model Configuration.** For training the eight methods – VulDeePecker, MMD, D2GAN, DIRT-T, DDAN, Dual-GD-DDAN, SCDAN and Dual-GD-SDDAN – we use one-layer bidirectional recurrent neural networks with LSTM cells where the size of hidden states is in $\{128, 256\}$ for the generators. For the source classifier and discriminators, we use deep feed-forward neural networks with two hidden layers in which the size of each hidden layer is in $\{200, 300\}$. We embed the opcode and statement information in the $\{150, 150\}$ dimensional embedding spaces respectively (see our Supplementary Material for Data Processing and Embedding, available at https://app.box.com/s/aijcavbcp). We employ the Adam optimizer with an initial learning rate in $\{10^{-3}, 10^{-4}\}$. The mini-batch size is 64. The trade-off parameters $\alpha, \beta, \gamma, \lambda$ are in $\{10^{-1}, 10^{-2}, 10^{-3}\}$, $\theta$ is in $\{0, 1\}$ and $1/(2\sigma^2)$ is in $\{2^{-10}, 2^{-9}\}$.

We split the data of the source domain into two random partitions containing 80% for training and 20% for validation. We also split the data of the target domain into two random partitions. The first partition contains 80% for training the models of VulDeePecker, MMD, D2GAN, DIRT-T, DDAN, Dual-GD-DDAN, SCDAN and Dual-GD-SDDAN without using any label information while the second partition contains 20% for testing the models. We additionally apply gradient clipping regularization to prevent over-fitting in the training process of each model. We implement eight mentioned methods in Python using Tensorflow which is an open-source software library for Machine Intelligence developed by the Google Brain Team.

**Table 1.** Performance results in terms of false negative rate (FNR), false positive rate (FPR), Recall, Precision and F1-measure of VulDeePecker (VULD), MMD, D2GAN, DIRT-T, DDAN and Dual-GD-DDAN for predicting vulnerable and non-vulnerable code functions on the testing set of the target domain (Best performance in **bold**).

| Source → Target | Methods | FNR | FPR | Recall | Precision | F1-measure |
|---|---|---|---|---|---|---|
| Pidgin → LibPNG | VULD | 42.86% | 1.08% | 57.14% | 80% | 66.67% |
| | MMD | 37.50% | **0%** | 62.50% | **100%** | 76.92% |
| | D2GAN | **33.33%** | 1.06% | **66.67%** | 80% | 72.73% |
| | DIRT-T | **33.33%** | 1.06% | **66.67%** | 80% | 72.73% |
| | DDAN | 37.50% | **0%** | 62.50% | **100%** | 76.92% |
| | Dual-GD-DDAN | **33.33%** | **0%** | **66.67%** | **100%** | **80%** |
| FFmpeg → LibTIFF | VULD | 43.75% | **6.72%** | 56.25% | 50% | 52.94% |
| | MMD | 28.57% | 12.79% | 71.43% | 47.62% | 57.14% |
| | D2GAN | 30.77% | 6.97% | 69.23% | **64.29%** | 66.67% |
| | DIRT-T | 25% | 9.09% | 75% | 52.94% | 62.07% |
| | DDAN | 35.71% | 6.98% | 64.29% | 60% | 62.07% |
| | Dual-GD-DDAN | **12.5%** | 8.2% | **87.5%** | 56% | **68.29%** |
| FFmpeg → LibPNG | VULD | 25% | **2.17%** | 75% | 75% | 75% |
| | MMD | 12.5% | 3.26% | 87.5% | 70% | 77.78% |
| | D2GAN | 14.29% | **2.17%** | 85.71% | 75% | 80% |
| | DIRT-T | 15.11% | 2.2% | 84.89% | **80%** | 84.21% |
| | DDAN | **0%** | 3.26% | **100%** | 72.73% | 84.21% |
| | Dual-GD-DDAN | **0%** | **2.17%** | **100%** | **80%** | **88.89%** |
| VLC→ LibPNG | VULD | 57.14% | **1.08%** | 42.86% | 75% | 54.55% |
| | MMD | 45% | 4.35% | 55% | 60% | 66.67% |
| | D2GAN | 28.57% | 4.3% | 71.43% | 55.56% | 62.5% |
| | DIRT-T | 50% | 1.09% | 50% | **80%** | 61.54% |
| | DDAN | 33.33% | 2.20% | 66.67% | 75% | 70.59% |
| | Dual-GD-DDAN | **28.57%** | 2.15% | **71.43%** | 71.43% | **71.43%** |
| Pidgin → LibTIFF | VULD | 35.29% | 8.27% | 64.71% | 50% | 56.41% |
| | MMD | 30.18% | 12.35% | 69.82% | 50% | 58.27% |
| | D2GAN | 40% | 7.95% | 60% | **60%** | 60% |
| | DIRT-T | 38.46% | 8.05% | 61.54% | 53.33% | 57.14% |
| | DDAN | **27.27%** | 8.99% | **72.73%** | 50% | 59.26% |
| | Dual-GD-DDAN | 29.41% | **6.76%** | 70.59% | 57.14% | **63.16%** |

### 4.2 Experimental Results

**Code Domain Adaptation for a Fully Non-labeled Target Project.** We investigate the performance of our proposed Dual-GD-DDAN compared with other methods including VulDeePecker (VULD) without domain adaptation [12], DDAN [16], MMD [14], D2GAN [15] and DIRT-T [18] with VAP applied in the joint feature layer using the architecture CDAN introduced in [16]. The VulDeePecker method is only trained on the source data and then tested on the target data, while the MMD, D2GAN, DIRT-T, DDAN and Dual-GD-DDAN methods employ the target data without using any label information for domain adaptation.

In Table 1, the experimental results *show that our proposed Dual-GD-DDAN achieves a higher performance for detecting vulnerable and non-vulnerable functions for most performance measures*, including FNR, FPR, Recall, Precision and F1-measure in almost cases of the source and target domains, especially for F1-measure. Particularly, our Dual-GD-DDAN always obtains the highest F1-measure in all cases. For example, for the case of the source domain (FFmpeg) and target domain (LibPNG), Dual-GD-DDAN achieves an F1-measure of 88.89% compared with an F1-measure of 84.21%, 84.21%, 80%, 77.78% and 75% obtained with DDAN, DIRT-T, D2GAN, MMD and VulDeePecker respectively.

**Boundary Distortion Analysis**

*Quantitative Results.* To quantitatively demonstrate the efficiency of our proposed Dual-GD-DDAN in alleviating the boundary distortion problem caused by using the GAN principle, we reuse the experimental setting in Sect. 5.2 [17]. The basic idea is, given two data sets $S_1$ and $S_2$, to quantify the degree of cover of these two data sets. We train a classifier $\mathcal{C}_1$ on $S_1$, then test on $S_2$ and another classifier $\mathcal{C}_2$ on $S_2$, then test on $S_1$. If these two data sets cover each other well with reduced boundary distortion, we expect that if $\mathcal{C}_1$ predicts well on $S_1$, then it should predict well on $S_2$ and vice versa if $\mathcal{C}_2$ predicts well on $S_2$, then it should predict well on $S_1$. This would seem reasonable since if boundary distortion occurs (i.e., assume that $S_2$ partly covers $S_1$), then $\mathcal{C}_2$ trained on $S_2$ would struggle to predict $S_1$ well which is much larger and possibly more complex. Therefore, we can utilize the magnitude of the accuracies and the accuracy gap of $\mathcal{C}_1$ and $\mathcal{C}_2$ when predicting their training and testing sets to assess the severity of the boundary distortion problem.

**Table 2.** Accuracies obtained by the DDAN and Dual-GD-DDAN methods when predicting vulnerable and non-vulnerable code functions on the source and target domains. Note that tr src, ts tar, tr tar, ts src, and acc gap are the shorthands of train source, test target, train target, test source, and accuracy gap respectively. For the accuracy gap, a smaller value is better.

| Source → Target | Methods | Accuracy | | | Accuracy | | |
|---|---|---|---|---|---|---|---|
| | | Tr src/Ts tar/acc gap | | | Tr tar/Ts src/ acc gap | | |
| Pidgin → LibPNG | DDAN | 98.8% | 96% | 2.8% | 97% | 92% | 5% |
| | Dual-GD-DDAN | 99% | 97% | **2%** | 97% | 95% | **2%** |
| FFmpeg → LibPNG | Methods | Accuracy | | | Accuracy | | |
| | | Tr src/Ts tar/acc gap | | | Tr tar/Te src/acc gap | | |
| | DDAN | 95.9% | 92% | 3.9% | 91% | 83.3% | 7.7% |
| | Dual-GD-DDAN | 97% | 96% | **1%** | 98% | 95.6% | **2.4%** |

Inspired by this observation, we compare our Dual-GD-DDAN with DDAN using the representations of the source and target samples in the joint feature space corresponding to their best models. In particular, for a given pair of source and target data sets and for comparing each method, we train a neural network classifier on the best

representations of the source data set in the joint space, then predict on the source and target data set and do the same but swap the role of the source and target data sets. We then measure the difference of the corresponding accuracies as a means of measuring the severity of the boundary distortion. We choose to conduct such a boundary distortion analysis for two pairs of the source (FFmpeg and Pidgin) and target (LibPNG) domains. As shown in Table 2, *all gaps obtained by our Dual-GD-DDAN are always smaller than those obtained by DDAN*, while the accuracies obtained by our proposed method are always larger. We can therefore conclude that our Dual-GD-DDAN method produces a better representation for source and target samples in the joint space and is less susceptible to boundary distortion compared with the DDAN method.

*Visualization.* We further demonstrate the efficiency of our proposed Dual-GD-DDAN in alleviating the boundary distortion problem caused by using the GAN principle. Using a t-SNE [9] projection, with perplexity equal to 30, we visualize the feature distributions of the source and target domains in the joint space. Specifically, we project the source and target data in the joint space (i.e., $G(x)$) into a 2D space with domain adaptation (DDAN) and with dual-domain adaptation (Dual-GD-DDAN). In Fig. 3, we observe these cases when performing domain adaptation from a software project (FFmpeg) to another (LibPNG). As shown in Fig. 3, with undertaking domain adaptation (DDAN, the left figure) and dual-domain adaptation (Dual-GD-DDAN, the right figure), *the source and target data sampled are intermingled especially for Dual-GD-DDAN*. However, it can be observed that DDAN when solely applying the GAN is seriously vulnerable to the boundary distortion issue. In particular, in the clusters/data modes 2, 3 and 4 (the left figure), the boundary distortion issue occurs since the blue data only partly cover the corresponding red ones (i.e., the source and target data do not totally mix up). Meanwhile, for our Dual-GD-DDAN, *the boundary distortion issue is much less vulnerable*, and the mixing-up level of source and target data is significantly higher in each cluster/data mode.

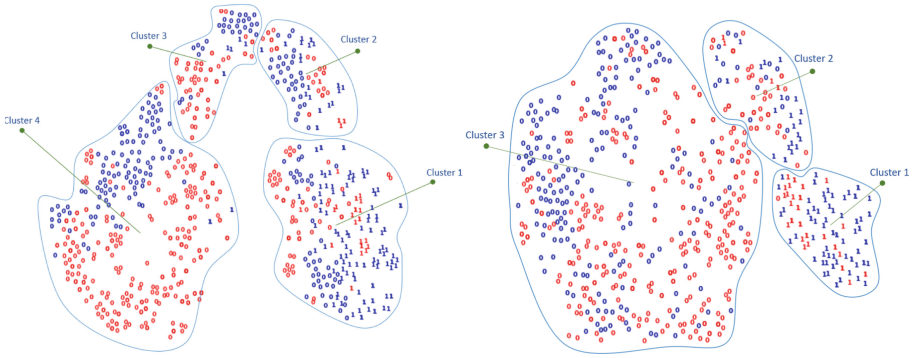


**Fig. 3.** A 2D t-SNE projection for the case of the FFmpeg → LibPNG domain adaptation. The blue and red points represent the source and target domains in the joint space respectively. In both cases of the source and target domains, data points labeled 0 stand for non-vulnerable samples and data points labeled 1 stand for vulnerable samples. (Color figure online)

**Quantitative Results of Dual Generator-Discriminator Semi-supervised Deep Code Domain Adaptation.** In this section, we compare the performance of our Dual-GD-SDDAN with Semi-supervised Deep Code Domain Adaptation (SCDAN) [16] on four pairs of the source and target domains. In Table 3, *the experimental results show that our Dual-GD-SDDAN achieves a higher performance than SCDAN for detecting vulnerable and non-vulnerable functions* in terms of FPR, Precision and F1-measure in almost cases of the source and target domains, especially for F1-measure. For example, to the case of the source domain (VLC) and target domain (LibPNG), our Dual-GD-SDDAN achieves an F1-measure of 76.19% compared with an F1-measure of 72.73% obtained with SCDAN. These results further demonstrate the ability of our Dual-GD-SDDAN for dealing with the mode collapsing problem better than SCDAN [16], hence obtaining better predictive performance in the context of software domain adaptation.

**Table 3.** Performance results in terms of false negative rate (FNR), false positive rate (FPR), Recall, Precision and F1-measure of SCDAN and Dual-GD-SDDAN for predicting vulnerable/non-vulnerable code functions on the testing set of the target domain (Best performance in **bold**).

| Source → Target | Methods | FPR | FNR | Recall | Precision | F1-measure |
|---|---|---|---|---|---|---|
| FFmpeg → LibTIFF | SCDAN | 5.38% | **14.29%** | **85.71%** | 57.14% | 68.57% |
| | Dual-GD-SDDAN | **3.01%** | 35.29% | 64.71% | **73.33%** | **68.75%** |
| FFmpeg → LibPNG | SCDAN | 1.08% | **12.5%** | **87.5%** | 87.5% | 87.5% |
| | Dual-GD-SDDAN | **0%** | 17.5% | 82.5% | **100%** | **90.41%** |
| VLC→ LibPNG | SCDAN | **1.06%** | 33.33% | 66.67% | **80%** | 72.73% |
| | Dual-GD-SDDAN | 4.39% | **11.11%** | **88.89%** | 66.67% | **76.19%** |
| Pidgin → LibTIFF | SCDAN | 5.56% | **30%** | **70%** | 58.33% | 63.64% |
| | Dual-GD-SDDAN | **2.98%** | 37.5% | 62.5% | **71.43%** | **66.67%** |

## 5 Conclusion

Software vulnerability detection (SVD) is an important problem in the software industry and in the field of computer security. One of the most crucial issues in SVD is to cope with the scarcity of labeled vulnerabilities in projects that require the laborious labeling of code by software security experts. In this paper, we propose the Dual Generator-Discriminator Deep Code Domain Adaptation Network (Dual-GD-DDAN) method to deal with the missing mode and boundary distortion problems which arise from the use of the GAN principle when reducing the discrepancy between source and target data in the joint space. We conducted experiments to compare our Dual-GD-DDAN method with the state-of-the-art baselines. The experimental results show that our proposed method outperforms these rival baselines by a wide margin in term of predictive performances.

**Acknowledgement.** This research was supported under the Defence Science and Technology Group's Next Generation Technologies Program.

## References


1. Almorsy, M., Grundy, J., Ibrahim, A.: Supporting automated vulnerability analysis using formalized vulnerability signatures. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, ASE, pp. 100–109. ACM (2012)
2. Chapelle, O., Zien, A.: Semi-supervised classification by low density separation. In: AISTATS, pp. 57–64. Citeseer (2005)
3. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: Proceedings of the 32nd International Conference on Machine Learning, vol. 37, pp. 1180–1189. ICML (2015)
4. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
5. Goodfellow, I.: Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160 (2016)
6. Gopalan, R., Ruonan, L., Chellappa, R.: Domain adaptation for object recognition: an unsupervised approach. In: Proceedings of the 2011 International Conference on Computer Vision, pp. 999–1006. ICCV (2011)
7. Hoang, Q., Nguyen, T.D., Le, T., Phung, D.: MGAN: training generative adversarial nets with multiple generators. In: International Conference on Learning Representation (2018)
8. Kim, S., Woo, S., Lee, H., Oh, H.: VUDDY: a scalable approach for vulnerable code clone discovery. In: IEEE Symposium on Security and Privacy, pp. 595–614. IEEE Computer Society (2017)
9. Laurens, V.M., Geoffrey, H.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
10. Le, T., Hoang, Q., Vu, H., Nguyen, T.D., Bui, H., Phung, D.: Learning generative adversarial networks from multiple data sources. In: 2019 International Joint Conference on Artificial Intelligence (2019)
11. Le, T., et al.: Maximal divergence sequential autoencoder for binary software vulnerability detection. In: International Conference on Learning Representations (2019)
12. Li, Z., et al.: VulDeePecker: a deep learning-based system for vulnerability detection. CoRR abs/1801.01681 (2018)
13. Lin, G., et al.: Cross-project transfer representation learning for vulnerable function discovery. IEEE Trans. Industr. Inform. **14**, 3289–3297 (2018)
14. Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, Lille, France, vol. 37, pp. 97–105 (2015)
15. Nguyen, T.D., Le, T., Vu, H., Phung, D.: Dual discriminator generative adversarial nets. In: Advances in Neural Information Processing (2017)
16. Nguyen, V., et al.: Deep domain adaptation for vulnerable code function identification. In: The International Joint Conference on Neural Networks (IJCNN) (2019)
17. Santurkar, S., Schmidt, L., Madry, A.: A classification-based study of covariate shift in GAN distributions. In: Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4480–4489. PMLR (2018)
18. Shu, R., Bui, H., Narui, H., Ermon, S.: A DIRT-t approach to unsupervised domain adaptation. In: International Conference on Learning Representations (2018)