# Enhancing Multivariate Time-Series Domain Adaptation via Contrastive Frequency Graph Discovery and Language-Guided Adversary Alignment

**Haoren Guo**[1], **Haiyue Zhu**[2*], **Jiahui Wang**[1], **Vadakkepat Prahlad**[1], **Weng Khuen Ho**[1], **Tong Heng Lee**[1]

[1]National University of Singapore
[2]Agency for Science, Technology and Research (A*STAR)
{haorenguo_06, wjiahui}@u.nus.edu, {prahlad, wk.ho, eleleeth}@nus.edu.sg, zhu_haiyue@simtech.a-star.edu.sg

## Abstract

Unsupervised domain adaptation (UDA) is a machine learning approach designed to minimize reliance on labeled data by aligning features between a labeled source domain and an unlabeled target domain, thereby reducing feature discrepancies, which is efficient for multivariate time series (MTS) prediction. However, most MTS UDA methods focus solely on aligning intra-series temporal features, overlooking the valuable information in inter-series dependencies. Research has highlighted that analyzing decomposed frequency dependencies in time series can reveal significant trends, noise patterns, and intricate temporal details. To address these unexplored frequency dependencies, we introduce the **F**requency **G**raph **D**iscovery Module (**FGD**), which uncovers and aligns shared frequency information and correlations across domains. Additionally, we propose a **F**requency-**C**ontextual **C**ontrastive **L**earning (**FCCL**) framework to better capture and align frequency-contextual representations in multivariate time series, ensuring the extraction of label-invariant information for prediction. Furthermore, considering existing models overlooking the valuable and abundant information outside source and target dataset, we enhance the MTS UDA prediction model with a **L**anguage-guided **A**dversary **A**lignment (**LAA**) module, which leverages the advancement and capabilities of Large Language Models (LLMs) to get text-encoded labeled embeddings and align the classification features, thereby improving prediction accuracy. Our model achieves state-of-the-art results on three public multivariate time-series datasets for unsupervised domain adaptation, as demonstrated by empirical results.

## 1 Introduction

Multivariate Time Series (MTS) data are extensively applied and researched across various fields. The advancement of data-driven models, particularly deep learning methods, has significantly improved performance in MTS-related tasks due to their ability to model latent dependencies within data (Ragab et al. 2023). However, these methods often require a large amount of labeled data for training, which can be costly and sometimes even impossible, such as forcing every single patient to record and submit their daily activity sensor data and label with their specific activities and training individual models for every particular patient. Targeting this bottleneck, Unsupervised Domain Adaptation (UDA) methods have emerged. These methods transfer knowledge from a labeled source domain to an unlabeled target domain without using labels from the target domain (Wang et al. 2023).

Most of the current UDA methods attempt to reduce domain discrepancy by learning domain-invariant features, typically through metric-based, such as Recurrent Neural Networks (RNNs) (Purushotham et al. 2022a) and Long Short-Term Memory (LSTM) (da Costa et al. 2020) or adversarial-based approaches, such as domain adversarial neural network (DANN) (Ganin et al. 2016) and CALDA (Wilson, Doppa, and Cook 2023). These methods have proven effective in reducing label dependency, particularly in tasks involving MTS data. Additionally, transfer learning has emerged as another prominent approach in time series analysis, integrating Contrastive Learning (CL) to capture contextual representations for downstream tasks (Eldele et al. 2021; Tonekaboni, Eytan, and Goldenberg 2021). In the UDA task, CLUDA (Ozyurt, Feuerriegel, and Zhang 2022) has shown promise by leveraging CL to align contextual representations across source and target domains.

Despite advancements, existing MTS UDA methods primarily align features within the intra-series temporal signal space, neglecting inter-series dependencies and multi-frequency information. Domain shifts in time series can manifest as changes in temporal and frequency characteristics, where frequency features are more robust to small shifts and noise, offering better trend and turbulence extraction (He et al. 2023; Guo et al. 2024). Motivated by this, we hypothesize that frequency features and inter-frequency correlations between source and target domains should exhibit similarity. To leverage this, we propose the Frequency Graph Discovery Module (FGD), which identifies inherent frequency relationships and aligns features at the frequency graph level. Complementing this, we introduce the Frequency-Contextual Contrastive Learning (FCCL) framework to align frequency-contextual representations from augmented time series, extracting label-invariant information. Together with adversarial training to reduce domain discrepancies, our proposed model, ConFGD, integrates FGD and FCCL to enhance prediction accuracy.

Existing models extract features only from source and target data, overlooking abundant information outside. In line with the saying, “those who wanted to learn would seek out a

[*]Corresponding author

teacher, one who could propagate the doctrine, impart professional knowledge, and resolve doubt", we see large language models (LLMs) as this teacher, offering vast knowledge from books, articles, and other sources. The advancement and capabilities of LLMs inspired us to explore extending their potential to guide our MTS UDA task. Since dataset labels can serve as inherent prompts for generating LLM embeddings, we designed a time-efficient and computationally lightweight adversary module, Language-guided Adversary Alignment (LAA) module, that can be added not only to our ConFGD model but also to existing UDA models to improve the prediction ability and the upgraded model is denoted as ConFGD+.

We evaluate our method on the benchmark real-world MTS dataset HAR (Anguita et al. 2013), WISDM (Kwapisz, Weiss, and Moore 2011) and HHAR (Stisen et al. 2015), and achieve state-of-the-art (SOTA) performances. We further conduct experiments by introducing the LAA module to ConFGD and other existing models, such as CLUDA, to confirm its effectiveness and superior performance.

**Contributions:**

1. We propose and develop a novel Frequency Graph Discovery Module (FGD) to discover and align the inherent inter-series frequency channels information and relationship for Unsupervised Domain Adaptation (UDA) of time series.
2. To incorporate with the frequency embeddings, we capture the frequency-contextual representation by the novel design of a brand new Frequency-Contextual Contrastive Learning (FCCL) framework and further enhance the prediction capabilities by extracting the label-invariant information.
3. We are the first to introduce the LLM as a time-efficient and computationally lightweight adversarial language-guided model, which can be incorporated into not only our ConFGD model but also existing UDA models to enhance prediction performance.

## 2 Methodology

### 2.1 Problem Formulation

In this paper, the objective is to perform UDA on the multivariate time series classification tasks. There are two datasets sampled from two different distributions respectively which are given as $\mathcal{D}_{src} = \{(\mathbf{X}^i_{src}, y^i_{src})\}_{i=1}^{N_s}$ and $\mathcal{D}_{trg} = \{\mathbf{X}^i_{trg}\}_{i=1}^{N_t}$. $\mathcal{D}_{src}$ represents the **labeled** source domain dataset with $N_s$ number of samples. $\mathbf{X}^i_{src} = \{\mathbf{x}^{it}_{src}\}_{t=1}^{T} \in \mathbb{R}^{D\times T}$ is a sample of the source domain with $T$ time steps and $D$ sensor observations and $y^i_{src}$ is the label for the given sample. $\mathcal{D}_{trg}$ represents the **unlabeled** target domain dataset with $N_t$ number of samples. Similar as the samples from $\mathcal{D}_{src}$, the sample from the target domain is $\mathbf{X}^i_{trg} = \{\mathbf{x}^{it}_{trg}\}_{t=1}^{T} \in \mathbb{R}^{D\times T}$. In addition, the labels for the target domain are applicable during testing, therefore we specifically define the labeled testing target domain dataset as $\mathcal{D}^{test}_{trg} = \{(\mathbf{X}^i_{test\_trg}, y^i_{test\_trg})\}_{i=1}^{N_t^{test}}$ where $N_t^{test}$ is the number of samples in the test target domain dataset and $\mathbf{X}^i_{test\_trg} = \{\mathbf{x}^{it}_{test\_trg}\}_{t=1}^{T} \in \mathbb{R}^{D\times T}$. The same for both source and target domain samples, $\mathbf{x}^{it} = \{x^{itd}\}_{d=1}^{D} \in \mathbb{R}^D$. Although the source and target domains are drawn from different distributions, each representing distinct marginal distributions, the conditional distributions for both domains are identical. We assume the two domains share the same label space. The main goal of this work is to minimize the distribution shift between the source and target domains and achieve good generalization on the target domain $\mathcal{D}_{trg}$ by exploiting the labeled source domain samples.

### 2.2 Architecture Overview

The framework of our proposed **ConFGD** for multivariate time series unsupervised domain adaptation is shown in Fig. 1. First of all, both augmented time series from both domains are decomposed into multi-frequency level signals by implementing **Discrete Wavelet Transform (DWT)**, and the corresponding features, $\mathbf{H}_{src}$ and $\mathbf{H}_{trg}$, are extracted by the **temporal projection** network respectively. After that, the **frequency graph discovery module (FGD)** comprising the encoder, aggregate module, and decoder, is trained to capture and align the graph, including both edge and node attributes, across various frequency levels. The classification feature $\mathbf{v}^S_{node}$ extracted from $\mathbf{H}_{src}$ by the **graph discovery encoder** is utilized to predict the label $y_{src}$ of time series $\mathbf{X}_{src}$. The **domain discrimination** is trained to distinguish domains by utilizing in the encoder node embeddings $\mathbf{v}^S_{node}$ and $\mathbf{v}^T_{node}$. We arbitrarily labeled the source domain as 0 and the target domain as 1 for the training. To enhance the capture of contextual representations, **frequency-contextual contrastive learning (FCCL)** is employed across each domain through the utilization of a **momentum-updated** temporal projection and encoder. The ConFGD+ introduces an **Language-Guided Alignment (LAA)** module which is shown in Fig. 2 to align the embeddings gotten from label prompts with the classification features $\mathbf{v}^S_{node}$. This is explained in Sec. 2.5.

### 2.3 Frequency Graph Discovery Module

The information from time series signals is equally important in both the time and frequency domain. Signals in the time domain are vulnerable to noise and disruptions, making it challenging to discern trends and detailed information due to their volatility. In contrast, frequency-domain methods transform these signals to emphasize their spectral features, making it easier and more feasible to extract and recognize turbulence and trends. Higher frequency components usually include finer details and generally suggest random variations and noise. The lower frequency components usually offer insights into trend dynamics and more stable dependencies. Therefore, we hypothesize that if the source and target domains share the same label space, the information shared by the frequency channels and the correlation among the multi-frequency level signals should be similar.

**DWT Frequency Decomposed** Fourier Transform and Wavelet Transform are two prevailing methods for transforming signals between the time and frequency domain. Compared with Wavelet Transform, Fourier Transform mostly focuses on the overall dependencies of the time domain such as global seasonal and temporal information (Karlton 2020).

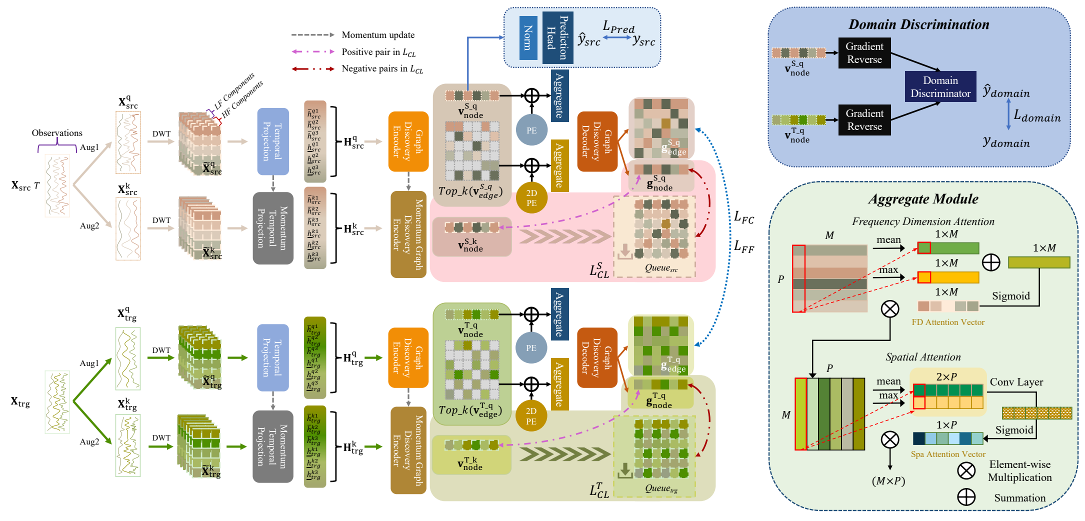


Figure 1: ConFGD Model Architecture (Best view in color). The left side graph is the overall structure. The source and target samples are augmented to be the query $\mathbf{X}^q$ and the key $\mathbf{X}^k$ and are decomposed by the DWT. After that, they are passed to the temporal projection layer and graph discovery encoder to get the encoder node and edge embeddings where the source node embedding $\mathbf{v}^{S_q}_{node}$ is utilized for prediction ($L_{pred}$). Next, the aggregate modules (detailed structure shown in the lower right corner) integrate the query node and edge embeddings respectively, along with their positional encodings. These combined embeddings are then passed to the graph discovery decoder to obtain the frequency correlation graph for $L_{FC}$, $L_{FF}$ and $L_{CL}$. The domain discrimination framework ($L_{domain}$) is in the upper right corner.

Even though the windowed Fourier Transform technique is optimized to capture the local feature, it is restricted by the constant window to manage the input shorter or longer than the window (Gabor 1946). Consequently, the Wavelet Transform is selected for frequency decomposition due to its scaled window, which addresses the limitation of inconsistent input lengths and enables the capture of local properties. More specifically, we employ the Discrete Wavelet Transform (DWT) in feature-wise. The decompose implementation is denoted as $DWT(\cdot)$ which can be illustrated as $\tilde{\mathbf{X}}_{coeff} = DWT(\mathbf{X})$. $\tilde{\mathbf{X}}_{coeff}$ is the concatenate coefficients of all the levels gotten by DWT. There is a number of $S$ sets of coefficients within $\tilde{\mathbf{X}}_{coeff}$. To simply extract the low and high-frequency part of the time series, we multiply each set of coefficients $\tilde{\mathbf{X}}_{coeff}[s]$ with a parameter $\lambda_s$. The low-frequency parameter $\lambda^{lf}_s$ exhibits a small value at high levels, *i.e.*, when s is small, conversely for the high-frequency parameter $\lambda^{hf}_s$. This can be denoted as

$$\tilde{\mathbf{X}}^{lf}_{coeff}[s], \tilde{\mathbf{X}}^{hf}_{coeff}[s] = \lambda^{lf}_s \tilde{\mathbf{X}}_{coeff}[s], \lambda^{hf}_s \tilde{\mathbf{X}}_{coeff}[s]. \quad (1)$$

Then, the filtered high- and low-frequency coefficient sets would be inverse back to the time domain by implementing the Inverse Discrete Wavelet Transform (IDWT),

$$\begin{aligned}\overline{\mathbf{X}} &= Concat(\{IDWT(\tilde{\mathbf{X}}^{hf}_{coeff}[s]\}^{S}_{s=1}),\\ \underline{\mathbf{X}} &= Concat(IDWT(\{\tilde{\mathbf{X}}^{lf}_{coeff}[s])\}^{S}_{s=1}),\\ \tilde{\mathbf{X}} &= Concat(\overline{\mathbf{X}}, \underline{\mathbf{X}}), \{\overline{\mathbf{X}}, \underline{\mathbf{X}}\} \in \mathbb{R}^{D\times T\times S}, \tilde{\mathbf{X}} \in \mathbb{R}^{D\times T\times 2S}\end{aligned} \quad (2)$$

where the *Concat* is a concatenate implementation along the frequency channel and $\overline{\mathbf{X}}$ and $\underline{\mathbf{X}}$ represent the decomposed high-frequency and low-frequency time series for each feature and $\tilde{\mathbf{X}}$ is the integration of the multi-frequency levels. Motivated by SASA (Cai et al. 2021), we allocate independent temporal projection layers for each frequency channel, $\mathbf{H} = Concat(\{T[s](\tilde{\mathbf{X}}[s])\}^{2S}_{s=1})$, where $T[s]$ is the independent temporal projection layer for each channel, and $\mathbf{H} \in \mathbb{R}^{2S\times P}$ is the hidden representation and $P$ is the hidden dimension of each $T[s]$.

**Graph Discovery Module** The objective of the graph discovery module is to find the intrinsic relationship among the frequency channels. The interaction among the frequency channels is modeled as a frequency correlation graph $\mathcal{G}$ which contains the edge and node information. The graph includes self-loop which means each node also points to itself. Inspired by the graph discovery architecture in the VCDN (Li et al. 2020), our Graph Discovery Module utilizes graph neural networks (see Appendix. C) as the encoder and decoder, further integrating an aggregation module between the encoder and decoder to capture the frequency channel and spatial channel information.

To simplify the notation, we let $Q$ represent $2S$ in the previous chapter. The fully connected graph encoder is denoted as $G^{enc}(\cdot)$, and the process can be described as

$$\mathbf{v}_{node}, \mathbf{v}_{edge} = G^{enc}(\mathbf{H}), \mathbf{v}_{node} \in \mathbb{R}^{Q\times P}, \mathbf{v}_{edge} \in \mathbb{R}^{Q^2\times P} \quad (3)$$

where the $\mathbf{v}_{node}, \mathbf{v}_{edge}$ are denoted as the encoder node and edge embedding respectively.

The encoder node embedding $\mathbf{v}_{node}$ is also the classification feature that is utilized for the prediction. We define the prediction head as $Pred(\cdot)$ and the *prediction loss* is expressed as

$$\hat{y}^i_{src} = Pred(\mathbf{v}^{Si}_{node}), \quad L_{pred} = \frac{1}{N_s}\sum_{i=1}^{N_s} L_{ce}(\hat{y}^i_{src}, y^i_{src}), \tag{4}$$

where the $L_{ce}$ is the cross-entropy loss.

Besides, the encoder node embedding $\mathbf{v}_{node}$ is also used to align the domain distributions. We introduce adversarial learning (Ganin et al. 2016), to enhance the indistinguishability of the domain discriminator $D_{disc}(\cdot)$. This encourages the model to classify both domains as the same class. To achieve this, the gradients of $\mathbf{v}^S_{node}$ and $\mathbf{v}^T_{node}$ are reversed by the gradient reversal layer $F(\cdot)$. This layer is designed to train $G^{enc}(\cdot)$ to maximize the *domain classification loss*, which is minimized during the training of $D_{disc}(\cdot)$. The gradient reverse process is defined as $F(x) = x, \quad \frac{dF}{dx} = -\mathbf{I}$. The pseudo domain labels are defined as $d_{src}$ and $d_{trg}$, the *domain classification loss* can be written as

$$\begin{aligned} L_{domain} =& \frac{1}{N_s}\sum_{i=1}^{N_s} L_{ce}(D_{disc}(R(v^{Si}_{node})), d_{src}) \\ &+ \frac{1}{N_t}\sum_{i=1}^{N_t} L_{ce}(D_{disc}(R(v^{Ti}_{node})), d_{trg}) \end{aligned} \tag{5}$$

Next, we implement $\mathbf{v}_{edge} = Top\_k(\mathbf{v}_{edge})$ to take the top $k$ important edge embeddings of each node to reduce the computational burden and the disturbance of the relatively irrelevant edge information and set the discarded feature into zeros. Then, the *1D* frequency positional encoding and *2D* frequency positional encoding is added to the node and edge embeddings respectively (See Appendix B).

To aggregate the encoder features ($\mathbf{v}_{edge}$ and $\mathbf{v}_{node}$) with their positional encoding and integrate both frequency and spatial information, we introduce the aggregate module. We utilize max pooling to capture the most prominent features and introduce average pooling to obtain smooth global information. By integrating these two pooling strategies, the module functions like residual layers, effectively capturing features from both frequency and spatial channels while preventing gradient vanishing. Additionally, to manage computational complexity, we include only one learnable layer at the end of this module, ensuring it remains lightweight and does not add significant computational overhead. The overall flow is denoted as below,

$$\begin{aligned} &\mathbf{v}' = \mathbf{v} \otimes \sigma(AP(\mathbf{v}) + MP(\mathbf{v})), \\ &\mathbf{w} = \mathbf{v}'^T \otimes \sigma(BN(Conv(AP(\mathbf{v}'^T) \oplus MP(\mathbf{v}'^T)))), \end{aligned} \tag{6}$$

where $\otimes$ is the feature-wise multiplication, $\oplus$ is the concatenation of two matrices, $\sigma$ is the sigmoid activation function. The AP and MP are average pooling and max pooling. Due to the dimension for $\mathbf{v}_{edge}$ and $\mathbf{v}_{node}$ being different, in Fig. 1, we use M and P to imply the dimension would remain the same after the aggregation. Then, the aggregate feature are denoted as $\mathbf{w}_{node} \in \mathbb{R}^{Q\times P}$ and $\mathbf{w}_{edge} \in \mathbb{R}^{Q^2\times P}$. After getting the aggregate features, a parameterized decoder $G^{dec}(\cdot)$ is applied to take in both the node and edge aggregated information to build up the frequency correlation graph

$$\mathcal{G} \sim \{\mathbf{g}_{node}, \mathbf{g}_{edge}\} = G^{dec}(\mathbf{w}_{node}, \mathbf{w}_{edge}), \tag{7}$$

where $\mathbf{g}_{node} \in \mathbb{R}^{Q\times P}$ is the node correlation projection and $\mathbf{g}_{edge} \in \mathbb{R}^{Q^2\times 1}$ is the edge correlation projection. The $\mathbf{g}_{node}$ is used for calculating the *frequency feature-wise loss* ($L_{FF}$) and $\mathbf{g}_{edge}$ is used for calculating the *frequency contrastive loss* ($L_{FC}$). The $L_{FF}$ is to calculate the expectation of the discrepancy among the frequency graph over feature-wise between domains and the $L_{FC}$ is to align the frequency features correlations where we assume the correlation between the same set of frequency pairs should have similar distribution and properties, and the features from the different frequency pairs should be different. These two losses are denoted as

$$\begin{aligned} &L_{FF} = \mathbb{E}(|\mathbf{g}^S_{node} - \mathbf{g}^T_{node}|), \\ &L_{FC} = -\frac{1}{Q}\sum_{i=1}^{Q} \log \frac{e^{\mathbf{g}^{Si}_{edge}(\mathbf{g}^{Ti}_{edge})^T}}{\sum_{j=1,j\neq i}^{Q} e^{\mathbf{g}^{Si}_{edge}(\mathbf{g}^{Tj}_{edge})}}. \end{aligned} \tag{8}$$

where the $i$ and $j$ are the frequency level vector in the edge correlation projection.

## 2.4 Frequency-Contextual Contrastive Learning

Contrastive learning (CL) has been widely proven to learn and capture contextual representation effectively in various unsupervised representation learning scenarios (Wu et al. 2024; Eldele et al. 2021) ; see related work (Appendix A). In the CLUDA (Ozyurt, Feuerriegel, and Zhang 2022), CL has been demonstrated to be efficient in capturing and aligning the contextual representation of multivariate time series data which preserves the label invariant information and makes the domain alignment and prediction tasks easier. To enhance the existing FGD module, we propose the FCCL approach. This method is designed to align the frequency-contextual information from two augmented views of the same sample and distinguish it from the frequency-contextual information from other samples.

The overall FCCL framework is shown in Fig. 1. Motivated by MoCo (He et al. 2019), we implement the CL with the momentum contrast technique. In addition, the semantic-preserving augmentation strategy is utilized to get two augmentations for each sample as the key $\mathbf{X}^k$ and query $\mathbf{X}^q$ respectively where $\mathbf{X}^q, \mathbf{X}^k \in \mathbb{R}^{D\times T}$. Both would undergo decomposition into multiple frequency channels using DWT following with as Sec. 2.3, $\{\mathbf{X}^q, \mathbf{X}^k\} \rightarrow \{\tilde{\mathbf{X}}^q, \tilde{\mathbf{X}}^k\}$. The $\tilde{\mathbf{X}}^q$ and $\tilde{\mathbf{X}}^k$ are passed to the temporal projection layer $T(\cdot)$ and $\tilde{T}(\cdot)$ to get the frequency embeddings $\mathbf{H}^q$ and $\mathbf{H}^k$ respectively. After that, according to Eqn. 3, frequency embeddings $\mathbf{H}^q$ and $\mathbf{H}^k$ are processed by the graph discovery encoder $G^{enc}(\cdot)$ and $\tilde{G}^{enc}(\cdot)$ to get their encoder node embeddings $\mathbf{v}^q_{node}$ and $\mathbf{v}^k_{node}$. The weights of $\tilde{T}(\cdot)$ and $\tilde{G}^{enc}(\cdot)$ are momentum updated via

$$\{\theta_{\tilde{T}}, \theta_{\tilde{G}^{enc}}\} \leftarrow \alpha\{\theta_{\tilde{T}}, \theta_{\tilde{G}^{enc}}\} + (1-\alpha)\{\theta_T, \theta_{G^{enc}}\}, \tag{9}$$

$\alpha \in [1, 0)$ is the momentum coefficient to update the weights. To avoid model collapse, where the query and key networks might converge to trivial or identical representations, the query encoder node embeddings are passed through $G^{dec}(\cdot)$ to obtain the node correlation projection $\mathbf{g}^q_{node}$. The objective of FCCL is to bring $\mathbf{g}^{qi}_{node}$ closer to its positive

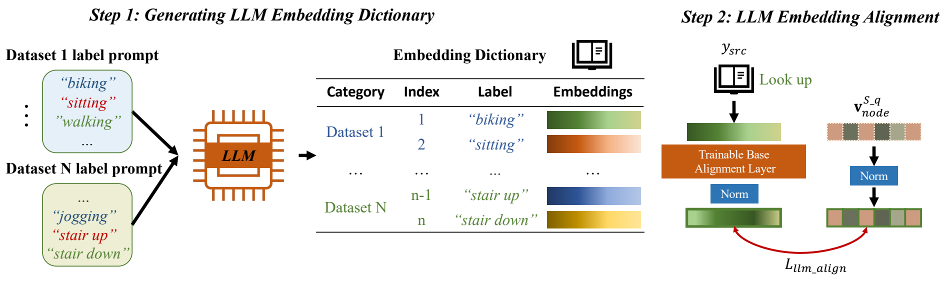


Figure 2: Language-guided Adversary Alignment (LAA).

sample, *i.e.*, $\mathbf{v}^{ki}_{node}$, but further to its negative samples in $Queue \leftarrow \{\mathbf{v}^{kj}_{node}\}_{j=1}^{J}$ where $J$ is the queue size of the large set of negative pairs collected from previous batch($J \gg B$, $J\%B = 0$, $B$ is the batch size), which is efficient to capture better frequency-contextual representations (Ozyurt, Feuerriegel, and Zhang 2022). The contrastive loss $L_{CL}$ is denoted as

$$-\frac{1}{B}\sum_{i=1}^{B}\log\frac{e^{\mathbf{g}^{qi}_{node}(\mathbf{v}^{ki}_{node})^T/\tau}}{e^{\mathbf{g}^{qi}_{node}(\mathbf{v}^{ki}_{node})^T/\tau}+\sum_{j=1}^{J}e^{\mathbf{g}^{qi}_{node}(\mathbf{v}^{kj}_{node})^T/\tau}}. \tag{10}$$

The $\tau$ is the temperature scale which is larger than 0. The source and target domains' contrastive losses are denoted as $L^S_{CL}$ and $L^T_{CL}$ respectively, and the queues for them are denoted as $Queue_{src}$ and $Queue_{trg}$.

## 2.5 ConFGD+: Language-guided Adversary Alignment

The **ConFGD+** framework extends our proposed **ConFGD** framework with a Language-guided Adversary Alignment (LAA) module, as illustrated in Fig. 2, to improve the prediction accuracy by aligning the classification feature $\mathbf{v}^{S}_{node}$ with the label embeddings guided by the pre-trained LLM text encoder. Th LAA can also be added to other MTS UDA model.

Given the large number of parameters in LLM, obtaining embeddings from label prompts can slow down inference. Instead of directly integrating the LLM into the model and extracting features for each sample during training, we store all label embeddings in a dictionary **Dict** before training starts,

$$\mathbf{Eb} = LLM(LP), \mathbf{Dict} \leftarrow \{Dataset_{name}, y^i_{src}, LP, \mathbf{Eb}\} \tag{11}$$

where the $LP$ is the label prompts, *i.e.*, the text label of the ground true $y^i_{src}$, and the $Dataset_{name}$ stored in the **Dict** is convenient to find the exact set of labels during loading data. Therefore, the dictionary only needs to be built once which is very time-saving and computationally efficient.

During the model training, the way to get embeddings is similar to looking up the dictionary where the index is $[Dataset_{name}, y^i_{src}]$ for each sample. Then, the source dataset $\mathcal{D}_{src}$ is denoted as $\mathcal{D}^+_{src} = \{([\mathbf{X}^i_{src}, \mathbf{Eb}^i], y^i_{src})\}_{i=1}^{N_s}$. As the dimension of **Eb** is high, we adopt the approach outlined in (OpenAI 2024) to reduce its dimension to match that of the flattened $\mathbf{v}^{S_q}_{node} \in \mathbb{R}^{1\times QP}$ by $\mathbf{Eb}' = \mathbf{Eb}[: Q\times P] \in \mathbb{R}^{1\times QP}$.

Considering the base of $\mathbf{Eb}'$ and the classification feature $\mathbf{v}^{S}_{node}$ are not consistent, the $\mathbf{Eb}'$ is multiplied by a learnable identity matrix $\mathbf{W}_{\mathbb{I}}$ to slightly align their basement. The *language-guided alignment loss* is expressed as

$$L_{llm\_align} = \frac{1}{N_s}\sum_{i=0}^{N_s}\left(1-\frac{\mathbf{v}^{Si}\cdot\mathbf{W}_{\mathbb{I}}\mathbf{Eb}'^{i}}{\max\left(\left\|\mathbf{v}^{Si}\right\|_2\cdot\left\|\mathbf{W}_{\mathbb{I}}\mathbf{Eb}'^{i}\right\|_2,\epsilon\right)}\right), \tag{12}$$

where the $\epsilon$ is a constant set to avoid a zero denominator.

## 2.6 Overall Loss

(a) **ConFGD**: $\min L_{ConFGD} = L_{pred} + \lambda_{domain}L_{domain} + \lambda_{freq}(L_{FF} + L_{FC}) + \lambda_{CL}(L^S_{CL} + L^T_{CL})$.

(b) **ConFGD+**: $\min L_{ConFGD+} = L_{ConFGD} + \lambda_{llm\_align}L_{llm\_align}$.

## 3 Experiment Preparation

To evaluate the effectiveness of our proposed ConFGD and ConFGD+, we conduct experiments on three **benchmark** datasets, HAR (Anguita et al. 2013), WISDM (Kwapisz, Weiss, and Moore 2011) and HHAR (Stisen et al. 2015). Besides, we utilize GPT3 from OpenAI (OpenAI 2024) to create the text embedding dictionary for the LAA module during data preparation. The dataset preparation details are specified in Appendix D. To demonstrate how our proposed ConFGD and ConFGD+ models significantly improve the accuracy on target domains, we randomly select 10 source-target domain pairs for HAR and WISDM, and 7 pairs for HHAR during the evaluation, where the domains are distinct by different participants. More dataset details are specified in Appendix D.

During the experiments, we choose 2 w/o UDA models and 11 UDA models as the **baseline** models. The 2 w/o UDA models are different from whether including the graph discovery encoder. The 11 baseline models are illustrated in Appendix E.1. To further assess the effectiveness of our proposed LAA module in ConFDG+, we incorporate this module into the w/o UDA and CLUDA models for experimentation. These models are only trained on the source domain and selected by the performance on the validation source domain dataset. More specific training details, time of execution, and contrastive learning augmentation strategy are specified in Appendix E.2 and E.3.

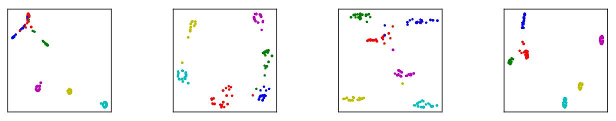
| Dataset | Metric | TCN(w/o UDA) | w/o UDA | VRADA | CoDATS | AdvSKM | CAN | CDAN | DDC | DeepCORAL | DSAN | HoMM | MMDA | CLUDA | ConFGD | ConFGD+ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **HAR** | Avg. Acc | 60.48 | 80.60 | 77.03 | 66.24 | 61.93 | 67.98 | 69.58 | 61.57 | 71.21 | 74.70 | 70.66 | 59.18 | 91.21 | 94.86 | **96.81** |
| | Std. | 18.79 | 15.07 | 6.65 | 19.33 | 19.58 | 14.26 | 15.80 | 18.81 | 11.07 | 11.23 | 13.99 | 18.66 | 6.78 | 4.63 | 3.87 |
| | Avg. F1 | 53.63 | 77.69 | 71.05 | 59.57 | 55.15 | 62.97 | 62.69 | 54.85 | 66.86 | 70.89 | 64.36 | 49.15 | 90.63 | 94.84 | **96.69** |
| | Std. | 19.56 | 17.45 | 9.05 | 19.32 | 20.25 | 16.35 | 19.94 | 19.56 | 12.58 | 12.39 | 16.19 | 19.58 | 7.54 | 5.12 | 4.09 |
| **WISDM** | Avg. Acc | 66.26 | 62.08 | 65.69 | 64.41 | 64.91 | 58.45 | 57.90 | 66.07 | 63.54 | 59.10 | 60.22 | 53.77 | 72.32 | 79.15 | **80.67** |
| | Std. | 11.69 | 8.68 | 11.40 | 12.88 | 10.36 | 12.15 | 17.93 | 10.05 | 14.03 | 17.29 | 14.19 | 17.93 | 7.55 | 4.50 | 5.03 |
| | Avg. F1 | 50.20 | 46.04 | 47.67 | 46.71 | 46.79 | 45.33 | 37.66 | 48.01 | 44.04 | 46.48 | 43.33 | 35.30 | 53.92 | 62.25 | **65.42** |
| | Std. | 9.56 | 10.68 | 17.39 | 10.54 | 9.04 | 13.08 | 14.79 | 8.37 | 9.59 | 18.02 | 13.16 | 14.09 | 15.50 | 16.16 | 15.25 |
| **HHAR** | Avg. Acc | 68.84 | 76.51 | 73.57 | 59.89 | 64.96 | 73.69 | 63.92 | 64.81 | 73.38 | 62.38 | 71.44 | 58.95 | 75.90 | 83.08 | **84.11** |
| | Std. | 12.72 | 16.31 | 19.63 | 11.12 | 15.75 | 16.64 | 20.27 | 16.83 | 13.64 | 18.58 | 14.48 | 12.68 | 14.61 | 12.46 | 12.24 |
| | Avg. F1 | 66.44 | 73.92 | 70.98 | 57.43 | 60.28 | 69.83 | 57.18 | 61.41 | 71.25 | 60.47 | 67.93 | 56.21 | 74.83 | 82.09 | **83.94** |
| | Std. | 13.92 | 20.26 | 17.56 | 11.43 | 18.19 | 21.01 | 22.52 | 18.98 | 17.85 | 20.42 | 17.95 | 12.33 | 15.68 | 14.50 | 12.31 |

Table 1: Average evaluation results (Accuracy (%) and Macro F1 (%)) of the baseline models over 10 pairs of source-target domains on the HAR and WISDM and 7 pairs of source-target domains on HHAR (Higher is better. The best is in **bold** and the second best is marked with underline) with the standard deviation (Std., lower is more stable) .

## 4 Experiment Results

### 4.1 Performance Comparison

The average results of evaluating the baseline models over 10 pairs of source-target domains on the HAR and WISDM and 7 pairs of source-target domains on HHAR are shown in Table. 1. We provide the full UDA results for each pair and each model in Appendix F. In the results for HAR, our ConFGD surpasses the best baseline model, CLUDA by 4.00% in accuracy and 4.44% in Macro F1 metrics (Accuracy: 94.86 vs. 91.21; Macro F1: 94.84 vs. 90.63). In the results for WISDM, our ConFGD surpasses the best baseline model, CLUDA by 9.40% in accuracy and 15.44% in Macro F1 metrics (Accuracy: 79.15 vs. 72.32; Macro F1: 62.25 vs. 53.92). Thus, our proposed ConFGD is more effective and better suited for handling the more challenging WISDM prediction. Regarding HHAR, our ConFGD surpasses the best baseline model, CLUDA by 9.46 % in accuracy and 9.70 % in Macro F1 metrics (Accuracy: 83.08 vs. 75.90; Macro F1: 82.09 vs. 74.83). By adding the LAA to the ConFGD, all the ConFGD+ results surpass the basic ConFGD models, proving the LAA module's effectiveness. A more specific evaluation of the LAA module is demonstrated in Sec. 4.2. Based on the standard deviation (Std.) in the accuracy metrics for HAR and WISDM, and Std. in the Macro-F1 metrics for HAR, our ConFGD and ConFGD+ models exhibit significantly smaller Std. compared to the other models. Overall, our models are not only effective but also achieve more stable prediction accuracy by a large margin compared to the baseline models.



(a) w/o UDA (b) CLUDA (c) ConFGD (d) ConFGD+

Figure 3: Embedding t-SNE visualizations of w/o UDA, CLUDA, ConFGD, and ConFGD+ on the benchmark dataset HAR (Anguita et al. 2013). The classes are differentiated by colors. The "star" shapes are from the source domain, and the "round" shapes are from the target domain.

**Embedding visualization:** Fig. 3 visualizes the embeddings to illustrate the domain discrepancies learned by different models. In Fig. 3a, w/o UDA exhibits a clear domain shift, where the green class splits into two clusters, and some classes overlap. Fig. 3b and Fig. 3c represent CLUDA and ConFGD, the best baseline CL framework models, which show clearer clusterings compared to w/o UDA, confirming the effectiveness of contrastive learning. However, slight domain shifts and occasional class mixing are still present. ConFGD, with its novel FCCL framework aligning contextual representations at the frequency level, further reduces class mixing compared to CLUDA. Fig. 3d, depicting ConFGD+, achieves the most distinct clusterings, with source and target domain embeddings well-aligned and no visible class mixing, thanks to its language-guided alignment. These findings, supported by additional t-SNE visualizations in Appendix I, validate the effectiveness of ConFGD and ConFGD+.

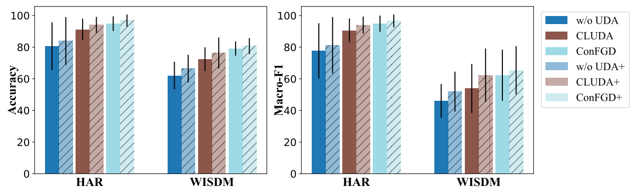


Figure 4: LAA Module Evaluation Results

### 4.2 LAA Module Evaluation

To validate the effectiveness of LAA across different LLMs, we tested it on five paired WISDM datasets using GPT-3 (Brown et al. 2020), BERT (Devlin et al. 2019), and LLaMA2 (Touvron et al. 2023) as text embedding frameworks. The results (Appendix G) confirmed its robustness, with GPT-3 showing slightly better accuracy and LLaMA2 excelling in F1 score. Ultimately, GPT-3 was chosen as the primary model for its advanced natural language understanding, high-quality embeddings, and extensive training on diverse textual data, making it ideal for our domain adaptation tasks.

To extensively prove the effectiveness of our introduced LAA module, we add it to not only our proposed ConFGD but also the w/o UDA model and the best baseline model CLUDA where the upgraded model is denoted with a "+"

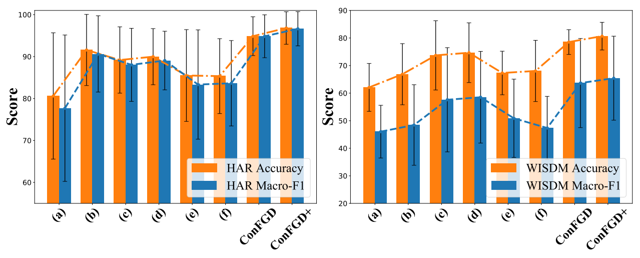


Figure 5: Ablation study: (a) w/o UDA, (b) w/o $L_{FC}\&L_{FF}$, (c) w/o $L_{CL}$, (d) w/o $L_{CL}\&L_{FC}\&L_{FF}$, (e) w/o $L_{domain}$ and (f) w/o PE.

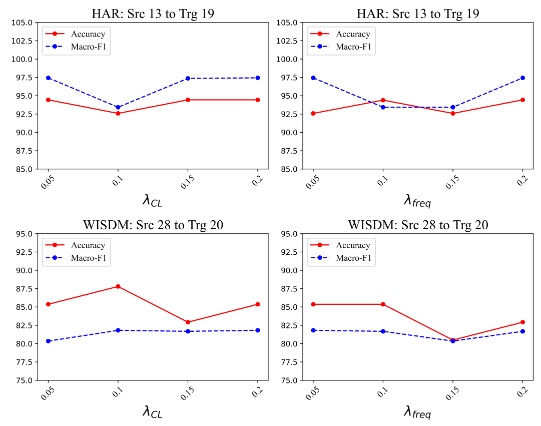


Figure 6: The sensitivity analysis for different for different $\lambda_{CL}$ and $\lambda_{freq}$

sign. Fig. 4 shows the performance comparisons between the basic models and the upgraded "+" models. The performances of all the upgraded models surpass their basic models. To be noticed, on WISDM, the Macro-F1 score of CLUDA+ increased by 15.50% from CLUDA which is considered as a significant improvement. Furthermore, it also proves the accuracy and effectiveness of the LAA module. We provide all the detailed results of the LAA module evaluations in Appendix H.

### 4.3 Ablation Study

We conduct an ablation study of our proposed ConFDG framework by comparing different ablation models (discarding some parts of the variants) to gain a deeper insight into the different components of our framework. The ablation models used for the evaluations are (a) w/o UDA, (b) w/o $L_{FC}\&L_{FF}$, (c) w/o $L_{CL}$, (d) w/o $L_{CL}\&L_{FC}\&L_{FF}$, (e) w/o $L_{domain}$ and (f) w/o PE. Fig. 5 shows the ablation study results on HAR and WISDM by averaging the 10 random initialization results. From the plotting graphs, all the components contribute improvements over the base w/o UDA which proves the effectiveness of the framework. In Appendix J, we provide more detailed results of the ablation study.

**Sensitivity Analysis:** The results of the sensitivity analysis for the HAR and WISDM datasets are presenting in Fig. 6. We tested various values of $\lambda_{CL}$ and $\lambda_{freq}$ ranging from 0.05 to 0.2. Our results show that the ConFGD model remains stable within a certain range for these hyperparameters. In addition, the standard deviation (std.) of the F1 score in WISDM case is only 0.71 ($\lambda_{CL}$) and 0.69 ($\lambda_{freq}$) which are considered very small values and imply the stable results with varying in these two parameters.

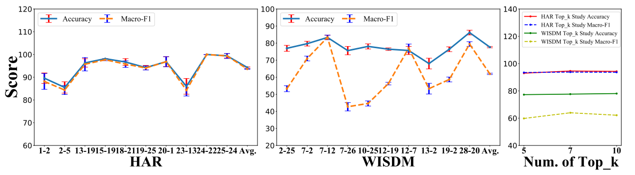


Figure 7: $Top\_k$ Study Evaluataion Results

**Top_k Study:** In the $\mathbf{v}_{edge}$, the number of $Top\_k$ nodes are a changeable parameters and smaller $Top\_k$ implies higher computational efficiency. Fig. 7 shows the $Top\_k$ study results on the chosen source-target domain pairs of HAR and WISDM with standard deviation (std.) bars, and the overall average results. The std. values are very small, where the std. value for HAR Macro-F1 is only 0.17 (0.18% of the average value), indicating that the performances remain strong and stable even when the number of $Top\_k$ changes. Therefore, reducing the number of $Top\_k$ can effectively speed up the computational efficiency while maintaining high prediction accuracy. In Appendix E.2, we provide the execution time for all the models which shows that by setting the $Top\_k$ to 5, our models achieve better results and faster execution time compared to the second-best baseline model, CLUDA. In Appendix K, we provide all the experiment results in detail of the $Top\_k$ study.

## 5 Conclusion

In this work, we have proposed and developed a noteworthy and novel MTS UDA contrastive learning-based framework, ConFGD. First, we proposed and developed a novel FGD module to identify and align inherent inter-series frequency channel information and relationships for UDA of time series. Additionally, we introduced an FCCL framework to cooperate with the frequency embeddings to capture frequency-contextual representations and enhance prediction capabilities by extracting label-invariant information. Furthermore, we are pioneers in integrating the LLM as a time-saving and computationally efficient LAA module to further improve prediction accuracy by aligning the classification feature with LLM text-encoded labeled embeddings. Comprehensive experiments were conducted to validate the effectiveness and robustness of the proposed ConFGD and ConFGD+. Additionally, we upgraded other SOTA models with the LAA module to verify its superiority and effectiveness. Comparisons with various alternative SOTA approaches were made using two evaluation criteria on three benchmark datasets, effectively demonstrating the superiority and accuracy of our proposed method.

## References

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; Reyes-Ortiz, J. L.; et al. 2013. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, 3.

Antonino-Daviu, J. A.; Riera-Guasp, M.; Pineda-Sanchez, M.; and Perez, R. B. 2009. A Critical Comparison Between DWT and Hilbert–Huang-Based Methods for the Diagnosis of Rotor Bar Failures in Induction Machines. *IEEE Transactions on Industry Applications*, 45(5): 1794–1803.

Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A theory of learning from different domains. *Machine learning*, 79: 151–175.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Cai, R.; Chen, J.; Li, Z.; Chen, W.; Zhang, K.; Ye, J.; Li, Z.; Yang, X.; and Zhang, Z. 2021. Time series domain adaptation via sparse associative structure alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 6859–6867.

Chen, C.; Fu, Z.; Chen, Z.; Jin, S.; Cheng, Z.; Jin, X.; and Hua, X.-S. 2020a. Homm: Higher-order moment matching for unsupervised domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 3422–3429.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020b. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

da Costa, P. R. d. O.; Akçay, A.; Zhang, Y.; and Kaymak, U. 2020. Remaining useful lifetime prediction via deep domain adaptation. *Reliability Engineering & System Safety*, 195: 106682.

Del Pup, F.; and Atzori, M. 2023. Applications of Self-Supervised Learning to Biomedical Signals: where are we now. *Authorea Preprints*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Eldele, E.; Ragab, M.; Chen, Z.; Wu, M.; Kwoh, C. K.; Li, X.; and Guan, C. 2021. Time-Series Representation Learning via Temporal and Contextual Contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2352–2359.

Gabor, D. 1946. Theory of communication. Part 3: Frequency compression and expansion. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, 93(26): 445–457.

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; March, M.; and Lempitsky, V. 2016. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59): 1–35.

Guo, H.; Pasunuru, R.; and Bansal, M. 2020. Multi-Source Domain Adaptation for Text Classification via DistanceNet-Bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 7830–7838.

Guo, H.; Zhu, H.; Wang, J.; Prahlad, V.; Ho, W. K.; de Silva, C. W.; and Lee, T. H. 2024. Remaining Useful Life Prediction via Frequency Emphasizing Mix-Up and Masked Reconstruction. *IEEE Transactions on Artificial Intelligence*.

He, H.; Queen, O.; Koker, T.; Cuevas, C.; Tsiligkaridis, T.; and Zitnik, M. 2023. Domain Adaptation for Time Series Under Feature and Label Shifts. In *International Conference on Machine Learning*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2019. Momentum Contrast for Unsupervised Visual Representation Learning. *arXiv preprint arXiv:1911.05722*.

Kang, G.; Jiang, L.; Yang, Y.; and Hauptmann, A. G. 2019. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4893–4902.

Karlton, W. 2020. Time Frequency Analysis of Wavelet and Fourier Transform.

Kumar, V.; Patil, H.; Lal, R.; and Chakraborty, A. 2023. Improving Domain Adaptation Through Class Aware Frequency Transformation. *International Journal of Computer Vision*, 131(11): 2888–2907.

Kwapisz, J. R.; Weiss, G. M.; and Moore, S. A. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2): 74–82.

Lai, K.-H.; Wang, L.; Chen, H.; Zhou, K.; Wang, F.; Yang, H.; and Hu, X. 2023. Context-aware domain adaptation for time series anomaly detection. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, 676–684. SIAM.

Li, Y.; Baldwin, T.; and Cohn, T. 2018. What's in a Domain? Learning Domain-Robust Text Representations using Adversarial Training. *ArXiv*, abs/1805.06088.

Li, Y.; Torralba, A.; Anandkumar, A.; Fox, D.; and Garg, A. 2020. Causal discovery in physical systems from videos. *Advances in Neural Information Processing Systems*, 33.

Liu, Q.; and Xue, H. 2021. Adversarial Spectral Kernel Matching for Unsupervised Time Series Domain Adaptation. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2744–2750. International Joint Conferences on Artificial Intelligence Organization.

Liu, S.; Kimura, T.; Liu, D.; Wang, R.; Li, J.; Diggavi, S.; Srivastava, M.; and Abdelzaher, T. 2023. FOCAL: Contrastive Learning for Multimodal Time-Series Sensing Signals in Factorized Orthogonal Latent Space. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 47309–47338. Curran Associates, Inc.

Long, M.; Cao, Z.; Wang, J.; and Jordan, M. I. 2018. Conditional adversarial domain adaptation. *Advances in neural information processing systems*, 31.

Luo, J.; Xiao, Z.; Wang, Y.; Luo, X.; Yuan, J.; Ju, W.; Liu, L.; and Zhang, M. 2024. Rank and Align: Towards Effective Source-free Graph Domain Adaptation. *arXiv preprint arXiv:2408.12185*.

Luo, Y.; Huang, Z.; Wang, Z.; Zhang, Z.; and Baktashmotlagh, M. 2020. Adversarial Bipartite Graph Learning for Video Domain Adaptation. In *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*, 19–27. ACM.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

OpenAI. 2024. Embeddings Guide. OpenAI Platform Documentation.

Ozyurt, Y.; Feuerriegel, S.; and Zhang, C. 2022. Contrastive learning for unsupervised domain adaptation of time series. *arXiv preprint arXiv:2206.06243*.

Purushotham, S.; Carvalho, W.; Nilanon, T.; and Liu, Y. 2022a. Variational recurrent adversarial deep domain adaptation. In *International Conference on Learning Representations*.

Purushotham, S.; Carvalho, W.; Nilanon, T.; and Liu, Y. 2022b. Variational recurrent adversarial deep domain adaptation. In *International Conference on Learning Representations*.

Ragab, M.; Eldele, E.; Chen, Z.; Wu, M.; Kwoh, C.-K.; and Li, X. 2024. Self-Supervised Autoregressive Domain Adaptation for Time Series Data. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1): 1341–1351.

Ragab, M.; Eldele, E.; Tan, W. L.; Foo, C.-S.; Chen, Z.; Wu, M.; Kwoh, C.-K.; and Li, X. 2023. ADATIME: A Benchmarking Suite for Domain Adaptation on Time Series Data. *ACM Trans. Knowl. Discov. Data*.

Rahman, M. M.; Fookes, C.; Baktashmotlagh, M.; and Sridharan, S. 2020. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, 81–94.

Ramponi, A.; and Plank, B. 2020. Neural unsupervised domain adaptation in NLP—a survey. *arXiv preprint arXiv:2006.00632*.

Singh, A. 2021. Clda: Contrastive learning for semi-supervised domain adaptation. *Advances in Neural Information Processing Systems*, 34: 5089–5101.

Stisen, A.; Blunck, H.; Bhattacharya, S.; Prentow, T. S.; Kjærgaard, M. B.; Dey, A.; Sonne, T.; and Jensen, M. M. 2015. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, 127–140.

Sun, B.; and Saenko, K. 2016. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. arXiv:1607.01719.

Tonekaboni, S.; Eytan, D.; and Goldenberg, A. 2021. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Tzeng, E.; Hoffman, J.; Saenko, K.; and Darrell, T. 2017. Adversarial Discriminative Domain Adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tzeng, E.; Hoffman, J.; Zhang, N.; Saenko, K.; and Darrell, T. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Wang, Y.; Song, X.; Wang, Y.; Xu, P.; Hu, R.; and Chai, H. 2021. Dual metric discriminator for open set video domain adaptation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8198–8202. IEEE.

Wang, Y.; Xu, Y.; Yang, J.; Chen, Z.; Wu, M.; Li, X.; and Xie, L. 2023. SEnsor Alignment for Multivariate Time-Series Unsupervised Domain Adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8): 10253–10261.

Wilson, G.; Doppa, J. R.; and Cook, D. J. 2020. Multi-Source Deep Domain Adaptation with Weak Supervision for Time-Series Sensor Data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, 1768–1778. New York, NY, USA: Association for Computing Machinery. ISBN 9781450379984.

Wilson, G.; Doppa, J. R.; and Cook, D. J. 2023. Calda: Improving multi-source time series domain adaptation with contrastive adversarial learning. *IEEE transactions on pattern analysis and machine intelligence*.

Wu, Y.; Meng, X.; He, Y.; Zhang, J.; Zhang, H.; Dong, Y.; and Lu, D. 2024. Multi-view Self-Supervised Contrastive Learning for Multivariate Time Series. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 9582–9590.

Xu, Y.; Cao, H.; Chen, Z.; Li, X.; Xie, L.; and Yang, J. 2022. Video unsupervised domain adaptation with deep learning: A comprehensive survey. *arXiv preprint arXiv:2211.10412*.

Yang, L.; and Hong, S. 2022. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *International conference on machine learning*, 25038–25054. PMLR.

Yang, Y.; and Soatto, S. 2020. Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4085–4095.

Zhang, X.; Zhao, Z.; Tsiligkaridis, T.; and Zitnik, M. 2022. Self-supervised contrastive pre-training for time series via time-frequency consistency. *Advances in Neural Information Processing Systems*, 35: 3988–4003.

Zhang, Y. 2021. A survey of unsupervised domain adaptation for visual recognition. *arXiv preprint arXiv:2112.06745*.

Zhu, Y.; Zhuang, F.; Wang, J.; Ke, G.; Chen, J.; Bian, J.; Xiong, H.; and He, Q. 2020. Deep subdomain adaptation network for image classification. *IEEE transactions on neural networks and learning systems*, 32(4): 1713–1722.

## A Related Work

**Contrastive Learning:** Contrastive learning (CL) is a method that seeks to learn representations through self-supervised learning (SSL). This means that it aims to place similar samples (positive pairs) close to each other in the embedding space while pushing dissimilar samples (negative pairs) further apart (He et al. 2019). This method utilizes the two augmented views and optimizes their mutual information lower bound to capture the semantic information of the input samples. There are some typical works in the CL. Contrastive predictive coding (CPC) (Oord, Li, and Vinyals 2018) designed to capture the global information across different time series segments by the predictive coding technique. SimCLR (Chen et al. 2020b) minimized the NT-Xent loss for the generated augmented views in the embedding space. Moment contrast (MoCo) (He et al. 2019) is considered a significant work in CL which generates two augmented views from the input samples as the query and key, and gets the embedding of the key by a momentum-updated encoder and stored in a queue to further used as the negative pairs for the other samples.

CL is also widely applied in many UDA tasks. CL reduces the domain discrepancy by minimizing a contrastive loss and aligning the source and target embeddings which are from the same class. Due to the labels of the target samples are unknown, these methods rely on pseudo-labels produced by a clustering algorithm (Ozyurt, Feuerriegel, and Zhang 2022). CAN (Kang et al. 2019) designed a discrepancy-metric-based CL framework to align features in the class domain. CLDA (Singh 2021) introduced a simple single-stage CL training framework and aligned features at class centroids and instance levels. The studies discussed provide valuable insights into integrating CL frameworks into unsupervised domain adaptation.

**Contrastive Learning for Time Series:** There is a growing number of research focused on creating contrastive learning frameworks tailored for time-series data. There are some notable works which are mainly on time series forecasting, classification, and anomaly detection (Tonekaboni, Eytan, and Goldenberg 2021). TNC (Tonekaboni, Eytan, and Goldenberg 2021) proposed a debiased contrastive framework to distinguish the representation space, signals in the local neighborhood from non-neighboring signals. TFC (Zhang et al. 2022) aligned the time-frequency consistency for forecasting. TS-TCC enhanced the consistency between strong and weak augmentations of the same sample within the CPC framework. FOCAL (Liu et al. 2023) extract comprehensive information from the factorized orthogonal latent space of multi-modal time series. The aforementioned studies demonstrate the significant effectiveness of CL frameworks in time-series tasks. In the MTS UDA task, CLUDA (Ozyurt, Feuerriegel, and Zhang 2022) is the first CL framework that is incorporated with the MoCo framework to align the label invariant information by capturing the contextual representations. CALDA (Wilson, Doppa, and Cook 2023) combined the CL framework with adversarial learning. These works of CL frameworks in the MTS UDA show their effectiveness and great potential.

**Unsupervised Domain Adaptation:** UDA is a machine learning technique that aims to reduce label dependency by aligning the features from the labeled source domain with those of unlabeled target domain features, thus mitigating the feature discrepancy between the two domains and predicting the unlabeled target domain. UDA has been widely applied in many fields, such as computer vision (CV) (Zhang 2021), natural language processing (NLP) (Ramponi and Plank 2020), video (Xu et al. 2022), and also time series (Ragab et al. 2023). These methods can be generally categorized into two paradigms. The first one is *adversarial-based* methods which reduce the domain discrepancy by learning the domain-invariant features from training the domain discriminator networks. The examples are **CV**: DANN (Ganin et al. 2016), adversarial discriminative domain adaptation (ADDA) (Tzeng et al. 2017), **NLP**: language identification (Li, Baldwin, and Cohn 2018), natural language inference, **video**: adversarial bipartite graph (ABG) (Luo et al. 2020) and **time series**: CoDATS (Wilson, Doppa, and Cook 2020) and CALDA (Wilson, Doppa, and Cook 2023). The second one is *metric-based* methods which aim to mitigate the domain statistical distribution shift between source and domains. The examples are **CV**: deep subdomain adaptation network (DSAN) (Zhu et al. 2020), Deep Coral (Sun and Saenko 2016), and RNA (Luo et al. 2024), **NLP**: distant-bandits (Guo, Pasunuru, and Bansal 2020), **video**: duel metric domain adaptation (Wang et al. 2021) and **time series**: sparse associative structure alignment (SASA) (Cai et al. 2021) and adversarial spectral kernel matching (AdvSKM) (Liu and Xue 2021). These UDA methods have demonstrated their efficacy in reducing the reliance on labeled data for Deep Learning training. Frequency transformation has proven highly effective for domain adaptation in areas like CV. For instance, FDA (Yang and Soatto 2020) employed low-frequency spectral swapping for semantic segmentation, while CAFT++ (Kumar et al. 2023) used class-aware frequency transformation to swap low-frequency information between classes, reducing domain shift.

**Unsupervised Domain Adaptation for Time Series:** Several works have been designed for MTS UDA, which broaden the range of applications for time series. In Variational recurrent adversarial deep domain adaptation, (VRADA) (Purushotham et al. 2022b), they utilized adversarial learning to align temporal features across domains, employing a variational recurrent neural network as the feature extractor. CoDATS (Wilson, Doppa, and Cook 2020) built upon VRADA's adversarial training but used a convolutional neural network for feature extraction. SASA (Cai et al. 2021) and AdvSKM (Liu and Xue 2021) are metric-based methods that align intra- and inter-variable attention mechanisms or spectral kernel mappings, respectively, to minimize domain discrepancy. These methods focus on aligning features across source and target domains. In addition to these, other UDA methods have been proposed. For example, Lai et al. (Lai et al. 2023) employed a Markov decision process formulation and deep reinforcement learning for aligning context information between different time series domains for anomaly detection. He et al. (He et al. 2023) addressed feature and label shifts between domains using temporal and frequency features. ADATIME (Ragab et al. 2023) have evaluated various existing time series UDA models where they captured temporal dynamics by CNN. Furthermore, autoregressive models (Ragab et al. 2024), contrastive learning (Ozyurt, Feuerriegel, and Zhang 2022; Wilson, Doppa, and Cook 2023), sensor alignment model (Wang et al. 2023) and temporal-spectral fusion (Yang and Hong 2022) have also been explored in the context of UDA for time-series data. These methods are valuable as they do not rely on labels in the target domain, highlighting their potential for various applications involving time-series data.

**Research Gap:** In MTS UDA, the majority of current studies focus solely on aligning intra-series features within the time-domain signal. These approaches neglect valuable information from inter-series dependencies, including frequency information. Additionally, research in MTS UDA regarding the capture of contextual representations is limited, even with no exploration of frequency-contextual representations where the label-invariant information in the contextual representations is a better alignment across domains for each class. Therefore, we develop a novel framework, ConFGD, to get a better generalization over the unlabeled target domain.

## B Positional Encoding

In Sec. 2.3, we apply two different positional encoding strategies targeting the encoder node $\mathbf{v}_{node}$ and edge $\mathbf{v}_{edge}$ embeddings respectively. $\mathbf{v}_{node}$ is added with 1D frequency positional encoding which is denoted as

$$\begin{aligned} PE_{(q,2p)} &= \sin\left(\frac{q}{10000^{2p/P}}\right), \\ PE_{(q,2p+1)} &= \cos\left(\frac{q}{110000^{2p/P}}\right), \end{aligned} \tag{13}$$

where $q$ is the position of the frequency channels in $\mathbf{v}_{node}$. $\mathbf{v}_{edge}$ is added with 2D frequency positional encoding which is denoted as

$$\begin{aligned} 2DPE_{(q1,q2,2p)} &= \sin\left(\frac{q1}{10000^{2p/P}} + \frac{q2}{10000^{2p/P}}\right), \\ 2DPE_{(q1,q2,2p+1)} &= \cos\left(\frac{q1}{10000^{2p/P}} + \frac{q2}{10000^{2p/P}}\right), \end{aligned} \tag{14}$$

where $(q1, q2)$ is the frequency position in the edge embedding $Top\_k(\mathbf{v}_{edge})$. $2p$ is referring to the features at even positions, and $2p+1$ is the features at odd positions.

## C Graph Neural Network

In our model, the GNN utilized in the graph discovery encoder $G^{enc}(\cdot)$ and decoder $G^{dec}(\cdot)$ is to capture the inter-frequency interactive correlations. For input embeddings with $2S$ frequency channels, the directed graph can be denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where the vertices $\mathcal{V} = \{f_i\}$ is embeddings of each frequency channels and the edge $\mathcal{E} = \{(f_i, f_j, f_{ij})\}$ is the directed interactive correlation from frequency channel $i$ to $j$ with the associate edge attribute $f_{ij}$. Our GNN uses a similar structure as the Interaction Network (IN) [1] to generate the node and edge embeddings for each frequency channel and their paired relations which are shown below,

$$\begin{aligned} \mathbf{h}_{ij} &= Edge(f_i, f_j, f_{ij}) \quad (f_i, f_j, f_{ij}) \in \mathcal{E}, \\ \mathbf{h}_i &= Node(f_i, \sum_{j \in S_i} \mathbf{h}_{ij}) \quad f_i \in \mathcal{V}. \end{aligned} \tag{15}$$

The $Edge(\cdot)$ and $Node(\cdot)$ are the edge and node embeddings encoders respectively which are just one simply linear layer. $S_i$ is the gathering of all vertices that have an edge pointing to the frequency channel i.

In addition, the edge of the directed graph to $G^{enc}(\cdot)$ is predicted from the input frequency hidden representation $\mathbf{H} \in \mathbb{R}^{2S \times P}$. Each frequency channels hidden representation $\mathbf{H}_i \in \mathbb{R}^P$ is repeated to be 2-D dimension as $\hat{\mathbf{H}}_i \in \mathbb{R}^{P \times P}$. Then the edge information prediction is denoted as

$$\mathbf{H}_{node}, \mathbf{H}_{edge} = Edge_{inf}(\{\hat{\mathbf{H}}_i\}_{i \in 2S}), \tag{16}$$

where the $Edge_{inf}(\cdot)$ is the function embedded inside $G^{enc}(\cdot)$ and is two linear layers. The output of $G^{enc}(\cdot)$ and $G^{dec}(\cdot)$ is the aggregation of all the frequency channels edge and node embeddings. We denote them in the general form as $\mathbf{h}_{edge} = \{\mathbf{h}_{ij}\}_{i,j \in S}$ and $\mathbf{h}_{node} = \{\mathbf{h}_i\}_{i \in S}$ where the $\mathbf{h}$ are $\mathbf{v}$ and $\mathbf{g}$ for $G^{enc}(\cdot)$ and $G^{dec}(\cdot)$ respectively.

## D Dataset Details

We chose three sensor datasets that have been widely utilized in previous studies. In these datasets, participants engage in different activities while wearing smartphones and/or smartwatches. The objective is to predict the activity being performed based on the sensor data. Table 2 provides the summary statistics for all the datasets. Further details about each dataset are provided below.

**HAR (Anguita et al. 2013):** The dataset includes measurements from 3-axis accelerometer, 3-axis gyroscope, and 3-axis body acceleration of 30 participants. The data is recorded at 50 Hz, and we use non-overlapping segments of 128 time steps to predict the activity type. There are 6 activities including walking, walking upstairs, walking downstairs, sitting, standing, and lying down. The label statistics for each subjects are shown in Fig. 8. In our experiments, we randomly selected 10 cross-domain scenarios out of a large number of domain combinations.

**WISDM (Kwapisz, Weiss, and Moore 2011):** The dataset comprises 3-axis accelerometer measurements from 30 participants, recorded at 20 Hz. We utilize non-overlapping segments of 128 time steps to predict the activity type, which includes walking, jogging, sitting, standing, walking upstairs, and walking downstairs. This dataset poses a challenge due to class imbalance across participants, with some participants not performing all activities (Fig. 9). Similar to HAR, in the experiments, we also randomly combined 10 sets of cross-domain scenarios out of all the domain combinations.

**HHAR (Stisen et al. 2015)**: HHAR includes measurements from 3-axis accelerometer sensors of 9 participants and the data is recorded at 50 Hz where we use non-overlapping segments of 128 time steps to predict the activity type. The activity types are biking, sitting, standing, walking, walking upstairs, and walking downstairs. The label statistics for each subjects are shown in Fig. 10. In our experiments, we randomly selected 7 cross-domain scenarios out of a large number of domain combinations.

**ConFGD+ Data Preparation:** We need to generate the embedding dictionary from the labels prompts for the LAA based on the description in Sec. 2.5 and Fig. 2. We choose the

| Dataset | Domains/Subjects | Channels | Classes | Sequence Length | Training Samples | Validation Samples | Test Samples |
|---|---|---|---|---|---|---|---|
| HAR | 30 | 9 | 6 | 128 | 7194 | 1542 | 1563 |
| WISDM | 30 | 3 | 6 | 128 | 3870 | 1043 | 1052 |
| HHAR | 9 | 3 | 6 | 128 | 10336 | 2214 | 2222 |

Table 2: Benchmark datasets details

text embedding models from OpenAI (OpenAI 2024) for the text embedding generation. They offered three embedding models which are *text-embedding-3-small*, *text-embedding-3-large* and *text-embedding-ada-002*. The label prompts for above benchmark dataset are:

- *HAR Label Prompt:* {"biking", "sitting", "standing", "walking", "stair up", "stair down"}
- *WISDM Label Prompt:* {"walking", "jogging", "stair up", "stair down", "sitting", "standing"}
- *HHAR Label Prompt:* {"biking", "sitting", "standing", "walking", "walking upstairs", "walking downstairs"}

In the embedding dictionary, each prompt and their embeddings would be store as:

$$\{\text{HAR, 0, ``biking'', } \mathbf{Eb}_{biking}\}$$

where the **Eb** is the text embedding of "biking". This step is off-line prepared and "once for all" implementation. The text embeddings will be added as an entry for each sample (the same as loading ground true for each sample) during the data loading which does not need computational effort. Therefore, it is a low-effort but high-profit module.

**Performance metrics:** We apply two metrics as the performance metrics which are accuracy and Macro-F1. More training and evaluation details are shown in Sec. E.

## E Training Details

In this section, the training details are provided which include the briefings of all the baseline models, and the training process details. As the CL framework requires to implement augmentation strategy, the augmentation details are also provided in this section.

### E.1 Baseline Models

**TCN (w/o UDA) & w/o UDA:** The difference between these two is, that TCN (w/o UDA) is only trained on source domain dataset without UDA and utilizes the temporal convolutional network (TCN) as the feature extractor, and w/o UDA is our proposed ConFGD model trained without UDA but the feature extractor has an additional graph discovery encoder as shown in Fig. 1. As TCN is the feature extractor for some of our baseline models, such as CoDATS, AdvSKM and CLUDA, we take TCN (w/o UDA) into the evaluation.

**VRADA: (Purushotham et al. 2022b)** build a variational recurrent neural network (VRNN) and train adversarial to capture complex temporal relationships that are domain-invariant.

**CoDATS: (Wilson, Doppa, and Cook 2020)** utilizes adversarial training combined with weak supervision from a CNN network to enhance performance on MTS UDA.

**AdvSKM: (Liu and Xue 2021)** aligns spectral kernel feature to emphasize the non-monotonic and non-stationary problem in MTS by metric-based methods,

**CAN: (Kang et al. 2019)** originally designed for vision and aligns features in the class domain by a discrepancy-metric-based cl framework.

**CDAN: (Long et al. 2018)** stands for conditional adversarial domain adaptation and was originally designed for vision and suggests a conditional adversarial alignment approach by integrating task knowledge with features during the domain alignment process

**DDC: (Tzeng et al. 2014)** originally designed for vision and align the source and target domains by minimizing the Maximum Mean Discrepancy (MMD) distance.

**DeepCORAL: (Sun and Saenko 2016)** originally designed for vision and to mitigate the domain shift by aligning source and target distributions from their second-order statistics.

**DSAN: (Zhu et al. 2020)** initially designed for vision applications, employs a Local Maximum Mean Discrepancy (LMMD) to align sub-domain distributions, thereby reducing the discrepancy between the source and target domains.

**HoMM: (Chen et al. 2020a)** stands for higher-order moment matching and was originally designed for vision. Matching higher-order moments to minimize the discrepancy between domains.

**MMDA: (Rahman et al. 2020)** stands for minimum discrepancy estimation for deep domain adaptation and was originally designed for vision. MMDA incorporates both MMD and CORAL and mitigate the domain shift by introducing conditional entropy minimization into the training.

**CLUDA: (Ozyurt, Feuerriegel, and Zhang 2022)** introduces contrastive learning into UDA to capture the contextual representations for obtaining the label invariant information and enhance the prediction.

### E.2 Training Process

In Table 3, we provide the details of the hyperparameters for each baseline model individually. It is important to note that the first row, *"TCN (w/o UDA), all models"*, indicates that these hyperparameters are applied to all models with the same tuning strategy. We performed a grid search strategy for tuning and evaluations.

We implemented all the experiments in PyTorch 2.2.1 on the NVIDIA GeForce RTX 3090 with 24GB GPU. All the models including our proposed models include TCN temporal projection layer $T(\cdot)$. To keep the consistency, we set the kernel size to be 3 and the dilation factor to be 2. In addition, we use 3 layers with 64 channels. We maintain a consistent configuration for the TCN model across all baseline models

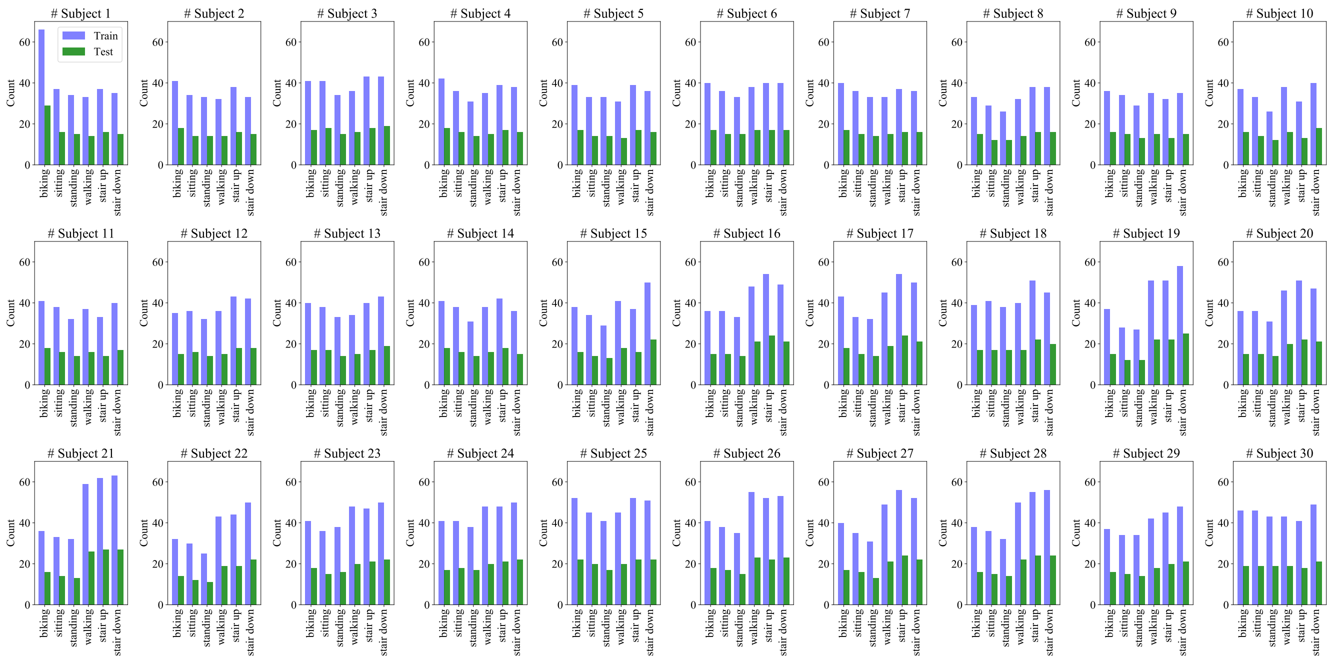


Figure 8: HAR: Labels statistics summary for individual subjects

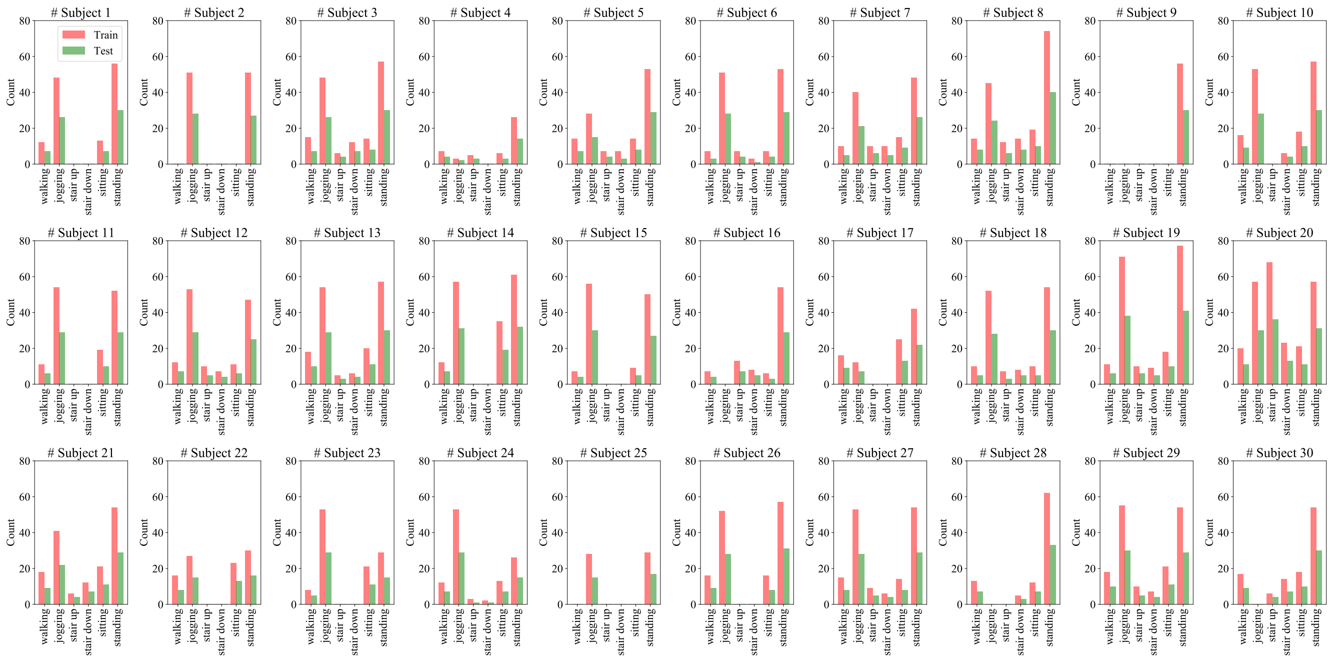


Figure 9: WISDM: Labels statistics summary for individual subjects

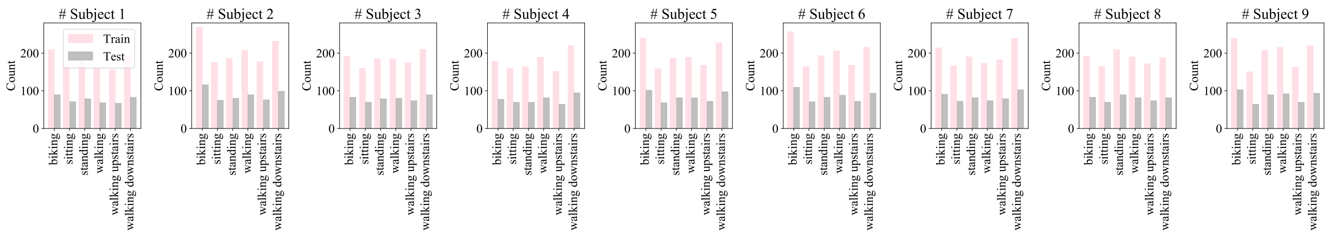


Figure 10: HHAR: Labels statistics summary for individual subjects

and benchmark datasets. We chose a batch size of 32 for all the benchmark datasets and we trained all methods for a max of 1600 epochs.

To report the performance results to resemble the real-world scenario of UDA closely, we avoided using labels from target domains and introduced an early stopping strategy into our training based on the validation loss. Learning from (Ben-David et al. 2010), we selected models that have better performance metrics (Appendix. D) on the validation dataset of the source domain, as the target domain loss is theoretically bounded by the source domain loss and along with additional factors that are restricted and passive. Besides, when the model repeatedly achieves the best validation predictions on the source domain, we will then save the model that achieves better validation performance metrics on the target domain among these instances. After finalizing the model selection, we assess its predictions on the labeled test dataset from both the target and source domains, which were not part of the training or model selection phases. The model selection criteria were applied consistently across all baseline methods in E.1 to ensure equitable comparisons. To address the variance in test performance across methods, we conducted the individual experiment with 10 unique and random initialization, and presented the results using error bars.

To report the time of execution, we average the runtime for running 100 steps for each models. **w/o UDA**: 8.58 seconds, **w/o UDA+**: 8.80 seconds, **TCN (w/o UDA)**: 8.20 seconds, **VRADA**: 56.00 seconds, **CoDATS**: 22.60 seconds, **AdvSKM**: 23.20 seconds, **CAN**:23.30 seconds, **CDAN**: 22.80 seconds, **DDC**: 23.81 seconds, **DeepCORAL**: 22.35 seconds, **DSAN**: 26.70 seconds, **HoMM**: 21.50 seconds, **MMDA**: 23.52 seconds, **CLUDA**: 35.74 seconds, **CLUDA+**: 36.68 seconds, **ConFDG** ($Top\_k$ is 5): 35.23 seconds, **ConFDG+** ($Top\_k$ is 5): 35.55 seconds, **ConFDG** ($Top\_k$ is 7): 38.58 seconds, **ConFDG+** ($Top\_k$ is 7): 38.64 seconds,**ConFDG** ($Top\_k$ is 10): 44.37 seconds, **ConFDG+** ($Top\_k$ is 10): 45.33 seconds.

### E.3 Augmentation Strategy

In the model architecture shown in 1, we generate two augmentation views for each input sample in order to capture the frequency-contextual representation of the time series. To achieve this, we leverage semantic-preserving augmentation strategy (Del Pup and Atzori 2023). The augmentation hyperparameter details are listed below:

**Gaussian noise:** The Gaussian noises are added to each sensor measurement independently and the standard deviation is 0.1.

**History crop:** The history crop masks out a minimum 20% of the initial time series with a 50% probability.

**History cutout:** We randomly select a time window representing 15% of the total time series length, with a probability 50%.

**Channel dropout:** We independently mask out each sensor measurement with a probability 10%, applied to each channel separately.

## F UDA Results on the Benchmark Datasets

We conduct the UDA task of prediction on the three benchmark datasets, HAR, WISDM and HHAR (Sec. D). For each dataset, we evaluate 11 Baseline models, 2 w/o UDA models and two of our proposed models (ConFGD and ConFGD+) where the baseline model details can be checked from Sec. E.1. Besides, for justice, we randomly select 10 pairs of source and target domains for each dataset and conduct 10 random initialized experiments for each pair on HAR and WISDM, and 7 pairs of source and target domains for each dataset and conduct 10 random initialized experiments for each pair on HHAR. We take the mean values of the 10 experiments for each pair. We evaluate each model from two metrics, Accuracy and Macro-F1. Table. 4 are the results of HAR, Table. 5 are the results of WISDM and Table. 6 are the results of HHAR. Fig. 11 shows the evaluation results.

Regarding the HAR dataset (Table. 4), our ConFGD model outperforms the baseline models in 18 out of 20 instances. The ConFGD+ model surpasses the baseline models in all 20 instances. Additionally, ConFGD+ delivers better results than ConFGD in 18 out of 20 comparisons.

Regarding the WISDM dataset (Table. 5), our ConFGD model outperforms the baseline models in 16 out of 20 instances. The ConFGD+ model surpasses the baseline models in 17 out of 20 instances. Additionally, ConFGD+ delivers better results than ConFGD in 18 out of 20 comparisons.

Regarding the HHAR dataset (Table. 6), our ConFGD model outperforms the baseline models in 17 out of 20 instances. The ConFGD+ model surpasses the baseline models in 17 out of 20 instances. Additionally, ConFGD+ delivers better results than ConFGD in 19 out of 20 comparisons.

## G LAA with different pre-trained models

We performed experiments on five paired WISDM datasets by selecting three mainstream pre-trained large-scale models–

| Method | Hyperparameter Name | Tuning Range |
|---|---|---|
| TCN (w/o UDA), all models | Optimizer | Adam |
| | Learning rate | $2 \cdot 10^{-4}, 1 \cdot 10^{-4}$ |
| | Dropout Rate | 0.1, 0.2 |
| | Weight decay | $1 \cdot 10^{-4}, 1 \cdot 10^{-3}$ |
| | Classifier hidden dim | 64, 128 256 |
| VRADA (Purushotham et al. 2022b) | VRNN layers | 1,2,3 |
| | Discriminator loss weight | 0.1, 0.5, 1 |
| | KL divergence weight | 0.1, 0.5, 1 |
| | Neg. log-likelihood weight | 0.1, 0.5, 1 |
| CoDATS (Wilson, Doppa, and Cook 2020) | Discriminator loss weight | 0.1, 0.5, 1 |
| AdvSKM (Liu and Xue 2021) | Spectral kernel | Linear, Gaussian |
| | Number of kernels (Gaussian) | 3, 5, 7 |
| | MMD loss weight | 0.1, 0.5, 1 |
| CAN (Kang et al. 2019) | Kernel type | Linear, Gaussian |
| | Number of kernels (Gaussian) | 3, 5, 7 |
| | K-means iterations | 1, 3, 5 |
| | MMD loss weight | 0.1, 0.5, 1 |
| CDAN (Long et al. 2018) | Multiplier discriminator update | 0.1, 1, 10 |
| | Discriminator loss weight | 0.1, 0.5, 1 |
| | Conditional entropy loss weight | 0.1, 0.5, 1 |
| DDC (Tzeng et al. 2014) | Kernel type | Linear, Gaussian |
| | Number of kernels (Gaussian) | 3, 5, 7 |
| | MMD loss weight | 0.1, 0.5, 1 |
| DeepCORAL (Sun and Saenko 2016) | CORAL loss weight | 0.1, 0.3, 0.5, 1 |
| DSAN (Zhu et al. 2020) | Kernel multiplier | 1, 2, 3 |
| | Number of kernels | 3, 5, 7 |
| | Domain loss weight | 0.1, 0.5, 1 |
| HoMM (Chen et al. 2020a) | Domain discrepancy loss weight | 0.1, 0.5, 1 |
| | Discriminative clustering loss weight | 0.1, 0.5, 1 |
| MMDA (Rahman et al. 2020) | Kernel type | Linear, Gaussian |
| | Number of kernels (Gaussian) | 3, 5, 7 |
| | MMD loss weight | 0.1, 0.5, 1 |
| | CORAL loss weight | 0.1, 0.5, 1 |
| | Entropy loss weight | 0.1, 0.5, 1 |
| CLUDA (Ozyurt, Feuerriegel, and Zhang 2022) | Momentum | 0.9, 0.95, 0.99 |
| | Queue size | 8192, 12288, 24576 |
| | Discriminator loss weight | 0.1, 0.5, 1 |
| | Contrastive loss weight | 0.05, 0.1, 0.2 |
| | Nearest-neighbor contrastive learning loss weight | 0.05, 0.1, 0.2 |
| ConFGD (ours) | Number of frequency decomposition | 5 |
| | Momentum | 0.9, 0.95, 0.99 |
| | Queue size | 8192, 12288, 24576 |
| | Number of $Top_k$ | 5, 7, 10 |
| | $\lambda_{domain}$ | 0.1, 0.5, 1 |
| | $\lambda_{freq}$ | 0.5, 0.1, 0.2 |
| | $\lambda_{CL}$ | 0.05, 0.1, 0.2 |
| | $\epsilon$ | $1 \cdot 10^{-8}$ |
| ConFGD+ (ours) | $\lambda_{llm\_align}$ | 0.05, 0.1, 0.2 |

Table 3: Hyperparameter tuning setup

| Model | 1→2 | 2→5 | 13 → 19 | 15 → 19 | 18 → 21 | 19 → 25 | 20 → 1 | 23 → 13 | 24 → 22 | 25 → 24 | Avg. | Std. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCN(w/o UDA) | 56.52 | 28.26 | 79.60 | 72.20 | 55.20 | 46.10 | 86.79 | 44.80 | 80.80 | 54.50 | 60.48 | 18.79 |
| w/o UDA | 82.61 | 54.45 | 83.33 | 96.30 | 91.94 | 59.68 | 94.34 | 66.00 | 87.76 | 89.66 | 80.60 | 15.07 |
| VRADA | 60.87 | 80.43 | 75.20 | 75.60 | 79.40 | 76.80 | 83.02 | 73.60 | 83.70 | 81.70 | 77.03 | 6.65 |
| CoDATS | 71.74 | 39.13 | 79.30 | 73.33 | 55.20 | 46.80 | 96.23 | 50.40 | 92.00 | 58.30 | 66.24 | 19.33 |
| AdvSKM | 58.70 | 26.09 | 80.70 | 74.10 | 55.50 | 45.20 | 88.68 | 50.40 | 83.30 | 56.60 | 61.93 | 19.58 |
| CAN | 67.39 | 50.00 | 78.50 | 68.50 | 55.20 | 66.10 | 92.45 | 47.60 | 82.00 | 72.10 | 67.98 | 14.26 |
| CDAN | 36.96 | 52.17 | 84.10 | 75.90 | 80.30 | 77.10 | 56.60 | 70.00 | 83.70 | 79.00 | 69.58 | 15.80 |
| DDC | 58.70 | 26.09 | 80.00 | 73.33 | 54.80 | 45.50 | 86.79 | 50.40 | 80.80 | 59.30 | 61.57 | 18.81 |
| DeepCORAL | 63.04 | 71.74 | 79.30 | 75.90 | 61.00 | 59.00 | 96.23 | 66.80 | 74.30 | 64.80 | 71.21 | 11.07 |
| DSAN | 73.91 | 63.04 | 72.60 | 87.40 | 55.80 | 77.40 | 84.91 | 62.80 | 80.80 | 88.30 | 74.70 | 11.23 |
| HoMM | 69.57 | 73.91 | 81.50 | 74.08 | 58.10 | 48.70 | 94.34 | 60.40 | 85.30 | 60.70 | 70.66 | 13.99 |
| MMDA | 36.96 | 36.96 | 80.00 | 72.60 | 55.50 | 44.80 | 43.40 | 57.20 | 82.90 | 81.50 | 59.18 | 18.66 |
| CLUDA | 86.96 | 82.61 | 90.40 | 96.70 | 91.00 | 93.20 | 94.34 | 78.80 | 98.80 | 99.30 | 91.21 | 6.78 |
| ConFGD | 91.30 | 86.96 | 94.44 | **98.15** | **98.15** | 95.16 | 98.11 | 88.00 | **100** | 98.28 | 94.86 | 4.63 |
| ConFGD+ | **91.50** | **89.13** | **100** | **98.15** | **98.15** | **100** | **100** | **96.00** | **100** | **100** | **96.81** | 3.87 |
| TCN(w/o UDA) | 50.14 | 23.89 | 74.30 | 64.70 | 43.10 | 36.90 | 85.56 | 37.70 | 71.20 | 48.80 | 53.63 | 19.56 |
| w/o UDA | 83.12 | 45.32 | 82.88 | 97.14 | 87.64 | 56.38 | 94.09 | 60.29 | 81.25 | 88.75 | 77.69 | 17.46 |
| VRADA | 50.48 | 79.92 | 69.60 | 65.70 | 66.80 | 73.70 | 81.62 | 69.60 | 74.90 | 78.20 | 71.05 | 9.05 |
| CoDATS | 72.51 | 39.07 | 73.80 | 66.30 | 42.80 | 38.10 | 96.13 | 44.00 | 71.40 | 51.60 | 59.57 | 19.32 |
| AdvSKM | 52.25 | 21.83 | 76.90 | 66.40 | 44.50 | 35.90 | 87.22 | 43.60 | 72.60 | 50.30 | 55.15 | 20.25 |
| CAN | 65.57 | 44.46 | 72.90 | 59.30 | 43.40 | 64.00 | 91.66 | 41.00 | 77.20 | 70.20 | 62.97 | 16.35 |
| CDAN | 23.58 | 38.08 | 83.70 | 69.60 | 71.80 | 76.80 | 45.25 | 66.00 | 75.60 | 76.50 | 62.69 | 19.94 |
| DDC | 52.97 | 21.83 | 75.20 | 65.80 | 42.70 | 36.00 | 86.79 | 44.70 | 71.00 | 52.70 | 54.85 | 19.56 |
| DeepCORAL | 59.33 | 69.87 | 76.30 | 70.80 | 53.90 | 53.50 | 96.13 | 61.60 | 64.70 | 62.50 | 66.86 | 12.58 |
| DSAN | 72.95 | 63.17 | 66.20 | 83.10 | 45.80 | 75.40 | 81.78 | 60.60 | 72.60 | 87.30 | 70.89 | 12.39 |
| HoMM | 58.71 | 69.44 | 79.80 | 68.60 | 48.60 | 39.70 | 93.21 | 54.90 | 76.80 | 53.80 | 64.36 | 16.19 |
| MMDA | 23.48 | 23.05 | 75.20 | 65.60 | 44.00 | 34.80 | 36.40 | 52.70 | 72.20 | 64.10 | 49.15 | 19.58 |
| CLUDA | 85.32 | 80.86 | 91.10 | 95.70 | 92.30 | 93.20 | 94.10 | 76.20 | 98.30 | 99.20 | 90.63 | 7.54 |
| ConFGD | 91.22 | 86.18 | 97.44 | 97.44 | 97.44 | 94.97 | 98.44 | 86.40 | **100** | 98.91 | 94.84 | 5.12 |
| ConFGD+ | **91.33** | **88.35** | **100** | **98.59** | **97.50** | **95.00** | **100** | **95.54** | **100** | **100** | **96.63** | 4.09 |

Table 4: Evaluation Results on HAR Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 10 sets of random initialization (Higher is better. The best is in **bold** and the second best is marked with underline).

| Model | 2→25 | 7→2 | 7 → 12 | 7 → 26 | 10 → 25 | 12 → 19 | 12 → 7 | 13 → 2 | 19 → 2 | 28 → 20 | Avg. | Std. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCN(w/o UDA) | 71.05 | 62.00 | 81.82 | 72.20 | 68.29 | 74.50 | 65.40 | 53.66 | 41.00 | 72.70 | 66.26 | 11.69 |
| w/o UDA | 65.79 | 63.41 | 63.64 | **78.05** | 68.42 | 54.55 | 56.25 | 51.22 | 51.22 | 68.29 | 62.08 | 8.68 |
| VRADA | 53.16 | 60.50 | 86.36 | 69.30 | 73.68 | 55.80 | 70.80 | 48.78 | 64.40 | 74.10 | 65.69 | 11.40 |
| CoDATS | 65.79 | 61.00 | 86.36 | 70.20 | 60.53 | 63.30 | 72.10 | 51.22 | 39.50 | 74.10 | 64.41 | 12.88 |
| AdvSKM | 68.42 | 61.00 | 75.00 | 70.20 | 71.05 | 63.90 | 74.20 | 51.22 | 43.40 | 70.70 | 64.91 | 10.36 |
| CAN | 60.53 | 57.10 | 70.45 | 71.70 | 63.16 | 59.40 | 58.80 | 43.90 | 32.20 | 67.30 | 58.45 | 12.15 |
| CDAN | 78.93 | 64.90 | 38.64 | 72.20 | 44.74 | 48.80 | 77.10 | 41.46 | 34.60 | 77.60 | 57.90 | 17.93 |
| DDC | 78.95 | 62.00 | 77.27 | 71.70 | 63.16 | 56.40 | 69.20 | 63.41 | 45.90 | 72.70 | 66.07 | 10.05 |
| DeepCORAL | 76.32 | 62.40 | 79.55 | 68.30 | **78.95** | 43.30 | 59.20 | 46.34 | 47.30 | 73.70 | 63.54 | 14.03 |
| DSAN | 63.16 | 62.00 | 83.36 | 69.80 | 26.32 | 63.90 | 62.50 | 48.78 | 36.60 | 74.60 | 59.10 | 17.29 |
| HoMM | 76.32 | 60.50 | 75.00 | 69.80 | 55.26 | 41.50 | 54.60 | 41.46 | 48.80 | 79.00 | 60.22 | 14.19 |
| MMDA | 78.95 | 60.50 | 38.64 | 71.20 | 36.84 | 35.80 | 67.90 | 36.50 | 38.50 | 72.20 | 53.70 | 17.93 |
| CLUDA | 76.32 | 71.20 | 79.55 | 72.70 | 68.48 | 69.40 | **79.20** | 68.29 | 56.10 | 82.00 | 72.32 | 7.55 |
| ConFGD | 78.95 | 80.49 | 84.09 | **78.05** | **78.95** | 77.27 | 77.08 | 70.73 | **78.05** | 87.80 | 79.15 | 4.50 |
| ConFGD+ | **79.55** | **82.93** | **87.80** | **78.05** | **78.95** | **83.33** | 79.17 | **71.05** | **78.05** | **87.85** | **80.67** | 5.03 |
| TCN(w/o UDA) | 47.65 | 44.30 | 54.07 | 40.70 | **67.96** | 57.70 | 54.30 | 35.70 | 43.60 | 56.00 | 50.20 | 9.56 |
| w/o UDA | 40.64 | 48.72 | 55.16 | 42.46 | 47.38 | 39.05 | 33.07 | 33.55 | 52.31 | 67.96 | 46.03 | 10.68 |
| VRADA | 36.96 | 39.90 | 83.22 | 30.80 | 43.42 | 41.00 | 43.70 | 29.02 | 61.50 | 67.20 | 47.67 | 17.39 |
| CoDATS | 40.74 | 49.40 | 61.90 | 40.50 | 38.64 | 45.60 | 61.20 | 30.98 | 40.30 | 57.80 | 46.71 | 10.54 |
| AdvSKM | 42.14 | 47.60 | 43.65 | 41.60 | 42.31 | 51.00 | 65.50 | 32.37 | 46.00 | 55.70 | 46.79 | 9.04 |
| CAN | 38.64 | 49.00 | 44.44 | 39.50 | 46.49 | 50.80 | 63.60 | 22.62 | 32.70 | 65.50 | 45.33 | 13.08 |
| CDAN | 45.00 | 54.30 | 22.26 | 34.40 | 23.90 | 29.80 | 54.60 | 20.65 | 31.20 | 60.50 | 37.66 | 14.79 |
| DDC | 54.74 | 49.60 | 46.80 | 41.20 | 45.84 | 39.60 | 63.20 | 36.11 | 45.90 | 57.10 | 48.01 | 8.37 |
| DeepCORAL | 35.79 | 49.00 | 47.62 | 39.60 | 45.00 | 31.70 | 48.60 | 31.00 | 50.10 | 62.00 | 44.04 | 9.59 |
| DSAN | 40.15 | 48.10 | 80.42 | 40.10 | 17.50 | 51.80 | 57.40 | 25.00 | 42.80 | 61.50 | 46.48 | 18.02 |
| HoMM | 44.32 | 49.40 | 47.04 | 40.60 | 34.20 | 28.10 | 44.20 | 23.35 | 52.20 | 69.90 | 43.33 | 13.16 |
| MMDA | 45.00 | 45.90 | 25.44 | 38.50 | 13.46 | 23.30 | 53.90 | 23.24 | 30.60 | 53.70 | 35.30 | 14.09 |
| CLUDA | 43.59 | 57.60 | 80.44 | 40.30 | 55.95 | 53.20 | 67.80 | 27.26 | 45.80 | 67.30 | 53.92 | 15.50 |
| ConFGD | 54.74 | 72.12 | 85.34 | 44.95 | 45.00 | 59.45 | **79.37** | 43.73 | 55.94 | 81.82 | 62.25 | 16.16 |
| ConFGD+ | **55.95** | **77.70** | **86.60** | **45.80** | 51.56 | **70.03** | 77.11 | **45.80** | **61.76** | **81.86** | **65.42** | 15.25 |

Table 5: Evaluation Results on WISDM Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 10 sets of random initialization (Higher is better. The best is in **bold** and the second best is marked with underline).

| Model | 1→4 | 2→4 | 3→ 1 | 5→ 3 | 5→ 8 | 7→ 3 | 8 → 6 | Avg. | Std. |
|---|---|---|---|---|---|---|---|---|---|
| TCN(w/o UDA) | 64.14 | 49.80 | 60.05 | 69.00 | 84.44 | 85.15 | 69.32 | 68.84 | 12.72 |
| w/o UDA | 60.16 | 49.40 | 80.09 | **93.89** | 92.22 | 82.10 | 77.73 | 76.51 | 16.31 |
| VRADA | 49.80 | 44.22 | 79.48 | 84.72 | <u>96.50</u> | 87.73 | 72.51 | 73.57 | 19.63 |
| CoDATS | 41.04 | 50.20 | 60.45 | 69.00 | 73.93 | 62.88 | 61.75 | 59.89 | 11.12 |
| AdvSKM | 59.76 | 37.05 | 68.66 | 63.32 | 83.66 | 82.10 | 60.16 | 64.96 | 15.75 |
| CAN | 54.18 | 54.58 | 81.72 | 80.79 | 93.39 | 89.08 | 61.35 | 73.69 | 16.64 |
| CDAN | 67.33 | 41.04 | 67.54 | 50.66 | 98.05 | 78.17 | 44.62 | 63.92 | 20.27 |
| DDC | 55.38 | 35.21 | 62.69 | 68.56 | 85.60 | 81.66 | 64.54 | 64.81 | 16.83 |
| DeepCORAL | 67.73 | 45.02 | 77.25 | 77.43 | 85.60 | 81.72 | 78.88 | 73.38 | 13.64 |
| DSAN | 61.35 | 34.66 | 61.35 | 79.48 | 87.89 | 67.69 | 44.22 | 62.38 | 18.58 |
| HoMM | 62.69 | 45.82 | 69.03 | 77.73 | 84.44 | 89.08 | 71.31 | 71.44 | 14.48 |
| MMDA | 54.58 | 41.04 | 68.28 | 54.18 | 80.93 | 60.26 | 53.39 | 58.95 | 12.68 |
| CLUDA | 73.71 | 58.50 | 64.55 | 87.34 | <u>96.50</u> | 87.34 | 63.35 | 75.90 | 14.61 |
| ConFGD | <u>80.08</u> | <u>59.46</u> | <u>82.84</u> | 90.39 | **99.22** | <u>89.52</u> | <u>80.08</u> | <u>83.08</u> | 12.46 |
| ConFGD+ | **81.50** | **60.40** | **83.21** | <u>92.58</u> | **99.22** | **89.83** | **82.00** | **84.11** | 12.24 |
| TCN(w/o UDA) | 58.33 | 43.57 | 68.58 | 60.45 | 78.07 | 86.05 | 70.04 | 66.44 | 13.92 |
| w/o UDA | 56.61 | 37.37 | 78.5 | **93.95** | 92.00 | 82.30 | 76.71 | 73.92 | 20.26 |
| VRADA | 51.18 | 41.58 | 78.88 | 83.68 | 79.40 | 87.95 | 74.21 | 70.98 | 17.56 |
| CoDATS | 36.99 | 45.98 | 61.04 | 64.83 | 68.26 | 62.74 | 62.14 | 57.43 | 11.43 |
| AdvSKM | 57.74 | 26.06 | 65.84 | 52.04 | 76.72 | 81.09 | 62.49 | 60.28 | 18.19 |
| CAN | 47.69 | 40.16 | 78.50 | 81.43 | 93.35 | **89.96** | 58.30 | 69.83 | 21.01 |
| CDAN | 41.58 | 47.69 | 58.46 | 36.50 | 98.08 | 76.30 | 41.62 | 57.18 | 22.52 |
| DDC | 55.16 | 24.32 | 60.62 | 61.82 | 79.02 | 81.33 | 67.58 | 61.41 | 18.98 |
| DeepCORAL | 63.79 | 33.83 | 75.16 | 78.5 | 83.68 | 83.68 | 80.10 | 71.25 | 17.85 |
| DSAN | 58.42 | 32.52 | 58.42 | 78.88 | 87.89 | 69.81 | 37.37 | 60.47 | 20.42 |
| HoMM | 58.30 | 33.04 | 68.95 | 76.71 | 77.41 | 88.84 | 72.27 | 67.93 | 17.95 |
| MMDA | 55.38 | 36.99 | 66.94 | 47.69 | 74.64 | 58.66 | 53.14 | 56.21 | 12.33 |
| CLUDA | 67.38 | 52.60 | 70.15 | 85.52 | 96.84 | 87.95 | 63.34 | 74.83 | 15.68 |
| ConFGD | <u>80.65</u> | <u>53.13</u> | <u>80.60</u> | 90.56 | **99.22** | 89.32 | <u>81.13</u> | <u>82.09</u> | 14.50 |
| ConFGD+ | **80.86** | **60.29** | **83.40** | <u>92.54</u> | <u>98.91</u> | <u>89.37</u> | **81.62** | **83.94** | 12.31 |

Table 6: Evaluation Results on HHAR Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 7 sets of random initialization (Higher is better. The best is in **bold** and the second best is marked with <u>underline</u>).

GPT3 (Brown et al. 2020), BERT (Devlin et al. 2019), and LLaMA2 (Touvron et al. 2023)—as our text embedding frameworks. The results shown in Table 7 demonstrate that our proposed LAA module remains effective regardless of the specific LLM used for text embedding, with GPT3 showing slightly better accuracy (Up) and LLaMA2 slightly better performance in F1 score (Down). After a comprehensive evaluation, GPT-3 was selected as the primary model for the LAA module, given its leading-edge natural language understanding capabilities and its ability to produce high-quality embeddings. In addition, GPT-3's extensive training on diverse textual data makes it exceptionally well-suited for generating the word embeddings required in our domain adaptation tasks.

## H LAA Module Added for Existing Model

To further prove the effectiveness of our proposed LAA module, we add it to not only ConFGD models but w/o UDA and CLUDA models (denoted with a "+" sign) where the GPT3 is chosen as the pre-trained large model. We evaluate the experiments on both of the benchmark datasets HAR (Table. 8) and WISDM (Table. 9) with the two evaluation metrics, Accuracy and Macro-F1. Similar to Sec. F, we randomly initialize each model 10 times and take the mean values as the results. The results of adding LAA module as an adversary module to other models show the significant effectiveness and accuracy of the "low effort and high profit" module, LAA.

Regarding the HAR dataset (Table. 8), the upgraded model archives better results than the basic model in 50 out of 60 instances, and 7 of the remaining instances maintain the performance of the basic model.

Regarding the WISDM dataset (Table. 9), the upgraded model archives better results than the basic model in 52 out of 60 instances, and 5 of the remaining instances maintain the performance of the basic model.

## I Embedding Visualization of Benchmark Models

We provide the embeddings t-SNE visualization of all the baseline models on the benchmark dataset HAR (Anguita et al. 2013) which is shown in Fig. 12 where the source and target domain embeddings are differentiate by "star" and "round" shapes. Without domain adaptation (refer to Fig. 12a and Fig. 12b), TCN w/o UDA which is the most basic w/o UDA model exists significant domain shift be-

| Model | 2→25 | 7→2 | 7→12 | 12→7 | 28→20 | Avg. | Std. |
|---|---|---|---|---|---|---|---|
| ConFGD | 78.95 | 80.49 | 84.09 | 77.08 | 87.80 | 81.68 | 4.28 |
| +GPT 3 | 79.55 | 82.93 | 87.80 | 79.17 | 87.85 | 83.46 | 4.25 |
| Δ | +0.60 | +2.44 | +3.71 | +2.09 | +0.05 | +1.78 | |
| +Bert | 78.95 | 80.49 | 84.12 | 78.95 | 87.80 | 82.06 | 3.84 |
| Δ | - | - | +0.03 | +1.87 | - | +0.38 | |
| +LLaMA 2 | 81.58 | 82.00 | 85.12 | 79.17 | 87.85 | 83.14 | 3.38 |
| Δ | +2.63 | +1.51 | +1.03 | +2.09 | +0.05 | +1.46 | |
| ConFGD | 54.74 | 72.12 | 85.34 | 79.37 | 81.82 | 74.68 | 12.15 |
| +GPT 3 | 55.95 | 77.70 | 86.60 | 77.11 | 81.86 | 75.84 | 11.75 |
| Δ | +1.21 | +5.58 | +1.26 | -2.26 | +0.04 | +1.17 | |
| +Bert | 58.98 | 72.12 | 85.37 | 78.95 | 81.82 | 75.45 | 10.41 |
| Δ | +4.24 | - | +0.03 | -0.42 | - | +0.77 | |
| +LLaMA 2 | 67.5 | 72.70 | 86.60 | 79.55 | 81.86 | 77.64 | 7.57 |
| Δ | +12.76 | +0.58 | +1.26 | +0.18 | +0.04 | +2.96 | |

Table 7: Experiments conducted using various pre-trained large-scale models in LAA on the WISDM dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%).

| Model | 1→2 | 2→5 | 13 → 19 | 15 → 19 | 18 → 21 | 19 → 25 | 20 → 1 | 23 → 13 | 24 → 22 | 25 → 24 | Avg. | Std. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| w/o UDA | 82.61 | 54.45 | 83.33 | 96.30 | 91.94 | 59.68 | 94.34 | 66.00 | 87.76 | 89.66 | 80.61 | 15.07 |
| w/o UDA+ | 82.61 | 56.38 | 87.64 | 100.00 | 93.20 | 66.10 | 96.23 | 68.00 | 96.70 | 91.94 | 83.88 | 15.16 |
| Δ | 0 | +1.93 | +4.31 | +3.70 | +1.26 | +6.42 | +1.89 | +2.00 | +8.94 | +2.28 | +3.27 | |
| CLUDA | 86.96 | 82.61 | 90.40 | 96.70 | 91.00 | 93.20 | 94.34 | 78.80 | 98.80 | 99.30 | 91.21 | 6.78 |
| CLUDA+ | 89.13 | 89.13 | 88.89 | 96.70 | 91.94 | 98.39 | 98.11 | 88.00 | 100.00 | 100.00 | 94.03 | 5.05 |
| Δ | +2.17 | +6.52 | -1.51 | 0 | +0.94 | +5.19 | +3.77 | +9.20 | +1.20 | +0.70 | +2.82 | |
| ConFGD | 91.30 | 86.96 | 94.44 | 98.15 | 98.15 | 95.16 | 98.11 | 88.00 | 100.00 | 98.28 | 94.86 | 4.63 |
| ConFGD+ | 91.50 | 89.13 | 100.00 | 98.15 | 98.15 | 95.16 | 100.00 | 96.00 | 100.00 | 100.00 | 96.81 | 3.87 |
| Δ | +0.20 | +2.17 | +5.56 | 0 | 0 | 0 | +1.89 | +8.00 | 0 | +1.72 | +1.95 | |
| w/o UDA | 83.12 | 45.32 | 82.88 | 97.14 | 87.64 | 56.38 | 94.09 | 60.29 | 81.25 | 88.75 | 77.69 | 17.46 |
| w/o UDA+ | 83.12 | 45.8 | 83.33 | 100 | 93.20 | 64 | 96.67 | 62.79 | 95.70 | 87.64 | 81.23 | 17.90 |
| Δ | 0 | +0.48 | +0.45 | +2.86 | +5.56 | +7.62 | +2.58 | +2.50 | +14.45 | -1.11 | +3.54 | |
| CLUDA | 85.32 | 80.86 | 91.10 | 95.70 | 92.30 | 93.20 | 94.10 | 76.20 | 98.30 | 99.20 | 90.63 | 7.54 |
| CLUDA+ | 86.77 | 88.69 | 90.96 | 96.79 | 93.51 | 98.40 | 97.78 | 86.49 | 100.00 | 100.00 | 93.94 | 5.37 |
| Δ | +1.45 | +7.83 | -0.14 | +1.09 | +1.21 | +5.20 | +3.68 | +10.29 | +1.70 | +0.80 | +3.31 | |
| ConFGD | 91.22 | 86.18 | 97.44 | 97.44 | 97.44 | 94.97 | 98.44 | 86.40 | 100.00 | 98.91 | 94.84 | 5.12 |
| ConFGD+ | 91.33 | 88.35 | 100.00 | 98.59 | 97.50 | 95.00 | 100.00 | 95.54 | 100.00 | 100.00 | 96.63 | 4.09 |
| Δ | +0.11 | +2.17 | +2.56 | +1.15 | +0.06 | +0.03 | +1.56 | +9.14 | 0 | +1.09 | +1.79 | |

Table 8: Results for exiting models added on with LAA module on HAR Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 10 sets of random initialization.

tween the source and target domains, for example the red color class is separated and merged with blue and cyan color classes. However, our w/o UDA model has slightly improved from the TCN but still has obvious class overlapping. Consequently, embeddings from one class in the target domain intersect with those of another class in the source domain, leading to challenges for the classifier trained on the source domain to effectively generalize to the target domain. While UDA baselines reduce the domain shift, they still have significant overlapping or class shifting. In contrast, our ConFGD method can effectively align embeddings from the same class across different domains, resulting in better generalization in the target domain, though a bit overlappings still happen.

To be noticed, the embeddings of our ConFGD+ which incorporates with LAA module have no obvious class shifting and overlapping according to Fig. 12o, which proves the effectiveness and accuracy of the LAA module.

| Model | 2→25 | 7→2 | 7 → 12 | 7 → 26 | 10 → 25 | 12 → 7 | 13 → 2 | 19 → 2 | 28 → 20 | 28 → 20 | Avg. | Std. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| w/o UDA | 65.79 | 63.41 | 63.64 | 78.05 | 68.42 | 54.55 | 56.25 | 51.22 | 51.22 | 68.29 | 62.08 | 8.68 |
| w/o UDA+ | 71.05 | 65.85 | 70.45 | 78.95 | 68.48 | 56.06 | 64.58 | 54.55 | 56.1 | 78.05 | 66.41 | 8.77 |
| Δ | +5.26 | +2.44 | +6.81 | +0.90 | +0.06 | +1.51 | +8.33 | +3.33 | +4.88 | +9.76 | +4.33 | |
| CLUDA | 76.32 | 71.20 | 79.55 | 72.70 | 68.48 | 69.40 | 79.20 | 68.29 | 56.10 | 82.00 | 72.32 | 7.55 |
| CLUDA+ | 77.27 | 72.70 | 86.36 | 75.62 | 71.05 | 69.70 | 87.50 | 69.40 | 60.98 | 92.68 | 76.33 | 9.79 |
| Δ | +0.95 | +1.50 | +6.81 | +2.92 | +2.57 | +0.30 | +8.30 | +1.11 | +4.88 | +10.68 | +4.00 | |
| ConFGD | 78.95 | 80.49 | 84.09 | 78.05 | 78.95 | 77.27 | 77.08 | 70.73 | 78.05 | 87.80 | 79.15 | 4.50 |
| ConFGD+ | 79.55 | 82.93 | 87.80 | 78.05 | 78.95 | 83.33 | 79.17 | 71.05 | 78.05 | 87.85 | 80.67 | 5.03 |
| Δ | +0.60 | +2.44 | +3.71 | 0 | 0 | +6.06 | +2.09 | +0.32 | 0 | +0.05 | +1.53 | |
| w/o UDA | 40.64 | 48.72 | 55.16 | 42.46 | 47.38 | 39.05 | 33.07 | 33.55 | 52.31 | 67.96 | 46.03 | 10.68 |
| w/o UDA+ | 41.96 | 52.39 | 65.27 | 44.95 | 47.38 | 40.49 | 51.4 | 39.05 | 56.01 | 79.87 | 51.88 | 12.65 |
| Δ | +1.32 | +3.67 | +10.11 | +2.49 | 0 | +1.44 | +18.33 | +5.50 | +3.70 | +11.91 | +5.85 | |
| CLUDA | 43.59 | 57.60 | 80.44 | 40.30 | 55.95 | 53.20 | 67.80 | 27.26 | 45.80 | 67.30 | 53.92 | 15.50 |
| CLUDA+ | 55.16 | 57.60 | 79.29 | 52.62 | 65.52 | 67.89 | 75.45 | 33.55 | 45.36 | 90.39 | 62.28 | 16.91 |
| Δ | +11.57 | 0 | -1.15 | +12.32 | +9.57 | +14.69 | +7.65 | +6.29 | -0.44 | +23.09 | +8.36 | |
| ConFGD | 54.74 | 72.12 | 85.34 | 44.95 | 45.00 | 59.45 | 79.37 | 43.73 | 55.94 | 81.82 | 62.25 | 16.16 |
| ConFGD+ | 55.95 | 77.70 | 86.60 | 45.80 | 51.56 | 70.03 | 77.11 | 45.80 | 61.76 | 81.86 | 65.42 | 15.25 |
| Δ | +1.21 | +5.58 | +1.26 | +0.85 | +6.56 | +10.58 | -2.26 | +2.07 | +5.82 | +0.04 | +3.17 | |

Table 9: Results for exiting models added on with LAA module on WISDM Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 10 sets of random initialization.

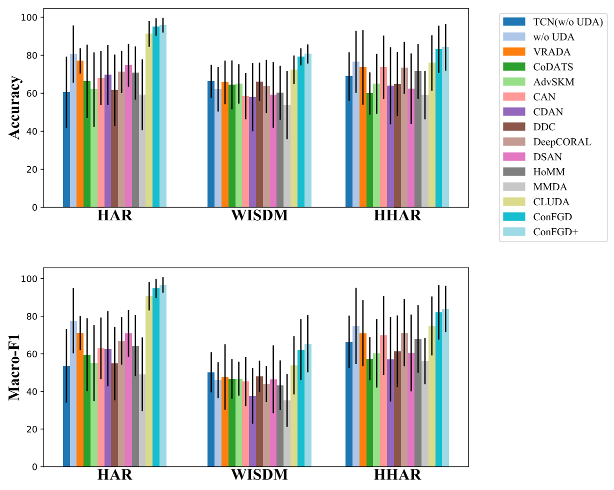


Figure 11: UDA baseline evaluation results on benchmark datasets

## J Ablation Study

We conduct an ablation study of our proposed ConFDG framework by comparing different ablation models (discarding some parts of the variants). We also evaluate the ablation models on these two benchmark datasets HAR and WISDM. The ablation models used for the evaluations are (1) w/o UDA, (2) w/o $L_{FC}\&L_{FF}$, (3) w/o $L_{CL}$, (4) w/o $L_{CL}\&L_{FC}\&L_{FF}$ and (5) w/o $L_{domain}$. Similar to Sec. F, we randomly initialize each model 10 times and take the mean values as the results. Table. 10 shows the ablation study results on HAR and Table. 11 shows the results on WISDM. We use both Accuracy and Macro-F1 as the evaluation metrics. The results prove the effectiveness and accuracy of our proposed complete ConFGD and ConFGD+.

## K $Top\_k$ Study of Encoder Edge Embedding $\mathbf{v}_{edge}$

This section elaborates on the experiments of different numbers of $Top\_k$. In the $\mathbf{v}_{edge}$, there are 10 nodes. Therefore, 10 $Top\_k$ means using all the edge information for the aggregating, and 5 means only using the top 5 important edge embeddings into the aggregating and setting the remaining entries into 0. Therefore, a smaller $Top\_k$ number means higher computational efficiency. Table. 12 and Table. 13 are the evaluation results for the $Top\_k$ study on the benchmark dataset HAR and WISDM respectively. Similar to Sec. F, we randomly initialize each model 10 times and take the mean values as the results. We also apply both Accuracy and Macro-F1 for the evaluation.

The results show that when the choice of $Top\_k$ is changed, our proposed model still archives very stable and significant performance. In addition, choosing the top 7 embeddings for each pair of nodes achieved 3 best performances out of 4 evaluation metrics. In addition, the standard deviation is very small (HAR: Accuracy: 0.328% of the best average value

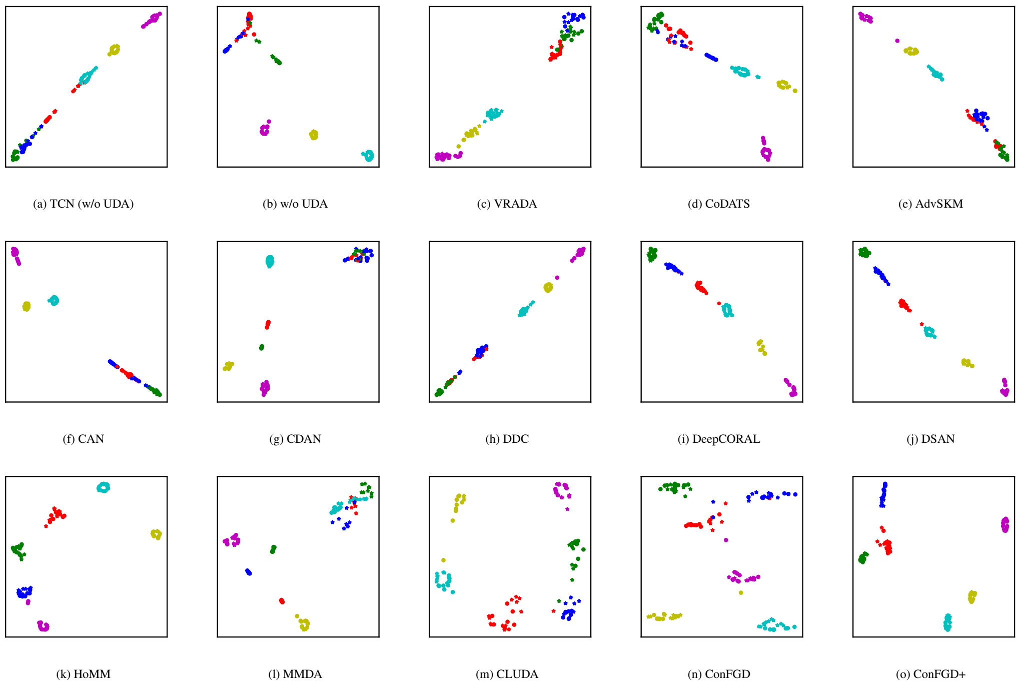


Figure 12: Embedding t-SNE visualizations of all the baseline models on the benchmark dataset HAR (Anguita et al. 2013). The classes are differentiated by colors. The "star" shapes are from the source domain and the "round" shapes are from the target domain

| Model | 1→2 | 2→5 | 13 → 19 | 15 → 19 | 18 → 21 | 19 → 25 | 20 → 1 | 23 → 13 | 24 → 22 | 25 → 24 | Avg. | Std. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| w/o UDA | 82.61 | 54.45 | 83.33 | 96.3 | 91.94 | 59.68 | 94.34 | 66.00 | 87.76 | 89.66 | 80.61 | 15.07 |
| w/o $L_{FC}$&$L_{FF}$ | 86.96 | 78.26 | 98.15 | 98.15 | 90.32 | 77.42 | 98.11 | 90.00 | 100.00 | <u>98.28</u> | 91.57 | 8.49 |
| w/o $L_{CL}$ | 86.96 | 78.26 | 88.89 | 96.30 | 88.71 | 75.81 | 94.34 | 86.00 | 100.00 | 96.55 | 89.18 | 7.90 |
| w/o $L_{CL}$&$L_{FC}$&$L_{FF}$ | 86.96 | 78.26 | 88.89 | 96.30 | 88.71 | 85.48 | 98.11 | 84.00 | 98.28 | 94.83 | 89.98 | 6.70 |
| w/o $L_{domain}$ | 78.26 | 76.09 | 85.19 | 96.30 | 90.32 | 67.74 | 90.57 | 74.00 | 98.15 | <u>98.28</u> | 85.49 | 10.94 |
| w/o PE | 76.09 | 75.81 | 83.33 | 94.34 | 85.48 | 74.00 | 90.32 | 80.00 | 95.92 | 98.11 | 85.34 | 8.93 |
| ConFGD | <u>91.30</u> | <u>86.96</u> | <u>94.44</u> | **98.15** | **98.15** | <u>95.16</u> | <u>98.11</u> | <u>88.00</u> | **100** | <u>98.28</u> | <u>94.86</u> | 4.63 |
| ConFGD+ | **91.50** | **89.13** | **100** | **98.15** | **98.15** | **100** | **100** | **96.00** | **100** | **100** | **96.81** | 3.87 |
| w/o UDA | 83.12 | 45.32 | 82.88 | 97.14 | 87.64 | 56.38 | 94.09 | 60.29 | 81.25 | 88.75 | 77.69 | 17.46 |
| w/o $L_{FC}$&$L_{FF}$ | 84.61 | 76.98 | 98.59 | 97.44 | 90.94 | 75.02 | 97.74 | 89.03 | 100.00 | 95.95 | 90.63 | 9.10 |
| w/o $L_{CL}$ | 82.95 | 75.93 | 90.62 | 94.58 | 88.36 | 73.88 | 94.03 | 83.89 | 100.00 | 96.06 | 88.03 | 8.71 |
| w/o $L_{CL}$&$L_{FC}$&$L_{FF}$ | 84.61 | 76.88 | 90.62 | 95.98 | 88.35 | 84.99 | 97.74 | 81.49 | 95.95 | 94.02 | 89.06 | 6.99 |
| w/o $L_{domain}$ | 75.34 | 69.03 | 85.47 | 94.22 | 90.94 | 63.19 | 90.60 | 69.02 | 97.44 | 98.04 | 83.33 | 13.03 |
| w/o PE | 76.88 | 73.88 | 76.56 | 94.03 | 86.41 | 69.02 | 90.60 | 76.77 | 94.49 | 97.74 | 83.64 | 10.18 |
| ConFGD | <u>91.22</u> | <u>86.18</u> | <u>97.44</u> | <u>97.44</u> | <u>97.44</u> | <u>94.97</u> | <u>98.44</u> | <u>86.40</u> | **100** | <u>98.91</u> | <u>94.84</u> | 5.12 |
| ConFGD+ | **91.33** | **88.35** | **100** | **98.59** | **97.50** | **95.00** | **100** | **95.54** | **100** | **100** | **96.63** | 4.09 |

Table 10: Ablation Results on HAR Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 10 sets of random initialization (Higher is better. The best is in **bold** and the second best is marked with <u>underline</u>).

94.58, Macro-F1: 0.18% of the best average value 93.79; WISDM: Accuracy: 0.513% of the best average value 78.04, Macro-F1: 3.25% of the best average value 93.96). Thus, our model offers an option to enhance computational efficiency without sacrificing accuracy.

## L Limitations

This paper has two primary limitations. Firstly, based on the runtime results of different models, a smaller $Top\_K$ significantly reduces execution time. In our evaluations, we fixed the frequency decomposition components at 5, with 10 nodes each. We opted for 5 frequency decompositions (filter level is 4db) instead of a larger number because, for finite-length signals, higher levels of filtering can cause a

| Model | 2→25 | 7→2 | $7 \rightarrow 12$ | $7 \rightarrow 26$ | 10→ 25 | $12 \rightarrow 19$ | $12 \rightarrow 7$ | $13 \rightarrow 2$ | $19 \rightarrow 2$ | $28 \rightarrow 20$ | Avg. | Std. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| w/o UDA | 65.79 | 63.41 | 63.64 | **78.05** | 68.42 | 54.55 | 56.25 | 51.22 | 51.22 | 68.29 | 62.08 | 8.68 |
| w/o $L_{FC}$&$L_{FF}$ | 71.05 | 65.85 | 59.09 | 75.61 | 65.79 | 54.55 | 64.58 | 48.78 | **78.05** | 85.37 | 66.87 | 11.10 |
| w/o $L_{CL}$ | 68.42 | $\underline{78.05}$ | $\underline{86.36}$ | 75.61 | 71.05 | **83.33** | 68.75 | 43.90 | 73.71 | $\underline{87.80}$ | 73.70 | 12.57 |
| w/o $L_{CL}$&$L_{FC}$&$L_{FF}$ | 76.32 | 75.61 | 81.82 | **78.05** | 71.05 | 72.73 | **81.25** | 46.34 | **78.05** | 85.37 | 74.66 | 10.83 |
| w/o $L_{domain}$ | 63.16 | $\underline{78.05}$ | 63.64 | 75.61 | 71.05 | 68.18 | 72.92 | 51.22 | 60.98 | 68.29 | 67.31 | 7.90 |
| w/o PE | 76.32 | 63.41 | 81.82 | 75.61 | 52.63 | 77.27 | 72.92 | 48.78 | 70.73 | 60.98 | 68.05 | 11.11 |
| ConFGD | $\underline{78.95}$ | 73.17 | 84.09 | **78.05** | **78.95** | 83.30 | 77.08 | $\underline{70.73}$ | 75.61 | 85.37 | $\underline{78.53}$ | 4.50 |
| ConFGD+ | **79.55** | **82.93** | **87.80** | **78.05** | **78.95** | **83.33** | $\underline{79.17}$ | **71.05** | **78.05** | **87.85** | **80.67** | 5.03 |
| w/o UDA | 40.64 | 48.72 | 55.16 | 42.46 | 47.38 | 39.05 | 33.07 | 33.55 | 52.31 | 67.96 | 46.03 | 10.68 |
| w/o $L_{FC}$&$L_{FF}$ | 40.59 | 51.60 | 39.56 | 43.85 | 37.75 | 45.34 | 50.22 | 32.50 | 59.02 | **83.95** | 48.44 | 14.63 |
| w/o $L_{CL}$ | 41.27 | 60.38 | 82.85 | 43.02 | 41.83 | $\underline{70.03}$ | 65.96 | 26.83 | 60.42 | $\underline{83.37}$ | 57.60 | 18.90 |
| w/o $L_{CL}$&$L_{FC}$&$L_{FF}$ | 43.87 | 61.25 | 81.92 | **54.93** | 40.51 | 50.07 | 75.29 | 35.19 | **61.76** | 80.35 | 58.51 | 16.64 |
| w/o $L_{domain}$ | 39.87 | 68.30 | 56.07 | 42.47 | 40.02 | 43.45 | 76.49 | 30.48 | 55.83 | 55.71 | 50.87 | 14.20 |
| w/o PE | 51.90 | 48.02 | 59.71 | 42.45 | 30.67 | 57.23 | 56.50 | 30.74 | 36.90 | 59.33 | 47.35 | 11.48 |
| ConFGD | $\underline{54.74}$ | $\underline{70.69}$ | $\underline{85.34}$ | 44.95 | **55.00** | **70.50** | **79.37** | $\underline{43.73}$ | 55.94 | 76.20 | $\underline{63.65}$ | 16.16 |
| ConFGD+ | **55.95** | **77.70** | **86.60** | $\underline{45.80}$ | $\underline{51.56}$ | $\underline{70.03}$ | $\underline{77.11}$ | **45.80** | **61.76** | 81.86 | **65.42** | 15.25 |

Table 11: Ablation Results on WISDM Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 10 sets of random initialization (Higher is better. The best is in **bold** and the second best is marked with $\underline{\text{underline}}$).

| Num. of $Top\_k$ | 1→2 | 2→5 | $13 \rightarrow 19$ | $15 \rightarrow 19$ | $18 \rightarrow 21$ | $19 \rightarrow 25$ | $20 \rightarrow 1$ | $23 \rightarrow 13$ | $24 \rightarrow 22$ | $25 \rightarrow 24$ | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 90.52 | 82.61 | **98.15** | **98.15** | 95.16 | 94.34 | 94.34 | **88.00** | **100.00** | 98.28 | 93.96 |
| 7 | 86.96 | **86.96** | 96.30 | **98.15** | **98.15** | 93.20 | **98.11** | **88.00** | **100.00** | **100.00** | **94.58** |
| 10 | **91.30** | **86.96** | 94.44 | **98.15** | 96.77 | **95.16** | **98.11** | 82.00 | **100.00** | **100.00** | 94.29 |
| Avg. | 89.59 | 85.51 | 96.30 | 98.15 | 96.69 | 94.23 | 96.85 | 86.00 | 100.00 | 99.43 | 94.28 |
| Std.dev | 2.31 | 2.51 | 1.86 | 0.00 | 1.50 | 0.98 | 2.18 | 3.46 | 0 | 0.99 | 0.31 |
| 5 | 88.89 | 82.29 | 97.37 | **98.15** | 95.07 | 94.03 | 94.30 | **86.40** | **100.00** | 98.04 | 93.45 |
| 7 | 84.38 | 84.61 | **97.44** | 97.44 | **97.44** | 93.20 | **98.44** | 84.90 | **100.00** | **100.00** | **93.79** |
| 10 | **91.22** | **86.18** | 92.31 | 97.44 | 94.84 | **94.97** | 97.74 | 81.49 | **100.00** | **100.00** | 93.62 |
| Avg. | 88.16 | 84.36 | 95.71 | 97.68 | 95.78 | 94.07 | 96.83 | 84.26 | 100.00 | 99.35 | 93.62 |
| Std.dev | 3.48 | 1.96 | 2.94 | 0.41 | 1.44 | 0.89 | 2.22 | 2.52 | 0 | 1.13 | 0.17 |

Table 12: $Top\_k$ study of encoder edge embedding $\mathbf{v}_{edge}$ on HAR Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 10 sets of random initialization (Higher is better. The best is in **bold**).

| Num. of $Top\_k$ | 2→25 | 7→2 | $7 \rightarrow 12$ | $7 \rightarrow 26$ | 10→ 25 | $12 \rightarrow 19$ | $12 \rightarrow 7$ | $13 \rightarrow 2$ | $19 \rightarrow 2$ | $28 \rightarrow 20$ | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 76.32 | 78.05 | **84.09** | 73.17 | 76.32 | 75.76 | **77.08** | **70.73** | 75.61 | 85.37 | 77.25 |
| 7 | 75.61 | **80.49** | 81.82 | 75.61 | **78.95** | 76.32 | 72.92 | 68.75 | **78.05** | **87.80** | 77.63 |
| 10 | **78.95** | **80.49** | **84.09** | **78.05** | **78.95** | **77.27** | **77.08** | 64.58 | 75.61 | 85.37 | **78.04** |
| Avg. | 76.96 | 79.68 | 83.33 | 75.61 | 78.07 | 76.45 | 75.69 | 68.02 | 76.42 | 86.18 | 77.64 |
| Std.dev | 1.76 | 1.41 | 1.31 | 2.44 | 1.52 | 0.76 | 2.40 | 3.14 | 1.41 | 1.40 | 0.40 |
| 5 | 43.87 | 70.39 | 82.85 | 41.06 | 44.23 | 55.14 | 75.29 | 43.73 | **61.11** | 80.35 | 59.80 |
| 7 | **61.25** | **72.12** | 81.92 | 42.10 | 44.65 | 54.23 | 76.49 | **65.96** | 59.02 | **81.82** | **63.96** |
| 10 | 54.74 | 70.39 | **85.34** | **44.95** | **45.00** | **59.45** | **79.37** | 50.22 | 55.94 | 76.20 | 62.16 |
| Avg. | 53.29 | 70.97 | 83.37 | 42.70 | 44.63 | 56.27 | 77.05 | 53.30 | 58.69 | 79.46 | 61.97 |
| Std.dev | 8.78 | 1.00 | 1.77 | 2.01 | 0.39 | 2.79 | 2.10 | 11.43 | 2.60 | 2.91 | 2.08 |

Table 13: $Top\_k$ study of encoder edge embedding $\mathbf{v}_{edge}$ on WISDM Dataset. Up: Mean Accuracy (%), Down: Mean Macro F1 (%). The results are the mean of the accuracy over 10 sets of random initialization (Higher is better. The best is in **bold**).

boundary effect. This means the coefficients near the edges might contain incorrect information, leading to artifacts or distortions in the reconstructed signal. Additionally, at the boundaries, the absence of data on one side results in incomplete convolution outcomes (Antonino-Daviu et al. 2009). In our case, with an input length of 128, the highest filter level that can avoid boundary effects is 4db, which corresponds to 5 decomposition components. Therefore, the DWT boundary effects are limited to decomposing the input sequence into more components. The second limitation is, that since the benchmark datasets are UDA classification tasks, the LAA module we developed is currently applicable only to classification tasks as well. For other tasks, such as regression, a different specific label prompting strategy should be further designed, which is also part of our future work.