*applied sciences*

MDPI

*Article*

# Progressive Prototype Alignment with Entropy Regularization for Cross-Project Software Vulnerability Detection

**Yuze Ding, Jinheng Zhang, Yimang Li * and Guozhen Li**

School of Mechanical Engineering and Rail Transit, Changzhou University, Changzhou 213164, China; 2400890234@smail.cczu.edu.cn (J.Z.)
* Correspondence: liyimang@cczu.edu.cn; Tel.: +86-135-8452-7366

**Abstract**

Cross-project software vulnerability detection must cope with pronounced domain shift and severe class imbalance, while the target project is typically unlabeled. Existing unsupervised domain adaptation techniques either focus on marginal alignment and overlook class-conditional mismatch, or depend on noisy pseudolabels, which can induce negative transfer in imbalanced settings. To address these challenges we propose DAP2ER, a progressive domain adaptation framework that couples adversarial domain confusion with entropy regularization and prototype-guided high-confidence pseudolabel optimization. Specifically, DAP2ER constructs source class prototypes, selects reliable target samples via confidence-aware pseudolabeling, and performs class-conditional alignment by pulling target features toward the corresponding prototypes. A progressive weighting schedule gradually increases the strength of domain and self-training objectives, stabilizing optimization in early epochs. Experiments on two real-world vulnerability datasets demonstrate that DAP2ER consistently outperforms strong baselines, improving the F1-score by up to 21 percentage points and achieving substantial gains in AUC for bidirectional transfer.

**Keywords:** software vulnerability detection; unsupervised domain adaptation; class imbalance; pseudolabeling; prototype alignment; progressive training

## 1. Introduction

Software vulnerabilities are flaws or oversights in software implementation that can be exploited by attackers, leading to severe security consequences such as information disclosure, denial of service, and privilege escalation attacks. As the scale and complexity of software systems continue to grow, efficiently and accurately identifying vulnerabilities during the development and maintenance stages has become a significant research challenge in software engineering and cybersecurity. In recent years, automated vulnerability detection methods based on deep representation learning [1,2] have demonstrated the ability to learn feature representations from source code, thereby reducing reliance on manual feature engineering and expert knowledge. These methods have shown great potential in large-scale detection scenarios. Concurrently, real-world vulnerability datasets [3,4] are expanding, with mechanisms such as reproducible vulnerability knowledge bases and automatic update mechanisms promoting more systematic evaluation and research.

However, despite the advancements made by deep learning and pretrained models, two prominent challenges persist in real engineering environments, primarily due to the nature of the task and the data. First, vulnerability detection often requires distinguishing



Academic Editor: George Drosatos

Received: 19 January 2026
Revised: 29 January 2026
Accepted: 2 February 2026
Published: 4 February 2026

between multiple vulnerability types or weakness categories, which is inherently a multi-class task. For instance, in edit-time or function-level detection scenarios [5,6], models must identify various vulnerability patterns while maintaining generalization capabilities. This need for multi-class classification adds complexity, making it difficult for existing methods to maintain stable performance across various vulnerability types. Second, in cross-project scenarios, codebases exhibit significant differences in coding style, API usage patterns, dependency environments, and vulnerability distributions. These discrepancies lead to performance degradation when the same model is transferred across projects. Even with the introduction of advanced structured representations such as Code Property Graphs and large model reasoning capabilities [7,8], cross-codebase robustness remains vulnerable to minor code modifications, data shifts, and changes in evaluation distributions. Additionally, issues such as class imbalance and label noise in training data [9] exacerbate these challenges, making it harder for models to generalize across domains and leading to the learning of spurious correlation patterns unrelated to vulnerabilities that can impair real-world performance.

From a data perspective, high-quality vulnerability annotation is resource-intensive. Vulnerability localization, reproduction, and validation typically require significant time investments from security experts. This has led to the long-standing problem of having a few labeled projects and many unlabeled projects. While recent research has explored using large models to enhance vulnerability detection [10], challenges such as insufficient annotations in target projects, cross-project distribution shifts, and difficulties in learning from imbalanced samples remain. Consequently, achieving stable cross-project transfer and multi-class vulnerability identification while using only source project annotations and no target project annotations remains a crucial and unsolved problem.

Unsupervised domain adaptation provides a natural solution to this problem by leveraging knowledge from the source domain to reduce performance degradation on the target domain. Domain adversarial learning [11] is one of the representative approaches. Through adversarial training with a gradient reversal layer and a domain discriminator, this method encourages the feature extractor to learn more domain-invariant representations as a way to mitigate the impact of source/target distribution differences. However, in tasks such as vulnerability detection that are characterized by high noise, strong imbalance, and complex class semantics, relying solely on global alignment is often insufficient. On one hand, multi-class tasks necessitate finer-grained class-level alignment; on the other hand, class-level alignment often depends on target domain pseudolabels. Pseudolabel selection methods such as [12] can introduce significant noise if a fixed confidence threshold is used. This results in misalignment and erroneous reinforcement, particularly during the early stages of training or when class boundaries are unclear. Furthermore, the entropy minimization approach [13–15] is often employed to encourage the model to produce more certain predictions on the target domain, thereby improving self-training effectiveness; however, its success heavily depends on training stability and noise control strategies. Recently, prototype alignment [16–18] has been proposed as an effective method to address class-level mismatches. By constructing source domain class prototypes and aligning target domain samples of the same class towards the corresponding prototypes, this approach improves class consistency and alignment robustness to some extent.

Building on these observations, this paper presents a progressive domain adaptation framework designed for multi-class and cross-project software vulnerability detection. Operating under a unified network structure consisting of a feature extractor, label classifier, and domain discriminator, the proposed framework employs a progressive loss weight scheduling strategy. This strategy first solidifies the discriminative capabilities of the source domain, then strengthens cross-domain generalization. Additionally, the proposed frame-

work incorporates prototype-guided high-confidence pseudolabel optimization, leveraging source domain annotations to construct class prototypes and selecting high-confidence samples in the target domain for class-level prototype alignment. This process reduces the interference of pseudolabel noise. To further enhance the decision boundaries on the target domain, entropy minimization is combined to improve multi-class vulnerability identification performance under cross-project transfer.

The main contributions of this paper are as follows:

1. A progressive domain adaptation framework for cross-project multi-class vulnerability detection is proposed, alleviating the optimization conflict between classification learning and alignment learning through progressive scheduling of multiple adaptive losses.
2. A prototype-guided high-confidence pseudo-label optimization strategy is introduced, achieving robust class-level alignment by relying solely on unlabeled target domain data.
3. Systematic experiments and ablation analysis are conducted in the context of cross-project vulnerability detection, demonstrating the effectiveness and robustness of the proposed framework in imbalanced and distribution-shift scenarios.

## 2. Related Work

Unsupervised domain adaptation aims to transfer knowledge learned from a labeled source domain to an unlabeled target domain, thereby mitigating performance degradation on the target domain. Extensive research has been conducted in this area [19–30]. Domain adversarial training, a classic UDA method, learns domain-invariant features through an adversarial objective to align the feature distributions between the source and target domains. The pioneering work in this area, DANN [19], utilizes a gradient reversal layer to force the feature extractor to generate representations that confuse the domain discriminator while maintaining the discriminative features required by the task classifier. Recent studies [20,21] have advanced adversarial domain adaptation by incorporating self-supervised learning, Generative Adversarial Networks (GANs), and other techniques to improve performance under complex domain shifts.

Recently, progressive training and pseudolabel selection have become widely adopted techniques in UDA. Although pseudo-labeling plays a crucial role in domain adaptation, the quality of pseudolabels is often compromised by noise. To address this issue, high-confidence pseudolabeling has emerged as a research focus. Studies [22,23] have improved pseudolabel quality by introducing confidence thresholds and class prototype alignment, which reduces the impact of noisy pseudolabels and enhances model performance and stability. In [24], the authors addressed domain differences by using a statistically-driven feature augmentation strategy. SAFA-DA dynamically selects high-prediction and high-confidence samples from both the source and target domains, progressively building intermediate representations. This framework uses statistical insights to align the source and target domains through mean-based feature comparison and covariance-based feature augmentation. SAFA-DA also employs an adaptive control mechanism based on the maximum mean discrepancy metric to regulate augmentation intensity, ensuring stable convergence and preventing overfitting. Extensive experiments on the Case Western Reserve University and Paderborn University datasets have shown that SAFA-DA significantly outperforms existing methods, improving average accuracy by 18.11%. Furthermore, SAFA-DA demonstrates strong robustness under industrial conditions, maintaining high accuracy despite severe noise and class imbalance, making it highly valuable for industrial informatics applications. MLSL [25] further improves the reliability of target

domain data through three progressively fine-tuned feature views combined with selective pseudolabel screening.

In domain adaptation research, class prototype alignment has been proposed to better align class boundaries between source and target domains. In [26,27], the authors were able to improve inter-class discriminability by calculating representative feature centers for each class in the source domain and aligning target domain samples to these prototypes. This method reduces class confusion and improves model performance in the target domain. SPADA [28] combines prototype alignment with an attention mechanism, ensuring a more consistent feature space between source and target domains through active adaptation and pseudolabel optimization. MLRGL [29] enhances prototype alignment by constructing high-order graph structures and multi-view features, leveraging contextual information within the target domain to improve alignment accuracy.

Multi-view learning and subspace alignment are also important directions in domain adaptation. CAMSA [30] proposes a multi-view enhanced alignment method that improves target domain classification performance through multi-view masking and subspace alignment strategies. This approach improves model generalization by integrating information from different perspectives and using low-rank constraints for feature alignment.

In this context, the DAP2ER method proposed in this paper combines domain adversarial training, entropy minimization, and prototype alignment techniques. It first aligns features between the source and target domains through domain adversarial training to learn domain-invariant features. Unlike traditional adversarial training methods, DAP2ER introduces a progressive training strategy that gradually increases the roles of adversarial alignment, entropy minimization, and prototype alignment. This ensures that the model adapts progressively to the target domain's feature distribution. To enhance robustness, DAP2ER introduces a high-confidence pseudolabel optimization strategy, mitigating the impact of pseudolabel noise by selectively introducing high-confidence pseudolabels. Additionally, prototype alignment ensures consistent class representations between the source and target domains, further improving classification performance on the target domain.

## 3. Methodology

### 3.1. Problem Formulation

Let $\mathcal{D}_s = \{(x_s^{(j)}, y_s^{(j)})\}_{j=1}^{m}$ denote a labeled source-domain dataset, where $x_s^{(j)} \in \mathcal{X} \subseteq \mathbb{R}^d$ is a $d$-dimensional feature vector and $y_s^{(j)} \in \mathcal{Y}$ is its class label. Let $\mathcal{D}_t = \{x_t^{(j)}\}_{j=1}^{n}$ denote an unlabeled target-domain dataset drawn from a different distribution. We consider a $C$-class classification problem with label space $\mathcal{Y} = \{1, 2, \ldots, C\}$. In unsupervised domain adaptation, the objective is to learn a predictor $\xi : \mathcal{X} \to \mathcal{Y}$ from $\mathcal{D}_s$ and $\mathcal{D}_t$ that achieves strong generalization on the target domain despite the absence of target labels during training. This setting is particularly relevant to cross-project software vulnerability detection, where distribution shifts arise from project-specific coding conventions, divergent functional implementations, and heterogeneous vulnerability patterns. Consequently, an effective UDA model must explicitly mitigate the source–target discrepancy to transfer discriminative knowledge learned from $\mathcal{D}_s$ to $\mathcal{D}_t$.

### 3.2. Overview of the DAP2ER Framework

The proposed approach to cross-domain software vulnerability detection addresses distribution shift between a labeled source project and an unlabeled target project using the Domain Adaptation with Progressive Prototype and Entropy Regularization (DAP2ER) framework. DAP2ER trains a deep model that combines discriminative representation learning on the source domain with unsupervised domain adaptation to improve classification on the target domain. As illustrated in Figure 1, the framework has three stages: (1) network con-

struction and feature extraction; (2) progressive domain alignment via adversarial learning; and (3) joint objective with scheduled loss weights and prototype-guided high-confidence pseudolabel optimization.

Stage 1: Network construction and feature extraction. The network contains three components: a feature extractor $G(\cdot)$, a label classifier $C(\cdot)$, and a domain discriminator $D(\cdot)$. Given an input feature vector $x$ from either domain, $G$ maps it to a 128-dimensional embedding $f = G(x) \in \mathbb{R}^{128}$. Concretely, $G$ consists of two fully connected layers, each followed by batch normalization and a ReLU nonlinearity. The classifier $C$ takes $f$ as input and outputs logits over $K$ vulnerability categories, while a softmax layer converts logits into class probabilities. These probabilities supervise learning on labeled source samples and are used to form pseudolabels for target samples.

Stage 2: Progressive adversarial training and domain adaptation. To encourage domain-invariant representations, we insert a gradient reversal layer between $G$ and the domain discriminator $D$. The discriminator receives features from both domains and predicts their domain labels. During backpropagation, the GRL multiplies the gradient by $-\alpha(p)$, where $\alpha(p) \in [0, 1]$ is increased progressively with training progress $p$. This drives $G$ to generate features that confuse $D$, thereby reducing cross-domain discrepancy. We further adopt a progressive optimization schedule in which early training emphasizes supervised learning on the source domain while domain-adversarial alignment, target entropy minimization, and prototype-based class-conditional alignment are strengthened gradually as training proceeds. This curriculum follows the principle of stabilizing discrimination first, then enhancing transferability.

Stage 3: Loss design and progressive weighting. Training is guided by a joint objective that combines the source classification loss $\mathcal{L}_{\text{cls}}$, the domain-adversarial loss $\mathcal{L}_{\text{adv}}$, the target entropy regularizer $\mathcal{L}_{\text{ent}}$, and the prototype-alignment loss $\mathcal{L}_{\text{proto}}$. The overall objective minimizes a weighted sum of these terms, where the weights $\lambda_{\text{cls}}(p)$, $\lambda_{\text{adv}}(p)$, $\lambda_{\text{ent}}(p)$, and $\lambda_{\text{proto}}(p)$ are scheduled as functions of training progress $p$ to control the tradeoff between source discrimination and target adaptation. Model parameters are optimized with Adam. The symbols are described in Table 1.

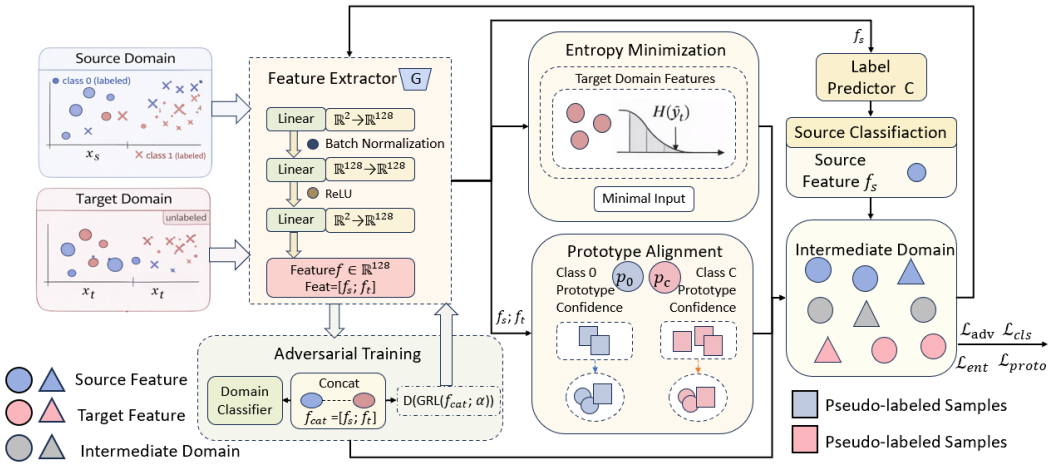



**Figure 1.** Overall architecture of the proposed DAP2ER framework for cross-domain software vulnerability detection.

**Table 1.** Symbols and descriptions used in the proposed method.

| Symbol | Description |
|---|---|
| $\mathcal{D}_s$ | Labeled source-domain dataset |
| $\mathcal{D}_t$ | Unlabeled target-domain dataset (unlabeled during training) |
| $K$ | Number of vulnerability categories (classes) |
| $G(\cdot)$ | Feature extractor (maps inputs to embeddings) |
| $C(\cdot)$ | Label classifier (predicts class logits/probabilities) |
| $D(\cdot)$ | Domain discriminator (predicts domain label) |
| $x_s^{(i)}$ | Source-domain input feature vector (sample index $i$) |
| $y_s^{(i)}$ | Source-domain class label for $x_s^{(i)}$ |
| $f_s, f_t$ | Embedding vectors of source/target samples, $f = G(x)$ |
| $\mu_s^c, \mu_t^c$ | Source/target prototype (class-mean embedding) for class $c$ |
| $N_s^c$ | Number of source samples in class $c$ |
| $\hat{y}_t$ | Pseudo-label of a target sample |
| $\hat{p}_t$ | Maximum predicted probability of a target sample |
| $\mathrm{conf}_t$ | Confidence score of a target sample |
| $\nabla$ | Confidence threshold, $\nabla \in [0, 1]$ |
| $p$ | Normalized training progress in $[0, 1]$ |
| $\alpha(p)$ | GRL reversal strength schedule |
| $\lambda_{\mathrm{cls}}(p)$ | Weight for $\mathcal{L} * \mathrm{cls}$ |
| $\lambda_{\mathrm{adv}}(p)$ | Weight for $\mathcal{L} * \mathrm{adv}$ |
| $\lambda_{\mathrm{ent}}(p)$ | Weight for $\mathcal{L} * \mathrm{ent}$ |
| $\lambda_{\mathrm{proto}}(p)$ | Weight for $\mathcal{L} * \mathrm{proto}$ |
| $\mathcal{L}_{\mathrm{cls}}$ | Source-domain classification loss |
| $\mathcal{L}_{\mathrm{adv}}$ | Domain-adversarial loss |
| $\mathcal{L}_{\mathrm{ent}}$ | Target entropy minimization loss |
| $\mathcal{L}_{\mathrm{proto}}$ | Prototype alignment loss |

### *3.3. Prototype-Guided High-Confidence Pseudolabel Optimization*

In unsupervised domain adaptation, target-domain samples are unlabeled; consequently, naive class-conditional alignment can be easily corrupted by noisy pseudolabels. To improve robustness, we propose a prototype-guided high-confidence pseudolabel optimization strategy. The core idea is to (i) compute class prototypes from labeled source data, (ii) assign pseudolabels to target samples while retaining only high-confidence predictions, and (iii) align the per-class mean features of the selected target samples to the corresponding source prototypes, enabling stable class-level cross-domain alignment.

To obtain reliable class prototypes, we set the feature extractor $G(\cdot)$ to evaluation mode at the beginning of each epoch to disable stochastic layers. We then iterate over the labeled source dataset $\mathcal{D}_s = \{(x_s^{(i)}, y_s^{(i)})\}_{i=1}^{n_s}$ and extract embeddings $f_s^{(i)} = G(x_s^{(i)})$. For each class $c \in \{1, \ldots, K\}$, we accumulate the feature sum and the sample count using the indicator $\mathbb{I}(y_s^{(i)} = c)$, yielding the source prototype

$$\mu_s^c = \frac{1}{N_s^c} \sum_{i=1}^{n_s} \mathbb{I}\left(y_s^{(i)} = c\right), \tag{1}$$

where $N_s^c = \sum_{i=1}^{n_s} \mathbb{I}(y_s^{(i)} = c)$ is the number of source samples in class $c$. After prototype computation, we switch the network back to training mode for subsequent optimization.

Given an unlabeled target sample $x_t \in \mathcal{D}_t$, we compute its embedding $f_t = G(x_t)$ and obtain logits $z_t = C(f_t)$. The predicted class probabilities are

$$p_t = \mathrm{softmax}(z_t), \tag{2}$$

and the pseudolabel is assigned by the maximum-probability rule:

$$\hat{y}_t = \arg\max_k p_t^{(k)}. \tag{3}$$

To suppress unreliable pseudolabels, we score prediction confidence using the maximum probability $\hat{p}_t = \max_k p_t^{(k)}$ and define

$$\mathrm{conf}_t = 2|\hat{p}_t - 0.5| \in [0, 1]. \tag{4}$$

Given a threshold $\nabla$, we select high-confidence target samples via the mask. We adopt a relatively high confidence threshold to prioritize pseudolabel precision, as target-domain pseudolabels are inherently noisy in unsupervised adaptation; such noise can be amplified by self-training and class-conditional objectives, leading to misalignment and negative transfer. With a fixed $\nabla$, the number of selected target samples still increases naturally as training progresses and the model becomes more confident on the target domain, which is consistent with our progressive weighting strategy: early epochs rely on a small set of highly reliable pseudolabels, while later epochs leverage more target samples without substantially sacrificing pseudolabel quality.

$$m_t = \mathbb{I}(\mathrm{conf}_t \geq \nabla) \tag{5}$$

For each class $c$, we further form the class-specific selected set:

$$T_c = \{x_t \in \mathcal{B}_t \mid m(x_t) = 1 \wedge \hat{y}(x_t) = c\}. \tag{6}$$

For each class with non-empty $T_c$, we compute the target class mean feature:

$$\mu_t^c = \frac{1}{|T_c|} \sum_{x_t \in T_c} G(x_t) \tag{7}$$

and align it with the corresponding source prototype $\mu_s^c$. Let $C_{\mathrm{val}} = \{c \mid |T_c| > 0 \wedge N_s^c > 0\}$ denote the valid classes. When $C_{\mathrm{val}} = \emptyset$, no class has any selected target sample; we set $\mathcal{L}_{\mathrm{proto}} = 0$ for that update to avoid undefined division.

$$\mathcal{L}_{\mathrm{proto}} = \frac{1}{\max(1, |C_{\mathrm{val}}|)} \sum_{c \in C_{\mathrm{val}}} |\mu_t^c - \mu_s^c|_2^2 \tag{8}$$

We introduce a scheduling weight $\lambda_{\mathrm{proto}}(p)$ to control the contribution of $\mathcal{L}_{\mathrm{proto}}$ along training progress $p \in [0, 1]$. Here, $p$ is the normalized progress, i.e., $p = e/E$ at epoch $e$ out of $E$. Specifically, we adopt the standard logistic ramp-up schedule:

$$\lambda_{\mathrm{proto}}(p) = \frac{2}{1 + \exp(-10p)} - 1, \qquad p \in [0, 1], \tag{9}$$

which monotonically increases from 0 to 1 as training proceeds. This design emphasizes stable source-domain discrimination at early epochs and progressively strengthens class-conditional alignment once pseudolabels become more reliable.

We choose the logistic curve as the progressive weighting function because it provides a smooth, monotonic, and bounded ramp-up that naturally matches the training dynamics of UDA. Compared with step or linear schedules, the logistic function increases slowly at the beginning, preventing adaptation losses from dominating when the classifier and pseudolabels are still unreliable. It then grows faster in the middle stage as predictions become more stable and finally saturates near the end, avoiding overly aggressive late-stage

updates that can cause oscillations. Moreover, its single slope parameter offers a simple and consistent way to control the transition speed across different objectives, yielding a unified curriculum that follows the principle of discrimination first and alignment later.

Most deep domain adaptation approaches jointly optimize source classification and domain alignment throughout training. In practice, however, aggressive alignment early in training can distort class-discriminative structure. To address this issue, we use progressive weight scheduling, where each loss weight is an explicit function of normalized training progress $p \in [0,1]$. In the same spirit, the GRL reversal strength $\alpha(p)$ is increased progressively. This design prioritizes supervised learning on the labeled source domain at early training and gradually strengthens adversarial alignment, target entropy minimization, and prototype-based class-conditional alignment.

Given a labeled source sample $x_s$ with label $y_s \in \{1, \ldots, C\}$, the classifier $C(\cdot)$ produces class logits $z_s = C(G(x_s))$ and probabilities $p_s = \mathrm{softmax}(z_s)$. We minimize the standard cross-entropy loss:

$$\mathcal{L}_{\mathrm{cls}} = -\sum_{k=1}^{K} \mathbf{y}_s^{(k)} \log p_s^{(k)}, \tag{10}$$

where $y_s^{(i)} \in \{0,1\}$ denotes the one-hot encoding of $y_s$. We modulate $\mathcal{L}_{\mathrm{cls}}$ by a decreasing coefficient $\lambda_{\mathrm{cls}}(p)$ to emphasize discriminative learning at the beginning of training:

$$\lambda_{\mathrm{cls}}(p) = 1 - \left( \frac{2}{1 + \exp(-10p)} - 1 \right), \qquad p \in [0,1] \tag{11}$$

where $p$ denotes the current epoch index or normalized progress and max_epoch is the total number of epochs. Consequently, $\lambda_{\mathrm{cls}}(p)$ starts near 1 and gradually decreases, preventing premature alignment from harming class separation.

To avoid unstable optimization caused by overly strong domain confusion at the early stage, we adopt a progressive schedule for the GRL reversal strength:

$$\alpha(p) = \frac{2}{1 + \exp(-10p)} - 1, \qquad p \in [0,1]. \tag{12}$$

We adopt a logistic ramp-up as the progressive weighting function because it provides a smooth and monotonic transition from weak to strong adaptation. In early epochs, the classifier is not yet stable and pseudolabels can be noisy; in this case, keeping adaptation-related objectives small helps preserve discriminative learning and avoids negative transfer. As training proceeds, the logistic curve gradually increases the weights, strengthening domain alignment and self-training only when predictions become more reliable. Compared with step-wise or linear schedules, the logistic ramp-up changes slowly at the beginning and the end, which reduces abrupt shifts in the optimization landscape and improves training stability. The slope controls how fast the transition happens around mid-training.

This design follows a curriculum of discrimination first and alignment later; the model focuses on learning class-discriminative representations on the source domain at early epochs, then progressively strengthens adversarial domain alignment in later training to improve cross-domain transferability.

To learn domain-invariant features, we employ a domain discriminator $D(\cdot)$ coupled with a gradient reversal layer. Let $f_s = G(x_s)$ for $x_s \sim \mathcal{D}_s$ and $f_t = G(x_t)$ for $x_t \sim \mathcal{D}_t$, and let the discriminator output $D(f) \in (0,1)$ be the predicted probability that $f$ comes from the source domain. The GRL applies the identity transform in the forward pass and multiplies the gradient by $-\alpha(p)$ in the backward pass, where $\alpha(p) \in [0,1]$ is increased

progressively with $p$. The discriminator is trained to distinguish domains, while $G$ is trained to confuse it; the adversarial objective is formulated as:

$$\mathcal{L}_{\text{adv}} = -\mathbb{E}_{x_s \sim \mathcal{D}_s}[\log D(G(x_s))] - \mathbb{E}_{x_t \sim \mathcal{D}_t}[\log(1 - D(G(x_t)))]. \tag{13}$$

We further introduce an increasing weight for adversarial alignment:

$$\lambda_{\text{adv}}(p) = \frac{2}{1 + \exp(-10p)} - 1, \qquad p \in [0, 1] \tag{14}$$

which is close to 0 at early epochs and gradually approaches 1, thereby activating alignment only after the classifier becomes sufficiently discriminative.

To encourage confident predictions on unlabeled target data, we minimize the conditional entropy of the target predictive distribution. For a target sample $x_t$, let $p_t = \text{softmax}(C(G(x_t)))$. The entropy loss is defined as:

$$\mathcal{L}_{\text{ent}} = \mathbb{E}_{x_t \sim \mathcal{D}_t}\left[ -\sum_{k=1}^{K} p_t^{(k)} \log\left(p_t^{(k)} + \epsilon\right)\right], \tag{15}$$

where $\epsilon = 10^{-8}$ avoids numerical instability. Similar to adversarial alignment, we progressively increase its contribution using:

$$\lambda_{\text{ent}}(p) = \frac{2}{1 + \exp(-10p)} - 1, \qquad p \in [0, 1]. \tag{16}$$

Combining the above objectives with the prototype alignment term in Section 3.3, the total loss is:

$$\mathcal{L} = \lambda_{\text{cls}}(p)\mathcal{L}_{\text{cls}} + \lambda_{\text{adv}}(p)\mathcal{L}_{\text{adv}} + \lambda_{\text{ent}}(p)\mathcal{L}_{\text{ent}} + \lambda_{\text{proto}}(p)\mathcal{L}_{\text{proto}}, \qquad p \in [0, 1]. \tag{17}$$

Overall, progressive domain alignment mitigates optimization conflicts by delaying strong alignment until discriminative structures are formed on the source domain. As training advances, adversarial alignment, entropy minimization, and prototype-based class-conditional constraints are gradually strengthened, leading to improved target-domain generalization under substantial cross-project distribution shifts.

Algorithm 1 outlines the proposed progressive domain alignment training pipeline, comprising scheduled loss weighting, adversarial feature alignment, target-entropy minimization, and prototype-guided high-confidence pseudolabel optimization.

**Algorithm 1** Progressive Domain Alignment Training (DAP2ER)

**Require:** Labeled source dataset $\mathcal{D}_s = (x_s^{(i)}, y_s^{(i)})_{i=1}^{n_s}$; unlabeled target dataset $\mathcal{D}_t = x_t^{(j)}{}_{j=1}^{n_t}$; feature extractor $G$, label classifier $C$, domain discriminator $D$; maximum epochs $E$; confidence threshold $\nabla$; $\epsilon = 10^{-8}$; scheduling functions $\lambda_{\text{cls}}(p)$, $\lambda_{\text{adv}}(p)$, $\lambda_{\text{ent}}(p)$, $\lambda_{\text{proto}}(p)$ and GRL strength $\alpha(p)$; optimizer (e.g., Adam).

**Ensure:** Predicted labels for target samples $\{\hat{y}_t^{(j)}\}_{j=1}^{n_t}$

1: Initialize $G$, $C$, and $D$.
2: Set training hyperparameters (learning rate, batch size, etc.).
3: **for** $e = 1$ to $E$ **do**
4: $p \leftarrow e/E$ ▷ normalized training progress $p \in (0, 1]$
5: Update $\lambda_{\text{cls}} \leftarrow \lambda_{\text{cls}}(p)$, $\lambda_{\text{adv}} \leftarrow \lambda_{\text{adv}}(p)$, $\lambda_{\text{ent}} \leftarrow \lambda_{\text{ent}}(p)$, $\lambda_{\text{proto}} \leftarrow \lambda_{\text{proto}}(p)$.
6: Update GRL strength $\alpha \leftarrow \alpha(p)$.
7: Compute source prototypes $\mu_{sc=1}^{cK}$ from $\mathcal{D}_s$ (Equation (1)).
8: **for all** mini-batches $\mathcal{B}_s = (x_s, y_s) \sim \mathcal{D}_s$ and $\mathcal{B}_t = x_t \sim \mathcal{D}_t$ **do**

**Algorithm 1** *Cont.*

$f_s \leftarrow G(x_s)$; $f_t \leftarrow G(x_t)$.
$p_s \leftarrow \text{softmax}(C(f_s))$; $p_t \leftarrow \text{softmax}(C(f_t))$.
$\mathcal{L}_{\text{cls}} \leftarrow \text{CE}(p_s, y_s)$.
$f_{\text{cat}} \leftarrow [f_s; f_t]$.
$d_{\text{cat}} \leftarrow [\mathbf{1}_{|\mathcal{B}_s|}; \mathbf{0}_{|\mathcal{B}_t|}]$.
$\mathcal{L}_{\text{adv}} \leftarrow \text{BCE}\big(D(\text{GRL}(f_{\text{cat}}, \alpha)), d_{\text{cat}}\big)$.
$\mathcal{L}_{\text{ent}} \leftarrow -\frac{1}{|\mathcal{B}_t|} \sum_{x_t \in \mathcal{B}_t} \sum * k = 1^K p_t^{(k)} \log\left(p_t^{(k)} + \epsilon\right)$.
$\hat{y}_t \leftarrow \arg\max_k p_t^{(k)}$.
$\hat{p}_t \leftarrow \max_k p_t^{(k)}$; $\text{conf}_t \leftarrow 2|\hat{p}_t - 0.5|$.
$m_t \leftarrow \mathbb{I}(\text{conf}_t \geq \nabla)$.
Construct $T_c = x_t \in \mathcal{B}_t \mid m_t = 1 \wedge \hat{y}_t = c$ for each class $c$.
Compute $\mu_t^c = \frac{1}{|T_c|} \sum_{x_t \in T_c} G(x_t)$ for non-empty $T_c$.
$C_{\text{val}} \leftarrow c \mid |T_c| > 0 \wedge N_s^c > 0$.
$\mathcal{L}_{\text{proto}} \leftarrow \frac{1}{\max(1, |C_{\text{val}}|)} \sum_{c \in C_{\text{val}}} |\mu_t^c - \mu_s^c|_2^2$.
$\mathcal{L} \leftarrow \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{ent}}\mathcal{L}_{\text{ent}} + \lambda_{\text{proto}}\mathcal{L}_{\text{proto}}$.
Backpropagate $\mathcal{L}$ and update parameters of $G$, $C$, and $D$.
**end for**
**end for**
**return** $\{\hat{y}_t^{(j)}\}_{j=1}^{n_t}$.

## 4. Experiments

### *4.1. Experimental Design*

The primary objective of the experiments in this section is to evaluate the performance of the proposed DAP2ER method and compare it with state-of-the-art techniques in the domain of cross-project imbalanced software vulnerability detection. Specifically, the focus is on the task of cross-domain vulnerability detection, where the imbalance between vulnerable and non-vulnerable functions is pronounced.

For experimental datasets, this study uses two real-world source code datasets, each containing both vulnerable and non-vulnerable functions. The first dataset, FFmpeg, includes 187 vulnerable functions and 5427 non-vulnerable functions. The second dataset, LibPNG, contains 43 vulnerable functions and 551 non-vulnerable functions.

It is important to note that these datasets exhibit extreme class imbalance, with the proportion of vulnerable data representing only about 0.51% to 11.65% of the non-vulnerable data. Our observations suggest that in cross-domain vulnerability detection, the smaller the proportion of vulnerable samples relative to non-vulnerable samples within a given source-target domain pair, the more severe the imbalance problem. This issue can also arise when transitioning between different source–target domain pairs.

To explicitly address the severe imbalance that varies across projects and transfer directions, we adopt a unified imbalance-handling strategy for each source→target experiment based on the labeled source training split. Specifically, we use a cost-sensitive classification objective by applying class weights in the source-domain classification loss, where the weight of each class is computed from the source training data. This makes misclassification of minority classes more costly and alleviates the dominance of the majority class during optimization. In addition, to ensure sufficient exposure of minority samples during training, mini-batches are formed with class-balanced sampling on the source training set. Notably, because the target domain is unlabeled during training under the UDA setting, imbalance handling is performed only on the labeled source side, while target samples are incorporated through domain alignment and the proposed high-confidence selection mechanism.

In this experiment, we aim to demonstrate the transfer learning capability of the proposed method for imbalanced cross-domain software vulnerability detection. We use FFmpeg, a multimedia application dataset, as the source domain and LibPNG, an image processing application dataset, as the target domain. It is noteworthy that the labels of the target domain dataset are hidden during training and are only revealed during the testing phase to evaluate the model's performance.

In the data processing and embedding stage, we perform a series of preprocessing steps before feeding the source code datasets into the neural network. First, we standardize the source code by removing comments, blank lines, and non-ASCII characters. Then, we map user-defined variables to symbolic variable names and user-defined functions to symbolic function names. Additionally, integers, real numbers, and hexadecimal numbers are replaced with a generic number token, while strings are replaced with a generic Strings token. Subsequently, we embed source code statements into numerical vectors by tokenizing each code statement into a sequence of code tokens, constructing a frequency vector representing the information of that statement, and multiplying this frequency vector by a learnable embedding matrix $W^{st}$.

To comprehensively evaluate the effectiveness of the proposed DAP2ER method in cross-domain multi-classification tasks, we compare it against five widely used and representative baseline methods from the domain adaptation literature: SourceOnly [19], DANN [19], PseudoLabeling [31], MMD [32], and CORAL [33]. These methods cover the mainstream technological approaches and can verify the advantages and necessity of DAP2ER from different perspectives.

All comparison methods share the same backbone network architecture in the experiments, with consistent training epochs, optimizers, and data partition strategies. The only difference lies in whether the corresponding domain alignment/pseudolabeling modules and loss terms are introduced. Additionally, all domain adaptation methods strictly follow the UDA setup: during training, only the true labels of the source domain are used, and no true label information is used from the target domain data. The source-only method trains the classifier using only the labeled source domain data without any domain alignment strategy and directly transfers the model to the target domain for testing, serving as the baseline for cross-domain performance. The pseudolabeling method first trains an initial model on the source domain, then generates pseudolabels for the target domain samples; in subsequent training, the high-confidence target samples are used as supervision signals for optimization, improving the model's adaptation to target domain data. This strategy is generally effective when predictions are more reliable in the early stages of the model but is sensitive to pseudolabel noise. MMD is a statistical distribution alignment method that minimizes the distance between the source and target domain feature distributions in the RKHS. Its advantages are simplicity and stability, but its limitation is that it mainly performs marginal distribution alignment, which can lead to alignment with class confusion when there is class imbalance or significant differences in class-conditional distributions. CORAL aligns the covariance matrices of source and target domain features to achieve second-order statistical matching, and is a lightweight distribution alignment strategy. Compared to MMD, CORAL does not rely on kernel functions and has lower computational overhead; like MMD, however, it tends to align global statistics, with limited ability for class-level alignment. DANN is a classic deep adversarial domain adaptation method that uses a gradient reversal layer and domain discriminator for adversarial training, enabling the feature extractor to learn domain-invariant representations, thereby reducing the discrepancy between source and target domains. This method has shown robust performance in multiple UDA tasks and is one of the key adversarial baselines.

Building upon these baselines, our proposed DAP2ER method introduces mechanisms such as progressive weight scheduling and high-confidence pseudolabel/prototype alignment. The method prioritizes preserving the source domain's discriminative capability during the early stages of training, gradually increasing domain adversarial and target domain constraints to achieve more stable and superior performance on the target domain. This approach is especially beneficial in scenarios involving cross-domain transfer, class-conditional alignment, and imbalance.

In the model configuration, all experiments were implemented using the PyTorch version: 1.7.1 framework and accelerated using NVIDIA GPUs. For reproducibility, we fixed the random seed to 42 for all experiments. In all methods (including baselines and DAP2ER), we applied the same class-imbalance handling strategy (class-weighted source classification loss and class-balanced sampling on the labeled source training data) to ensure fair comparison across different domain adaptation objectives.

During training, the Adam optimizer was employed to update model parameters, with the learning rate set to $2 \times 10^{-3}$ and the weight decay coefficient set to $1 \times 10^{-4}$. The models were optimized over 150 training epochs. The source domain data were split into 80% for training and 20% for testing, while the target domain data were split into 50% for training and 50% for testing. Importantly, the target domain training set did not utilize real labels. The batch size was set to 128 in each training epoch to ensure sufficient sample support for gradient updates.

In the early stages of training, domain adaptation-related losses were progressively increased through a scheduling mechanism in order to balance the conflict between classification learning and cross-domain alignment tasks. This progressive approach ensures that the relevant loss weights start at lower values, gradually increasing to avoid oscillations and ensuring model stability and generalization in the later stages of training.

For each run, we first randomly shuffled samples in both the source and target domains using a fixed random seed, then partitioned the data as follows: for the source domain, we split the dataset into 80% for training and 20% for evaluation; for the target domain, we split the dataset into 50% for training and 50% for evaluation. During training, target-domain labels were not used; the target training split was treated as unlabeled data, and only served to compute the domain-adversarial, entropy minimization, and prototype alignment objectives. The held-out target split was used exclusively for performance evaluation with ground truth labels.

The experimental evaluation employs the standard classification evaluation metrics of Accuracy, Precision, Recall, F1-score, and AUC to thoroughly assess the performance of the proposed method on the target domain. These metrics provide a comprehensive assessment of classification effectiveness, particularly in scenarios involving class imbalance. The detailed calculation procedures are outlined below.

For each sample in the target domain, its feature representation was first extracted, then the corresponding outputs were generated by the prediction model. These outputs were processed and converted into a probability distribution, with the probability of the vulnerability class being taken as the predicted probability of the sample belonging to the positive class.

We determined the predicted label for each sample by setting a threshold of 0.5. If the predicted probability of vulnerability was greater than or equal to 0.5, the sample was classified as a positive class; otherwise, it was classified as a negative class. The specific calculation formula is as follows:

$$\hat{y} = \begin{cases} 1, & \text{if probs}[:, 1] \geq 0.5, \\ 0, & \text{if probs}[:, 1] < 0.5. \end{cases} \tag{18}$$

Based on the predicted labels, we evaluate the model performance using the following standard classification metrics: Accuracy: The proportion of correctly classified samples, computed as follows:

$$\text{Accuracy} = \frac{\text{Number of correctly predicted samples}}{\text{Total number of samples}}. \tag{19}$$

Precision: The proportion of true positive samples among all samples predicted as positive, computed as follows:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}. \tag{20}$$

Recall: The proportion of true positive samples among all actual positive samples, computed as follows:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}. \tag{21}$$

F1-score: The harmonic mean of Precision and Recall, balancing both metrics, computed as follows:

$$F1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{22}$$

AUC: The area under the Receiver Operating Characteristic (ROC) curve, reflecting the classifier's performance across various thresholds. A higher AUC indicates better model performance. AUC is calculated based on the true labels and predicted probabilities:

$$\text{AUC} = \int_0^1 \text{True Positive Rate}\, d(\text{False Positive Rate}). \tag{23}$$

In the above equations, True Positive refers to the number of samples correctly predicted as positive, False Positive refers to the number of negative samples incorrectly predicted as positive, and False Negative refers to the number of positive samples incorrectly predicted as negative.

### *4.2. Comparative Experiments*

To evaluate the effectiveness of DAP2ER in cross-project imbalanced software vulnerability detection tasks, we establish a bidirectional transfer setup between two projects and compare it with five representative unsupervised/weakly-supervised domain adaptation baseline methods: Source-Only, Pseudolabel, MMD, CORAL, and DANN. The comparison results are presented in Tables 2 and 3. The evaluation metrics include Accuracy, Precision, Recall, F1-score, and AUC, which comprehensively assess detection performance under class-imbalanced scenarios.

**Table 2.** Performance comparison of the transfer task FFmpeg → LibPNG.

| Method | Accuracy | Precision | Recall | F1-Score | AUC |
|---|---|---|---|---|---|
| Source-Only | 54.63% | 51.15% | 59.79% | 55.13% | 56.22% |
| Pseudolabel | 46.88% | 46.98% | 99.20% | 63.77% | 54.35% |
| MMD | 58.25% | 56.60% | 85.37% | 68.07% | 66.51% |
| CORAL | 47.38% | 47.79% | 56.75% | 51.89% | 48.84% |
| DANN | 61.62% | 57.83% | 82.28% | 67.92% | 70.18% |
| DAP2ER | **87.30%** | **83.27%** | **93.16%** | **87.94%** | **95.93%** |

**Table 3.** Performance comparison of the transfer task LibPNG → FFmpeg.

| Method | Accuracy | Precision | Recall | F1-Score | AUC |
|---|---|---|---|---|---|
| Source-Only | 54.13% | 54.03% | 76.09% | 63.19% | 55.66% |
| Pseudolabel | 48.63% | 46.46% | 55.41% | 50.54% | 48.66% |
| MMD | 55.13% | 56.31% | 60.24% | 58.21% | 58.34% |
| CORAL | 54.63% | 57.60% | 38.52% | 44.17% | 47.52% |
| DANN | 65.50% | 64.02% | 71.96% | 67.76% | 72.35% |
| DAP2ER | **89.30%** | **91.18%** | **86.53%** | **88.80%** | **95.92%** |

As shown in Tables 2 and 3, DAP2ER outperforms all other methods across all core metrics on the source-to-target transfer task, achieving an F1-score of 0.8794 and an AUC of 0.9593, which are the highest among the compared methods. Specifically, the strongest baseline, MMD, achieves an F1-score of 0.6807, while DAP2ER improves upon this by 0.1987. Similarly, DAP2ER achieves a 0.2575 higher AUC compared to DANN's 0.7018. Notably, while the Pseudolabel category exhibits extremely high Recall, its Precision is only 0.4698, indicating that it tends to generate noisy pseudolabels and over-predict the positive class under significant domain shift and imbalance. In contrast, DAP2ER maintains high Recall while significantly improving Precision, resulting in a more balanced and stable detection performance.

In the target-to-source transfer task, DAP2ER also demonstrates substantial advantages, achieving Accuracy = 89.30%, Precision = 91.18%, Recall = 86.53%, F1-score = 88.80%, and AUC = 95.92%. Compared to the best baseline, DANN, DAP2ER improves F1-score by 0.2104 and AUC by 0.2357. While DANN achieves relatively high Recall on the Source-Only category, it significantly lags behind DAP2ER in terms of F1-score, indicating that relying solely on source-domain supervision cannot effectively overcome cross-project distribution discrepancies.

Overall, DAP2ER demonstrates superior F1-score and AUC in both transfer directions, highlighting its ability to not only improve the identification of vulnerable samples in the target domain but also enhance the control of false positives and false negatives in class-imbalanced settings. This showcases the robustness and effectiveness of the proposed progressive alignment strategy and multi-constraint collaborative optimization in cross-domain vulnerability detection tasks.

We observe that vulnerabilities with weaker lexical/structural cues and higher context dependence are generally harder to detect in the cross-project setting. In particular, logic-related flaws and subtle semantic misuse patterns often manifest through dispersed code semantics rather than localized tokens, making their representations less transferable across projects. Similarly, boundary-condition and rare corner-case vulnerabilities tend to be underrepresented in the training data and to exhibit high intra-class diversity, which increases the uncertainty of pseudolabeling and weakens class-conditional alignment. In contrast, vulnerabilities with more explicit local signatures are comparatively easier to capture. Overall, these observations suggest that detection performance degrades when vulnerability categories are rare, semantically subtle, or heavily reliant on long-range program context, which is further amplified by domain shift and severe class imbalance in cross-project vulnerability detection.

### *4.3. Ablation Study*

To further validate the contribution of each module in DAP2ER, we conducted ablation experiments by systematically removing or disabling key components of the model: w/o DANN, removal of the domain adversarial training module; w/o Entropy, removal of the

target-domain entropy minimization constraint; w/o Prototype, removal of the prototype alignment module; and Full DAP2ER, the complete model.

In this section, we evaluate the following four model configurations: Only DANN, utilizing only the domain adversarial module from DANN while removing entropy minimization and prototype alignment; w/o Entropy, removing the entropy minimization module while retaining both domain adversarial and prototype alignment; w/o Prototype, removing the prototype alignment module while retaining both domain adversarial and entropy minimization; and DAP2ER, the full DAP2ER method incorporating all three modules.

As shown in Table 4, DAP2ER achieves the best overall performance; Accuracy increases from 0.870 to 0.873, F1-score from 0.8780 to 0.8785, and Precision from 0.8225 to 0.8376. However, Recall slightly decreases from 0.9416 to 0.9235, indicating that the model adopts a more conservative approach to positive-class predictions with the introduction of the entropy constraint and prototype guidance. This reduces false positives and improves the Precision, albeit at the expense of a slight reduction in Recall. This observation aligns with the design goal of high-confidence screening and class-conditional alignment to suppress noise propagation. It is worth noting that the performance metrics for w/o Entropy and Only DANN are virtually identical in this setup, with minimal fluctuations in AUC. Similarly, the difference between w/o Prototype and Only DANN is marginal, indicating that adversarial alignment plays a dominant role in performance improvement. Meanwhile, the benefits of entropy minimization and prototype-guided alignment are primarily observed in enhancing prediction reliability and decision boundary robustness, leading to a modest but stable improvement in the overall metrics for DAP2ER.

**Table 4.** Ablation study results.

| Source→Target | Methods | Accuracy | Precision | Recall | F1 | AUC |
|---|---|---|---|---|---|---|
| FFmpeg→LibPNG | Only DANN | 87.0% | 82.25% | 94.16% | 87.8% | 96.19% |
| | w/o Entropy | 87.2% | 82.25% | 94.16% | 87.8% | 96.19% |
| | w/o Prototype | 87.2% | 82.25% | 94.16% | 87.8% | 96.20% |
| | Full DAP2ER | 87.3% | 87.85% | 92.35% | 87.85% | 95.75% |

To further visually assess the contribution of each key module, we present the ablation results in the form of a heatmap (Figure 2). From the overall color distribution, it is evident that the three variants (Only DANN, w/o Entropy, and w/o Prototype) exhibit nearly identical color intensities across the four metrics of Accuracy, Precision, F1-score, and Recall. This suggests that domain adversarial alignment is the main driving force behind model performance under the current transfer setting, providing a stable foundational alignment capability for target-domain transfer. In contrast, the removal of either entropy minimization or prototype guidance does not lead to significant performance degradation, indicating that these modules primarily serve as fine-tuning mechanisms for decision boundaries and class-conditional structures, rather than being the sole contributors to baseline performance.

Notably, the heatmap reveals a deeper color in the Precision and F1-score dimensions for DAP2ER, along with a slight improvement in Accuracy. This suggests that the improvements primarily stem from enhanced prediction reliability and discriminability in the target domain. Specifically, entropy minimization reduces uncertainty in target-domain predictions, promoting clearer classification boundaries. On the other hand, the prototype-guided strategy enhances intra-class sample aggregation and minimizes inter-class confusion through class-conditional constraints, thereby improving the Precision. The heatmap also shows a slight decrease in Recall for DAP2ER; combined with the improved Precision, this reflects a more robust but relatively conservative decision tendency in the

target domain. This tradeoff, in which a reduction in Recall is compensated for by a higher Precision, results in an overall improvement in F1-score.

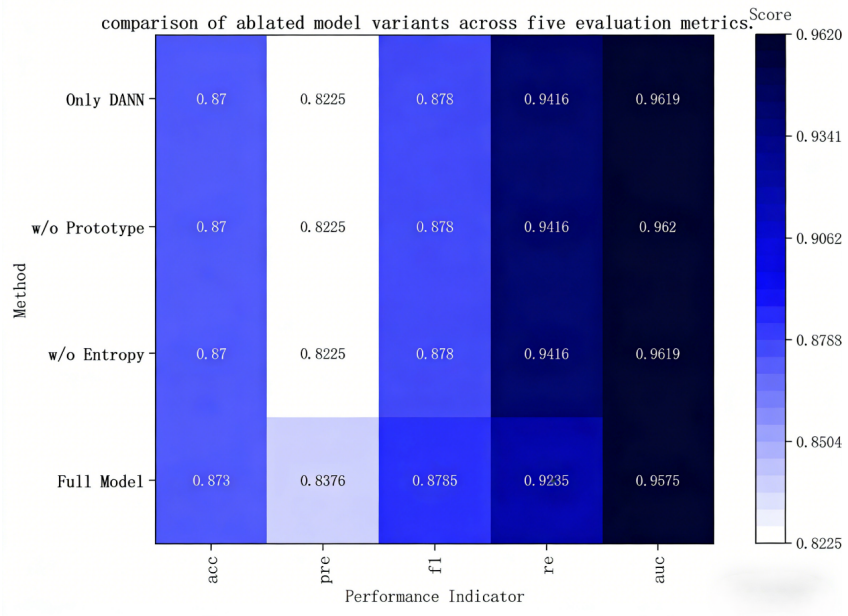



**Figure 2.** Heatmap visualization of ablation study results across different model configurations.

### *4.4. Hyperparameter Sensitivity Analysis*

To evaluate the sensitivity of our method to changes in the key hyperparameters, we conducted a series of experiments focusing on the impact of $\alpha$, $\nabla$, $h$, and $\lambda_p^{\max}$ on model performance. In these experiments, the target-domain AUC was used as the primary evaluation metric.

Specifically, we independently varied $\alpha$, $\nabla$, $h$, and $\lambda_p^{\max}$ while keeping all other hyperparameters fixed. For each setting, the model was trained using the same protocol and the corresponding AUC on the target domain was recorded. The results cover a range of values for $\alpha$ and different settings of $\lambda_p^{\max}$ as well as variations in the hidden layer size $h$ and the confidence threshold. The hyperparameter sensitivity analysis results are shown in Table 5.

**Table 5.** Hyperparameter sensitivity analysis on the target domain (AUC, seed = 42). The best result for each hyperparameter is highlighted in bold.

| Hyperparameter | Value | Target AUC |
|---|---|---|
| $\alpha$ | 0.1 | 0.808 |
| | 0.3 | 0.888 |
| | 0.5 | 0.883 |
| | 0.7 | **0.909** |
| | 1.0 | 0.839 |
| $\nabla$ | 0.50 | 0.915 |
| | 0.60 | 0.919 |
| | 0.70 | 0.839 |
| | 0.80 | **0.947** |
| | 0.90 | 0.908 |
| $h$ | 32 | 0.882 |
| | 64 | 0.815 |
| | 128 | 0.839 |
| | 256 | **0.927** |
| | 512 | 0.871 |

**Table 5.** *Cont.*

| Hyperparameter | Value | Target AUC |
|---|---|---|
| $\lambda_p^{\max}$ | 0.1 | **0.946** |
| | 0.3 | 0.918 |
| | 0.5 | 0.839 |
| | 0.7 | 0.777 |
| | 0.9 | 0.859 |

As shown in Figure 3, increasing $\alpha$ from 0.1 to 0.3 leads to a clear improvement in target AUC, with the best performance achieved at $\alpha = 0.7$. When $\alpha$ is further increased to 1.0, the AUC decreases. This trend indicates that a small $\alpha$ results in insufficient adversarial signals leading to inadequate feature alignment, whereas an overly large $\alpha$ may cause excessive domain confusion, weakening class-discriminative information and destabilizing optimization. Therefore, we set $\alpha = 0.7$ as the default value in order to strike a better tradeoff between domain invariance and class separability.

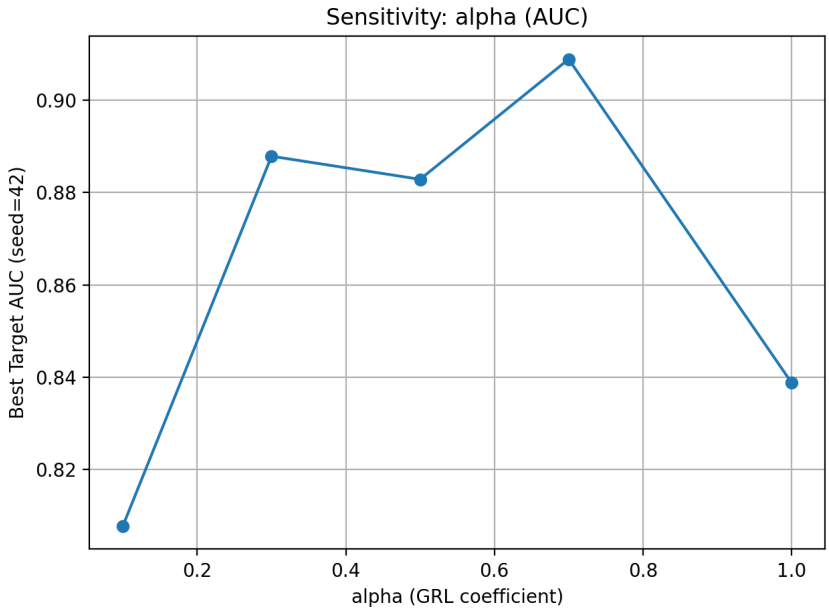



**Figure 3.** Sensitivity analysis of the GRL coefficient $\alpha$ on the target domain.

As shown in Figure 4, the confidence threshold has a significant impact on performance, with the best AUC obtained at $\nabla = 0.8$. When the threshold is reduced to 0.7, the AUC drops noticeably to 0.84, suggesting that more noisy pseudolabels are retained and the resulting class-conditional alignment can be corrupted, leading to negative transfer. When the threshold is increased to 0.9, the AUC also slightly decreases, which is likely because overly strict filtering retains too few target samples and yields unreliable class-wise statistics. Overall, these results support choosing $\nabla = 0.8$ as the default threshold.

As shown in Figure 5, the model achieves the highest AUC at $h = 256$. With a smaller hidden layer size, the performance is lower, indicating that limited model capacity may restrict representation learning and hinder the learning of robust decision boundaries under domain shift. When $h$ becomes too large, the AUC also decreases, which may be attributed to increased overfitting risk or more difficult optimization due to the enlarged parameter space. Considering both performance and complexity, we adopt $h = 256$ as the default hidden layer width.

As shown in Figure 6, we observe an overall trend that larger values of $\lambda_p^{\max}$ lead to worse performance; the best AUC is achieved at $\lambda_p^{\max} = 0.1$, while the performance

degrades substantially when $\lambda_p^{\max}$ increases to 0.5 and 0.7. This observation suggests that prototype alignment acts as a strong class-conditional constraint and is highly sensitive to pseudolabel quality. When the prototype loss is overly weighted, target samples with noisy pseudolabels may be forcibly pulled towards incorrect prototypes. This distorts class clusters and amplifies the effect of residual noise, causing negative transfer. Hence, we set $\lambda_p^{\max} = 0.1$ and combine it with a progressive weighting strategy to improve training stability in early epochs.

In summary, the above sensitivity analyses indicate that our method remains effective within a reasonable range of hyperparameter values, although clear optima exist for the examined parameters. Based on the target-domain AUC results, we subsequently used $\alpha = 0.7$, $\nabla = 0.8$, $h = 256$, and $\lambda_p^{\max} = 0.1$ as the default settings in our experiments to obtain stable and strong cross-domain detection performance.

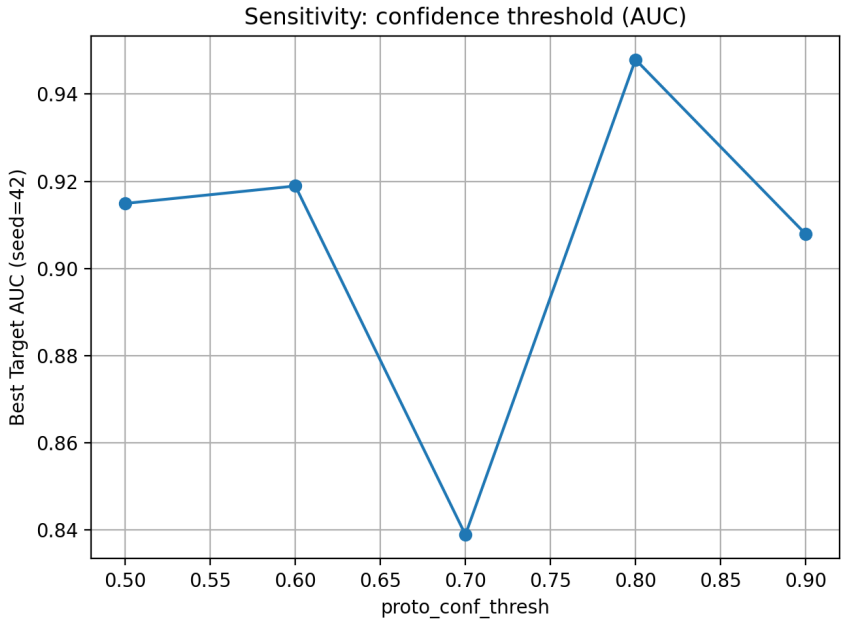



**Figure 4.** Sensitivity analysis of the pseudolabel confidence threshold $\nabla$ on the target domain.

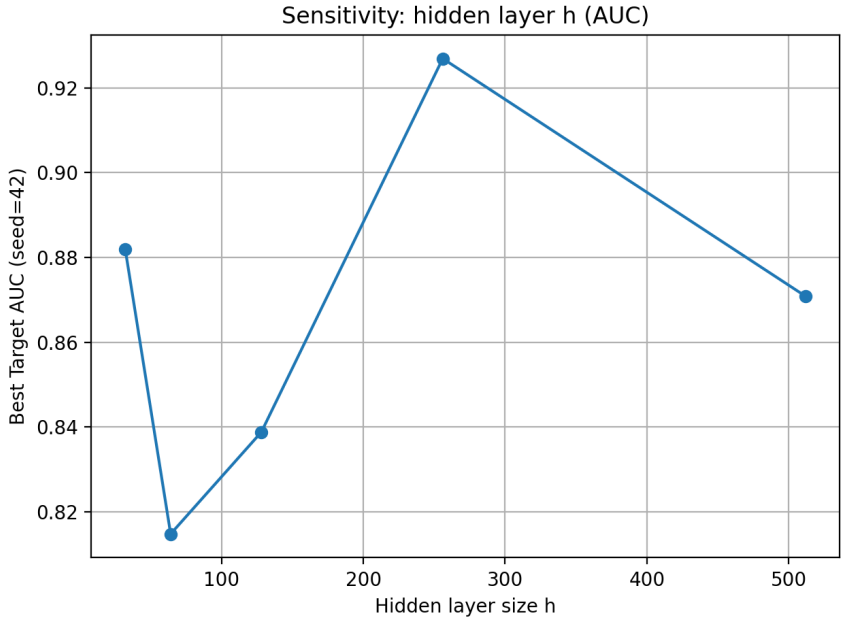



**Figure 5.** Sensitivity analysis of the hidden layer size $h$ on the target domain.

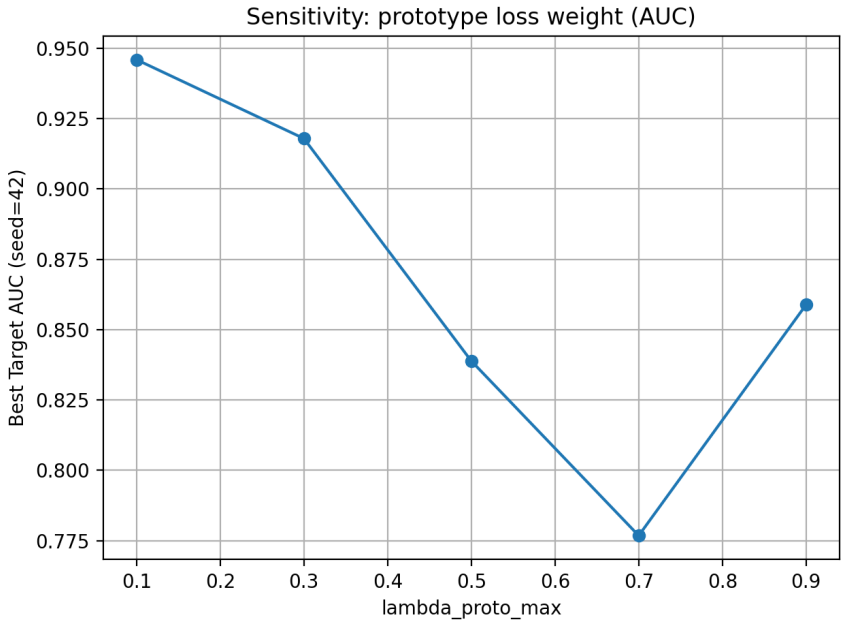



**Figure 6.** Sensitivity analysis of the maximum prototype loss weight $\lambda_p^{\max}$ on the target domain.

### *4.5. Feature Visualization Results*

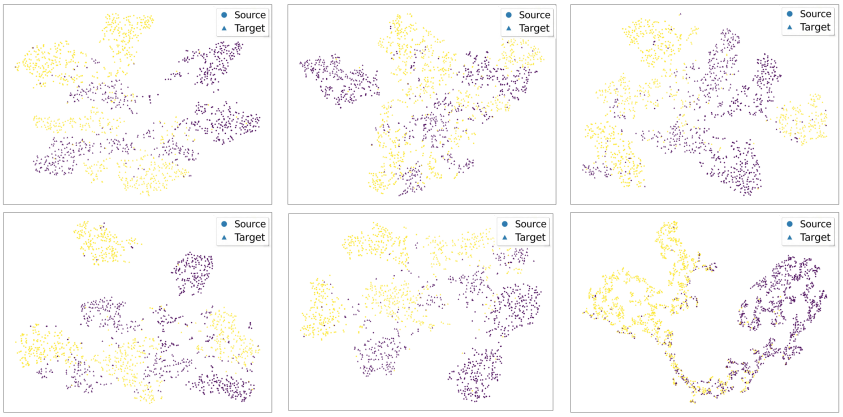
To qualitatively evaluate the feature alignment achieved by different methods, we visualize the learned representations using t-SNE, a common practice in domain adaptation. Specifically, we project the joint features learned by each model into a two-dimensional space and show the distributions of source- and target-domain samples for both transfer directions. The t-SNE visualizations are shown in Figure 7.

Source Only: The source and target features are clearly separated, indicating that domain shift has not been mitigated. This results in limited generalization to the target domain.

MMD, CORAL, and DANN: These methods reduce the inter-domain discrepancy to some extent; however, overlapping class clusters and/or incomplete alignment are still observable. This suggests that purely global distribution alignment or adversarial alignment alone may not sufficiently address class-level ambiguity under significant domain shift and class imbalance.

DAP2ER: DAP2ER produces a substantially stronger overlap between source and target samples, resulting in more compact clusters with clearer class boundaries. This indicates that DAP2ER improves domain alignment while maintaining class discriminability, which is consistent with the quantitative results, where DAP2ER achieves the highest F1-score and AUC in both transfer directions.

In summary, both the quantitative and visualization results demonstrate that the advantage of DAP2ER arises not from a single technique but from the stabilizing effect of its progressive training strategy. The complementary roles of entropy minimization and prototype alignment continually refine class decision boundaries in the unlabeled target domain, leading to more reliable generalization for cross-project imbalanced vulnerability detection.




(**a**) t-SNE visualization for FFmpeg → LibPNG


(**b**) t-SNE visualization for LibPNG → FFmpeg

**Figure 7.** t-SNE visualization of learned feature representations for different domain adaptation methods. The yellow points represent the source-domain samples, while the purple points represent the target-domain samples.

## 5. Conclusions

This work addresses the challenge of cross-project and imbalanced multi-class software vulnerability detection in a realistic setting where the source project is labeled but the target project remains unlabeled. To simultaneously mitigate domain shift and label scarcity, we propose DAP2ER, a progressive unsupervised domain adaptation framework that integrates adversarial domain alignment, entropy regularization, and prototype-guided high-confidence pseudolabel optimization. The core innovation of DAP2ER is a curriculum-based weighting schedule, which postpones strong domain alignment until the classifier is sufficiently reliable and leverages class-wise prototype alignment that utilizes only high-confidence target samples to reduce confirmation bias and category mismatch.

Empirical results demonstrate that DAP2ER significantly outperforms strong baseline methods across two real-world vulnerability datasets. DAP2ER consistently achieves superior performance in bidirectional transfer experiments, with the best Accuracy, F1-score, and AUC, outperforming representative alignment and self-training techniques by substantial margins. Ablation studies confirm the contributions of progressive training, entropy regularization, and prototype-guided alignment to the overall improvements in performance.

Future work can extend the evaluation to additional projects and vulnerability taxonomies to better understand generalization across diverse domain shifts. Further enhancement of transferable representations could be achieved by incorporating richer program semantics or leveraging large pretrained code models, which can naturally complement DAP2ER's progressive alignment strategy. Additionally, exploring open-set, partial, and

continual adaptation scenarios in which the target domain may contain unseen categories or evolve over time presents important avenues for further research.

In conclusion, DAP2ER provides a robust and extensible solution for unsupervised domain adaptation in cross-project vulnerability detection under class imbalance. Notably, its primary novelty lies in the progressive curriculum scheduling that prioritizes early source-domain discriminative learning and only later activates strong alignment, rather than merely combining adversarial training, entropy regularization, and prototype alignment. As such, it offers a practical foundation for the deployment of adaptation-based detectors in real-world software engineering pipelines.

**Author Contributions:** Conceptualization, Y.D. and J.Z.; methodology, Y.D. and J.Z.; software, Y.D., J.Z., Y.L., and G.L.; validation, Y.D. and G.L.; formal analysis, Y.D. and J.Z.; investigation, Y.D. and J.Z.; data curation, Y.L.; writing—original draft preparation, J.Z. and Y.L.; writing—review and editing, Y.D.; supervision, Y.D. and G.L.; project administration, Y.D.; funding acquisition, G.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The datasets used in this study, FFmpeg and LibPNG, are from publicly available software projects and have been described in the research by [34]. These datasets are accessible and have been utilized in software vulnerability detection research.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| MMD | Maximum Mean Discrepancy |
| CORAL | CORelation ALignment |
| DANN | Domain-Adversarial Training of Neural Network |
| UDA | Unsupervised Domain Adaptation |
| GRL | Gradient Reversal Layer |
| RKHS | Reproducing Kernel Hilbert Space |
| AUC | Area Under Curve |
| SAFA-DA | Statistically-Aligned Feature Augmentation for Domain Adaptation |
| MLRGL | Multiview Low-Rank High-Order Graph Learning |
| CAMSA | Consensus Augmented Masking for Subspace Alignment |
| SPADA | Spectral Prototype Attention Domain Adaptation |
| DAP2ER | Domain Adaptation with Progressive Prototype and Entropy Regularization |
| MLSL | Multiview Latent Space Learning |

## References

1. Li, Z.; Zou, D.; Xu, S.; Ou, X.; Jin, H.; Wang, S.; Deng, Z.; Zhong, Y. VulDeePecker: A Deep Learning-Based System for Vulnerability Detection. *arXiv* **2018**, arXiv:1801.01681.
2. Zhou, Y.; Liu, S.; Siow, J.; Du, X.; Liu, Y. Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
3. Booth, H.; Rike, D.; Witte, G.A. *The National Vulnerability Database (NVD): Overview*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2013.
4. Zhang, S.; Caragea, D.; Ou, X. An Empirical Study on Using the National Vulnerability Database to Predict Software Vulnerabilities. In *International Conference on Database and Expert Systems Applications*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 217–231.
5. Li, Z.; Zou, D.; Xu, S.; Jin, H.; Zhu, Y.; Chen, Z. SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 2244–2258. [CrossRef]

6. Kasula, V.K.; Yadulla, A.R.; Yenugula, M.; Konda, B. Enhancing Smart Contract Vulnerability Detection Using Graph-Based Deep Learning Approaches. In Proceedings of the 2024 International Conference on Integrated Intelligence and Communication Systems (ICIICS), Kalaburagi, India, 22–23 November 2024; pp. 1–6.
7. Yamaguchi, F.; Golde, N.; Arp, D.; Rieck, K. Modeling and Discovering Vulnerabilities with Code Property Graphs. In Proceedings of the 2014 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 18–21 May 2014; pp. 590–604.
8. Lekssays, A.; Mouhcine, H.; Tran, K.; Yu, T.; Khalil, I. LLMxCPG: Context-Aware Vulnerability Detection Through Code Property Graph-Guided Large Language Models. In Proceedings of the 34th USENIX Security Symposium (USENIX Security 25), Seattle, WA, USA, 13–15 August 2025; pp. 489–507.
9. Croft, R.; Babar, M.A.; Chen, H. Noisy Label Learning for Security Defects. In Proceedings of the 19th International Conference on Mining Software Repositories, Pittsburgh, PA, USA, 23–24 May 2022; pp. 435–447.
10. Mathews, N.S.; Brus, Y.; Aafer, Y.; Nagappan, M.; McIntosh, S. Llbezpeky: Leveraging Large Language Models for Vulnerability Detection. *arXiv* **2024**, arXiv:2401.01269. [CrossRef]
11. Long, M.; Cao, Z.; Wang, J.; Jordan, M.I. Conditional Adversarial Domain Adaptation. *Adv. Neural Inf. Process. Syst.* **2018**, *31*.
12. Sohn, K.; Berthelot, D.; Carlini, N.; Zhang, Z.; Zhang, H.; Raffel, C.A.; Cubuk, E.D.; Kurakin, A.; Li, C.-L. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 596–608.
13. Grandvalet, Y.; Bengio, Y. Semi-Supervised Learning by Entropy Minimization. *Adv. Neural Inf. Process. Syst.* **2004**, *17*.
14. Vu, T.-H.; Jain, H.; Bucher, M.; Cord, M.; Pérez, P. ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 2517–2526.
15. Wu, X.; Zhou, Q.; Yang, Z.; Zhao, C.; Latecki, L.J. Entropy Minimization vs. Diversity Maximization for Domain Adaptation. *arXiv* **2020**, arXiv:2002.01690. [CrossRef]
16. Kundu, J.N.; Bhambri, S.; Kulkarni, A.R.; Sarkar, H.; Jampani, V.; Venkatesh Babu, R. Subsidiary Prototype Alignment for Universal Domain Adaptation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 29649–29662.
17. Yan, Y.; Guo, Y. Partial Label Unsupervised Domain Adaptation with Class-Prototype Alignment. In Proceedings of the Eleventh International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
18. Pan, Y.; Yao, T.; Li, Y.; Wang, Y.; Ngo, C.-W.; Mei, T. Transferrable Prototypical Networks for Unsupervised Domain Adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 2239–2247.
19. Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; March, M.; Lempitsky, V. Domain-Adversarial Training of Neural Networks. *J. Mach. Learn. Res.* **2016**, *17*, 1–35.
20. Tzeng, E.; Hoffman, J.; Saenko, K.; Darrell, T. Adversarial Discriminative Domain Adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7167–7176.
21. Pei, Z.; Cao, Z.; Long, M.; Wang, J. Multi-Adversarial Domain Adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
22. Chen, H.; Li, L.; Chen, J.; Lin, K.-Y. Unsupervised Domain Adaptation via Double Classifiers Based on High Confidence Pseudo Label. *arXiv* **2021**, arXiv:2105.04729. [CrossRef]
23. Zhang, P.; Zhang, B.; Zhang, T.; Chen, D.; Wang, Y.; Wen, F. Prototypical Pseudo Label Denoising and Target Structure Learning for Domain Adaptive Semantic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 12414–12424.
24. Zhu, C.; Zhu, H.; Zhang, L.; Wang, F.; Zhu, Z. Statistically-Aligned Feature Augmentation for Robust Unsupervised Domain Adaptation in Industrial Fault Diagnosis. *J. Intell. Manuf.* **2025**, 1–17. [CrossRef]
25. Zhu, C.; Wang, Q.; Xie, Y.; Xu, S. Multiview Latent Space Learning with Progressively Fine-Tuned Deep Features for Unsupervised Domain Adaptation. *Inf. Sci.* **2024**, *662*, 120223. [CrossRef]
26. Nath Kundu, J.; Bhambri, S.; Kulkarni, A.; Sarkar, H.; Jampani, V.; Venkatesh Babu, R. Subsidiary Prototype Alignment for Universal Domain Adaptation. *arXiv* **2022**, arXiv:2210.15909. [CrossRef]
27. Li, Y.; Long, S.; Wang, S.; Zhao, X.; Li, Y. Prompt-Induced Prototype Alignment for Few-Shot Unsupervised Domain Adaptation. *Expert Syst. Appl.* **2025**, *269*, 126400. [CrossRef]
28. Zhang, W.; Hu, R.; Wang, J.; Zhang, L.; Zhu, C. Spectral Prototype Attention Domain Adaptation for Hyperspectral Image Classification. *Remote. Sens.* **2025**, *17*, 3901. [CrossRef]
29. Zhu, C.; Zhang, L.; Luo, W.; Jiang, G.; Wang, Q. Tensorial Multiview Low-Rank High-Order Graph Learning for Context-Enhanced Domain Adaptation. *Neural Netw.* **2025**, *181*, 106859. [CrossRef] [PubMed]
30. Zhu, C.; Luo, W.; Xie, Y.; Fu, L. Multiview Unsupervised Domain Adaptation through Consensus Augmented Masking for Subspace Alignment. *Appl. Intell.* **2025**, *55*, 1–17. [CrossRef]
31. Lee, D.H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Proceedings of the ICML Workshop on Challenges in Representation Learning, Atlanta, GA, USA, 16–21 June 2013.

32. Gretton, A.; Borgwardt, K.M.; Rasch, M.J.; Schölkopf, B.; Smola, A. A kernel two-sample test. *J. Mach. Learn. Res.* **2012**, *13*, 723–773.
33. Sun, B.; Feng, J.; Saenko, K. Return of frustratingly easy domain adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
34. Nguyen, V.; Le, T.; Tantithamthavorn, C.; Grundy, J.; Phung, D. Deep Domain Adaptation With Max-Margin Principle for Cross-Project Imbalanced Software Vulnerability Detection. *ACM Trans. Softw. Eng. Methodol.* **2024**, *33*, 1–34. [CrossRef]