Diverse Intra- and Inter-Domain Activity Style Fusion for Cross-Person Generalization in Activity Recognition

Junru Zhang
Zhejiang University
Hangzhou, China
junruzhang@zju.edu.cn

Lang Feng
Zhejiang University
Hangzhou, China
langfeng@zju.edu.cn

Zhidan Liu
Shenzhen University
Shenzhen, China
liuzhidan@szu.edu.cn

Yuhan Wu
Zhejiang University
Hangzhou, China
wuyuhan@zju.edu.cn

Yang He
Zhejiang University
Hangzhou, China
he_yang@zju.edu.cn

Yabo Dong*
Zhejiang University
Hangzhou, China
dongyb@zju.edu.cn

Duanqing Xu
Zhejiang University
Hangzhou, China
xdq@zju.edu.cn

ABSTRACT

Existing domain generalization (DG) methods for cross-person generalization tasks often face challenges in capturing intra- and inter-domain style diversity, resulting in domain gaps with the target domain. In this study, we explore a novel perspective to tackle this problem, a process conceptualized as domain padding. This proposal aims to enrich the domain diversity by synthesizing intra- and inter-domain style data while maintaining robustness to class labels. We instantiate this concept using a conditional diffusion model and introduce a style-fused sampling strategy to enhance data generation diversity. In contrast to traditional condition-guided sampling, our style-fused sampling strategy allows for the flexible use of one or more random styles to guide data synthesis. This feature presents a notable advancement: it allows for the maximum utilization of possible permutations and combinations among existing styles to generate a broad spectrum of new style instances. Empirical evaluations on a board of datasets demonstrate that our generated data achieves remarkable diversity within the domain space. Both intra- and inter-domain generated data have proven to be significant and valuable, contributing to varying degrees of performance enhancements. Notably, our approach outperforms state-of-the-art DG methods in all human activity recognition tasks.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous computing**; • **Computing methodologies** → **Transfer learning**.

*Corresponding author

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08…$15.00
https://doi.org/10.1145/3637528.3671828

KEYWORDS

Human Activity Recognition; Domain Generalization; Diffusion Model; Domain Padding

ACM Reference Format:
Junru Zhang, Lang Feng, Zhidan Liu, Yuhan Wu, Yang He, Yabo Dong, and Duanqing Xu. 2024. Diverse Intra- and Inter-Domain Activity Style Fusion for Cross-Person Generalization in Activity Recognition. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3637528.3671828

1 INTRODUCTION

Human activity recognition (HAR) is a crucial application of time series data collected from wearable devices like smartphones and smartwatches, garnering substantial attention in recent years [6, 22, 55]. Deep learning (DL) techniques have proven effective in time series classification (TSC) for HAR tasks [53, 64, 66]. However, a common assumption underpinning these models is that training and test data distributions are identically and independently distributed (i.i.d.) [48], a condition that does not often hold up in real life due to *individual differences in activity styles* influenced by factors such as age and gender [33, 36]. For instance, sensor data distributions can diverge significantly between younger and older individuals due to variations in walking speed and frequency, leading to challenges in achieving cross-person generalization with standard DL models.

Domain generalization (DG) seeks to address this issue [48]. Approaches such as domain-invariant [1, 10, 13, 32, 33, 68] and domain-specific [5, 30, 49, 65] methods are designed to extract robust inter-domain and intra-domain features that can withstand data distribution shifts across various domains. However, their effectiveness is reliant on the diversity and breadth of the training data [56]. The challenge arises in HAR tasks, where the collected training data is often small-scale and lacks the necessary diversity due to resource constraints on edge devices [36, 50]. This inherent *diversity scarcity in source domain training data* can lead to overfitting to local and narrow inter- or intra-domain features, resulting in poor generalization to new, unseen domains. As shown in Fig. 1

(a) and (b), the learned features lack required intra- or inter-domain feature robustness, thereby impeding their generalization to target domains (red circles).

One promising solution is to enrich training distributions by data generation. Recent research [36] has focused on enhancing training data richness through standard data augmentation like rotation and scaling; however, it primarily enhances *intra-domain diversity and falls short of addressing inter-domain variability*. As shown in Fig. 1 (c), the augmented data (stars) for source domains (orange and blue circles) tends to cluster tightly, yet fails to generate the necessary inter-domain data. The target domain (red circles) thus cannot be comprehensively represented.

In this work, we focus on the generation of highly diverse data distributions to address the issue of limited domain diversity in HAR. We explore a novel perspective to tackle this problem. As depicted in Fig. 1 (d), the core idea involves enabling synthetic data (stars) to fill the empty spaces within and across source domains while maintaining robustness to class labels, a process we conceptualize as "domain padding". For instance, as illustrated in Fig. 1(e), we can combine multiple walking styles of an elderly man and a young man to create a novel inter-domain style or merge multiple walking styles of a young man to generate a new intra-domain style. Compared to existing DG methods, our domain padding holds great potential to generate a more extensive range of unknown style distributions. This enables TSC models to comprehensively explore a wide array of intra- and inter-domain variations, contributing to enhanced generalization in HAR scenarios.

We instantiate our concept using conditional diffusion probabilistic models [16, 41]. To generate samples with instance-level diversity, we first design a contrastive learning pipeline [9]. It aims to extract the activity style representations of the available data in the source domains while preserving their robustness for classification tasks. The resulting style representation, denoted as S_i, can be interpreted as "*a [class] activity performed in [S_i] style*". We then propose a novel style-fused sampling strategy for the diffusion model to achieve domain padding requirements. This involves randomly combining one or multiple style representations of training samples within the same class. Styles in each combination are then utilized to jointly guide the diffusion to generate novel activity samples that fuse the styles. This innovation presents a notable advancement: the randomness of the combination (whether originating from different or the same domains) ensures diversity in both inter-domain and intra-domain, thereby achieving the domain padding, as shown in Fig. 1 (d) and (e). Moreover, it allows for the maximum utilization of possible permutations and combinations among existing styles to generate a broad spectrum of new style instances. Hence, we term our approach as **D**iversified **I**ntra- and **I**nter-domain distributions via activity **S**tyle-fused **Diff**usion modeling (DI2SDiff)[1]. We summarize our main contributions as follows:

- We explore a pivotal challenge hampering the effectiveness of current DG methods in HAR: diversity scarcity of source domain features. In response, we introduce the concept of "domain padding", offering a fresh perspective for enhancing domain diversity and ultimately improving DG models' performance.

[1]Our code is available at https://github.com/jrzhang33/DI2SDiff.

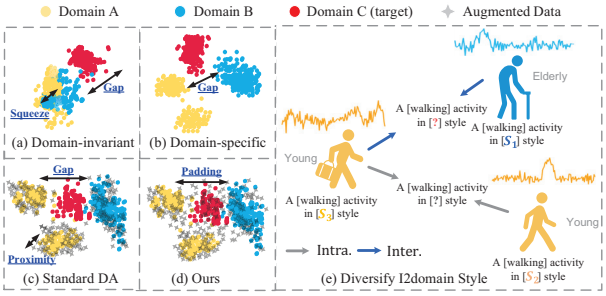


Figure 1: T-SNE visualization [46] of time-series features extracted by various methods across three domains in HAR. Existing representation learning methods result in domain gaps as in both (a) and (b), covering a small portion of target domain (red circles). Standard data augmentation (DA) leads to augmented data (stars), with source domains (orange/blue circles) remaining in close proximity to each other and failing to fill gaps. Our method (d) creates a comprehensive feature space by padding domain gaps via the idea of (e).

- We propose to use activity style features as conditions to guide the diffusion process, extending the information available at the instance-level beyond mere class labels.
- We propose a novel style-fused sampling strategy, which can flexibly fuse one or more style conditions to generate new, unseen samples. This strategy achieves data synthesis diversity both within and across domains, enabling DI2SDiff to instantiate the concept of domain padding.
- We conduct extensive empirical evaluations of DI2SDiff across a board of HAR tasks. Our findings reveal that it markedly diversifies the intra- and inter-domain distribution without introducing class label noise. Leveraging these high-quality samples, DI2SDiff outperforms existing solutions, achieving state-of-the-art results across all cross-person activity recognition tasks.

2 RELATED WORK

Human activity recognition (HAR) uses sensors like ECGs for recognizing activities in healthcare and human-computer interaction [7, 54]. The complexity of daily activities, varying among individuals with different personal styles, makes recognition challenging [33, 64]. With the rise of deep learning, deep neural networks have been increasingly utilized to extract informative features from activity signals [20, 59, 62, 64]. For example, DeepConvLSTM incorporates convolutional and LSTM units for multimodal wearable sensors [34]. MultitaskLSTM extracts features using shared weights, then classifies activities and estimates intensity separately [3].

Domain generalization (DG) aims to improve model performance across different domains. Early works [1, 10, 13, 32, 68] focused on utilizing multiple source domains and enforcing domain alignment constraints to extract robust features. For example, DANN [1] employed adversarial training to accomplish this task, but it requires target data during training. To recall more beneficial features, several methods [5, 30, 49, 65] such as mDSDI [5] have been proposed to preserve domain-specific features. Another line

of research in DG focuses on data augmentation techniques [14, 15, 23, 47, 58, 67] to explore more robust patterns for improved generalization, such as generating adversarial examples [14].

Given the practical significance of DG learning for HAR tasks, researchers [27, 33, 35, 36, 52] have turned their focus to studying DG problems in this field. For instance, Wilson et al. [52] proposed an adversarial approach to learn domain-invariant features, which requires labeled data in the target domain during training. Qian et al. [33] improved variational autoencoder (VAE) framework [21] to disentangle domain-agnostic and domain-specific features automatically, but domain labels are required. DDLearn [36] is a recent advanced approach that enriches feature diversity by contrasting augmented views but is limited to standard augmentation techniques that only enrich intra-domain features.

Diffusion models have showcased their remarkable potential in generating diverse and high-quality samples in various domains, like computer vision [28], natural language processing [24], and decision-making [11]. Furthermore, classifier-free guidance models [17] have achieved impressive outcomes in multimodal modeling, with wide applications in tasks such as text-to-image synthesis [39] and text-to-motion [44]. Considering the potential non-stationary distribution of time-series data [18], we propose harnessing the power of diffusion models to generate diverse data in HAR tasks, and thereby enhancing the model's generalization ability. Intriguingly, diffusion models have received limited attention in HAR tasks. A recent survey on time-series diffusion models [25] indicates that although some successful attempts have been made to apply diffusion models to time-series tasks like interpolation [43] and forecasting [4, 37], comprehensive investigations in time-series generation tasks are still lacking. Our study not only establishes diffusion models for time-series generation but also guides the diffusion model to produce diverse samples, effectively addressing the challenges of DG in HAR tasks. Our work thus presents a novel and challenging contribution to the field.

3 PRELIMINARIES

3.1 Problem Statement

In cross-person activity recognition [36], a domain is characterized by a joint probability distribution $P_{X,Y}$ across the product space of time-series instances $\mathcal{X}$, and the corresponding label space $\mathcal{Y}$. Each instance $\mathbf{X}_i \in \mathbb{R}^{K\times L}$ represents the values of each time series obtained from sensors, where K is the dimensionality of features, and L is the temporal length of the series. Moreover, each instance $\mathbf{X}_i$ corresponds to an activity class label $y_i \in \{1, 2, \cdots, C\}$, indicating the specific activity category performed by the subjects, with C denoting the total number of activity categories. The domain generalization challenge lies in the nonintersection and domain differences between the training and testing sets. Typically, the training set $D^s = \{(\mathbf{X}_i, y_i)\}_{i=1}^{n^s}$ is collected from the labeled source-domain subjects, where n^s represents the number of training instances. Importantly, n^s is often small in cross-person activity recognition scenarios, presenting the *small-scale* challenge. On the other hand, the test set $D^t = \{(\mathbf{X}_i, y_i)\}_{i=1}^{n^t}$ consists of n^t instances obtained from the unseen target-domain subjects and satisfies the condition $D^s \cap D^t = \emptyset$. In addition, source and target domains have different joint probability distributions while sharing the identical feature space and class label space, i.e., $P^s(\mathbf{X}_i, y_i) \neq P^t(\mathbf{X}_i, y_i)$, and $\mathcal{X}^s = \mathcal{X}^t$, $\mathcal{Y}^s = \mathcal{Y}^t$. The primary objective is to leverage the available data in D^s to train a TSC model $f : \mathcal{X} \to \mathcal{Y}$ capable of effectively generalizing to an inaccessible, unseen test domain D^t, without any prior exposure to target domain data or domain labels during training. This task is inherently more challenging than conventional transfer learning settings [33, 36] due to the disparate distributions across source and target domains, compounded by the small-scale settings of the training data.

3.2 Diffusion Probabilistic Model

Diffusion model [41] involves training a model distribution $p_\theta(x)$ to closely approximate the target ground-truth data distribution $q(x)$. It assumes distribution $p_\theta(x)$ as a Markov chain of Gaussian transitions: $p_\theta(x_0) = \int p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) dx_{1:K}$, where $x_1, \ldots, x_T$ denote the latent variables with the same dimensionality as original (noiseless) data x_0. $p_\theta(x_T) \sim \mathcal{N}(0, \mathbf{I})$ is the Gaussian prior. $p_\theta(x_{t-1}|x_t)$ is the trainable reverse process given by

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)). \tag{1}$$

Diffusion predefines a forward process that progressively adds Gaussian noise to x_0 in T steps, defined as

$$q(x_t|x_{t-1}) := \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}\right), \tag{2}$$

where $\beta_t \in (0, 1)$ is variance schedule for noise control.

Training procedure. The diffusion model's parameters θ are optimized by maximizing the evidence lower bound of the log-likelihood of the data, i.e., $\log p_\theta(x_0)$, which can be further simplified as a surrogate loss [16]:

$$\mathcal{L}(\theta) := \mathbb{E}_{x_0, t\sim\mathcal{U}, \epsilon\sim\mathcal{N}(0,\mathbf{I})}\left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right], \tag{3}$$

where $\mathcal{U}$ is the uniform distribution and the noise predictor $\epsilon_\theta(x_t, t)$, parameterized with a deep neural network, aims to estimate the noise ϵ at time t given x_t. As $\mu_\theta(x_t, t)$ is determined by $\epsilon_\theta(x_t, t)$, the target $p_\theta(x_{t-1}|x_t)$ can be consequently derived.

Sampling procedure. Given a well-trained p_θ, the data generation procedure begins with a Gaussian noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and proceeds by iteratively denoising x_t for $t = T, \cdots, 1$ through $p_\theta(x_{t-1}|x_t)$, culminating in the generation of the new data x_0.

4 DOMAIN PADDING

A major obstacle to achieving domain generalization in HAR tasks is the limited data diversity of the source domain. This presents representation learning methods from extracting robust features to distribution shifts across domains. Moreover, data augmentation also shows insufficient data richness within the domain space, particularly the inter-domain distribution.

In response, our work aims to achieve highly diverse data generation to enrich the training distributions. We propose a novel perspective, which we refer to as "domain padding". The core idea is to achieve the richness of the domain space by "padding" the distributional gaps within and between source domains, as demonstrated in Fig. 1 (d). To ensure the generation of high-quality, diverse data that can effectively augment the training datasets for HAR models, domain padding adheres to two key criteria:

- **Class-Preserved Generation**: The generated data should maintain alignment with the original data in terms of class labels, ensuring consistency.
- **Intra- and Inter-Domain Diversity**: The generated data should not only boost diversity within individual domains (i.e., intra-domain diversity) but should also enrich distribution between distinct domains (i.e., inter-domain diversity).

The first criterion ensures that the enhanced diversity does not compromise the semantic integrity of the data. By maintaining consistency in class labels, domain padding contributes meaningfully to model learning without introducing label noise or confusion. The second criterion guarantees that the models are exposed to a wide range of domain variations, thereby enhancing their robustness against shifts in data distribution.

5 METHODOLOGY

We implement domain padding using conditional diffusion models given their highly-expressive generative capabilities [16, 60]. The iterative denoising process of diffusion models makes them exceptionally suited for flexible conditioning mechanisms. In this framework, given the original dataset D^s, we generate a new sample $\tilde{x}_0 \sim \tilde{\mathcal{X}}^s$ using conditional information $s \in \mathcal{X}^{\text{cond}}$ to guide the generation process. The ensemble of all generated data constitutes a synthetic dataset, denoted as $\tilde{D}^s = \{(\tilde{\mathbf{X}}_\mathbf{i}, y_i)\}_{i=1}^{\tilde{n}^s}$, with $\tilde{n}^s$ denoting the total count of generated samples. Next, we use $\tilde{x}_0$ to denote an example of the synthetic samples. The generation objective is to estimate the conditional data distribution $q(\tilde{x}|s)$. This allows us to generate a synthetic sample $\tilde{x}_0$ given a specific constraint s. The conditional diffusion process can be described by:

$$q(\tilde{x}_t|\tilde{x}_{t-1}, s), \quad p_\theta(\tilde{x}_{t-1}|\tilde{x}_t, s). \tag{4}$$

Sequentially performing p_θ enables the generation of new samples to capture the attributes of s. However, realizing domain padding is not a trivial task due to a key aspect: *how to guide the diffusion model to generate diverse activity samples meeting two criteria of domain padding.*

In the following, we introduce the DI2SDiff framework, designed to enable diffusion to achieve domain padding. In §5.1, we present a contrastive learning pipeline that extracts style features to serve as conditions for the diffusion model. Given a style condition, we employ classifier-free guidance [17] to generate new samples that meet the first criterion in §5.2. For the second criterion, we construct a diverse style combination space for the condition space $\mathcal{X}^{\text{cond}}$ and introduce a style-fused sampling strategy to generate highly diverse intra and inter-domain data in §5.3. We finally provide the workflow of DI2SDiff in §5.4.

5.1 Activity Style Condition

Conditional diffusion models are typically guided by label or text prompts that provide task-specific knowledge, such as "create a [cartoonish] [cat] image" [31, 51, 61]. However, the generation of instance-level time-series data introduces distinct challenges. It is difficult to capture the complex patterns solely through label or text prompts due to the inherently high-dimensional and non-stationary nature [18]. To address this issue, we propose the development of a style conditioner using a contrastive learning approach [9]. This approach has demonstrated robustness in extracting representations from unlabeled time-series data. The transformed data can retain the distinctive characteristics of the original data while preserving the semantic information of the classes. Thus, it is well-suited for extracting robust *instance-level representation*, termed as "*style*", which can serve as conditions to guide diffusion models.

Delving into specifics, the contrastive learning pipeline consists of a feature encoder and Transformer on the available training data. The objective is to maximize the similarity between different contexts of the same sample and minimize the similarity between contexts of different samples. Once the module is trained, we utilize it as *style conditioner* denoted as f_{style}. When extracting the style from the original data $\mathbf{X}_i$, the style conditioner produces a style vector $S_i = f_{\text{style}}(\mathbf{X}_i) \in \mathbb{R}^H$, where H denotes the length of the vector. Consequently, each activity style condition can be interpreted as "*a [y_i] activity performed in [S_i] style*", where y_i denotes the class of the original data. This approach takes an important step towards the first criteria of domain padding due to the preservation of class semantics. The aggregation of all context vectors from n^s training instances constitutes a set $\mathcal{S} = \{S_i\}_{i=1}^{n^s}$, which can be further divided into C class-specific subsets corresponding to C classes. Each subset contains style vectors pertaining to a specific class, expressed as $\mathcal{S} = \{\mathcal{S}^1 \cup \mathcal{S}^2 \cup \cdots \cup \mathcal{S}^C\}$. In Appendix A, we provide the details of the contrastive learning approach [9].

5.2 Synthesizing with Classifier-Free Guidance

To control the generation of time-series samples, we can leverage the style in $\mathcal{S}$ to guide the conditional sampling process $p_\theta(\tilde{x}_{t-1}|\tilde{x}_t, s)$ presented in Eq. (4). To this end, we adopt the classifier-free guidance [17], which has proven to be effective in generating data with specific characteristics. In this framework, the training process is modified to learn a conditional $\epsilon_\theta(\tilde{x}_t, t, s)$ and an unconditional $\epsilon_\theta(\tilde{x}_t, t, \emptyset)$, where $\emptyset$ symbolizes the absence of the condition s. The loss function is formulated as follows:

$$\mathcal{L}(\theta) := \mathbb{E}_{x_0 \sim \mathcal{X}^s, \epsilon \sim \mathcal{N}(0,\boldsymbol{I}), t \sim \mathcal{U}, s \sim \mathcal{S}} \left[\|\epsilon - \epsilon_\theta(\tilde{x}_t, t, s)\|^2 \right], \tag{5}$$

where condition s is one style feature in $\mathcal{S}$ derived from the pre-trained conditioner, and it is randomly dropped during the training.

During the sampling phase, a sequence of samples $\tilde{x}_T, \ldots, \tilde{x}_0$ is generated starting from $\tilde{x}_T \sim \mathcal{N}(0, \mathbf{I})$. For each timestep t, the model refines the process of denoising $\tilde{x}_{t-1}$ based on $\tilde{x}_t$ through the following operation:

$$\hat{\epsilon}_\theta = \epsilon_\theta(\tilde{x}_t, t, \emptyset) + \omega\big(\epsilon_\theta(\tilde{x}_t, t, s) - \epsilon_\theta(\tilde{x}_t, t, \emptyset)\big), \tag{6}$$

where ω is a scalar hyperparameter that controls alignment between the guidance signal and the sample [17]. Through the iterative application of Eq. (6), the diffusion model is capable of sampling new time-series samples that conform to specific styles $s \in \mathcal{S}$. It is worth noting that the styles $S_1, \ldots, S_{n^s}$ are robust to the class labels, the generation process guarantees the first criterion of domain padding: each generated sample belongs to a known class under the guidance of a single condition.

5.3 Beyond One Activity Style

So far our approach has not yet achieved the second criterion of domain padding, as samples conditioned on a singular style $s \in \mathcal{S}$

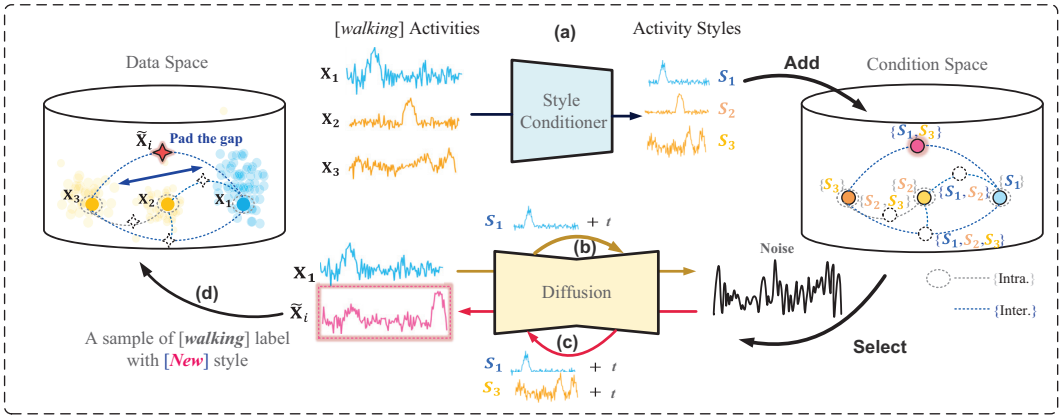


Figure 2: Illustration of the diffusion within DI2SDiff. It contains a style conditioner to produce styles and a conditional diffusion for data generation. Suppose we have three original walking samples: X_1, X_2, and X_3, where X_1 is from a different domain while X_2 and X_3 come from the same domain. (a) The style conditioner generates style features from the original data. The style features are randomly combined to build the condition space, in which the combination of inter-domain styles is indicated by blue brackets and the combination of intra-domain styles is indicated by grey brackets. (b) During training, the diffusion retrieves each data sample with one style for the forward process. (c) During sampling, the diffusion receives noise and a style combination, e.g., $[S_1, S_3]$, for the reverse process. (d) The generated sample $\tilde{X}_i$ is used to diversify the data space.

could demonstrate a limited range of variation within the intra-domain space. Therefore, we then propose a style-fused sampling strategy to further enhance the diversity. This strategy guides the diffusion to generate new data that satisfy any number and combination of styles (conditions), rather than just one. By doing so, the generated data fuse diverse inter and intra-domain styles, effectively meeting the second criterion of domain padding.

Random style combination. Random style combination entails the ensemble of *one or multiple* style features under a unified class to establish a new diffusion sampling condition. Importantly, the ensemble styles must belong to the same class to preserve class consistency. For each class label c, we randomly select any number of style features from the class-specific set $\mathcal{S}^c$ and combine them in all possible ways. This will end up with $2^k - 1$ different style combinations (excluding the empty set), where $k = |\mathcal{S}^c|$ is the number of styles in $\mathcal{S}^c$. Mathematically, the collection of all possible style combinations for class c can be expressed by the power set $\mathcal{P}(\mathcal{S}^c)$ of $\mathcal{S}^c$:

$$\mathcal{P}(\mathcal{S}^c) = \{\mathcal{D}_j | \mathcal{D}_j \subseteq \mathcal{S}^c, \mathcal{D}_j \neq \emptyset\} \tag{7}$$

For instance in Fig. 2 (c), three styles in $\mathcal{S}^c = \{S_1, S_2, S_3\}$ results in 7 different combinations[2]. This operation is replicated across all classes $1, \ldots, C$, integrating them into a comprehensive style combination set $\mathcal{D} = \{\mathcal{P}(\mathcal{S}^1) \cup \cdots \cup \mathcal{P}(\mathcal{S}^C)\}$. The randomness in selecting style combinations can significantly foster diversity within and between domains, and maximize the exploitation of existing styles to generate highly diverse condition space $\mathcal{X}^{\text{cond}}$.

Style-fused sampling. Subsequently, our efforts are directed towards empowering the diffusion model to fuse multiple styles during the data generation conditioned on a specific style combination $\mathcal{D}_j \in \mathcal{D}$. Assuming the diffusion has learned the data distributions $\{\epsilon_\theta(\tilde{x}_t, t, s)\}_{i=1}^{n_s}$ through Eq. (5), sampling from the composed data distribution $q(\tilde{x}_0|\mathcal{D}_j)$ for any given style combination $\mathcal{D}_j \in \mathcal{D}$ is achieved using the below perturbed noise:

$$\hat{\epsilon}_\theta = \epsilon_\theta(\tilde{x}_t, t, \emptyset) + \omega \sum_{s \in \mathcal{D}_j} \Big(\epsilon_\theta(\tilde{x}_t, t, s) - \epsilon_\theta(\tilde{x}_t, t, \emptyset)\Big). \tag{8}$$

The derivation of Eq. (8) is provided in Appendix B. This indicates that while the diffusion training process primarily focuses on an individual style, we can flexibly combine these styles during sampling. For instance, consider the combination of $\mathcal{D}_j = \{S_1, S_3\}$ in Fig. 2 (c) and (d). Each element represents a style associated with the [*walking*] activity. Eq. (8) can generate new samples with the [*walking*] label possesses unique characteristics that fuse these two styles. This is critical for inter and intra-domain diversity in domain padding: diffusion can flexibly incorporate class-specific instance-level styles from different or the same domains to generate new samples with novel domain distribution. Moreover, given the existence of sub-domains within each domain, our diffusion model is capable of synthesizing novel domains, even from sampling instances within the same domain (we verify this later in the experiments).

5.4 Workflow of DI2SDiff

Finally, we elaborate on the comprehensive workflow of our approach, which we refer to as **D**iversified **I**ntra- and **I**nter-domain distributions via activity **S**tyle-fused **Diff**usion modeling (DI2SDiff).

Architectural design. The diffsuion model $\epsilon_\theta : \tilde{\mathcal{X}}^s \times \mathbb{N} \times \mathcal{X}^{\text{cond}} \to \tilde{\mathcal{X}}^s$ is built upon a UNet architecture [17] with repeated convolutional residual blocks. To accommodate the characteristics of time series input, we adapt 2D convolution to 1D temporal convolution. The model incorporates a timestep embedding module and

[2] $\mathcal{P}(\mathcal{S}^c) = \{\{S_1\}, \{S_2\}, \{S_3\}, \{S_1, S_2\}, \{S_1, S_3\}, \{S_2, S_3\}, \{S_1, S_2, S_3\}\}$

a condition embedding module, each of which is a multi-layer perceptron (MLP). The condition embedding module is used to encode each activity style $s \in \mathcal{S}$, and in the unconditional case $s = \emptyset$, we zero out the entries of s. These embeddings are then concatenated and fed into each block of the UNet.

Training. During the training stage, as shown in Fig. 2 (a) and (b), the pre-trained style conditioner extracts style features $\{S_i\}_{i=1}^{n^s}$ for the training instances $\{\mathbf{X_i}\}_{i=1}^{n^s}$. Each data instance $\mathbf{X_i}$, paired with its style S_i and a randomly sampled timestep $t \sim \mathcal{U}$, forms a tripartite input $(\mathbf{X_i}, t, S_i)$. This setup facilitates the optimization of the model against a loss function defined by Eq. (5).

Sampling. During the sampling stage, we construct the style combination set $\mathcal{D}$ by Eq. (7). As shown in Fig. 2 (c), a specific style combination $\mathcal{D}_j \in \mathcal{D}$ is then selected to guide the diffusion process, generating the new sample that fuses the styles in $\mathcal{D}_j$. The sampling operates under single-condition guidance when the style combination comprises a single style, i.e., $|\mathcal{D}_j| = 1$. Conversely, when the style combination comprises multiple styles, i.e., $|\mathcal{D}_j| > 1$, the sampling proceeds under multiple-condition guidance.

Domain space diversity. Through the iterative execution of the sampling procedure, we can generate a diverse range of new, unseen samples that meet domain padding criteria. These synthetic samples collectively form a synthetic dataset $\tilde{D}^s$ for TSC models' training. This process involves two hyperparameters κ and o. κ denotes the proportion of synthetic to original training samples, effectively managing the volume of synthetic samples. o denotes the maximum number of style features that can be combined in each style set.

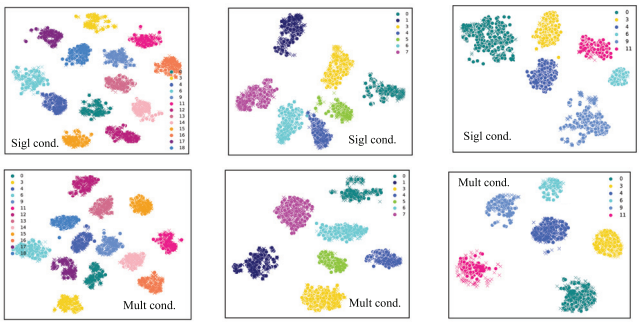
Training TSC model. Utilizing the synthetic dataset $\tilde{D}^s$, we are able to augment the training dataset to $\{\tilde{D}^s \cup D^s\} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{n^s+\tilde{n}^s}$. This augmented dataset can be straightforwardly utilized for standard TSC task training. To extract more value from the dataset, we consider the diversity learning strategy from [36] for the TSC model's training. The main thought goes beyond just minimizing not only the standard cross-entropy loss for correct classification. It also involves minimizing the additional cross-entropy loss to effectively differentiate between synthetic and original samples. For more details, please refer to Appendix C and Appendix D.

6 EXPERIMENTS

In this section, we conduct a comprehensive evaluation of DI2SDiff across various cross-person activity recognition tasks to demonstrate **(1)** its ability to achieve domain padding and significantly diversify the domain space; **(2)** its outstanding performance in domain generalization; **(3)** a detailed ablation and sensitivity analysis; and **(4)** its versatility in boosting existing DG baselines.

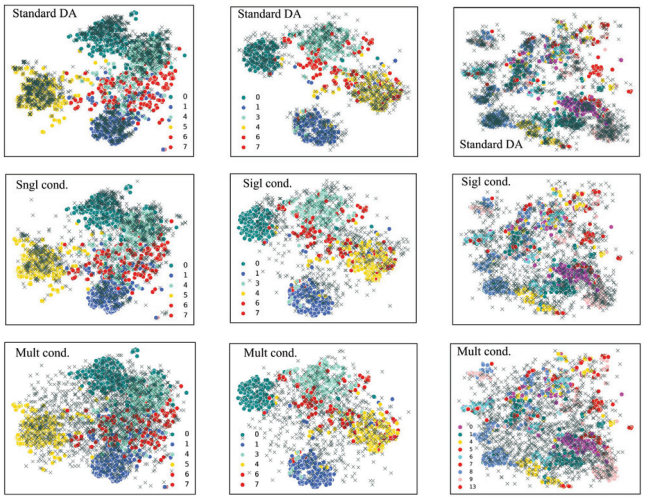
6.1 Experimental Setup

Datasets. We assess our method on three widely used HAR datasets: UCI Daily and Sports Dataset (DSADS) [2], PAMAP2 dataset [38] and USC-HAD dataset [63]. We follow the same experimental settings in [36] that provided a generalizable cross-person scenario. Specifically, the subjects are organized into separate groups for leave-one-out validation. We assign the data of one group as the target domain and utilize the remaining subjects' data as the source domain. Each subject is treated as an independent task.



(a) DSADS (b) PAMAP2 (c) USC-HAD

Figure 3: T-SNE visualization of DSADS, PAMAP2, USC-HAD datasets. Each method generates the same amount of synthetic data. The original and synthetic data are represented by shapes dots and crosses, and each class is denoted by a color. Best viewed in color and zoom in.



(a) DSADS (b) PAMAP2 (c) USC-HAD

Figure 4: T-SNE visualization of DSADS, PAMAP2, and USC-HAR datasets. Each method generates the same amount of synthetic data. Each domain category is represented by a color, and the target domain is represented by a red dot. The original and synthetic data are represented by shapes dots and crosses, respectively. Best viewed in color and zoom in.

Baselines. We compare our approach with a wide range of closely related, strong baselines adapted to TSC tasks. We first select `Mixup` [57], `RSC` [19], `SimCLR` [8], `Fish` [40], and `DDLearn` [36], given their outstanding performance in most recent study [36]. Notably, `DDLearn` [36] is ranked as the top-performing method. We also include `TS-TCC` [9] for its remarkable generalization performance in self-supervised learning. Additionally, we incorporate `DANN` [12] and `mDSDI` [5], which are designed to address domain-invariant and domain-specific feature learning, respectively. In our analysis, the standard data augmentation (DA) techniques [45] are identical to those employed in [36], such as scaling and jittering.

Architecture and implementation. For fairness, we adopt the same feature extractor as described in [36], which consists of two

Table 1: Classification accuracy (%) (± standard deviation) on three public datasets, where each task only comprises 20% of training data. The best results are marked in bold. "T0-T4" represent different cross-person activity recognition tasks.

	Tar	Mixup [57]	RSC [19]	SimCLR [8]	Fish [40]	DANN [12]	mDSDI [5]	TS-TCC [9]	DDLearn [36]	Ours
DSADS	T0	74.77 (±1.76)	54.32 (±2.19)	72.48 (±3.18)	55.06 (±1.60)	72.49(±3.21)	76.91(±2.34)	80.47(±0.53)	87.88 (±1.92)	**89.93** (±2.57)
	T1	75.78 (±3.95)	63.62 (±10.56)	76.61 (±2.56)	62.28 (±3.13)	69.61(±1.96)	76.02(±1.56)	79.68(±0.42)	88.80 (±1.11)	**90.17** (±0.84)
	T2	74.18 (±4.36)	66.48 (±1.80)	78.25 (±0.92)	68.15 (±1.60)	78.97(±4.06)	72.71(±0.98)	84.37(±1.87)	89.21 (±1.23)	**91.39** (±1.31)
	T3	75.85 (±3.45)	64.29 (±3.37)	76.49 (±0.91)	68.83 (±3.83)	78.54(±2.14)	79.58(±1.29)	82.09(±2.51)	85.63 (±1.13)	**88.95** (±1.79)
	Avg	75.15 (±2.36)	62.18 (±4.32)	75.96 (±1.25)	63.58 (±0.37)	74.90(±2.63)	76.31(±1.56)	81.65(±1.33)	87.88 (±0.82)	**90.11** (±1.63)
PAMAP2	T0	57.81 (±0.55)	55.99 (±1.29)	63.28 (±3.33)	58.70 (±4.31)	54.02(±3.52)	42.70(±3.14)	64.08(±1.98)	75.55 (±0.79)	**79.58** (±2.46)
	T1	81.51 (±3.94)	83.08 (±2.42)	81.25 (±1.59)	85.16 (±1.39)	77.21(±3.79)	83.82(±1.62)	86.55(±2.28)	90.07 (±2.40)	**94.12** (±1.20)
	T2	77.34 (±3.33)	78.65 (±3.99)	78.65 (±1.87)	79.69 (±4.00)	78.80(±1.87)	79.15(±2.72)	80.21(±0.52)	85.51 (±0.76)	**89.57** (±2.48)
	T3	70.31 (±5.64)	68.10 (±6.27)	71.09 (±1.99)	72.53 (±0.49)	61.96(±2.11)	78.61(±0.49)	77.32(±0.47)	80.67 (±1.78)	**84.75** (±3.72)
	Avg	71.74 (±1.37)	71.45 (±2.55)	73.57 (±1.21)	72.85 (±0.37)	67.99(±2.66)	75.07(±1.99)	77.04(±1.29)	82.95 (±0.60)	**87.01** (±1.94)
USC-HAD	T0	68.66 (±4.67)	75.69 (±4.28)	69.36 (±2.34)	73.70 (±3.97)	57.79(±4.73)	59.71 (±1.23)	78.96(±0.79)	79.06 (±2.11)	**88.33** (±1.70)
	T1	68.75 (±1.29)	72.40 (±2.88)	66.62 (±1.44)	72.05 (±2.93)	64.95(±2.68)	67.35 (±2.46)	79.55(±1.23)	80.15 (±1.11)	**81.64** (±0.28)
	T2	71.79 (±0.65)	72.83 (±3.62)	76.04 (±1.61)	69.10 (±2.93)	71.97(±3.23)	63.89 (±3.69)	78.15(±2.15)	80.81 (±0.74)	**88.37** (±1.46)
	T3	61.29 (±3.90)	63.19 (±5.30)	61.24 (±1.06)	58.51 (±3.66)	45.65(±2.18)	63.87 (±4.92)	64.35(±1.58)	70.93 (±1.87)	**77.84** (±1.10)
	T4	65.63 (±4.55)	66.75 (±3.25)	62.85 (±2.17)	63.72 (±8.31)	54.94(±3.56)	55.95 (±6.15)	70.25(±0.88)	75.87 (±2.99)	**83.84** (±0.88)
	Avg	67.22 (±2.41)	70.17 (±3.51)	67.22 (±0.39)	67.42 (±3.91)	59.06(±2.65)	62.15 (±3.08)	74.25(±1.16)	77.36 (±0.99)	**84.00** (±1.09)
Avg All		71.37	67.93	72.25	67.95	66.68	71.18	77.39	82.73	**87.06**

blocks for DSADS and PAMAP2, and three blocks for USC-HAD. Each block includes a convolution layer, a pooling layer, and a batch normalization layer. All baselines, except TS-TCC [9], employ this feature extractor. In each experiment, we report the average performance and standard deviation over three random seeds. For detailed experimental setups, including dataset details and training procedures, please refer to Appendix E.

6.2 Domain Padding and Diversity Evaluation

In this part, we demonstrate whether our approach can effectively diversify the domain space and generate diverse samples that meet domain padding criteria. To this end, we adopt T-SNE [46] to visualize the latent feature space in terms of class and domain dimensions.

(1) Class-Preserved Generation. Firstly, we evaluate the class consistency of synthetic data, i.e., the first criterion of domain padding. We employ a class feature extractor, trained with class labels, to map both original and synthetic data into a class-specific latent space. The results of single-condition guidance ($|\mathcal{D}_j| = 1$) and multiple-condition guidance ($|\mathcal{D}_j| > 1$) are shown in Fig. 3.

It can be observed that all synthetic samples (crosses) are closely clustered around their corresponding original instances and classes (dots). This clustering indicates that our method effectively maintains class information, avoiding the introduction of class noise; importantly, this holds true under both single and multiple-condition guidance. Moreover, the use of multiple-condition guidances appears to enhance class discriminability more than single-condition guidance in Fig. 3. This enhancement is likely because more guidance signals provide more robust class semantics (akin to ensemble learning), therefore resulting in a better class alignment.

(2) Intra- and Inter-Domain Diversity. Next, we evaluate the intra- and inter-domain diversity of the synthetic data, i.e., the second criterion of domain padding. We train the domain feature extractor on source domains with domain labels. We then map source and target data into a domain-specific latent space, and compare the synthetic data from the standard DA method, single-condition guidance ($|\mathcal{D}_j| = 1$), and multiple-condition guidance ($|\mathcal{D}_j| > 1$). The results are shown in Fig. 4.

The findings reveal that the standard DA method generates tightly clustered samples (crosses) around the original data (dots), falling short of diversifying the domain space, particularly the inter-domain space. Our single-condition guidance method offers a partial solution and generates sparse data between different domains thanks to diffusion's probabilistic nature. However, relying solely on a single-style guidance approach has limitations for domain padding. The introduction of our style combinations in Eq. (7) makes a substantial improvement: the multiple-condition guidance excels in "padding" the distributional gaps both within and between source domains, as demonstrated in Fig. 4.

Moreover, as indicated in Fig. 4, through multiple-condition guidance, the synthetic samples (crosses) closely resemble the target domain data (red dots) and demonstrate less dependence on the specific characteristics of source domains. This demonstrates that the fusion of multiple style features creates a new style. This is of importance for the domain generalization in TSC. Given the significant differences in individual styles and the small-scale nature of source domain data, our style-fused sampling demonstrates great potential to simulate various new and unseen distributions, from which the TSC model can better adapt to the target domains.

In addition, we can observe in Fig. 4 (c) that the USC-HAD dataset presents an additional challenge of intra-domain gaps due to its fragmented and sparsely distributed source domains with distinct sub-domains. These gaps contribute to an increased distribution shift, posing difficulties for existing DG methods to perform effectively (We show their results in Tab. 1 in later). Through random instance-level style fusion, our approach effectively addresses this sub-domain challenge, enabling the synthesis of new data distribution within sub-domains. As a result, our method can yield exceptional performance on the complex tasks like USC-HAD.

6.3 Generalization Performance

Now we conduct a series of experiments to evaluate the generalization performance of DI2SDiff against other strong DG baselines.

Overall performance. Tab. 1 presents a comparative analysis of the classification accuracies achieved by all DG methods across three datasets, each task of which comprises 20% of the training data. As we can see, representation learning baselines that focus solely on learning domain-invariant features, such as DANN [12],

Table 2: Classification accuracy (%) on three public datasets with varying percentages (%) of used training data.

	Perct.	Mixup	RSC	SimCLR	Fish	DANN	mDSDI	TS-TCC	DDLearn	Ours
DSADS	20%	75.15	62.18	75.96	63.58	74.90	76.31	81.65	87.88	**90.11**
	40%	82.48	67.70	75.76	65.82	75.45	76.55	82.54	89.71	**91.25**
	60%	82.70	69.98	75.61	67.65	76.55	77.89	83.78	90.43	**92.56**
	80%	81.58	75.37	74.69	66.03	76.89	79.25	84.12	90.97	**94.58**
	100%	83.44	75.58	76.22	69.35	80.52	79.58	86.57	91.95	**95.23**
	Avg	81.07	70.16	75.65	66.49	76.86	77.92	83.73	90.19	**92.75**
PAMAP2	20%	71.74	71.45	73.57	72.85	67.99	75.07	77.04	82.95	**87.01**
	40%	76.69	73.73	74.25	77.02	69.85	72.55	78.35	84.34	**87.66**
	60%	77.83	75.72	74.71	76.04	70.88	76.56	80.15	85.03	**88.75**
	80%	78.00	76.17	74.09	75.13	77.82	77.53	81.78	86.67	**89.92**
	100%	79.72	77.96	74.25	75.49	79.56	78.83	83.45	86.31	**90.96**
	Avg	76.80	75.01	74.17	75.31	73.22	76.11	80.15	85.06	**89.32**
USC-HAD	20%	67.22	70.17	67.22	67.42	59.06	62.15	74.25	77.36	**84.00**
	40%	75.30	77.31	69.16	73.54	61.52	68.85	75.32	80.72	**84.97**
	60%	78.14	77.59	71.38	76.09	64.71	76.75	77.84	80.88	**87.53**
	80%	79.76	78.65	71.99	77.21	68.52	77.72	78.91	82.49	**89.25**
	100%	81.27	79.41	72.14	78.92	72.05	78.59	79.15	82.51	**91.13**
	Avg	76.34	76.62	70.38	74.64	65.97	72.81	77.09	80.80	**87.38**

exhibit suboptimal performance due to the limited diversity of the training data in HAR. The method mDSDI [5], on the other hand, achieves improved performance by additionally learning domain-specific features. However, it does not match the performance of DDLearn [36], which utilizes data augmentation, underscoring the importance of training data diversity in enhancing generalization in HAR. In contrast, our DI2SDiff, leveraging advanced synthesis of highly diverse data across both intra- and inter-domain space, markedly surpasses all baseline methods in every task. In addition, we observe that all baselines, including DDLearn, demonstrate poor performance on the USC-HAD dataset. As we discussed in Fig. 4 (c), this decline is due to the presence of sub-domains within the source domain, which poses a highly challenging problem in DG. Nevertheless, DI2SDiff adeptly addresses this issue by integrating instance-level style fusion, thereby synthesizing new data distributions between the sub-domains. As a result, our approach achieves outstanding performance, outperforming the second-best method by a clear margin (6.64%) in USC-HAD.

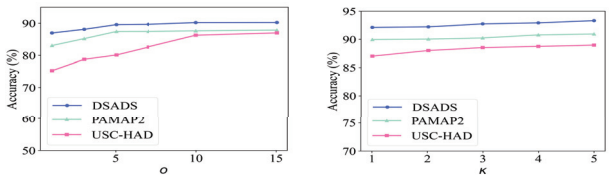
Data proportion analysis. In Tab. 2, we assess DI2SDiff's performance over a range of data volumes by adjusting the proportion of training data from 20% to 100%. The results demonstrate DI2SDiff's consistent superiority over the baseline methods across various proportions of training data. This highlights the ability of our approach to efficiently generate informative samples from varying amounts of available data and effectively learn from them. As the size of the training sample increases, the advantage of our method becomes more pronounced. For instance, as we increase the size from 20% to 100% of USC-HAD, the accuracy improvement grows from 6.64% to 8.62% compared to the second-best baseline (DDLearn). This is because the number of style combinations increases exponentially ($2^k - 1$) with larger training data volumes, as shown in Eq. (7). Hence, enlarging the training dataset can provide significant diversity enhancement of data synthesis, leading to more substantial gains in the model's generalization ability.

6.4 Ablation and Sensitivity Analysis

In this section, we perform an ablation study that focuses on the main step of DI2SDiff, i.e., generating diverse time-series data via diffusion for data augmentation. We keep the number of synthetic samples and the training strategy of TSC models the same for all variants. Additionally, we conduct a sensitivity analysis that focuses on its two hyperparameters: κ, which controls the volume of synthetic data, and o, which controls the maximum number of style features in each style combination.

Table 3: The results of the ablation study on three datasets and each task is averaged for an overall assessment.

Variants	DSADS		PAMAP2		USC-HAD	
	20%	100%	20%	100%	20%	100%
Standard DA	75.58	82.57	70.31	86.41	69.14	75.45
Class Label Guidance	76.25	84.67	72.78	88.52	70.85	76.26
Single Style Sampling	86.98	91.12	83.07	89.64	75.27	83.57
Style-Fused Sampling	**90.11**	**95.23**	**87.01**	**90.96**	**84.00**	**91.13**



(a) o (b) κ

Figure 5: Hyperparameter sensitivity analysis on o and κ.

Ablation on diffusion model. Our findings, as presented in Table 3, indicate that the standard DA method and class label guidance[3] falter in performance. The failure of the class label guidance sampling suggests that using class labels alone, without instance-related information, cannot generate high-quality data. In contrast, the diffusion model that utilizes a single style feature as a condition achieves better performance, suggesting that leveraging representation features for instance-specific sampling can boost the quality of generated data. Moreover, the incorporation of style-fused sampling can further improve generalization by producing samples with distinct features.

Hyperparameter sensitive analysis. We analyze the sensitivity of our hyperparameters by varying one parameter while maintaining the others constant. As shown in Fig. 5, increasing the complexity of style combinations (o) and the volume of synthetic data (κ) generally leads to performance improvement. It becomes non-sensitive when the value is too large. We find that an o value of 5 for the DSADS and PAMAP2 and an o value of 10 for the USC-HAD sufficiently ensure a diverse range of styles. κ values of 1 or 2 strike an effective balance between accuracy and training overhead for all three datasets. By flexibly tuning these hyperparameters, we can achieve even greater performance improvements for the TSC model across various tasks while meeting specific needs.

6.5 Benifits to other DG baselines

We demonstrate the versatility of our approach in boosting the performance of existing DG baselines. The results are shown in Fig. 6. By incorporating our synthetic data into the training datasets of baselines, we consistently observe performance improvements

[3]This involves directly using the class labels, rather than the style features, as the condition to guide diffusion.

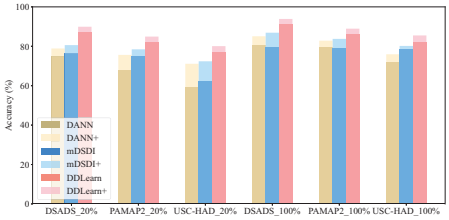


Figure 6: Enhancing the performance of DANN [12], mDSDI [5] and DDLearn [36] with our data generation (+) on 20% and 100% training data in three datasets.

across the board, including DANN [12], mDSDI [5][4], and DDLearn [36][5]. This demonstrates the versatility of integrating our method to provide additional gains, making it a practical solution for immediate application. The diverse synthetic data of DI2SDiff is thus ready for use, offering a straightforward way to bolster various baselines without necessitating further data generation.

7 CONCLUSION

In this paper, we tackle the key issue of DG in cross-person activity recognition, i.e., the limited diversity in source domain. We introduce a novel concept called "domain padding" and propose DI2SDiff to realize this concept. Our approach generates highly diverse inter- and intra-domain data distributions by utilizing random style fusion. Through extensive experimental analyses, we demonstrate that our generated samples effectively pad domain gaps. By leveraging these new samples, our DI2SDiff outperforms advanced DG methods in all HAR tasks. A notable advantage of our work is its efficient generation of diverse data from a limited number of labeled samples. This potential enables DI2SDiff to provide data-driven solutions to various models, thereby reducing the dependence on costly human data collection.

ACKNOWLEDGEMENTS

This work was supported in part by Zhejiang Science and Technology Plan Project under Grant 2023C03183, Key Scientific Research Base for Digital Conservation of Cave Temples (Zhejiang University), China NSFC under Grant 62172284, and Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515010155.

[4]In mDSDI, our synthetic data is treated as a new domain.
[5]In DDLearn, our synthetic data is treated as a new augmentation method.

REFERENCES

[1] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. 2014. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446* (2014).
[2] Billur Barshan and Murat Cihan Yüksek. 2014. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *Comput. J.* 57, 11 (2014), 1649–1667.
[3] Onur Barut, Li Zhou, and Yan Luo. 2020. Multitask LSTM model for human activity recognition and intensity estimation using wearable sensor data. *IEEE Internet of Things Journal* 7, 9 (2020), 8760–8768.
[4] Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. 2022. Modeling temporal data as continuous functions with process diffusion. *arXiv preprint arXiv:2211.02590* (2022).
[5] Manh-Ha Bui, Toan Tran, Anh Tran, and Dinh Phung. 2021. Exploiting domain-specific features to enhance domain generalization. *Advances in Neural Information Processing Systems* 34 (2021), 21189–21201.
[6] Andreas Bulling, Ulf Blanke, and Bernt Schiele. 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)* 46, 3 (2014), 1–33.
[7] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. 2021. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–40.
[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.
[9] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. 2021. Time-series representation learning via temporal and contextual contrasting. *arXiv preprint arXiv:2106.14112* (2021).
[10] Sarah Erfani, Mahsa Baktashmotlagh, Masud Moshtaghi, Xuan Nguyen, Christopher Leckie, James Bailey, and Rao Kotagiri. 2016. Robust domain generalisation by enforcing distribution invariance. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. AAAI Press, 1455–1461.
[11] Lang Feng, Pengjie Gu, Bo An, and Gang Pan. 2024. Resisting Stochastic Risks in Diffusion Planners with the Trajectory Aggregation Tree. *arXiv preprint arXiv:2405.17879* (2024).
[12] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.
[13] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. 2019. Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2477–2486.
[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[15] Kehua Guo, Rui Ding, Tian Qiu, Xiangyuan Zhu, Zheng Wu, Liwei Wang, and Hui Fang. 2023. Single domain generalization via unsupervised diversity probe. In *Proceedings of the 31st ACM International Conference on Multimedia*. 2101–2111.
[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.
[17] Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598* (2022).
[18] Biwei Huang, Kun Zhang, Jiji Zhang, Joseph Ramsey, Ruben Sanchez-Romero, Clark Glymour, and Bernhard Schölkopf. 2020. Causal discovery from heterogeneous/nonstationary data. *The Journal of Machine Learning Research* 21, 1 (2020), 3482–3534.
[19] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. 2020. Self-challenging improves cross-domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 124–140.
[20] Wenchao Jiang and Zhaozheng Yin. 2015. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia*. 1307–1310.
[21] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[22] Oscar D Lara and Miguel A Labrador. 2012. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials* 15, 3 (2012), 1192–1209.
[23] Lei Li, Ke Gao, Juan Cao, Ziyao Huang, Yepeng Weng, Xiaoyue Mi, Zhengze Yu, Xiaoya Li, and Boyang Xia. 2021. Progressive domain expansion network for single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 224–233.
[24] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, Vol. 35. 4328–4343.
[25] Lequan Lin, Zhengkun Li, Ruikun Li, Xuliang Li, and Junbin Gao. 2023. Diffusion models for time-series applications: a survey. *Frontiers of Information Technology*

& Electronic Engineering (2023), 1–23.
[26] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. 2022. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*. Springer, 423–439.
[27] Wang Lu, Jindong Wang, Xinwei Sun, Yiqiang Chen, and Xing Xie. 2022. Out-of-distribution Representation Learning for Time Series Classification. In *The Eleventh International Conference on Learning Representations*.
[28] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. 2022. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11461–11471.
[29] Calvin Luo. 2022. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970* (2022).
[30] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. 2018. Best sources forward: domain generalization through source-specific nets. In *2018 25th IEEE international conference on image processing (ICIP)*. IEEE, 1353–1357.
[31] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *Comput. Surveys* 56, 2 (2023), 1–40.
[32] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. 2013. Domain generalization via invariant feature representation. In *International conference on machine learning*. PMLR, 10–18.
[33] Hangwei Qian, Sinno Jialin Pan, and Chunyan Miao. 2021. Latent independent excitation for generalizable sensor-based cross-person activity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11921–11929.
[34] Meiying Qiao, Shuhao Yan, Xiaxia Tang, and Chengkuan Xu. 2020. Deep convolutional and LSTM recurrent neural networks for rolling bearing fault diagnosis under strong noises and variable loads. *Ieee Access* 8 (2020), 66257–66269.
[35] Xin Qin, Jindong Wang, Yiqiang Chen, Wang Lu, and Xinlong Jiang. 2022. Domain generalization for activity recognition via adaptive feature fusion. *ACM Transactions on Intelligent Systems and Technology* 14, 1 (2022), 1–21.
[36] Xin Qin, Jindong Wang, Shuo Ma, Wang Lu, Yongchun Zhu, Xing Xie, and Yiqiang Chen. 2023. Generalizable Low-Resource Activity Recognition with Diverse and Discriminative Representation Learning. *arXiv preprint arXiv:2306.04641* (2023).
[37] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. 2021. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*. PMLR, 8857–8868.
[38] Attila Reiss and Didier Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th international symposium on wearable computers*. IEEE, 108–109.
[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.
[40] Yuge Shi, Jeffrey Seely, Philip HS Torr, N Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. 2021. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937* (2021).
[41] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*. PMLR, 2256–2265.
[42] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020).
[43] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems* 34 (2021), 24804–24816.
[44] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. 2022. Human motion diffusion model. *arXiv preprint arXiv:2209.14916* (2022).
[45] Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. 2017. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM international conference on multimodal interaction*. 216–220.
[46] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[47] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. 2018. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems* 31 (2018).
[48] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. 2022. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering* (2022).
[49] Shujun Wang, Lequan Yu, Kang Li, Xin Yang, Chi-Wing Fu, and Pheng-Ann Heng. 2020. Dofe: Domain-oriented feature embedding for generalizable fundus image segmentation on unseen datasets. *IEEE Transactions on Medical Imaging* 39, 12 (2020), 4237–4248.
[50] Yucheng Wang, Yuecong Xu, Jianfei Yang, Zhenghua Chen, Min Wu, Xiaoli Li, and Lihua Xie. 2023. Sensor alignment for multivariate time-series unsupervised domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 10253–10261.
[51] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 139–149.
[52] Garrett Wilson, Janardhan Rao Doppa, and Diane J Cook. 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*. 1768–1778.
[53] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186* (2022).
[54] Huatao Xu, Pengfei Zhou, Rui Tan, and Mo Li. 2023. Practically Adopting Human Activity Recognition. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.
[55] Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. 2021. Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 220–233.
[56] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2020. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848* (2020).
[57] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. 2020. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 6502–6509.
[58] Zhenlin Xu, Deyi Liu, Junlin Yang, Colin Raffel, and Marc Niethammer. 2020. Robust and generalizable visual representation learning via random convolutions. *arXiv preprint arXiv:2007.13003* (2020).
[59] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition.. In *Ijcai*, Vol. 15. Buenos Aires, Argentina, 3995–4001.
[60] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2023. Diffusion models: A comprehensive survey of methods and applications. *Comput. Surveys* 56, 4 (2023), 1–39.
[61] Haotian Zhang, Pengchuan Zhang, Xiaowei Hu, Yen-Chun Chen, Liunian Li, Xiyang Dai, Lijuan Wang, Lu Yuan, Jenq-Neng Hwang, and Jianfeng Gao. 2022. Glipv2: Unifying localization and vision-language understanding. *Advances in Neural Information Processing Systems* 35 (2022), 36067–36080.
[62] Junru Zhang, Lang Feng, Yang He, Yuhan Wu, and Yabo Dong. 2023. Temporal Convolutional Explorer Helps Understand 1D-CNN's Learning Behavior in Time Series Classification from Frequency Domain. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3351–3360.
[63] Mi Zhang and Alexander A Sawchuk. 2012. USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*. 1036–1043.
[64] Shibo Zhang, Yaxuan Li, Shen Zhang, Farzad Shahabi, Stephen Xia, Yu Deng, and Nabil Alshurafa. 2022. Deep learning in human activity recognition with wearable sensors: A review on advances. *Sensors* 22, 4 (2022), 1476.
[65] Yi-Fan Zhang, Jindong Wang, Jian Liang, Zhang Zhang, Baosheng Yu, Liang Wang, Dacheng Tao, and Xing Xie. 2023. Domain-Specific Risk Minimization for Domain Generalization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3409–3421.
[66] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. 2019. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing* 115 (2019), 213–237.
[67] Guangtao Zheng, Mengdi Huai, and Aidong Zhang. 2024. AdvST: Revisiting Data Augmentations for Single Domain Generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 21832–21840.
[68] Fan Zhou, Zhuqing Jiang, Changjian Shui, Boyu Wang, and Brahim Chaib-draa. 2020. Domain generalization with optimal transport and metric learning. *arXiv preprint arXiv:2007.10573* 2 (2020).

A DETAILS OF STYLE CONDITIONER

Our style conditioner is adapted from the contrastive module called Time-Series representation learning framework via Temporal and Contextual Contrasting (TS-TCC), proposed in [9]. TS-TCC has demonstrated robust representation learning capability for the HAR task, resulting in each output containing both class-maintained information and unique features of the corresponding time-series instance. This makes it highly suitable for extracting distinctive activity styles from each activity sample. We implemented the model using the official code (https://github.com/emadeldeen24/TS-TCC/). Here is a brief introduction to the model and how to adjust it for our tasks.

Architecture. TS-TCC [9] consists of two components: a feature encoder denoted as f_{enc} and a Transformer denoted as f_{trans}. The feature encoder f_{enc} is a 3-block convolutional architecture. Each block comprises a convolutional layer, a batch normalization layer, and a ReLU activation function. The Transformer f_{trans} primarily consists of successive blocks of multi-headed attention (MHA) followed by an MLP block. The MHA block employs 8 attention heads, and the MLP block is composed of two fully-connected layers with a non-linearity ReLU function and dropout in between. The Transformer stacks L layers to generate the final features, where L is typically set to 4.

Contrastive learning pipeline . TS-TCC [9] uses strong and weak data augmentation techniques to enable contrastive learning of the feature encoder and Transformer. These generate two views for temporal and contextual contrasting, which minimize distance and pull views closer together. Thus, the self-supervised loss combines the temporal and contextual contrastive losses to encourage discriminative representations.

Adaptation for our style conditioner. We adjust the input channel to match the input channel of our training data and set the kernel size to 9. Other components remain consistent with the experimental settings used in the HAR dataset. We maintain the original training settings, such as setting the epoch to 40 and using a Adam optimizer with a learning rate of 3e-4. Once the module is trained, the trained encoder and Transformer are combined to form our style conditioner. When extracting the style from the corresponding original data $\mathbf{X}_i$, the style conditioner produces the representation $Z_i = f_{enc}(\mathbf{X}_i)$ and then outputs the corresponding context vector $S_i = f_{trans}(Z_i) \in \mathbb{R}^H$, where H is its length.

B PROOF

We elaborate on how the conditional diffusion model trained with single style condition $\{S_i\}_{i=1}^{n_s}$ fuse multiple style conditions. From the derivations in [29, 42], we have

$$\nabla_{\tilde{x}_t} \log q(\tilde{x}_t|s) \propto -\epsilon_\theta(\tilde{x}_t, t, s). \tag{9}$$

Hence, single style conditional data distribution $\{q(\tilde{x}_t|S_i)\}_{i=1}^{n_s}$ can be modeled with a singular denoising model $\{\epsilon_\theta(\tilde{x}_t, t, S_i)\}_{i=1}^{n_s}$ that conditions on the respective style S_i.

To integrate multiple styles in a specific sytle combination $\mathcal{D}_j \in \mathcal{D}$, we aim is to model data distribution $q(\tilde{x}_t|\mathcal{D}_j) = q(\tilde{x}_t|\{S_i\}_{i=1}^L)$. Here, for convenience, we use L to denote the number of styles in $\mathcal{D}_j$ and use $\{S_i\}_{i=1}^L$ to denote all styles in $\mathcal{D}_j$. We assume that $\{S_i\}_{i=1}^L$ are conditionally independent given $\tilde{x}_t$. Thus, it can be factorized as follows:

$$q(\tilde{x}_t|\{S_i\}_{i=1}^L) \propto q(\tilde{x}_t) \prod_{i=1}^L \frac{q(\tilde{x}_t|S_i)}{q(\tilde{x}_t)} \quad \text{(Bayes Rule)}$$

$$\Rightarrow \log q(\tilde{x}_t|\{S_i\}_{i=1}^L) \propto \log q(\tilde{x}_t) + \sum_{i=1}^L (\log q(\tilde{x}_t|S_i) - \log q(\tilde{x}_t))$$

$$\Rightarrow \nabla_{\tilde{x}_t} \log q(\tilde{x}_t|\{S_i\}_{i=1}^L) = \nabla_{\tilde{x}_t} \log q(\tilde{x}_t) + \sum_{i=1}^L (\nabla_{\tilde{x}_t} \log q(\tilde{x}_t|S_i) - \nabla_{\tilde{x}_t} \log q(\tilde{x}_t))$$

$$\Rightarrow \epsilon_\theta(\tilde{x}_t, t, \{S_i\}_{i=1}^L) = \epsilon_\theta(\tilde{x}_t, t, \emptyset) + \sum_{i=1}^L \Big(\epsilon_\theta(\tilde{x}_t, t, S_i) - \epsilon_\theta(\tilde{x}_t, t, \emptyset)\Big).$$

By these equations, we can sample from $q(\tilde{x}_0|\{S_i\}_{i=1}^L)$ with classifier-free guidance using the perturbed noise:

$$\begin{aligned}\epsilon^* &:= \epsilon_\theta(\tilde{x}_t, t, \emptyset) + \omega\Big(\epsilon_\theta(\tilde{x}_t, t, \{S_i\}_{i=1}^L) - \epsilon_\theta(\tilde{x}_t, t, \emptyset)\Big) \\ &= \epsilon_\theta(\tilde{x}_t, t, \emptyset) + \omega \sum_{s \in \mathcal{D}_j}^L \Big(\epsilon_\theta(\tilde{x}_t, t, s) - \epsilon_\theta(\tilde{x}_t, t, \emptyset)\Big).\end{aligned}$$

We borrowed this derivation from [26] to support the completeness of our work. Although the integration of conditioning styles $\{S_i\}_{i=1}^L$ requires them to be conditionally independent given the generated sample $\tilde{x}_0$, it has been observed that this condition does not strictly need to be strictly met in practice.

C DETAILS OF DIFFUSION MODEL

We present a 1D UNet implementation that includes key components such as style embedding layer. During training, the model takes in a 1D time-series sample, an activity style vector, and a timestep to produce noise of the same dimension as the input. During sampling, the model uses noise, concatenated activity style vectors, and a timestep to generate a new time-series sample. Our diffusion model operates according to these specifications.

C.1 Architecture and Training Details

Architecture. The model begins with an initialization convolution layer, followed by a series of downsampling blocks. Each downsampling block comprises two residual blocks and an attention layer, executed using a 1D convolutional layer with a kernel size of 3. The output of each downsampling block is saved in a list, which is later used in the upsampling process. After the downsampling blocks, the model has a middle block consisting of a residual block and an attention layer. The upsampling blocks are then applied in reverse order, with each block consisting of two residual blocks and an attention layer. The upsampling operation is performed using a transposed convolutional layer with a kernel size of 3. In addition to the convolutional layers and residual blocks, the model also includes a time embedding layer, which consists of a sinusoidal positional embedding and two linear layers with a channel size of 256. We borrow the code for the 1D UNet from https://github.com/lucidrains/denoising-diffusion-pytorch. Differently, we add a style embedding layer, which consists of a linear layer with a channel size of 100 and a linear layer with a channel size of 64. Both of these embedding layers are concatenated along

the channel dimension, resulting in a tensor that is used as the condition for each residual block in the UNet.

Training details. Our diffusion training settings primarily adhere to the guidelines outlined in [16, 17]. The batch size is 64, with a learning rate of 2×10^{-4} using the Adam optimizer. The diffusion step is set to $T = 100$. We choose the probability of dropping the conditioning information to be 0.5.

C.2 Sampling Procedure

Once our diffusion is trained, we can use the style-fused sampling strategy to diversify the condition space for our task. During the selection of style sets for generating samples, we specify the ratio κ of new samples to original samples, as well as the maximum number o of styles that can be fused for each new sample and the number of fused styles is determined by a random variable that follows a specified distribution.

For example, if we have 1000 training samples and $\kappa = 1$ and $o = 5$, we will generate 1000 training samples. For each new sample, we can constrain it to be fused with up to 5 styles, and the number of fused styles is determined by a random variable that follows a specified distribution. The distribution used to determine the number of fused styles can be customized based on the specific task and dataset. For example, we can use a uniform distribution to ensure an equal probability of fusing any number of styles, or we can use a Poisson distribution to favor fewer fused styles. In our code, we have set a default probability distribution for the number of fused styles. For example, when $o = 5$, we set the probability distribution for mixing from 1 to 5 styles to {0.2, 0.2, 0.2, 0.2, 0.2}, with a total sum of 1. This means that each new sample can be fused with up to 5 styles, and the number of fused styles is determined randomly based on this distribution. However, this probability distribution can be flexibly adjusted to potentially achieve better results. By adjusting these hyperparameters, we can control the number of fused styles for each new sample and explore different combinations of styles. This could help us generate more diverse and representative samples for our task.

C.3 Implementation of Generation in TSC models

To leverage parallel computing for training time series classification (TSC) models, we simultaneously process a batch of training data containing B samples and generate $\kappa \times B$ new samples. Within each batch, we select an appropriate number of style sets for each class in a class-balanced manner and aggregate all $\kappa \times B$ style sets as conditional inputs. Next, we use our diffusion model to generate $\kappa \times B$ new samples. Once this batch generation process is complete, we utilize the generated samples to train the feature extractor, while the generated data with its class labels are stored. This generation process is repeated for each batch of training data, requiring execution only once. The stored data will be directly used to train the feature extractor in subsequent epochs without the need for data regeneration.

Budget. Overall, our approach offers a cost-effective solution to the challenges associated with human activity data collection in HAR scenarios. For instance, the cost of a three-axis accelerometer typically ranges from several tens of dollars, and collecting human activity data for 30,000 samples of various activities can take several weeks and cost approximately $1,000 per participant. Additionally, manual annotation of the data in subsequent stages can lead to additional expenses. In contrast, our diffusion model only uses a GPU like RTX 3090 to create over 30,000 labeled activity samples in just one hour [6] . These generated samples can provide significant performance gains for various baselines. Importantly, our method only requires a one-time expansion without the need for re-generation. Moreover, the generated samples may simulate new and unseen users, making the trained deep learning models more likely to be effectively deployed on new edge devices for real-world applications.

D DIVERSITY LEARNING STRATEGY

D.1 Details of training TSC models

After generating synthetic dataset $\tilde{D}^s$, the data space expands to $D^s = \{\tilde{D}^s \cup D^s\} = \{(\mathbf{X}_i, y_i^c)\}_{i=1}^{n^s+\tilde{n}^s}$ [7]. To enhance the diversity of learned features from D^s, we present a simple yet effective diversity learning strategy that is adapted from the representation learning method proposed in [36].Our approach involves differentiating between the origin of each sample, whether it is "synthetic" or "original," and its corresponding class label. Since our augmented data is highly diverse, we adopt a simplified learning objective that removes complex computations, such as measuring the distance and similarity between synthetic and original data. Instead, we depend entirely on the cross-entropy loss, where each loss is determined by different classification criteria. To this end, we employ a multi-objective method that consists of three sequential steps to train the classifier on D^s.

Specifically, there are three fundamental components of a TSC model: the feature extractor G_f, the projection layer G_{proj^*}, and the classifier G_{y^*}. The function $G_f(\cdot)$ maps the inputs to their respective representations and is updated throughout all three steps. The function $G_{\text{proj}^*}(\cdot)$ is a fully connected layer that maps the representations to vectors of length Z. The function $G_{y^*}(\cdot)$ is a classifier responsible for predicting the designed label. The superscripts (*) indicate that these components are utilized in different steps. The three steps are then described as follows:

(i) Class-origin feature learning. To learn more detailed representations, we label each sample based on its origin and class. The original data is labeled as $y_i^{\text{o}} = 1$ and the augmented data as $y_i^{\text{o}} = 0$. We then combine the origin and class labels to create new labels, represented as $y_i^{\text{co}} = (y_i^{\text{c}} + y_i^{\text{o}} \times C) \in \mathbb{N}$. By using the classifier $G_{y^{\text{co}}} : \mathbb{R}^Z \to \mathbb{R}^{2\times C}$, we train the model to predict these new labels using cross-entropy, which can be expressed as:

$$\mathcal{L}_{\text{cls-ori}} = \frac{1}{n^s + \tilde{n}^s} \sum_{i=1}^{n^s+\tilde{n}^s} \ell\left(G_{y^{\text{co}}}\left(G_{\text{proj}^{\text{co}}}\left(G_f(\mathbf{X}_i)\right)\right), y_i^{\text{co}}\right). \quad (10)$$

(ii) Origin-specific feature learning. To further enhance the distinction between synthetic and original data, we encourage the model to differentiate between origin labels using the loss function

[6] Renting a single RTX 3090 GPU for one hour typically costs less than $0.2

[7] Here, y_i^c denotes the class label, which has the same meaning as y_i in the paper.

$L_{\text{ori-spe}}$, which is defined as follows:

$$\mathcal{L}_{\text{ori-spe}} = \frac{1}{n^s + \tilde{n}^s} \sum_{i=1}^{n^s+\tilde{n}^s} \ell\left(G_{y^{\text{o}}}\left(G_{\text{proj}^{\text{o}}}\left(G_f(\mathbf{X}_i)\right)\right), y_i^{\text{o}}\right). \quad (11)$$

Here, $G_{y^{\text{o}}} : \mathbb{R}^Z \to \mathbb{R}^2$ serves as an origin classifier to distinguish whether the input features originate from a synthetic or an original sample.

(iii) Class-specific feature learning. The feature extractor finally undergoes a training process on D^s to correctly predict the class labels. This step allows the model to separate clusters belonging to different classes. We employ a class classifier $G_{y^{\text{c}}} : \mathbb{R}^Z \to \mathbb{R}^C$ by minimizing the following loss:

$$\mathcal{L}_{\text{cls-spe}} = \frac{1}{n^s + \tilde{n}^s} \sum_{i=1}^{n^s+\tilde{n}^s} \ell\left(G_{y^{\text{c}}}\left(G_{\text{proj}^{\text{c}}}\left(G_f(\mathbf{X}_i)\right)\right), y_i^{\text{c}}\right). \quad (12)$$

Overall. During the training process, as Algorithm 1 shows, the three steps are iteratively repeated in each epoch until termination. In the inference phase, as Algorithm 2 shows, we only use the trained feature extractor G_f, projection layer $G_{\text{proj}^{\text{c}}}$ and the class classifier $G_{y^{\text{c}}}$ from step (iii). They are stacked together to classify the input time-series samples in the test dataset D^t.

D.2 Pseudo-code

Algorithm 1 Training Algorithm for Diversity Learning Strategy

Initialize: G_f (feature extractor), $G_{\text{proj}^{\text{co}}}$ (projection layer), $G_{y^{\text{co}}}$ (classifier)
Input: D^s.
Output: G_f, $G_{\text{proj}^{\text{c}}}$, $G_{y^{\text{c}}}$
for each epoch **do**
  for each batch $\mathbf{B}_i$ in training dataset D^s **do**
    if epoch is 0 **then**
      Generate $B \times \kappa$ **new samples** $\tilde{\mathbf{B}}_i$.
      Expand **data space** $D^s = D^s \cup \tilde{\mathbf{B}}_i$ **and** $\mathbf{B}_i = \mathbf{B}_i \cup \tilde{\mathbf{B}}_i$.
    end if
    Predict class-origin label: $\tilde{y}_i^{\text{co}} = G_{y^{\text{co}}}(G_{\text{proj}^{\text{co}}}(G_f(\mathbf{B}_i)))$.
    Calculate class-origin loss: $\mathcal{L}_{\text{cls-ori}}$.
    Update G_f, $G_{\mathbf{proj}^{\text{co}}}$, **and** $G_{y^{\text{co}}}$ **using** $\mathcal{L}_{\mathbf{cls\text{-}ori}}$.
    Predict origin label: $\tilde{y}_i^{\text{o}} = G_{y^{\text{o}}}(G_{\text{proj}^{\text{o}}}(G_f(\mathbf{B}_i)))$.
    Calculate original-specific loss: $\mathcal{L}_{\text{ori-spe}}$.
    Update G_f, $G_{\mathbf{proj}^{\text{o}}}$, **and** $G_{y^{\text{o}}}$ **using** $\mathcal{L}_{\mathbf{ori\text{-}spe}}$.
    Predict class label: $\tilde{y}_i^{\text{c}} = G_{y^{\text{c}}}(G_{\text{proj}^{\text{c}}}(G_f(\mathbf{B}_i)))$.
    Calculate class-specific loss: $\mathcal{L}_{\text{cls-spe}}$.
    Update G_f, $G_{\mathbf{proj}^{\text{c}}}$, **and** $G_{y^{\text{c}}}$ **using** $\mathcal{L}_{\mathbf{cls\text{-}spe}}$.
  end for
end for

Algorithm 2 Inference

Input: Trained models G_f, $G_{\text{proj}^{\text{c}}}$, $G_{y^{\text{c}}}$; Input data D^t.
Output: Predicted labels $\hat{Y}$.
for each input sample $\mathbf{x}$ in D^t **do**
  Predict class label: $\tilde{y}_i^{\text{c}} = G_{y^{\text{c}}}(G_{\text{proj}^{\text{c}}}(G_f(\mathbf{x})))$.
end for

E DETAILS OF EXPERIMENTAL SETUP

E.1 Datasets

To ensure fairness and reproducibility, we conduct our experiments by the same experimental setup as that provided in [36] to compare the performance of our method on HAR tasks. Our datasets for this study include DSADS [8], PAMAP2 [9], and USC-HAD [10]. We follow the same processing steps involving the domain split and randomly choose remaning data used for each dataset, as detailed in the official code [11]. The corresponding processing information is as follows:

Dataset information and pre-processing. The information details of the three HAR datasets are presented in Table 4. To segment the data, we employ a sliding window approach. For DSADS, the window duration is set to 5 seconds as per [2], while for PAMAP2, it is set to 5.12 seconds [38]. Similarly, for USC-HAD, a 5-second window is utilized with a 50% overlap between consecutive windows. Given the sampling rates of each dataset (25Hz for DSADS, 100Hz for PAMAP2, and 100Hz for USC-HAD), the window lengths are calculated to be 125 readings, 512 readings, and 500 readings, respectively. We normalize the data using normalization and reshape the 1D time series sample into a 2D format with a height of 1. Each batch is structured as (b, c, h, w), where b represents the mini-batch size, c denotes the number of channels corresponding to the total axes of sensors, h signifies the height, and w denotes the window length.

Domain split. We implement leave-one-out-validation by dividing subjects into multiple groups. In this approach, we designate one group of subjects' data as the target domain, while the remaining subjects' data serve as the source domain. Each group can be considered as an individual task. For DSADS and PAMAP2, we divide the 8 subjects into 4 groups. As for USC-HAD, we divide the 14 subjects into 5 groups, with groups 0-3 comprising 3 subjects each, and the last group consisting of 2 subjects. Subsequently, we partition the data within each group into training, validation, and test sets, maintaining a ratio of 6:2:2. To assess the impact of training data size on model performance, we conduct experiments where we randomly sample 20% to 100% of the training data with increments of 20%. This allows us to simulate the small-scale setting. During testing, we evaluate the trained model on the test set of the target domain.

E.2 Baselines

We compared our proposed model with several closely related DG baselines that are also suitable for time series input:

- `Mixup` [57]: Through appwalking domain mixup at both pixel and feature levels, it is a data augmentation-based DG method.
- `RSC` [19]: A training method enhances out-of-domain generalization by discarding dominant features in the training data.
- `SimCLR` [8]: It leverages data augmentation to generate positive samples and a learnable transformation for contrastive learning.

[8] https://archive.ics.uci.edu/dataset/256/daily+and+sports+activities
[9] https://archive.ics.uci.edu/dataset/231/pamap2+physical+activity+monitoring
[10] https://sipi.usc.edu/had/
[11] https://github.com/microsoft/robustlearn/tree/main/ddlearn

Table 4: Summary of information and feature network settings for three HAR datasets

Dataset	Task	Subjects	Activities (C)	Sample Shape	Kernel Size	Network (feature and projection)	Output Channel
DSADS	4	8	19	(45, 1, 125)	9	2 Conv1D Layers 1 FC Layer	(16, 32, 64)
PAMAP2	4	9	8	(27, 1, 512)	9	2 Conv1D Layers 1 FC Layer	(16, 32, 64)
USC-HAD	5	14	12	(6, 1, 500)	6	3 Conv1D Layers 1 FC Layer	(16, 32, 64, 128)

- `Fish` [40]: By maximizing the gradient inner product between domains, it learns domain-invariant features in DG issues.
- `DANN` [12]: It uses adversarial training to learn domain-invariant features from data with accessible target labels.
- `mDSDI` [5]: It optimizes domain-specific features via meta-learning from source domains, where the domain labels are known.
- `TS-TCC` [9]: As a recent self-supervised method, it leverages a small set of labeled time-series data for model generalization.
- `DDLearn` [36]: A latest DG method that leverages standard data augmentation baselines, designed for low-resource scenarios.

In our experiments, we use the same data augmentation techniques [45] as those used in [36] to ensure a fair comparison. These techniques include rotation, permutation, time-warping, scaling, magnitude warping, jittering, and random sampling. For example, jittering involves applying different types of noise to the samples, which increases the diversity of data magnitude. Scaling, on the other hand, rescales the samples to different magnitudes.

E.3 Architecture

All DG baselines use the same network architecture for feature extraction, except for TS-TCC [9]. Specifically, we adopt an architecture consisting of a feature extractor, a projection layer, and a classifier, as described in [36]. The feature extractor is composed of two or three Conv1D layers, depending on the dataset being used. For DSADS and PAMAP2, we use two Conv1D layers with a kernel size of 9, while for USC-HAD, we use three Conv1D layers with a kernel size of 6. Each Conv1D layer is followed by a ReLU activation function and a maxpool1d operation. The output is then connected to a projection layer, which is a fully-connected layer with output feature dimensions of 64 for DSADS and PAMAP2, and 128 for USC-HAD. Finally, we employ a fully-connected layer as the classifier, which takes the extracted features as input and outputs C-dimensional logits. By applying a softmax operation, we obtain prediction probabilities for each class, which sum up to 1. In our diversity learning strategy, we use different projection layers and classifiers at different training steps, while keeping the overall architecture of the model the same. Specifically, we use different projection layers with the same architecture for different steps, and classifiers with the same input channel but output $(2 \times C)$-dimensional, 2-dimensional, and C-dimensional logits for specific classification goals. During the inference phase, we utilize only a trained feature extractor, a projection layer, and a classifier. Since TS-TCC is designed for contrastive learning, we retain its architecture and make slight modifications to adapt it to the input data channels and lengths for use in HAR tasks.

E.4 Training Details

All methods in our experiments use PyTorch. We utilize Adam optimization with a scheduler. The batch size is fixed at 64. The experiments are conducted on one GeForce RTX 3090 Ti GPU. To ensure its performance, we fine-tune the training configurations for each baseline. Furthermore, the results for `Mixup` [57], `RSC` [19], `SimCLR` [8], `Fish` [40], and `DDLearn` [36] reported in Tables 1 and 2 are obtained from the paper [36].