WMN-CDA: Contrastive Domain Adaptation for Wireless Mesh Network Configuration

Aitian Ma
Knight Foundation School of Computing and Information Sciences
Florida International University
Miami, Florida, USA
aima@fiu.edu

Mo Sha
Knight Foundation School of Computing and Information Sciences
Florida International University
Miami, Florida, USA
msha@fiu.edu

Abstract

Wireless Mesh Networks (WMNs) have become essential for a wide range of applications, such as industrial automation, environmental monitoring, and smart cities. Network operators encounter significant challenges when selecting WMN parameters to ensure good network performance under various ambient operating conditions. Current domain adaptation methods designed for WMN configurations transfer the network configuration knowledge learned from simulations (source domain) to the physical deployment (target domain) at the domain level and fail to consider the simulation-to-reality gap variance under different network configurations, which causes misalignment and overfitting issues. To address such issues, we introduce the WMN Contrastive Domain Adaptation (WMN-CDA) framework, which leverages contrastive learning to transfer the network configuration knowledge learned from simulations to the physical deployment in a discriminative way at a granular scale. WMN-CDA employs the Network Configuration and Simulation-to-Reality contrastive losses to align feature representations and provide good network configuration predictions for physical deployments. We have implemented WMN-CDA and evaluated it with the data collected from a physical testbed with 50 devices and four wireless simulators. Experimental results show significant improvements over the baseline.

CCS Concepts

• **Networks** → **Network management**; **Sensor networks**; • **Computing methodologies** → **Machine learning approaches**.

Keywords

Cyber-Physical Systems, Wireless Mesh Networks, Network Configuration, Contrastive Learning, Domain Adaptation

ACM Reference Format:
Aitian Ma and Mo Sha. 2025. WMN-CDA: Contrastive Domain Adaptation for Wireless Mesh Network Configuration. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3672608.3707804

SAC '25, March 31-April 4, 2025, Catania, Italy

ACM ISBN 979-8-4007-0629-5/25/03
https://doi.org/10.1145/3672608.3707804

1 Introduction

Wireless Mesh Networks (WMNs) have become essential for a wide range of applications, such as industrial automation [1], environmental monitoring [2], and smart cities [3]. Thanks to decades of research, current WMNs provide flexible, scalable, and self-healing communication infrastructures by dynamically routing data through multiple wireless nodes [4]. However, WMN configuration remains challenging due to the diverse network conditions and stringent performance requirements on reliability, latency, and energy efficiency. In addition, the strict real-time communication demands in many applications, such as industrial automation, impose additional constraints, making the configuration even more challenging [5]. Over the past few years, simulations have been widely used to identify good network configurations for WMNs, because simulating a WMN provides distinct advantages over experimenting on a physical deployment when it comes to identifying a good network configuration: a simulation can be set up in less time, introduce less overhead, and allow for different configurations to be tested under exactly the same conditions. However, it is very challenging to set up a simulation that captures extensive uncertainties, variations, and dynamics in real-world WMN deployments. Recent studies show that the models for network configuration prediction learned from simulations cannot always help physical networks meet performance requirements because of the simulation-to-reality gap and propose to use domain adaptation to close the gap [6–8].

Domain adaptation has emerged as a promising approach when we have labeled training data collected from a source domain (simulations) and aim to learn a classifier, that performs well on a target domain (a physical deployment) with different distributions. Many domain adaptation approaches have been developed in the context of shallow learning, e.g. in the situation when data representation and features are given and fixed [9]. To solve the WMN configuration problem, current methods aim to train a prediction model with a large amount of simulation data and a few physical data by aligning simulation and physical representations. Research shows that the simulation-to-reality gap varies under different network configurations [6]. Current domain adaptation approaches for WMN configurations fail to consider the simulation-reality gap difference across different network configurations. This hinders the model's ability to transfer knowledge across domains due to two reasons. First, samples under different network configurations may be aligned incorrectly and the domain loss can be minimized even when the source domain samples (simulation data) are misaligned

with the target domain samples (physical data) of a different network configuration. Second, the learned decision boundary may generalize poorly for the target domain (physical data). There exist many suboptimal solutions near the decision boundary. Such solutions may overfit the source domain (simulation data) well but are less discriminative for the target domain (physical data).

In this paper, we introduce WMN-CDA, a novel framework that employs contrastive learning for domain adaptation in WMN configurations. WMN-CDA enables the model to transfer the network configuration knowledge from simulations to the physical deployment more effectively and discriminatively by bringing similar samples closer together while pushing dissimilar samples apart. Furthermore, it facilitates adaptation at a finer granularity, progressing from domain-level (simulation and reality) to configuration-level adaptation within network configurations. Specifically, WMN-CDA introduces both Network Configuration and Simulation-to-Reality contrastive losses to the existing teacher-student model, facilitating the alignment of feature representations across different network configurations in both simulation and reality. Those contrastive losses enable WMN-CDA to maintain consistent representations between domains, thereby preventing the misalignment of samples under varying network configurations. The Simulation-to-Reality contrastive loss, in particular, encourages the model to focus on robust, Simulation-to-Reality-invariant features, rather than overfitting to specific details or noise within the simulation. Such an approach reduces the model's dependency on the nuances of source domain data (simulation), thereby improving its prediction performance to target domain data (reality).

- We introduce a novel contrastive domain adaptation framework, WMN-CDA, which effectively bridges the network configuration gap between simulations and real-world deployments by learning discriminative feature representations at a more granular level, specifically under varying network configurations;
- To our knowledge, WMN-CDA is the first to use Network Configuration and Simulation-to-Reality contrastive losses to achieve robust feature alignment across different domains, effectively addressing the domain shift challenge inherent in WMN configurations;
- We have implemented WMN-CDA and showed its efficacy in enhancing prediction accuracy with the data collected from a physical WMN testbed and multiple wireless simulators.

The remainder of this paper is structured as follows: Section 2 reviews the related work in WMN, domain adaptation, and contrastive learning. Section 3 introduces our design of WMN-CDA. Section 4 presents our experimental results. Section 5 concludes the paper.

2 Related Work

WMNs have become essential for reliable, scalable, and flexible communication in industrial and commercial applications, but they pose several technical challenges. Ensuring reliable communication in industrial settings, such as those using WirelessHART, is difficult due to interference, multipath fading, and variable traffic loads [10]. Current network configuration practices on industrial WMNs often rely on manual, experience-based approaches, such as blacklisting noisy channels [11] in WirelessHART networks or using fixed Packet Reception Ratio (PRR) thresholds to select routing links. These methods have limitations, as recent studies suggest that using more channels or fixed PRR thresholds is not always optimal [12, 13]. While mathematical models and runtime adaptation techniques have been developed to optimize network configurations, they are typically insufficient for handling the complexity of modern, hierarchical wireless networks [14, 15]. Advanced machine learning techniques, including deep learning and reinforcement learning, have shown promise in optimizing network performance by using a large number of parameters and addressing the uncertainties inherent in real-world deployments [16–18]. However, the difficulty and cost of collecting sufficient data, especially from industrial environments, have hindered their widespread adoption. Simulations have been proposed as an alternative for training models, but significant performance gaps between simulation and real-world deployment persist, highlighting the challenge of bridging the simulation-to-reality gap [6].

Domain adaptation has become a crucial technique in machine learning, enabling models trained on one domain (source) to generalize to a new, unseen domain (target) with differing data distributions. Early methods focused on reweighting source samples to minimize the distribution gap between domains, but these approaches struggled with large domain shifts [19]. More recent techniques focus on learning domain-invariant representations by aligning feature spaces, such as using Maximum Mean Discrepancy (MMD) to bridge the gap between domains [20, 21]. Adversarial learning approaches like Domain-Adversarial Neural Networks (DANN) [22] train feature extractors to confuse a domain classifier, helping the model generalize across domains. In unsupervised Domain adaptation, where the target domain lacks labeled data, methods like self-training, pseudo-labeling, and contrastive learning have improved generalization [23]. Integrating contrastive learning into domain adaptation shows promise in further reducing the distribution gap, and improving performance when labeled data in the target domain is limited.

Contrastive learning has been widely used in representation learning, achieving superior results across various domains [24, 25]. It has recently gained prominence for its effectiveness in self-supervised learning, especially when labeled data is limited or unavailable. The core idea is to learn representations by pulling similar data points (positive pairs) closer in latent space while pushing dissimilar points (negative pairs) apart [26], typically optimized using loss functions like Noise Contrastive Estimation (NCE) or InfoNCE [27]. A foundational work in this area, SimCLR [28], demonstrated that contrastive learning could achieve state-of-the-art performance in representation learning by augmenting data to create positive pairs. This method has since been applied across tasks such as image classification, NLP, and time series analysis. Recent advancements, like MoCo [29], introduced a momentum encoder to handle large numbers of negative samples, improving scalability efficiently. BYOL [30] further innovated by eliminating negative pairs while maintaining competitive performance.

3 WMN-CDA

In this section, we first formulate the industrial MWN configuration problem as a contrastive domain adaptation problem, followed by an overview of WMN-CDA and a detailed description on the training algorithm. We then provide an in-depth discussion of our Network Configuration and Simulation-to-Reality contrastive losses.

3.1 Problem Formulation

3.1.1 Industrial WMN Configuration. The primary task in industrial WMN configurations is to select the WMN configuration that allows the network to meet the performance requirements. The parameter selection should be as accurate as possible with minimal physical data collection overhead.

The network configuration of an industrial WMN involves three adjustable parameters, including [12]:

- **PRR threshold for link selection** (R) – the Packet Reception Ratio threshold used for selecting links;
- **Number of channels used in the network** (C) – the total number of communication channels utilized;
- **Number of transmission attempts per packet** (A) – the number of times a packet is transmitted before being considered dropped.

The network performance consists of three key metrics, including:

- **End-to-end latency** (L): The total time taken for a packet to traverse from the source to the destination.
- **Battery lifetime** (B): The total energy consumption by network devices.
- **End-to-end reliability** (E): Measured using the Packet Delivery Ratio (PDR), which represents the ratio of successfully delivered packets to the total number of packets transmitted.

Industrial WMN configurations present two key challenges: (1) training an effective Deep Neural Network (DNN) model that utilizes network performance to predict network configurations, and (2) improving the accuracy of the DNN model while minimizing the reliance on physical data.

3.1.2 Industrial WMN Configuration from a Supervised Contrastive Learning Perspective. The first challenge in configuring an industrial WMN is selecting good configuration parameters $\mathbf{y} = \{\mathbf{R}, \mathbf{C}, \mathbf{A}\}$ that satisfy the network's performance requirements $\mathbf{x} = \{\mathbf{L}, \mathbf{B}, \mathbf{E}\}$, as dictated by the application. Here, $\mathbf{x}$ represents the performance requirements, which are the concatenation of metrics such as $\mathbf{L}, \mathbf{B}, \mathbf{E}$. Likewise, $\mathbf{y}$ denotes the network configuration parameters, which are the concatenation of variables $\mathbf{R}, \mathbf{C}, \mathbf{A}$. The objective is to learn a nonlinear mapping $f_\theta(\cdot) : \mathbf{x} \rightarrow \mathbf{y}$, where θ are the model parameters learned from labeled data.

From Supervised contrastive learning perspective, given a dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$, where each sample (network performance requirement) $\mathbf{x}_i \in \mathcal{X}$ is associated with a corresponding label (network configuration) $\mathbf{y}_i$, we define:

- **Positive samples** $\{\mathbf{x}_i^+\}$, which share the same label $\mathbf{y}_i$ as the anchor sample $\mathbf{x}_i$ and are considered similar.
- **Negative samples** $\{\mathbf{x}_i^-\}$, which have different labels from $\mathbf{x}_i$ and are considered dissimilar.

The goal of supervised contrastive learning is to learn a representation function $f(\mathbf{x}; \theta)$, parameterized by θ, that maps data samples into a latent space where samples sharing the same label are close together, while those with different labels are pushed farther apart.

This is achieved by minimizing the supervised contrastive loss. Let $\mathbf{z}_i = f(\mathbf{x}_i)$ represent the embedding of the sample $\mathbf{x}_i$ in the latent space, where $f(\cdot)$ is the representation function. The supervised contrastive loss for a single sample $\mathbf{x}_i$ can be expressed as:

$$\mathcal{L}_i = -\frac{1}{|P_i|} \sum_{\mathbf{x}_i^+ \in P_i} \log \frac{\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_i^+)/\tau\right)}{\sum_{\mathbf{x}_j \in P_i \cup N_i} \exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau\right)}, \tag{1}$$

where:

- $P(i)$ is the set of positive samples that share the same label as $\mathbf{x}_i$,
- $N(i)$ is the set of negative samples with different labels from $\mathbf{x}_i$,
- $\text{sim}(\mathbf{u}, \mathbf{v})$ denotes the similarity between two representations, typically computed using cosine similarity:

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|},$$

- τ is a temperature hyperparameter controlling the concentration of the distribution.

The overall objective of supervised contrastive learning is to solve the following optimization problem:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i,$$

where the goal is to learn the model parameters θ that minimize the supervised contrastive loss across the dataset. This process results in a representation function $f(\mathbf{x}; \theta)$ that tightly clusters samples with the same label in the latent space while distancing samples with different labels. Such representations are more robust and discriminative, enabling the model to better tackle the challenges of network configuration.

3.1.3 Industrial WMN Configuration Domain Adpation at a Granular Scale. The second key challenge in industrial WMN configurations is to improve the accuracy of the DNN model while minimizing the reliance on physical data, which can be formulated as a domain adaptation problem. Domain adaptation aims to reduce the domain discrepancy by learning domain-invariant features or by facilitating knowledge transfer from the source domain to the target domain by using a large amount of source domain (simulation) data and a few target domain (physical) data. In the industrial WMN configuration scenario, the source domain refers to the simulation and the target domain refers to the reality. Let the **source domain** be defined as $\mathcal{D}_S = \{(\mathbf{x}_i^S, y_i^S)\}_{i=1}^{n_S}$, where $\mathbf{x}_i^S \in \mathcal{X}_S$ are network performance requirements and $y_i^S \in \mathcal{Y}_S$ are corresponding network configurations in simulatioin, and let the **target domain** be $\mathcal{D}_T = \{\mathbf{x}_i^T\}_{i=1}^{n_T}$, where $\mathbf{x}_i^T \in \mathcal{X}_T$ are network performance requirements in reality.

In domain adaptation, the data distributions of the source and target domains differ, i.e., $P(\mathbf{x}^S) \neq P(\mathbf{x}^T)$. The goal is to learn a function $f(\mathbf{x})$ that performs well to the target domain $\mathcal{D}_T$ despite this domain shift. Specifically, we aim to minimize target domain prediction uncertainty $\mathcal{R}_T(f)$, defined as:

$$\mathcal{R}_T(f) = \mathbb{E}_{(\mathbf{x},y)\sim P(\mathbf{x}^T,y^T)}[\ell(f(\mathbf{x}),y)] \tag{2}$$

where ℓ is the loss function. Since the target domain lacks sufficient physical data, the challenge is to leverage labeled source domain (simulation) data and a small amount of target domain data to reduce uncertainty in the target domain.

We reformulate the domain adaptation problem from a domain scale to a granular network configuration scale. In the network configuration level domain adaptation, the data distributions of the source and target domains differ not only in their overall marginal distributions $P(\mathbf{x}^S)$ and $P(\mathbf{x}^T)$, but also in their configuration-specific distributions. We reformulate the problem to learn a function $f(\mathbf{x})$ that performs well to the target domain $\mathcal{D}_T$ by aligning the network configuration-level distributions across domains. Specifically, we aim to minimize the prediction uncertainty for each network configuration in the target domain, $\mathcal{R}_T^{\text{config}}(f)$, defined as:

$$\mathcal{R}_T^{\text{config}}(f) = \mathbb{E}_{(\mathbf{x},y)\sim P(\mathbf{x}^T,y^T|\,y=\text{config})}[\ell(f(\mathbf{x}),y)] \tag{3}$$

where ℓ is a loss function, and $f(\mathbf{x})$ is the prediction function for network configurations. This approach leverages both labeled source domain data $\mathcal{D}_S$ and a few labeled target domain samples for each network configuration to reduce uncertainty and improve adaptation at a more granular, configuration-specific level.

3.2 Overview

Recent advancements in domain adaptation for industrial WMNs have concentrated on mitigating the effects of domain shift by leveraging sophisticated feature alignment techniques. The core innovation of WMN-CDA lies in its dual application of Network Configuration and Simulation-to-Reality contrastive learning, enabling the model to efficiently adapt between domains with discriminative latent space representations at a granular scale. This not only enhances feature alignment but also improves the prediction capacity of the model by maximizing the mutual information between the source and target domains. Figure 1 shows the overall design of WMN-CDA. It has the teacher model trained on the source domain (simulation) data and the student model trained on the target domain (physical deployment) data with the classification loss. Also, a (Maximum Mean Discrepancy) MMD loss is applied to transfer knowledge from the source domain to the target domain. The Network Configuration Loss is used to learn the network requirement representations under different network configurations within the same domain and the Simulation-to-Reality Loss is used to learn the the network requirement representations under different network configurations across the different domains. The Network Configuration loss can help the model to learn better representations by contrasting network requirement differences under different network configurations. The Simulation-to-Reality loss can help the model to transfer knowledge from the source domain (simulation) to the target domain (physical deployment) by considering the simulation-to-reality gap difference under different network configurations.

3.3 WMN-CDA Training

The training process for the proposed *WMN-CDA* model follows a teacher-student framework, where knowledge is transferred from a teacher model trained in the source domain (simulation) to a student model in the target domain (physical deployment). The algorithm is designed to optimize the student model to predict wireless network configurations in reality with minimal physical data. As is shown in Algorithm 1, the training procedure consists of the following steps:

The first step involves pre-training the teacher model on the source domain dataset D_S. The teacher model is trained by minimizing the classification loss $\mathcal{L}_{\text{class}}^S$ over the labeled data from

Algorithm 1 Training Procedure for WMN-CDA

Require: Source domain dataset D_S, target domain dataset D_T
Require: Teacher model T, Student model S
Require: Hyperparameters: learning rate η, Network Configuration loss factor α, Simulation-to-Reality loss factor β, MMD loss factor λ
Ensure: Optimized student model S^*

Step 1: Pre-train Teacher Model on Source Domain
Initialize the teacher model T parameters randomly.
Train T using source domain dataset D_S by minimizing the source domain classification loss $\mathcal{L}_{\text{class}}^S$.
Save the learned teacher model parameters θ_T^*.
Step 2: Initialize and Transfer Knowledge to Student Model
Initialize the student model S parameters from the teacher model T.
Use the teacher model to generate labeled configuration representations from D_S, and transfer them to the target domain.
Step 3: Joint Training of Teacher-Student Model
while training not converged **do**
    Sample mini-batch of source domain data $x_S \in D_S$ and target domain data $x_T \in D_T$.
    Step 3.1: Compute Losses
    Compute the source domain classification loss $\mathcal{L}_{\text{class}}^S$ for teacher model T.
    Compute the target domain classification loss $\mathcal{L}_{\text{class}}^T$ for student model S.
    Compute the MMD loss $\mathcal{L}_{\text{MMD}}$ to align the feature distributions between the source and target domains.
    Compute the Network Configuration contrastive loss $\mathcal{L}_{\text{NC}}$ for both domains to ensure consistency within the domains.
    Compute the Simulation-to-Reality contrastive loss $\mathcal{L}_{\text{SR}}$ to align similar representations between the source and target domains.
    Step 3.2: Compute Total Loss
    Compute the total loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{class}}^S + \mathcal{L}_{\text{class}}^T + \lambda\mathcal{L}_{\text{MMD}} + \alpha\mathcal{L}_{\text{NC}} + \beta\mathcal{L}_{\text{SR}}$$

    Step 3.3: Update Model Parameters
    Update teacher model T and student model S parameters using gradient descent:

$$\theta_T \leftarrow \theta_T - \eta\nabla_{\theta_T}\mathcal{L}_{\text{total}}, \quad \theta_S \leftarrow \theta_S - \eta\nabla_{\theta_S}\mathcal{L}_{\text{total}}$$

end while
Step 4: Save the Optimized Student Model
Save the optimized student model S^*.

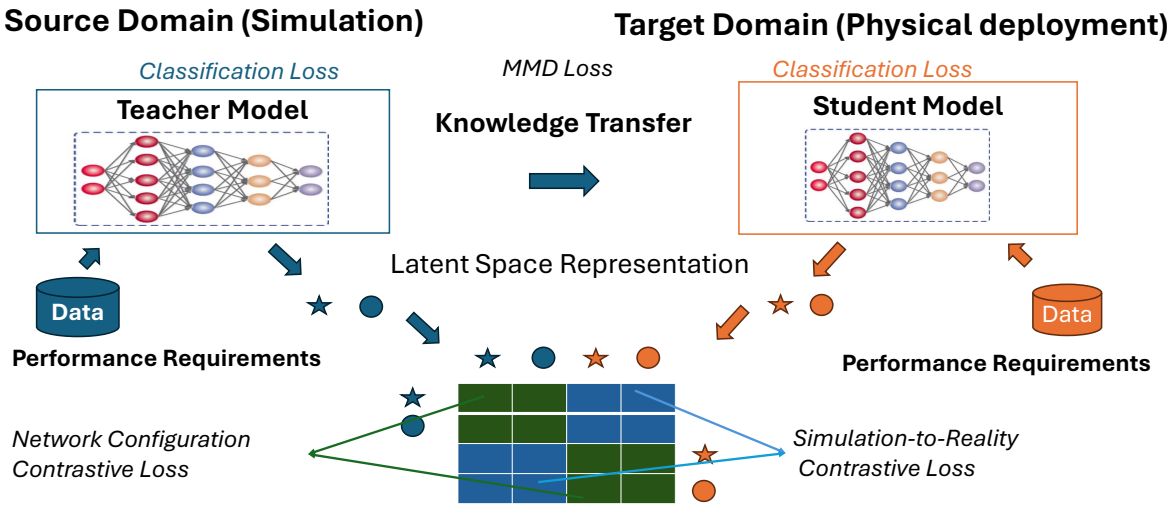

Figure 1: Architecture of WMN-CDA. The star and circle in the figure represent different representations of network performance requirements. The blue ones are the network requirements representations from the source domain (simulation) and The orange ones are the network requirements representations from the target domain (reality). By pushing the network requirements representations under similar network configurations closer and pushing away network requirements representations under different network configurations, WMN-CDA can better capture the features under different network configurations and transfer knowledge at the granular scale (network configuration scale instead of domain scale).

the source domain. Once trained, the parameters of the teacher model θ_T^* are saved for further use in transferring knowledge to the student model.

After pre-training the teacher model, the student model is initialized with the parameters of the teacher model. The teacher model generates labeled configuration representations from the source domain data, which are then transferred to the student model for use in the target domain. This knowledge transfer allows the student model to perform well in the target domain with minimal labeled data.

Then the teacher and student models are trained jointly by minimizing a total loss function that incorporates multiple components. During each iteration, mini-batches of data are sampled from both the source domain $x_S \in D_S$ and the target domain $x_T \in D_T$.

The following losses are computed:

- **Source domain classification loss** $\mathcal{L}_{\text{class}}^S$ for the teacher model.
- **Target domain classification loss** $\mathcal{L}_{\text{class}}^T$ for the student model.
- **MMD loss** $\mathcal{L}_{\text{MMD}}$, which aligns the feature distributions between the source and target domains.
- **Network Configuration contrastive loss** $\mathcal{L}_{\text{NC}}$, which help the model to learn discriminative representations hat capture relevant features and similarities within each domain.
- **Simulation-to-Reality contrastive loss** $\mathcal{L}_{\text{SR}}$, which discriminative representations hat capture relevant features and similarities across the source and target domains.

The total loss is computed as a weighted combination of these components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{class}}^S + \mathcal{L}_{\text{class}}^T + \lambda \mathcal{L}_{\text{MMD}} + \alpha \mathcal{L}_{\text{NC}} + \beta \mathcal{L}_{\text{SR}}, \tag{4}$$

where λ, α, and β are hyperparameters controlling the contributions of the MMD, Network Configuration, and Simulation-to-Reality losses, respectively.

Both the teacher and student models are updated using gradient descent to minimize the total loss and optimize their respective parameters.

Once the joint training process converges, i.e., when the total loss reaches a minimum, the optimized student model S^* is saved. This model is now capable of making accurate predictions in the target domain (reality) by leveraging the knowledge transferred from the source domain (simulation).

The WMN-CDA training procedure effectively allows the model to bridge the simulation-to-reality gap at a granular scale by ensuring consistent feature representations under different network configurations across different domains. The Network Configuration and Simulation-to-Reality contrastive losses, along with the MMD loss, ensure robust feature alignment within and across domains. This training method enables the student model to learn from the source domain and apply the learned configurations to the target domain, making WMN-CDA highly practical for real-world network configuration tasks with limited labeled physical data.

3.4 Network Configuration Contrastive Loss

The Network Configuration Contrastive Loss is designed to improve the consistency of representations within each domain (either

source or target), ensuring that samples under the same network configuration within the same domain are grouped together in the latent space. Given that both the source and target domains may contain diverse data points with varied characteristics, it is crucial to maintain the internal structure of each domain during the knowledge transfer process.

For each anchor sample $\mathbf{x}_i$ from either the source or target domain, we identify a positive sample $\mathbf{x}_i^+$ are the network requirements representations under same network configuration from the same domain. Simultaneously, we select negative samples $\mathbf{x}_i^-$ from dissimilar network configurations within the same domain. The Network Configuration Contrastive Loss encourages the anchor sample $\mathbf{x}_i$ and its positive counterpart $\mathbf{x}_i^+$ (network requirements representations under same network configuration) to be closer in the latent space while pushing the anchor away from the negative samples(network requirements representations different network configuration) $\mathbf{x}_i^-$.

The Network Configuration contrastive loss is used to align representations within the same domain. It encourages similarity between positive pairs (anchor and augmented views) and pushes apart dissimilar pairs (anchor and negative samples). Let $\mathbf{z}_i = f(\mathbf{x}_i)$ represent the embedding of sample $\mathbf{x}_i$. The loss function is formalized as:

$$\mathcal{L}_{\text{NC}} = -\log \frac{\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_i^+)/\tau\right)}{\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_i^+)/\tau\right) + \sum_{j=1}^{k} \exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_j^-)/\tau\right)}, \quad (5)$$

where τ is a temperature hyperparameter, and $\text{sim}(\cdot,\cdot)$ denotes the cosine similarity. This alignment ensures consistency within the domain, reducing Network Configuration variability and enhancing the robustness of the knowledge transfer process.

3.5 Simulation-to-Reality Contrastive Loss

While Network Configuration contrastive loss handles internal domain consistency, the Simulation-to-Reality Contrastive Loss is designed to align representations between the source (simulation) and target domains (reality). The key challenge in domain adaptation is the discrepancy between the data distributions of the source and target domains. The Simulation-to-Reality contrastive loss aims to reduce this domain shift by ensuring that corresponding configurations in the source and target domains are mapped to similar representations in the latent space, promoting cross-domain consistency.

For a given anchor sample $\mathbf{x}_i^S$ from the source domain and its positive counterpart $\mathbf{x}_i^T$ from the target domain (network requirements representations that are from network configurations but collected from different domains), the model is encouraged to bring their representations closer together. Meanwhile, samples under same network configuration that are dissimilar between the source and target domains are pushed apart. This is achieved through a contrastive loss function similar to the Network Configuration case but applied between domains.

The Simulation-to-Reality Contrastive Loss is defined to align the feature representations between the source and target domains. For a given source sample $\mathbf{x}_i^S$ and its positive counterpart from the target domain $\mathbf{x}_i^T$, the loss encourages their similarity while pushing away dissimilar samples. Let $\mathbf{z}_i^S = f(\mathbf{x}_i^S)$ and $\mathbf{z}_i^T = f(\mathbf{x}_i^T)$ represent the embeddings of the source and target samples, respectively. The loss is formulated as:

$$\mathcal{L}_{\text{SR}} = -\log \frac{\exp\left(\text{sim}(\mathbf{z}_i^S, \mathbf{z}_i^T)/\tau\right)}{\exp\left(\text{sim}(\mathbf{z}_i^S, \mathbf{z}_i^T)/\tau\right) + \sum_{j=1}^{k} \exp\left(\text{sim}(\mathbf{z}_i^S, \mathbf{z}_j^T)/\tau\right)}, \quad (6)$$

where $\text{sim}(\cdot,\cdot)$ denotes the similarity function, and τ is a temperature parameter that controls the scaling of similarities. The first term aligns the positive source-target pair, and the second ensures dissimilar pairs are separated.

By minimizing this Simulation-to-Reality loss, the model effectively reduces the domain shift between the source and target domains, allowing for a more effective transfer of knowledge from the well-labeled source domain (simulation) to the sparsely-labeled target domain (physical deployment). This combination of Simulation-to-Reality and Network Configuration contrastive learning ensures that the model is well-adapted to handle both local domain variability and cross-domain shifts, improving overall performance in WMN domain adaptation tasks.

4 Evaluation

In this section, we first introduce our experimental setup and the dataset used for our experiments. We then present the performance of WMN-CDA when we use four different simulators as well as our study on the effects of two key parameters on WMN-CDA's performance. Finally, we present the performance of WMN-CDA when different amounts of physical data are used to train the network configuration model.

4.1 Experimental Setup and Dataset

We implement WMN-CDA and our baseline TS-DA, the domain adaptation method developed by Shi el al. [6], using PyTorch [31]. We train WMN-CDA with the Adam optimizer [32] and run all experiments on a single NVIDIA A100 GPU with 80GB of memory. The physical data used in our experiment is collected from the WirelessHART network that runs on the testbed built by Shi et al. [6]. The testbed consists of 50 TelosB motes [33]. The data contains 88 distinct network configurations with three network parameters: $R \in \{0.7, 0.71, 0.72, \ldots, 0.9\}$, $C \in \{1, 2, 3, \ldots, 8\}$, and $A \in \{1, 2, 3\}$. Each network configuration corresponds to three performance metrics end-to-end latency (L), battery lifetime (B), and end-to-end reliability (E), In total, we have 6,600 data traces (88 network configurations × 75 traces per configuration). We implement the same WirelessHART network in four wireless simulators including NS3 [34], Cooja [35], Tossim [36], and OMNet [37]. In total, we collect 6,600 data traces (88 network configurations × 75 traces per configuration) from each simulator.

4.2 Performance of WMN-CDA

We first measure the performance of WMN-CDA on selecting configurations to achieve good network performance. The experiments are performed with 75 shots (each shot has one data sample under each of 88 network configurations) of simulation data and 5 shots of

Table 1: Prediction performance when using different simulators with various training methods. "Imp." represents the improvement of WMN-CDA compared to our baseline "TS-DA".

Simulator	NS3	Cooja	Tossim	OMNet
TS-DA	70.1%	69.8%	71.1%	60.9%
WSN-CD	**75.9**%	**74.1**%	**75.6**%	**68.0**%
Imp.	+5.8%	+4.3%	+4.5%	+7.1%

physical data. We repeat the experiments with the simulation data collected from different simulators. Table 1 presents the prediction performance of WMN-CDA and our baseline TS-DAs. As Table 1 lists, WMN-CDA consistently outperforms TS-DA. For instance, when we use the simulation data generated by the NS3 simulator, WMN-CDA achieves a prediction accuracy of 75.9%, a notable improvement of +5.8% over TS-DA. When we use the Cooja simulator, WMN-CDA achieves a prediction accuracy of 74.1%, a improvement of +4.3% over TS-DA. The most significant improvement is observed when we use OMNet, where WMN-CDA reaches 68.0% accuracy, representing a substantial +7.1% increase over TS-DA. The results highlight the effectiveness of WMN-CDA in selecting good network configurations across various simulators. The performance gains, particularly in simulators like NS3 and OMNet, underscore the robustness of the WMN-CDA method in addressing domain shifts and enhancing generalization across diverse network settings. This is achieved by learning discriminative feature representations at a more granular level, specifically under varying network configurations.

4.3 Effects of Key Parameters

Table 2: Prediction performance when using different Network Configuration contrastive loss factor α ($\beta = 1$).

α	NS3	Cooja	Tossim	OMNet
0.1	75.1%	**71.9%**	74.7%	52.5%
0.3	**75.9%**	71.6%	74.5%	**68.0%**
0.5	**75.9%**	**71.9%**	74.5%	60.1%
0.7	75.1%	**71.9%**	74.3%	56.7%
0.9	75.5%	71.4%	**75.0%**	60.0%

We then evaluate the effects of two key parameters (i.e., Network Configuration contrastive loss factor α and Simulation-to-Reality contrastive loss β) on WMN-CDA's performance. Table 2 presents the prediction performance of WMN-CDA under different simulators when we vary the Network Configuration contrastive loss factor (α) and keep using the same Simulation-to-Reality loss factor ($\beta = 1$). For instance, under NS3, the best performance is achieved at $\alpha = 0.3$ and $\alpha = 0.5$ with an accuracy of 75.9%. Under Cooja, the model performs best at $\alpha = 0.1$, $\alpha = 0.5$, and $\alpha = 0.7$ with 71.9% accuracy. Under Tossim, the highest accuracy of 75.0% is observed at $\alpha = 0.9$. Under OMNet, the performance peaks at $\alpha = 0.3$ with 68.0% accuracy. The results suggest that a moderate value of $\alpha = 0.3$ tends to yield better performance. As α goes beyond this range, the performance decreases in general, particularly under OMNet, where a $\alpha = 0.1$ results in the lowest accuracy of 52.5%. Table 3 shows the prediction performance of WMN-CDA under different simulators when we vary the Simulation-to-Reality contrastive loss factor (β) and keep using the same Network Configuration loss factor ($\alpha = 0.3$) constant. For example, under NS3, the best performance is achieved at $\beta = 1$, with an accuracy of 75.9%. Under Cooja, the model performs best at $\beta = 0.3$ with 74.1% accuracy. Under Tossim, the highest accuracy of 75.6% is observed at $\beta = 300$. Under OMNet, the performance peaks at $\beta = 1$ with 68.0% accuracy. The results suggest that a moderate value of $\beta = 1$ tends to yield better performance. As β goes beyond this range, performance generally decreases, particularly in OMNet, where a high $\beta = 0.3$ results in the lowest accuracy of 61.1%.

Table 3: Prediction performance when using different Simulation-to-Reality contrastive loss factor β ($\alpha = 0.3$).

β	NS3	Cooja	Tossim	OMNet
0.03	74.7%	70.9%	72.9%	67.2%
0.3	74.9%	**74.1%**	74.5%	61.1%
1	**75.9%**	71.6%	74.5%	**68.0**%
3	75.0%	71.5%	74.9%	64.8%
30	73.9%	70.4%	75.0%	62.9%
300	74.8%	71.1%	**75.6%**	61.3%

4.4 Performance When Using Different Amounts of Physical Training Data

Table 4: Prediction performance across different simulators with different shots of physical data.

Shot	NS3	Cooja	Tossim	OMNet
1	55.6%	62.7%	56.7%	47.7%
2	64.1%	64.1%	63.8%	47.3%
3	70.1%	67.9%	69.9%	51.4%
4	71.3%	70.2%	71.7%	53.5%
5	75.9%	74.1%	75.6%	68.0%

Finally, we examine the performance of WMN-CDA when we use different amounts of physical data to train the network configuration model. To investigate the effects of the training data. Table 4 lists the prediction performance of WMN-CDA when we vary the amount of physical data used for training (ranging from one shot to five shots). The results show that the performance of WMN-CDA improves as more physical data is used for training. For example, under NS3, the prediction accuracy starts at 55.6% with 1 shot and gradually increases to 75.9% with five shots. Similarly, in Cooja, the model's accuracy improves from 62.7% with one shot to 74.1% with five shots. Tossim shows a comparable trend, starting at 56.7% with one shot and reaching 75.6% with five shots. The most significant improvement is observed in OMNet, where the accuracy increases from 47.7% with one shot to 68.3% with

five shots. For most simulators, it is notable that the performance gain diminishes after using three shots of physical data, indicating the strong adaptation capability of WMN-CDA in transferring the network configuration knowledge under few-shot scenarios. This effectiveness can be attributed to the contrastive loss employed in WMN-CDA, which effectively addresses the issue of domain misalignment. However, in cases such as OMNet, where the source domain (simulation) data contains fewer domain-invariant features, additional physical data is required to achieve better domain adaptation performance. WMN-CDA demonstrates its adaptability by rapidly improving accuracy from 53.5% to 68.0%, with the addition of a single shot of data.

5 Conclusions

In this paper, we introduce WMN-CDA, a novel framework that enables effective and efficient domain adaptation for WMN configurations. By employing contrastive learning, WMN-CDA efficiently transfers the network configuration knowledge from simulations to a physical deployment, reduces the dependence on the labeled data, and leverages the Network Configuration and Simulation-to-Reality contrastive losses to ensure robust feature alignment. Experimental results show that WMN-CDA improves prediction accuracy when using the simulation data generated by various simulators compared to the baseline.

Acknowledgment

This work was supported in part by the National Science Foundation under grant CNS-2150010.

References

[1] B. Milic, S. Brack, and R. Naumann, "Hierarchical configuration system for wireless mesh networks in manufacturing industry," in *IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 2014, pp. 1–8.

[2] H.-C. Lee and H.-H. Lin, "Design and evaluation of an open-source wireless mesh networking module for environmental monitoring," *IEEE Sensors Journal*, vol. 16, no. 7, pp. 2162–2171, 2015.

[3] C. Silva, Y. Oliveira, C. Celes, R. Braga, and C. Oliveira, "Performance evaluation of wireless mesh networks in smart cities scenarios," in *Proceedings of the Euro American Conference on Telematics and Information Systems*, 2018, pp. 1–7.

[4] I. F. Akyildiz, X. Wang, and W. Wang, "A survey on wireless mesh networks," *IEEE communications magazine*, vol. 43, no. 9, pp. S23–S30, 2005.

[5] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-time wireless sensor-actuator networks for industrial cyber-physical systems," *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems*, vol. 104, no. 5, pp. 1013–1024, 2016.

[6] J. Shi, M. Sha, and X. Peng, "Adapting wireless mesh network configuration from simulation to reality via deep learning based domain adaptation," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021.

[7] X. Cheng, M. Sha, and D. Chen, "Configuring industrial wireless mesh networks via multi-source domain adaptation," in *ACM International Conference on Embedded Wireless Systems and Networks (EWSN)*. EWSN, 2024.

[8] J. Shi, A. Ma, X. Cheng, M. Sha, and X. Peng, "Adapting wireless network configuration from simulation to reality via deep learning based domain adaptation," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 1983–1998, 2024.

[9] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," *Advances in neural information processing systems*, vol. 19, 2006.

[10] WirelessHART, "Wirelesshart networks: A reliable real-time communication technology for industrial applications," *Wireless Communications and Networking Conference (WCNC)*, 2012.

[11] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE transactions on cognitive communications and networking*, vol. 4, no. 2, pp. 257–265, 2018.

[12] J. Shi and M. Sha, "Parameter self-configuration and self-adaptation in industrial wireless sensor-actuator networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 658–666.

[13] J. Shi and etc., "Parameter self-adaptation for industrial wireless sensor-actuator networks," *ACM Transactions on Internet Technology (TOIT)*, vol. 20, no. 3, pp. 1–28, 2020.

[14] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Journal on selected areas in communications*, vol. 24, no. 8, pp. 1426–1438, 2006.

[15] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter, "A survey on networking games in telecommunications," *Computers & Operations Research*, vol. 33, no. 2, pp. 286–311, 2006.

[16] K.-L. A. Yau, P. Komisarczuk, and P. D. Teal, "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 253–267, 2012.

[17] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE journal on selected areas in communications*, vol. 37, no. 6, pp. 1277–1290, 2019.

[18] L. Huang, S. Bi, and Y. J. Zhang, "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," *IEEE TMC*, vol. Early Access, 2020.

[19] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

[20] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.

[21] A. Ma, J. C. T. Rodriguez, and M. Sha, "Enabling reliable environmental sensing with lora, energy harvesting, and domain adaptation," in *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2024, pp. 1–9.

[22] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[23] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[24] X. Zheng, T. Wang, W. Cheng, A. Ma, H. Chen, M. Sha, and D. Luo, "Parametric augmentation for time series contrastive learning," in *International Conference on Learning Representations (ICLR)*, 2024.

[25] X. Zheng and T. Wang, "Auto tcl: Automated time series contrastive learning with adaptive augmentations," in *Proc. 32nd Int. Joint Conf. Artif. Intell.(IJCAI)*, 2023.

[26] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*. IEEE, 2006.

[27] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[28] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning (ICML)*, 2020.

[29] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020.

[30] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[32] P. K. Diederik, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2015.

[33] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*. IEEE, 2005, pp. 364–369.

[34] "NS-3 Shadowing Model." [Online]. Available: https://www.nsnam.org/docs/release/3.10/doxygen/classns3_1_1_shadowing_loss_model.html

[35] "Source Code of Cooja," https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja, Cooja.

[36] "Source Code of TOSSIM," https://github.com/tinyos/tinyos-main/tree/master/tos/lib/tossim, TOSSIM.

[37] "Source Code of OMNeT," https://github.com/omnetpp/omnetpp, OMNeT.