Augmented Adversarial Learning for Human Activity Recognition with Partial Sensor Sets

HUA KANG, The Hong Kong University of Science and Technology, China
QIANYI HUANG, Southern University of Science and Technology, China and Peng Cheng Laboratory, China
QIAN ZHANG*, The Hong Kong University of Science and Technology, China

Human activity recognition (HAR) plays an important role in a wide range of applications, such as health monitoring and gaming. Inertial sensors attached to body segments constitute a critical sensing system for HAR. Diverse inertial sensor datasets for HAR have been released with the intention of attracting collective efforts and saving the data collection burden. However, these datasets are heterogeneous in terms of subjects and sensor positions. The coupling of these two factors makes it hard to generalize the model to a new application scenario, where there are unseen subjects and new sensor position combinations. In this paper, we design a framework to combine heterogeneous data to learn a general representation for HAR, so that it can work for new applications. We propose an Augmented Adversarial Learning framework for HAR (AALH) to learn generalizable representations to deal with diverse combinations of sensor positions and subject discrepancies. We train an adversarial neural network to map various sensor sets' data into a common latent representation space which is domain-invariant and class-discriminative. We enrich the latent representation space by a hybrid missing strategy and complement each subject domain with a multi-domain mixup method, and they significantly improve model generalization. Experiment results on two HAR datasets demonstrate that the proposed method significantly outperforms previous methods on unseen subjects and new sensor position combinations.

CCS Concepts: • **Human-centered computing → Ubiquitous computing**; • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: Adversarial learning, augmentation, human activity recognition, domain mixup, generalization

ACM Reference Format:
Hua Kang, Qianyi Huang, and Qian Zhang. 2022. Augmented Adversarial Learning for Human Activity Recognition with Partial Sensor Sets. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 3, Article 122 (September 2022), 30 pages. https://doi.org/10.1145/3550285

1 INTRODUCTION

The past decade has witnessed the proliferation of wearable devices in various form factors, such as smartphones, smartwatches, smart necklaces, smart glasses, and smart shoes. These intelligent devices contribute to recognizing human activities for activity tracking, health monitoring, or entertainment (e.g., gaming, virtual reality). Meanwhile, a number of wearable sensor based public datasets for human activity recognition have been released,

*Corresponding Author

Authors' addresses: Hua Kang, The Hong Kong University of Science and Technology, Hong Kong, China, hkangae@cse.ust.hk; Qianyi Huang, Southern University of Science and Technology, Shenzhen, China and Peng Cheng Laboratory, Shenzhen, China, huangqy@sustech.edu.cn; Qian Zhang, The Hong Kong University of Science and Technology, China, qianzh@cse.ust.hk.

2474-9567/2022/9-ART122 $15.00
https://doi.org/10.1145/3550285

Table 1. Statistics of public wearable HAR datasets

Dataset	Subject	Activity	Body Position
DSADS [4]	8	19	Torso, Right Arm, Left Arm, Right Leg, Left Leg
PAMAP [26]	9	18	Wrist, Chest, Ankle
UCI-HAR [2]	30	6	Waist
REALDISP [3]	17	33	Left Calf, Right Calf, Left Thigh, Right Thigh, Left Lower Arm, Right Lower Arm, Left Upper Arm, Right Upper Arm, Back

with the prospect that collective efforts can enhance the recognition performance. We show a few examples in Table 1. These public resources save us great efforts for data collection. However, when a new application scenario emerges, these existing datasets fail to contribute their value. For example, when a user wears a smartphone on the waist and a smartwatch on the wrist to study his upper body activity, there is no such dataset that we can directly take advantage of. Although UCI-HAR [2] includes a sensor on the waist and PAMAP [26] includes a sensor on the wrist, there is no such dataset that includes sensors both on the waist and the wrist. We ask the question, can we combine the existing datasets to build a general representation for activity recognition so that it is general for new application scenarios which have unseen sensor position combinations?

As shown in Figure 1, we want to exploit existing datasets to learn a model, which can be used for new scenarios where we have unseen subjects and sensor position combinations. This is a practical problem. Many people own multiple devices in daily life, but different people may carry different numbers of devices and put them in different positions. For example, a subject A may have a smartphone in his pocket near his chest and wear a smartwatch on his left wrist, and a subject C may wear a pair of smart shoes and a smartwatch on her right wrist. If another subject a carries a smartphone in her pocket near the chest and wears a pair of smart shoes, it is hard to leverage the model trained with A and C 's data to the new subject a's new sensor position combination. This problem is challenging as it couples two dimensions of heterogeneity, user heterogeneity and sensor heterogeneity. Although multiple datasets may all contain samples corresponding to the "walk" activity, the samples are collected from different subjects. Even though the datasets may share some common sensors, we cannot directly align the samples across the datasets, as they are not collected from the same subject simultaneously. Furthermore, as existing samples only have partial sensor sets, likely without the target sensor position combination, there is no reference to explore the relationship between the source sensor combination and target sensor combination, making it difficult to transform the source sensor combination to the target sensor combination.

This problem has not been investigated in previous studies. Although there are some related works, existing methods cannot work for this problem. Previous works suggest feature concatenation[5] and model ensemble [38] for multi-sensor fusion. However, these works fuse a fixed set of sensors while failing to explore cases where different combinations of sensors are to be fused. Missing data imputation is a classical problem that targets completing the unobserved data in the input space. However, the missing model is usually assumed as missing completely at random (MCAR) [37]. Thus, traditional low-rank-based completion [24] is not applicable due to the block-wise missing mode of our problem. Methods leveraging the power of deep learning [13, 21] still need complete data to supervise the reconstruction of missing data learned from the observed data. Adversarial learning-based methods for data imputation [18, 37] can be used to generate missing components. It is still difficult for them to impute large block-wise data incompleteness, handle subject discrepancies, and adapt for classification tasks which may need to balance the tasks of reconstruction and classification.

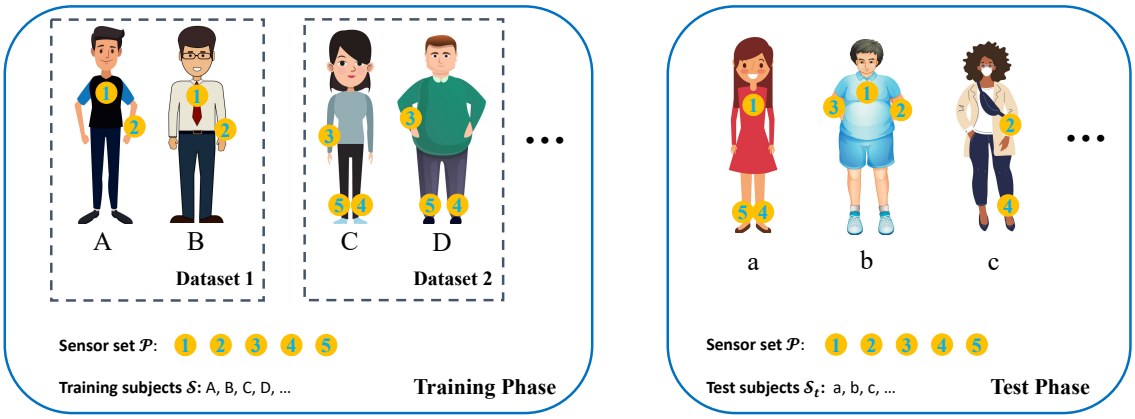
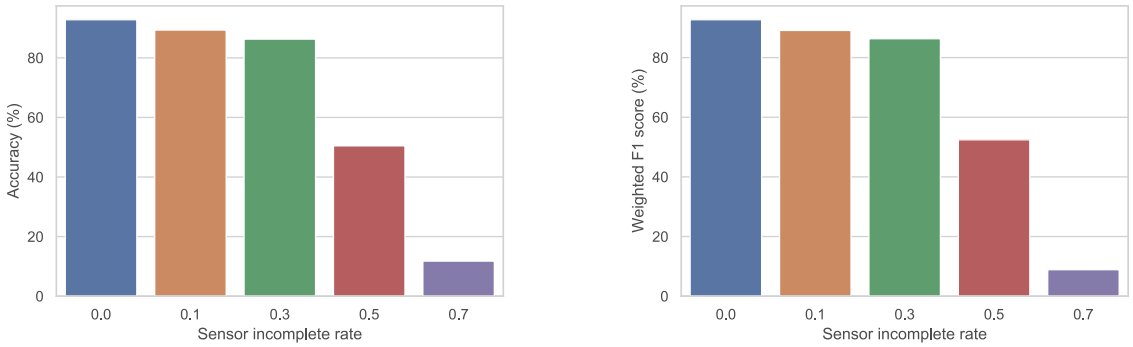


Fig. 1. Problem Illustration

We have summarized three main challenges of our problem.

- *Challenge 1*. There exist two coupling dimensions of heterogeneity in our dataset: subject discrepancy and sensor placement discrepancy.
- *Challenge 2*. We have no access to the test data in the training phase, meaning that we have no idea about the sensor placements and subjects.
- *Challenge 3*. There is no sample having all the possible available sensors, making it difficult to explore relationships between different positions' sensors.

To fully use all the source domains and achieve good classification performance, we want to project different incomplete feature spaces into a common generalizable and discriminative feature space. Domain adaptation [10] or domain generalization [17] complies with this principle since they are also committed to aligning different distributions. But the feature spaces of different domains in these problems are usually complete though they have different distributions. Thus we choose the adversarial domain adaptation method [10] as our backbone, which can partially solve *Challenge 1* and *Challenge 2*. However, deep neural networks are sensitive to incomplete data and are difficult to adjust for different incomplete data combinations. For instance, Figure 2 reveals that there has been a substantial decline in the test accuracy under different sensor incomplete rates of a multi-sensor fusion model [36] trained on complete sensor sets. In addition, missing data reduce the amount of information the network can learn from and make it prone to overfitting problem. To overcome *Challenge 1* and *Challenge 2*, we need to further improve model generalization. Thus, on top of the adversarial domain adaptation method, we propose the Augmented Adversarial learning framework for HAR (AALH) that includes two augmentation methods operating in the latent representation space to improve the model generalization ability. To prevent the model from memorizing the limited and fixed missing patterns provided by the training data (each subject only has a fixed set of partial sensors, in other words, one missing pattern) and increase representation diversity, we propose a hybrid missing strategy that manually and randomly ignores features in the original data. This method enables the model to experience diverse incomplete combinations and learn to adapt to various cases such that the model can maintain good performance on unseen sensor position combinations. We also present a multi-domain mixup strategy that mixes representation spaces of multiple source domains to complement each other. A combination of different domains' representations can provide guidance for classification. Thus it can



(a) Accuracy under different incomplete rates

(b) Weighted F1 score under different incomplete rates

Fig. 2. Decline of performance under different incomplete rates

leverage concerted efforts of partial sensor sets to overcome *Challenge 3*. Therefore, AALH learns a representation that embeds compact and effective information of all the source domains and is highly generalizable to the unseen target domain. We summarize our contributions as follows:

- We present a novel and practical framework for general human activity recognition. We learn from multiple data sources with heterogeneous subjects and sensor combinations and generalize to unseen target subjects and sensor combinations.
- We propose an Augmented Adversarial Learning framework for HAR (AALH) to solve the problem. Based on an adversarial domain adaptation framework, we design two augmentation strategies, i.e., hybrid missing and multi-domain mixup, that significantly improve model generalization.
- Experimental results on two multi-sensor multi-subject datasets show the superiority of our method. For the first dataset, our method obtains an average of 75.3% accuracy when the miss rate is as high as 0.7, and the average accuracy of all miss rates is 91.3% which is 10.3% higher than the best baseline. For the second dataset, our method obtains an average of 76.2% accuracy when the miss rate is as high as 0.6, and the average accuracy of all miss rates is 86.9% which is 10.3% higher than the best baseline.

The rest of this paper is organized as follows. Section 2 introduces related works and elaborates the differences between our problem setup and previous works. Section 3 gives a formal definition of the problem setup and defines the notations used in the paper. We present the technical details in Section 4 and present the evaluation results in Section 5. We conduct a discussion in Section 6. Finally, we conclude this paper in Section 7.

2 RELATED WORKS

This section briefly discusses related works, including multi-sensor fusion, domain adaptation and generalization, data imputation, and data augmentation methods. Table 2 specifies the differences between our problem setting and the common problem settings in these related works.

2.1 Multi-Sensor Fusion

To realize a more reliable and accurate HAR system, multiple sensors are employed to complement each other. DeepSense [36] designs a unified deep learning framework comprised of convolutional neural networks (CNNs)

Table 2. Comparison between our problem setting and related works. Note that more "✓" means more constraints of the problem.

Problem setting	Domain discrepancy	Incomplete feature space	No access to test data for training	No supervision for missing components	Task
Domain adaptation	✓	×	×	N/A	classification/regression
Domain generalization	✓	×	✓	N/A	classification/regression
Multi-sensor fusion	×	×	✓	N/A	classification/regression
Data imputation	×	✓	✓	×	reconstruction
Ours	✓	✓	✓	✓	classification/regression

to extract local interactions within each sensing modality and merge local interactions of different sensing modalities and recurrent neural networks (RNNs) to fuse multiple sensors' time-series information. GlobalFusion [23] introduces a lightweight attention mechanism called the global attention module, including spatial fusion and sensing modality fusion for multi-sensor information fusion. DeepFusion [35] designs a unified multi-sensor deep learning framework for learning informative representations of heterogeneous sensory data. However, these frameworks only consider fusing information from a fixed set of sensors for the same subject rather than various sensor combinations of different subjects. Most multi-sensor fusion works did not consider the heterogeneity of available sensor sets. SenseHAR [14] considers the heterogeneity among devices. In the proposed scheme, each device learns its own sensor fusion model, which maps the raw sensor data to a shared low dimensional latent space. The model is trained for a single person and can be directly used for a subset of sensors but still needs calibration for new devices. [30] considers building a unified model for multi-modal sensors and multi-class. They also consider the missing sensors and find that sensor dropout with a normalization ratio slightly performs better than using all six sensors under a single model situation. However, they did not clearly state that the train and test are conducted with different subjects, and only missing one sensor case is tested where the improvement is only 1%-3%. [15] proposes a method for single-sensor-based activity recognition using multiple sensors during training time. Nevertheless, our situation cannot provide the complete set of sensors for training and cannot get access to the target sensors in the training phase.

2.2 Domain Adaptation and Generalization

Domain adaptation and generalization aim to leverage the knowledge of label-rich source domains to help tasks in the target domain with fewer or no labels. The difference between domain adaptation and domain generalization lies in whether the target domain can participate in the training phase. Deep learning methods play an essential role by projecting the input data into a joint latent space to align the distributions of the source and target domains. One important approach is based on adversarial training. [7, 11, 40] are the pioneer works that adapt Generative Adversarial Networks (GANs) [12] to domain adaptation problems. [17] extends the adversarial autoencoders by introducing the Maximum Mean Discrepancy (MMD) distance along with an arbitrary prior distribution to align the distributions among the source domains. Data augmentation [28] is used to increase the diversity of existing training data. Typical augmentation operations commonly used for image data include flipping, rotation, scaling, cropping, etc. [40] proposes a conditional adversarial architecture for sleep stage classification that conditions on both extracted latent representation and discriminative information passed by the classifier predictions. [28] introduces a cross-gradient training method that augments source data by adversarial gradients from a domain classifier. Compared with our problem, all of these works rely on the assumption of covariate shift and availability of complete feature dimensions in all the source and target domains. [20] explores unsupervised domain adaptation with an incomplete target domain, but the source has complete data, and the test data are trained together.

2.3 Data Imputation

Data imputation is a classical problem that considers reconstructing the missing data in the input space. The utter goal is the performance of reconstruction. GAIN [37] is an adaptation of conditional GANs for imputation. The generator takes in both the incomplete data and the corresponding mask to impute the missing data. The discriminator is trained to infer whether each feature is imputed or original. MisGAN [18] learns a complete data generator as well as a mask generator that models the missing data distribution. MMAE [13] describes a specialized denoising autoencoder: the Multimodal Autoencoder, for handling missing multimodal data. [22] designs a Graph Neural Network (GNN) leveraging the topology relationship of sensors to reconstruct missing components and maintain high classification accuracy. However, they need the data with complete sensors for supervision in the training phase. Most approaches consider a supervised setting where complete data are available, and miss mode belongs to a stochastic process.

2.4 Data Augmentation

Data augmentation is an efficient way for model generalization. We investigate some common data augmentation methods. [32] summarizes different data augmentation methods for time series data. The augmentation methods are first divided into basic approaches and advanced approaches. Basic approaches include time domain, frequency domain, and time-frequency domain methods. For example, cropping, flipping, jittering, etc., reside in this category. Advanced approaches include decomposition methods, statistical generative models, and learning methods. [39] is a recently proposed data augmentation method through linearly interpolating random batch samples and has demonstrated effectiveness for image classification tasks. The aim of our paper is not to find the best data augmentation method but to find or design appropriate augmentation methods for the problem and demonstrate their effectiveness.

3 PROBLEM DEFINITION

In this section, we formally define our problem and the notations used. Suppose the training data are collected from a subject set $\mathcal{S}$ and comprise $|\mathcal{S}|$ source domains. We denote the source data by $\mathcal{X} = \{x_s\}_{s\in\mathcal{S}}$, where $x_s \in \mathbb{R}^{n_s,\cdot}$ are the data samples of domain s and n_s is the number of samples of the domain s. The corresponding labels are $\mathcal{Y} = \{y_s\}_{s\in\mathcal{S}}$. We denote the whole set of all the possibly available sensors by $\mathcal{P}$. Note that the term "sensor" indicates an IMU located at a body position. For each source domain $s \in \mathcal{S}$, the data only have a sensor set p_s which is a subset of $\mathcal{P}$, i.e., $p_s \subset \mathcal{P}$ and the union of all the source domains' sensor sets is exactly $\mathcal{P}$, i.e., $\bigcup_{s\in\mathcal{S}} p_s = \mathcal{P}$. For data samples x_s of domain s, we have $x_s = \{x_s^p\}_{p\in p_s}$, indicating that the data samples only have $|p_s|$ sensors' data. In this way, different source domains' data have different sets of sensors, thus different dimensions. To unify the expression, we denote data of domain s as $x_s = \{x_s^p\}_{p\in\mathcal{P}}$, and use an additional binary matrix $m_s \in Z^{n_s\times|\mathcal{P}|}$ to represent whether the samples of domain s contain the sensor data from position p. If $m_s^{ij} = 1$, the sample x_s^i has the j_{th} sensor's data; otherwise, the sample x_s^i doesn't have the j_{th} sensor's data. We call the ratio of the missing sensor number and the complete sensor number the "incomplete rate". To expand our method's application scope, we consider a challenging situation where there is no sample having the complete sensor set, i.e., $\sum_{j=1}^{|\mathcal{P}|} m_s^{ij} < |\mathcal{P}|, \forall i \in \{1, ..., n_s\}, s \in \mathcal{S}$. Note that our method can also be used directly in situations where samples with all sensors exist. Each sensor should have at least one sample containing its data, i.e., $\sum_{s\in\mathcal{S}} \sum_{i=1}^{n_s} m_s^{ij} \geq 1, \forall j \in \{1, ..., |\mathcal{P}|\}$. Thus, the valid information we have access to in the training phase are data samples $\mathcal{X}$, activity labels $\mathcal{Y}$, and incomplete masks $\mathcal{M} = \{m_s\}_{s\in\mathcal{S}}$.

The target data are collected from unseen subjects' set $\mathcal{S}_t$, denoted by $\mathcal{X}^T = \{x_t^T\}_{t\in\mathcal{S}_t}$. Each sample of the target dataset is collected from a subset of sensor positions $p_t \subset \mathcal{P}$. Each target subject has one such sensor combination and different target subjects can have different sensor combinations. We also know the incomplete masks $\mathcal{M}^T = \{m_t^T\}_{t\in\mathcal{S}_t}$ for each target subject in the test phase. Note that in the training phase, any information

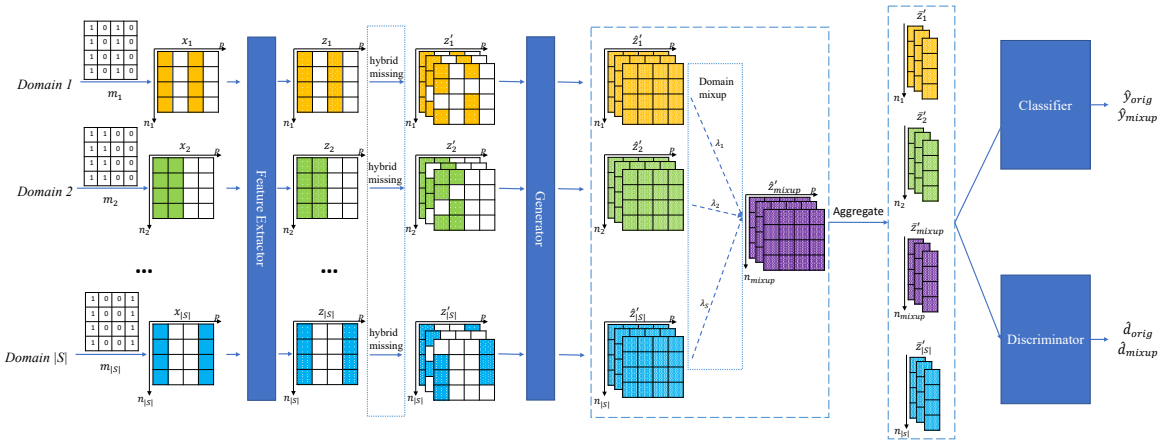


Fig. 3. Framework Overview

of the target data is unknown. We want to use the well-trained model of source domains to predict the activity labels of the unseen incomplete data.

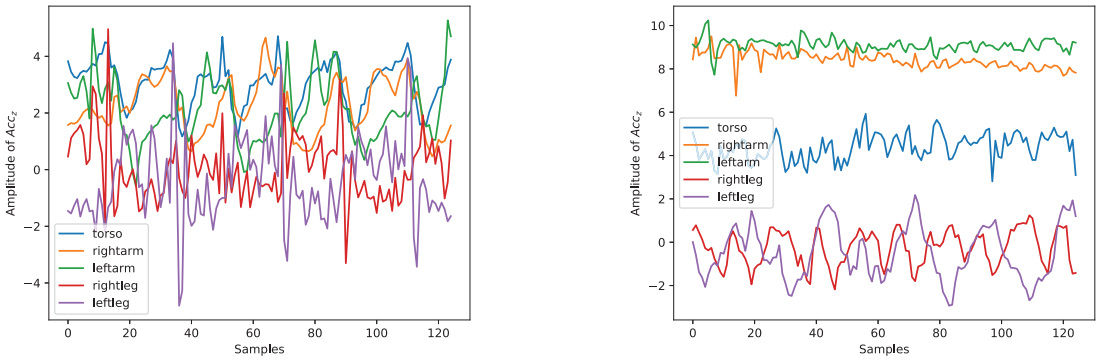
4 METHOD

4.1 Overview

An overview of our model is shown in Figure 3. The model takes in data from different sensor combinations of different subjects and aims to find a common latent representation space among subjects that can reflect the whole body movement pattern. Once trained, the model can achieve good performance on unseen subjects with different sensor sets. Each row of the grid rectangle represents a sample, and each column represents a sensor position.

In the training phase, the inputs are different subjects' data with different partial sensor sets. Each subject forms a branch. We use masks $\mathcal{M} = \{m_1, m_2, ..., m_{|S|}\}$ to indicate the incompleteness of the input data. Values of positions without data are set to 0. The inputs first pass through the common feature extractor to derive the latent representations $Z = \{z_s\}_{s \in \mathcal{S}}$. To increase the representations' diversity before feeding them into the generator, *hybrid missing strategy* is adopted which further manually misses data independently from samples with various probabilities based on the original incomplete data. This strategy can be regarded as a feature augmentation method. Then the augmented representations $Z' = \{z'_s\}_{s \in \mathcal{S}}$ are fed into the generator to generate incomplete components and the results are denoted as $\hat{Z}' = \{\hat{z}'_s\}_{s \in \mathcal{S}}$. All the source domains' representations are mixed to generate new data-label pairs to further enrich the intrinsic space and diversity of representations. Finally, different sensors' representations are aggregated as $\bar{Z} = \{\bar{z}_s\}_{s \in \mathcal{S}}$ and are fed into the common classifier and discriminator for classification and representation alignment.

As for the inference phase, the target data from unseen subjects with incomplete sensors simply pass through the feature extractor, generator, and classifier sequentially to get the predicted activities. Note that the *hybrid missing strategy* and *domain representation mixup* are not used in the inference phase.



(a) Comparison of different sensor positions' data of walking activity

(b) Comparison of different sensor positions' data of cycling activity

Fig. 4. Data comparison of different positions' sensor

4.2 Common Feature Extractor

The aim of our common feature extractor is to fuse the information of one sensor position and extract compact representations. It needs to be able to handle the data streams of different modalities and axes at a single sensor position. We choose a variant of the DeepSense framework [36] as the feature extractor F. Each body position has its own independent feature extractor F_p since the patterns reflected by the same motion at different positions may have great discrepancies, which can be observed from Figure 4.

Each body position's sensor usually includes an inertial measurement unit (IMU) comprised of three modalities, i.e., accelerometers, gyroscopes, and magnetometers, while each modality has three axes, i.e., axis x, axis y and axis z. We first use three convolutional layers to fuse the data of three axes of one modality. Then another three convolutional layers are deployed to fuse the representations of the three different modalities at the same body position. Figure 5 shows the detailed architecture of one position's feature extractor. Each position has one such feature extractor, and they constitute the common feature extractor F and $F = \{F_p\}_{p \in \mathcal{P}}$.

We concatenate the extracted features from each body position's feature extractor to obtain the latent representations $Z = \{z_s\}_{s \in \mathcal{S}}$. As there is no sensor at some body positions, the corresponding representations should remain zero thus we represent Z as

$$Z = F(\mathcal{X}) \odot \mathcal{M}$$

where $\mathcal{M}$ sets representations of unavailable body positions as all zero tensors.

4.3 Hybrid Missing Strategy

In the previous section, different body positions' data are transformed to latent representations Z, waiting to be fed into the following Generative Adversarial Network. However, each subject only has a fixed combination of sensors that have values. Thus the model may memorize the missing pattern of the inputs, and it is difficult to generalize to the unseen combination of sensors. Therefore, the model may be prone to overfitting problems and hard to train. [25] has demonstrated that creating multiple noisy samples during the training phase improves performance. [33] also shows that Dropout can be used as induced noise in Generative Adversarial Network (GAN) to prevent co-adaptation between hidden units and prevent overfitting, especially for cases where constraints in the input should be preserved, e.g., A-to-B translation. To introduce more diversity in each subject's representations, we

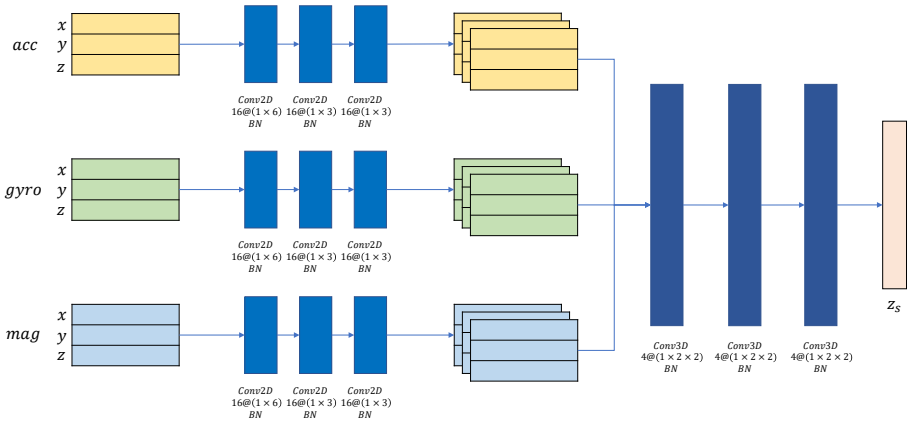


Fig. 5. Common Feature Extractor Architecture

randomly ignore some available sensors' representations in the initial incomplete data during the training phase to increase the representation diversity and model robustness for handling various incomplete cases. This is similar to adding a Dropout layer at the beginning of the following generator, but the difference lies in the way we ignore representations. Compared with Dropout, we adopt a hybrid missing strategy and drop the entire representations of a sensor. Specifically, we duplicate the representations Z into multiple copies $\{Z_1, Z_2, ..., Z_C\}$, multiplied with different additional missing masks $M^a = \{m_1^a, m_2^a, ..., m_C^a\}$ whose dimensions are the number of samples times the number of sensors. We keep the first copy the same as before, i.e., $Z_1 = Z$, which means all the values of this additional mask m_1^a are 1. For the other copies, the additional masks are generated with different missing probabilities. To be specific, each value of an additional mask m_c^a is first sampled i.i.d. from a uniform distribution $\mathcal{U}(0, 1)$. If we assume the additional missing probability is p_c, then the final additional mask is calculated as follows:

$$m_c^a(i,j) = \begin{cases} 1 & m_c^a(i,j) > p_c \\ 0 & otherwise \end{cases} \tag{1}$$

The overall incompleteness is represented by $m_s \odot m_c^a$, where m_s is the initial incompleteness mask for source domain s and the m_c^a is one additional incompleteness mask. In this way, the original vacant sensor data are still vacant while the original existing sensor is further missed with a certain probability. As said above, we use multiple probabilities to increase sample diversity. Then we stack these augmented additional missing data. The additional missing also makes more challenging examples and works as a way of regularization to make the model more effective in extracting information. We also discuss the impact of values of these miss probabilities in Section 5.7. The augmented representations are represented by Z', which is calculated as

$$Z' = Z \oplus [Z \odot (m_s \odot m_2^a)] \oplus ... \oplus [Z \odot (m_s \odot m_C^a)]. \tag{2}$$

where $\oplus$ means concatenation along the sample axis. Then the augmented Z' are fed into the generator G.

4.4 Generator

The generator takes in the extracted representations, preserves the inputs with values, and generates the representations for the missed inputs. In an analogy to the generative adversarial network, we substitute the

zero representations with a random noise sampled from the uniform distribution $\mathcal{U}(0, 1)$ and then feed the representations into G. As different sensor positions may have different signal patterns, similar to Section 4.2, we develop $|\mathcal{P}|$ independent generators for the $|\mathcal{P}|$ sensors, i.e., $G = \{G_p\}_{p\in\mathcal{P}}$. Note that these generators are shared across different subjects. The inputs to the generators are the augmented representations Z', which makes the generators experience various incomplete sensor combinations.

We find that three fully connected layers with ReLU activations are enough for the generators. The generated representations are denoted as $\hat{Z}' = \{\hat{z}'_s\}_{s\in\mathcal{S}}$.

4.5 Domain Representation Mixup

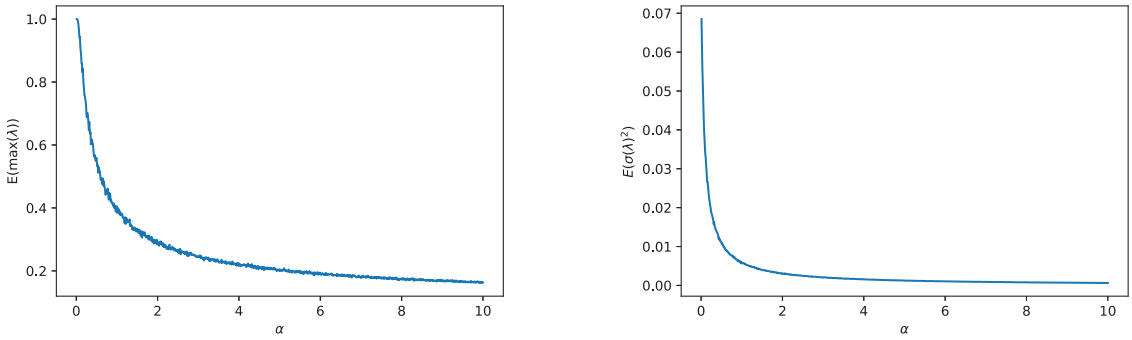
Mixup can be regarded as a kind of feature augmentation method. [39] proposed the mixup principle, which generates convex combinations of pairs of samples and their labels, and the weights of the pairs are usually sampled from a predefined Beta distribution. In our problem setting, each source domain only has a partial sensor set, but the union of all the source domains' partial sensor sets constitutes the complete set of sensors. Mixing the representations of source domains makes it possible for source domains to compensate for each other's missing components. However, the original mixup strategy only mixes two samples. To combine multiple source domains, we refer to [29] and use a Dirichlet distribution, an extension of Beta distribution to the multi-variant situation, to generate the random variable vector $\boldsymbol{\lambda}$. Then we introduce the procedure of the domain mixup method.

Suppose we have a batch of data and the batch size is $|B|$. Unlike the original mixup method, we only want to increase the variety of the feature space but do not want to change the activity label. In other words, we only mix the samples performed by different subjects of the same activity. For each sample i in the batch, we first find the samples with the same activity label and generate mixup weights for them. Thus, for each sample i, the random variable vector $\boldsymbol{\lambda}_i$ is calculated as follows:

$$\boldsymbol{\lambda}_i = Dirichlet(\boldsymbol{\alpha}_i) \tag{3}$$

Where $\boldsymbol{\alpha}_i = [\alpha_1, \alpha_2, ..., \alpha_{P_i}]$ is a predefined hyperparameter vector and P_i is the number of samples in the batch that have the same activity label as sample i. So the procedure is as follows, we traverse the mini-batch, and for each sample s we generate a corresponding mixup sample which is the combination of the original sample s and the other samples that have the same class as s. As we don't have access to the target domain and the similarities between the target domain and the source domains, we set $\alpha_1 = \alpha_2 = ... = \alpha_{P_i} = \alpha$ to pay equal attention to all the source domains. The relationships between α and the expectation of $max(\boldsymbol{\lambda})$ and $\sigma(\boldsymbol{\lambda})^2$ are shown in Figure 6. We can observe that with the increase of α, the generated combination weights are more deterministic and uniform. Figure 7 further visualizes the probability density function of a Dirichlet distribution. This figure shows the mixup sample value distribution of three original samples at the three corners of the triangle. The more red the color, the higher the probability. The distances between the point on the triangle and the three corners are inversely proportional to the weights of the three original samples. When the value of α is less than 1, the mixup samples are concentrated near the edge and corner. When $\alpha = 1$, all points on the triangular simplex are equally probable. When $\alpha > 1$, the distribution tends toward the center of the simplex. With the increase of α, the mixup values are more tightly concentrated. From the above analysis, we can find that with an $\alpha > 1$, the mixup samples are distributed across the triangular space. In this way, more diversity is introduced while more noisy samples are also introduced. We have a discussion of the impact of the value of α in Section 5.7.2.

As different samples have different numbers of samples with the same activity label, we generate mixup weights independently for each sample to increase the diversity of mixup results. All the samples and one-hot domain labels are mixed in the same way. For the i-th sample, the resulted mixup feature and domain label are calculated as follows:



(a) Relationship between α and the expectation of the maximum value of random variable vector $\boldsymbol{\lambda}$

(b) Relationship between α and the expectation of the variance of random variable vector $\boldsymbol{\lambda}$

Fig. 6. Relationship of α and $\boldsymbol{\lambda}$ under the condition of $|\mathcal{S}| = 12$.

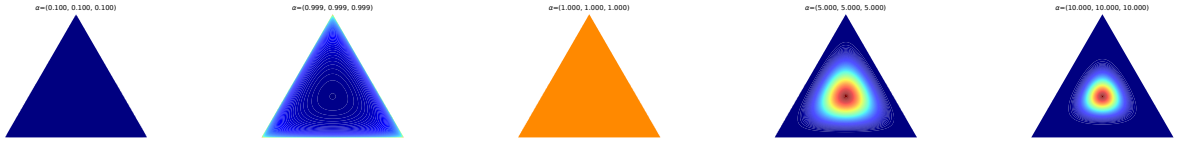


Fig. 7. Comparison of Dirichlet distributions with different α.

$$\hat{z}'^{i}_{mixup} = \sum_{y_j=y_i} \lambda_i \hat{z}'_j; \quad d^i_{mixup} = \sum_{y_j=y_i} \lambda_i d_j \tag{4}$$

And the sample class remains the same. Before passing through the following modules, all the original generated representations $\hat{Z}' = \{\hat{z}'_s\}_{s\in\mathcal{S}}$ and $\hat{z}'_{mixup}$ are aggregated respectively to fuse different sensors' data. Here we simply take the average along the sensor dimension to obtain $\bar{Z}' = \{\bar{z}'_s\}_{s\in\mathcal{S}}$ and $\bar{z}'_{mixup}$ as the fused representation for classification.

4.6 Discriminator and Classifier

To align the representations of different source domains into a common latent space, a discriminator is designed to distinguish which domain the data comes from and force the extracted representations to be subject-independent. The inputs to the discriminator are $\bar{Z}'$ and $\bar{z}'_{mixup}$ derived in the previous section. Here we let the discriminator discriminate each time step's sample. Thus for each time series sample with T time steps, the discriminator should output T results. The discriminator architecture adopts three fully connected layers with ReLU activation, and the last layer has $|\mathcal{S}|$ output units with a softmax activation function.

The aggregated representations are fed to the discriminator via a gradient reversal layer that multiplies the gradient by a certain negative constant during the backpropagation, which is the same as [10].

Our classifier takes the same architecture as that used in DeepSense [36]. It comprises two Gated Recurrent Unit (GRU) layers and one fully connected layer to extract the time-series information for classification.

4.7 Loss Function

Our loss function is comprised of four components: classification loss of original data, discrimination loss of original data, classification loss of mixup features, and discrimination loss of mixup features.

4.7.1 Classification loss. We choose the commonly used cross-entropy loss for the classification loss of original samples.

$$L_{cls}^{orig} = -\sum_{s\in\mathcal{S}} \frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{c=1}^{C} y_{i,c} log(\hat{y}_{i,c}) \tag{5}$$

We still adopt cross-entropy loss for classification for the mixup samples while the ground truth labels become the mixup labels.

$$L_{cls}^{mixup} = -\frac{1}{n_{mix}} \sum_{i=1}^{n_{mix}} \sum_{c=1}^{C} y_{i,c}^{mixup} log(\hat{y}_{i,c}^{mixup}) \tag{6}$$

4.7.2 Discriminator loss. The goal of the domain discriminator is to minimize the loss function L_{dis} to maximize the performance of domain label prediction, which contradicts our ultimate goal of learning domain-independent features of activities. As said in the last subsection, the discriminator is applied to each time step of the sample. Thus the discriminator loss takes the average cross-entropy of all the samples and time steps. The discrimination losses for original data and mixup data are shown as follows:

$$L_{dis}^{orig} = -\sum_{s\in\mathcal{S}} \frac{1}{n_s T} \sum_{i=1}^{n_s T} \sum_{j=1}^{|\mathcal{S}|} d_{i,j} log(\hat{d}_{i,j}) \tag{7}$$

$$L_{dis}^{mixup} = -\frac{1}{n_{mix} T} \sum_{i=1}^{n_{mix} T} \sum_{j=1}^{|\mathcal{S}|} d_{i,j}^{mixup} log(\hat{d}_{i,j}^{mixup}) \tag{8}$$

4.7.3 Total loss. The total loss function is as follows:

$$L = L_{cls}^{orig} + \beta L_{cls}^{mixup} + \gamma L_{dis}^{orig} + \beta\gamma L_{dis}^{mixup} \tag{9}$$

where β, γ are hyper-parameters indicating the importance of mixup samples' loss and discrimination loss, respectively. All the modules are trained together. Note that the features fed into the discriminator are reversed as mentioned in Section 4.6.

5 EVALUATION

In this section, we evaluate the performance of the proposed method on two wearable human activity recognition datasets and demonstrate its superiority over the state-of-the-art models.

5.1 Dataset and Data Preprocessing

For ease of convenience, we select datasets with multiple positions, subjects, and activity types and divide them into different parts to emulate different datasets of disjoint subject groups and sensor positions. We use two such kinds of datasets as follows.

The realistic sensor displacement activity recognition dataset (REALDISP) [3] covers 9 body positions, 17 subjects, and 33 activities. We only select the first 16 subjects and divide them into four groups, each containing four subjects. We alternatively make one group as the test data and the other three groups' subjects as the source domains. Note that each subject rather than each group forms a source domain. 90% of the source domain data are used for train and the rest 10% for validation. In our evaluation, we use ideal-placement sensor data to relieve training efforts for our main experiments. We specifically use the self-placement sensor data to verify our model's

robustness in section 5.4.4. The REALDISP dataset is sampled at 50Hz. Each sample lasts for 3 seconds with 150 data points. There are $|\mathcal{P}| = 9$ sensors in total, and each position's sensor is comprised of an inertial measurement unit (IMU) that has three modalities, and each modality has three axes. We divide each three-second sample into $T = 10$ fixed non-overlapped time steps [21]. Thus the dimension of the original signal is $x_s \in \mathbb{R}^{n_s \times |\mathcal{P}| \times T \times 3 \times 3 \times 15}$. As frequency features are more effective, we conduct Fourier transform to each time step's signal and concatenate the frequency magnitude (take the first half spectrogram) and phase as the final input data. Thus the input data become $x_s \in \mathbb{R}^{n_s \times |\mathcal{P}| \times T \times 3 \times 3 \times 14}$.

The Daily and Sports Activities Data Set (DSADS) [4] contains 19 daily and sports activities performed by eight subjects, each with five sensors attached to their body segments. This dataset is a little smaller than the REALDISP dataset. We divide the eight subjects into four groups, each containing two subjects. We alternatively make one group as the test data and the other three groups' subjects as the source domains where each subject represents a source domain. 90% of the source domain data are used for training, and the rest 10% for validation. The DSADS dataset has a sampling rate of 25Hz. Each subject performs each activity 60 times, and each time lasts for 5 seconds. There are $|\mathcal{P}| = 5$ sensors in total. Each position's sensor comprises an IMU with three modalities and three axes. We divide each five-second sample into $T = 5$ fixed non-overlapped time steps, and each timestep lasts one second. Thus the dimension of the original signal is $x_s \in \mathbb{R}^{n_s \times |\mathcal{P}| \times T \times 3 \times 3 \times 25}$. After conducting Fourier transform to each time step's signal and concatenating the frequency magnitude (take the first half spectrogram) and phase as the final input data, the input data become $x_s \in \mathbb{R}^{n_s \times |\mathcal{P}| \times T \times 3 \times 3 \times 24}$.

5.2 Evaluation Settings and Metrics

We conduct cross-validation on the datasets with different subject groups as the target domain and calculate the average results. To generate the initial incomplete data from the dataset, we generate an initial incomplete mask consisting of 0 and 1 for each subject. According to the incomplete rate, we calculate the number of unavailable sensors. We randomly select a subset of all the sensors with the size of the above number and set the corresponding values of the incomplete mask as 0. We multiply the incomplete mask with the complete data to generate the incomplete data. For the hybrid missing strategy, we set manual missing probabilities as $\{0, 0.2, 0.4, 0.6, 0.8\}$ to generate the additional missing masks for REALDISP dataset and $\{0, 0.2, 0.4, 0.6\}$ for DSADS dataset. For the domain mixup, as α is a hyperparameter that has an impact on the results, instead of tuning α we set it as a random number between 0 and 1 for each iteration. We load minibatch data from the whole dataset and set the batch size as 128. We adopt the default Adam optimizer with an initial learning rate of 0.001.

We report two metrics in our evaluation: accuracy and weighted F1 score. The weighted F1 score is the weighted average of the F1 score in each class. We just call it the F1 score for simplicity. The model is implemented with PyTorch 1.9 and Python 3.6. We conduct all the experiments on a cluster with NVIDIA Tesla V100 SXM2.

5.3 Methods

5.3.1 DeepSense. [36] We adopt a pervasively used multi-sensor fusion framework, DeepSense [36], as the backbone of our proposed framework. It is composed of a common feature extractor and a classifier that have the same architecture as the corresponding components of our proposed method. The results of this method demonstrate the sensitivity of the deep neural model faced with incomplete data.

5.3.2 Partial Ensemble. We train $|\mathcal{S}|$ independent models, each for a subject (source domain). In the test phase, we pass the target data into all these models and ensemble the prediction results as the final predictions. As the complete sensor set is the same across different subjects, we use the same feature extractors for all the subjects, i.e., each sensor position has a feature extractor, and it is shared among subjects. These feature extractors (each for one sensor position) constitute the common feature extractor which is the same as our proposed method. Each subject's data only pass through feature extractors that have the corresponding sensors' data, and the extracted

representations are averaged for classification. We train $|\mathcal{S}|$ independent classifiers, each for one subject. In the training phase, the loss is the sum of all the source domains' classification losses. In the inference phase, the data first pass through the common feature extractors to generate corresponding sensors' features and then pass through all the classifiers. The final prediction result simply takes the average of all the prediction results of source domains' classifiers.

5.3.3 Available. This method only uses the available sensor features. It adopts the same common feature extractor architecture. The original data first pass through the common feature extractor. For each source domain, the features fed into the classifier are the average of available sensors' features. The difference between this method and the partial ensemble method is that this method use one feature extractor and one classifier, and all the source domains' data are trained together.

5.3.4 DataAug. [8] Data augmentation is a standard method to enrich the diversity of data to overcome the domain shift between train and test. We adopt three kinds of transformations [8] (rescaling, axis rotation, and time-warping) to mimic the perturbations that commonly exist in accelerometer and gyroscope traces collected in the labeled dataset. We first add these transformations to the raw data and then conduct FFT to preprocess the data and get the corresponding magnitudes and phases. Then the augmented data are fed into the same network of the Available method.

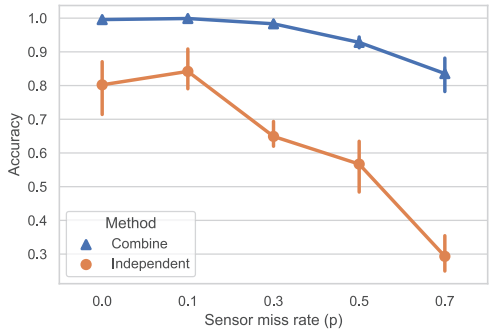
5.3.5 GAIN. [37] This method is an adaptation of [37]'s work for our problem. The original method is for data imputation on multi-variant datasets. We conduct the imputation on the latent representation space and add a classifier at the end. The feature extractor, generator, and classifier architectures are kept the same as our proposed method. Each sensor position has an independent discriminator with three fully connected layers, which is the same as our proposed method.

5.3.6 DANN. [10] This method is a classical domain adaptation method. It directly aligns the distributions of generated representations of all the source domains. Note that this baseline is adapted to the domain generalization setting, which does not need the data from the target domains. The input only has source domains' data, and we make the discriminator only classify which source domain the data come from. So the number of the discriminator's last layer's output neurons is equal to the number of the source domains. And the feature extractor is trained to make the discriminator unable to discriminate the domain where the data are from. This method has the same architecture as our proposed method, except that it doesn't take the augmentation strategies, i.e., hybrid missing and domain mixup strategies.

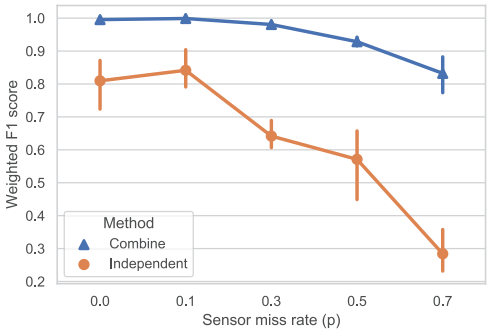
5.3.7 DGER. [41] This method argues that the adversarial training with a domain discriminator can only learn the invariant marginal distributions. In contrast, conditional marginal distributions are more critical for classification in new domains. With DANN[10] as the backbone, they propose an entropy regularization term that measures the dependency between the learned features and the class labels.

5.3.8 AALH. This is the proposed method with a loss function as shown in Equation 9. This method is used as the proposed method in our experiments by default unless otherwise specified.

5.3.9 AALH(aug). This is the combination of our proposed AALH method and data augmentation strategies. The input data are augmented with the same operations as DataAug, and then fed into the AALH network. This method shows whether our approach can combine with the common data augmentation methods to improve the performance jointly.



(a) Accuracy comparison between independently training and collective training under different miss rates



(b) F1 score comparison between independently training and collective training under different miss rates

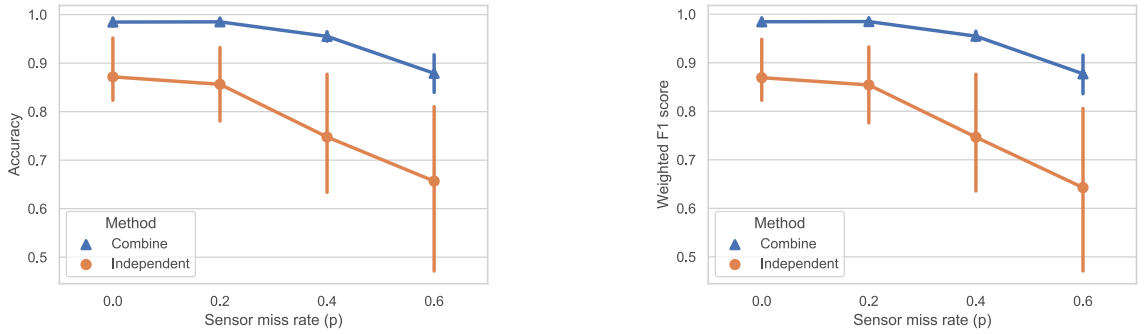
Fig. 8. Comparison between single model and fusion model on REALDISP dataset

5.4 Main Results

5.4.1 Single Model vs. Fusion Model. In this section, we compare the results of independently training each subject group's data and collectively training multiple subject groups' data under different miss rates. This experiment aims to show that combining different datasets' data, though with different subjects and sensor placements, can improve the classification performance compared with the independently training model for each subject group. Specifically, for each subject group, we take 90% of the data for training and the rest 10% for testing. The first way is to independently train models for each subject group and validate the performance on the 10% test data of this group. The other way is to first divide the training and testing data of the same subject group with the split rate of 9:1. For each subject group's training data, we combine them with the other two groups' training data and validate the subject group's testing data. Note that in both ways, we make sure that the subjects and the sensor numbers between train and test are the same while the sensor placements are different. As there are four subject groups in total, each subject group has three ways to combine with any two of the other three subject groups. Thus, we take the average of the validation results of the three combination ways. Figure 8 and Figure 9 show the validation results of independently training and collective training under different miss rates on REALDISP dataset and DSADS dataset, respectively. We can observe an obvious decrease in accuracy with the increase in incompleteness rate when independently training the model for one specific subject group. In contrast, when we combine the subject group's data with the other subject groups with different sensor placements, the performance only suffers from a much slighter decline. In addition, our proposed method has a larger improvement on the REALDISP dataset since there are more activity types such that our model is more likely to show the advantage of combining different sensor placements. Note that we don't test the model generalization performance when the same subjects wear a different number of sensors since we conduct the experiment in section 5.4.3 with a more challenging situation where both the subject and the sensor number are different.

5.4.2 Method Comparison. In this section, we compare the overall results of baseline methods and our proposed method on two datasets.

REALDISP Dataset We show the main results in Table 3. We report both the accuracy and weighted F1 score. For each target subject group, we conduct experiments on cases where both the source and target domains have



(a) Accuracy comparison between independently training and collective training under different miss rates

(b) F1 score comparison between independently training and collective training under different miss rates

Fig. 9. Comparison between single model and fusion model on DSADS dataset

a sensor miss rate of 0.1, 0.3, 0.5, and 0.7. Miss rate is the ratio of the number of unavailable sensors and total sensors. We also conduct experiments when the miss rate is 0 for reference. We only report the average and the standard deviation of the four cross-validation cases' results due to the page limit.

From the table, we can find that the partial ensemble method does not work at all. We observe that due to the limited data of each subject, an overfitting problem occurs in the training phase. Also, directly taking the average of all the prediction results may not be a good ensemble method.

Using available sensor data has significant improvement compared with the partial ensemble method since the feature extractor and classifier are shared among all the source domains. Thus, the data available for training the model are significantly increased, leading to better performance. Also, the ensemble of different sensors is conducted in the latent space, thus the classifier can be trained to adjust to the fused representation. From the results across different miss rates, we can find that the accuracy of low miss rates, i.e., 0.1, and 0.3, is similar. This is understandable since the input data may have some noisy or duplicate information, and appropriate missing may improve the model's robustness. However, when the miss rate increases to 0.5 and above, the accuracy substantially declines, indicating that the model cannot well handle the situation when the miss rate is high. Unfortunately, this situation may be common in daily life since the number of devices carried by persons usually resides in this range.

DeepSense method has similar results and trends as that of the Available method. These two methods are actually the same since once the miss rate is fixed, the number of available sensors is fixed. As the representation fed to the classifier is the average of all the sensors or available sensors, the difference between the extracted representations for classification is only a scalar. DataAug method performs similar to the Available method in terms of the average performance. It performs better than the Available method under the large miss rate, e.g., 0.7, demonstrating the improvement of generalization using standard data augmentation. But as it adds some noise to the original data, the performance slightly declines under lower miss rates.

With the attempt to reconstruct missing components in the latent space by adversarial learning, the GAIN method performs better than the method that only uses available sensors. DANN method has a similar performance as that of the Available method. It may result from the non-ideal alignment in the latent representation space. Both GAIN and DANN suffer from obvious performance degradation when the miss rate is large. DGER, as

Table 3. Average results of cross-validation for model comparison in REALDISP dataset. (Values in the cells represent "mean(std)"; "**Bold**" format for best performance in the column)

Miss rate		**0.0**	**0.1**	**0.3**	**0.5**	**0.7**	**Average**
DeepSense	Acc	0.902(0.035)	0.873(0.031)	0.813(0.072)	0.745(0.041)	0.551(0.136)	0.777(0.049)
	F1	0.901(0.034)	0.869(0.031)	0.810(0.072)	0.741(0.041)	0.541(0.140)	0.772(0.051)
Partial ensemble	Acc	0.330(0.125)	0.277(0.091)	0.241(0.118)	0.264(0.091)	0.227(0.058)	0.268(0.068)
	F1	0.306(0.130)	0.244(0.091)	0.192(0.119)	0.228(0.117)	0.170(0.053)	0.228(0.075)
Available	Acc	0.902(0.035)	0.846(0.010)	0.844(0.047)	0.714(0.066)	0.523(0.03)	0.766(0.027)
	F1	0.901(0.034)	0.845(0.011)	0.840(0.047)	0.700(0.072)	0.517(0.026)	0.761(0.029)
DataAug	Acc	0.889(0.034)	0.793(0.065)	0.784(0.042)	0.702(0.054)	0.614(0.073)	0.756(0.039)
	F1	0.887(0.035)	0.789(0.068)	0.781(0.040)	0.696(0.056)	0.615(0.072)	0.754(0.040)
GAIN	Acc	0.914(0.044)	0.905(0.023)	0.825(0.023)	0.734(0.107)	0.597(0.086)	0.795(0.041)
	F1	0.912(0.046)	0.901(0.022)	0.825(0.026)	0.728(0.109)	0.590(0.085)	0.791(0.041)
DANN	Acc	0.876(0.016)	0.862(0.044)	0.828(0.046)	0.718(0.017)	0.558(0.084)	0.768(0.023)
	F1	0.871(0.017)	0.863(0.043)	0.829(0.043)	0.711(0.013)	0.548(0.090)	0.765(0.025)
DGER	Acc	0.914(0.013)	0.907(0.024)	0.817(0.052)	0.786(0.077)	0.626(0.042)	0.810(0.035)
	F1	0.910(0.012)	0.904(0.024)	0.807(0.062)	0.780(0.082)	0.623(0.050)	0.805(0.040)
AALH	Acc	**0.979(0.006)**	**0.971(0.008)**	**0.955(0.016)**	**0.908(0.027)**	0.753(0.040)	**0.913(0.012)**
	F1	**0.978(0.006)**	**0.971(0.008)**	**0.955(0.016)**	**0.906(0.028)**	0.746(0.042)	**0.911(0.012)**
AALH(aug)	Acc	0.956(0.023)	0.930(0.035)	0.937(0.019)	0.845(0.063)	**0.804(0.035)**	0.895(0.035)
	F1	0.957(0.023)	0.929(0.037)	0.936(0.020)	0.839(0.067)	**0.799(0.037)**	0.892(0.039)

a recent domain generalization method, performs much better than DANN and GAIN and becomes the best baseline. However, as it is a general method that doesn't consider the specific characteristics of our problem, our approach still outperforms this baseline, especially for large miss rates.

Compared with the baselines, our proposed method has much better performance under all the evaluated miss rates. And we can observe that when the sensor set is complete (miss rate=0.0), our proposed method still achieves an improvement in accuracy compared with baseline methods. The model achieves similar performance for miss rates less than or equal to 0.3. When the miss rate reaches 0.7, the average performance has an explicit decline, but the performance remains above 75%, which is the accuracy achieved by other baselines at a miss rate of 0.5. The results demonstrate the robustness and generalization of our proposed model. Although AALH(aug) cannot surpass the average performance of our vanilla proposed method AALH, our method combined with traditional data augmentation strategies significantly improves under large miss rates, e.g., 0.7. This is consistent with the improvement achieved at the large miss rate of the DataAug method compared with its backbone, i.e., the Available method.

DSADS Dataset We show the main results in Table 4. We report both the accuracy and weighted F1 score. Miss rate is the ratio of the number of unavailable sensors and total sensors. For each target subject group, we conduct experiments on cases where both the source and target domains have a sensor miss rate of 0.2, 0.4, and 0.6, which results in the available number of sensors of 4, 3, and 2. We also conduct experiments when the miss rate is 0, i.e., remaining five sensors, for reference. We only report the average and the standard deviation of the four cross-validation cases' results due to the page limit.

From the table, we can find that the partial ensemble method performs extremely poorly. We observe that due to the limited data of each subject, an overfitting problem occurs in the training phase. For the other approaches, we have consistent findings as that in the REALDISP dataset, and we do not repeat them here. Compared with the baselines, our proposed method has much better performance under all the evaluated miss rates. And we can

Table 4. Average results of cross-validation for model comparison in DSADS dataset. (Values in the cells represent "mean(std)"; "**Bold**" format for best performance in the column)

Miss rate		0.0	0.2	0.4	0.6	Average
DeepSense	Acc	0.855(0.065)	0.806(0.065)	0.742(0.065)	0.564(0.062)	0.742(0.115)
	F1	0.848(0.071)	0.798(0.066)	0.738(0.066)	0.548(0.073)	0.733(0.119)
Partial ensemble	Acc	0.515(0.034)	0.512(0.072)	0.430(0.116)	0.372(0.028)	0.457(0.069)
	F1	0.500(0.037)	0.492(0.054)	0.403(0.124)	0.344(0.017)	0.435(0.075)
Available	Acc	0.855(0.065)	0.788(0.102)	0.707(0.012)	0.623(0.033)	0.743(0.081)
	F1	0.848(0.071)	0.781(0.108)	0.695(0.021)	0.614(0.035)	0.735(0.081)
DataAug	Acc	0.885(0.070)	0.789(0.054)	0.761(0.036)	0.628(0.121)	0.766(0.043)
	F1	0.884(0.048)	0.781(0.053)	0.747(0.039)	0.601(0.134)	0.753(0.047)
GAIN	Acc	0.835(0.071)	0.830(0.037)	0.691(0.039)	0.698(0.112)	0.761(0.082)
	F1	0.793(0.086)	0.821(0.043)	0.678(0.055)	0.682(0.118)	0.751(0.083)
DANN	Acc	0.770(0.084)	0.755(0.047)	0.690(0.011)	0.682(0.030)	0.724(0.045)
	F1	0.741(0.102)	0.738(0.044)	0.673(0.019)	0.672(0.041)	0.706(0.039)
DGER	Acc	0.856(0.029)	0.840(0.094)	0.648(0.096)	0.707(0.047)	0.752(0.042)
	F1	0.848(0.036)	0.834(0.103)	0.641(0.081)	0.703(0.053)	0.747(0.046)
AALH	Acc	**0.934(0.016)**	0.905(0.042)	0.874(0.039)	0.762(0.084)	0.869(0.034)
	F1	**0.933(0.017)**	0.900(0.045)	0.866(0.044)	0.754(0.089)	0.863(0.037)
AALH(aug)	Acc	0.925(0.025)	**0.910(0.023)**	**0.905(0.022)**	**0.787(0.087)**	**0.882(0.032)**
	F1	0.924(0.027)	**0.909(0.022)**	**0.903(0.023)**	**0.782(0.090)**	**0.879(0.032)**

observe that when the sensor set is complete (miss rate=0.0), our proposed model achieves a great improvement in accuracy. With the increase in miss rates, the accuracy steadily decreases. The reason is that this dataset has fewer sensors and less duplicate information, and the missing of each sensor may influence the final results. However, when the miss rate is as large as 0.6, the accuracy still remains above 75%. And the average accuracy of AALH is 10.3% higher than the best method of the baselines. AALH(aug), the combination of our method and the common data augmentation method, performs even better than our proposed method AALH in this dataset in terms of the average results and the results in large miss rates. Combining our method with the common data augmentation method is promising to achieve better results. The results demonstrate the robustness and generalization of our proposed model.

5.4.3 Model Generalization. As illustrated in our introduction, different users usually wear different combinations of sensors in daily life that may not have been seen in the training phase. The above experiments keep the miss rates of the train and test set the same. Here we deploy the above model on test data with different miss rates and show the confusion matrix of accuracy and F1 score for different train and test miss rate pairs on REALDISP dataset and DSADS dataset in Figure 10 and Figure 11, respectively. Each row of the matrix is the miss rate in the training phase, while each matrix column is the miss rate in the test phase. Each value of the matrix represents the test accuracy or F1 score under the miss rate of the corresponding column miss rate using the model trained with the corresponding row miss rate.

REALDISP dataset. We have some interesting findings from our model. First, for each row of the lower triangular matrix, we find that the model trained on miss rate p_s applied on test data with lower miss rate $p_t < p_s$ achieves better results than applied on test data with the same miss rate $p_t = p_s$ in most cases. This means that,

(a) Accuracy confusion matrix of the model applied on different test miss rate

(b) F1 score confusion matrix of the model applied on different test miss rate

Fig. 10. Performance of well-trained model applied on different test miss rates on REALDISP dataset

(a) Accuracy confusion matrix of the model applied on different test miss rate

(b) F1 score confusion matrix of the model applied on different test miss rate

Fig. 11. Performance of well-trained model applied on different test miss rates on DSADS dataset

with the same trained model, the more complete the test data, the better the performance. Second, for each row, the decrease in accuracy and F1 score due to the increase of test miss rate is smaller than 15.5% for all the models trained with different miss rates, demonstrating the effectiveness of our model for handling different sensor numbers. Third, for each column of the matrix, for the same test miss rate, the model trained with a smaller train miss rate usually achieves a better result. The reason is that with a smaller train miss rate, the data are more complete, and the model can extract more useful information. Finally, the colors of each column are similar, meaning that the models trained on different miss rates achieve similar results on the same target. The largest difference between the best and the other column values is 13.1%, indicating that even though the trained model

Table 5. Average results of cross-validation for model comparison in REALDISP dataset under "SELF" sensor placement. *(Values in the cells represent "mean(std)"; "**Bold**" format for best performance in the column)*

Miss rate		0.0	0.1	0.3	0.5	0.7	Average
Available	Acc	0.773(0.112)	0.688(0.175)	0.669(0.119)	0.583(0.065)	0.402(0.150)	0.623(0.108)
	F1	0.800(0.076)	0.710(0.143)	0.683(0.099)	0.595(0.049)	0.397(0.149)	0.637(0.079)
DataAug	Acc	0.749(0.117)	0.644(0.085)	0.663(0.117)	0.590(0.123)	0.399(0.056)	0.609(0.094)
	F1	0.770(0.095)	0.664(0.059)	0.685(0.093)	0.607(0.101)	0.406(0.033)	0.626(0.067)
GAIN	Acc	**0.818(0.150)**	0.771(0.166)	0.602(0.216)	0.504(0.110)	0.445(0.118)	0.628(0.143)
	F1	**0.849(0.114)**	0.801(0.133)	0.641(0.176)	0.511(0.097)	0.458(0.095)	0.652(0.112)
DANN	Acc	0.730(0.159)	0.644(0.154)	0.533(0.043)	0.475(0.075)	0.343(0.045)	0.545(0.077)
	F1	0.758(0.130)	0.673(0.129)	0.552(0.072)	0.491(0.076)	0.338(0.030)	0.562(0.051)
DGER	Acc	0.740(0.186)	0.685(0.206)	0.631(0.160)	0.594(0.082)	0.417(0.078)	0.613(0.139)
	F1	0.762(0.166)	0.710(0.179)	0.655(0.143)	0.626(0.052)	0.440(0.073)	0.639(0.118)
AALH	Acc	0.808(0.146)	**0.790(0.185)**	0.760(0.154)	0.710(0.107)	**0.537(0.105)**	**0.721(0.136)**
	F1	0.837(0.114)	**0.821(0.150)**	0.795(0.116)	0.733(0.072)	**0.533(0.115)**	**0.744(0.107)**
AALH(aug)	Acc	0.799(0.129)	0.772(0.129)	**0.762(0.121)**	**0.713(0.117)**	0.505(0.150)	0.710(0.115)
	F1	0.834(0.088)	0.803(0.097)	**0.797(0.086)**	**0.742(0.082)**	0.510(0.143)	0.737(0.078)

is not targeted at the test miss rate, it can still achieve satisfactory performance on the target, demonstrating model generalization.

DSADS dataset. As the dataset only has five sensors, we only conduct experiments on the same set of miss rates, i.e., 0, 0.2, 0.4, and 0.6. We have some consistent findings of our model on this dataset. First, similar to the REALDISP dataset, from each row of the matrix, we find that the model trained on miss rate p_s applied on test data with lower miss rate $p_t < p_s$ achieves better results than applied on test data with the same miss rate $p_t = p_s$ in most cases. This means that, with the same trained model, the more complete the test data, the better the performance. Second, the decrease in accuracy and F1 score due to the increase in miss rate is smaller than 17.3% for all the models trained with different miss rates, demonstrating the effectiveness of our model for handling different sensor numbers. Third, from each column of the matrix, for the same test miss rate, the model trained on a smaller train miss rate usually achieves a better result. Finally, the colors of most columns are similar, meaning that the models trained on different miss rates achieve similar results on the target. The largest difference between the largest and smallest values of one column is 12.9%, and the smallest difference is only 5.5%. This indicates that even though the trained model is not targeted at the test miss rate, it can still achieve satisfactory performance on the target, demonstrating model generalization.

5.4.4 Model Robustness. To further demonstrate our method's robustness, we compare our proposed approach and the other baselines under non-ideal sensor displacement situations. We do not include the Partial Ensemble method here as it has obvious bad performance. The DeepSense method is similar to the Available method, and we only include the Available method. The REALDISP dataset has different sensor displacement scenarios. In the self-placement scenario, the users are asked to position three sensors on the body parts specified by the instructor. Normally, self-placement will lead to on-body sensor setups that differ from ideal placement. We show the results in Table 5. We can observe that our proposed method still outperforms the other baselines in terms of the average result. The improvement starts to be obvious from the miss rate of 0.3. The improvement is as great as 9.3% in average accuracy compared to the best baseline.

5.5 Cross Dataset Evaluation

In this section, we conduct cross dataset experiments to evaluate the performance of our method under different data collection situations, which is important for real situations. DSADS dataset collects 19 activities through 8 subjects while REALDISP Dataset collects 33 activities through 17 subjects. Both datasets use body-worn sensor units, including a triaxial accelerometer, triaxial gyroscope, and triaxial magnetometer. Each sample in the REALDISP dataset lasts for 3 seconds, and the sampling rate is 50Hz, while each sample in DSADS dataset lasts for 5 seconds, and the sampling rate is 25Hz. To make the input share the same format, we follow the data format of the DSADS dataset, where there are five intervals, and each interval has 25 data points. For the REALDISP dataset, we also split the data into five segments where each segment has 30 data points, and we downsample each segment to 25 data points. Our method requires the same label space of source and target domains. We choose the five common activities from both datasets: Walking, Running, Jumping up, Rowing, and Cycling. We use all the 17 subjects in the REALDISP dataset as training subjects. Each subject composes a single domain, while all the subjects in the DSADS dataset compose the target domain. There are nine sensors in the REALDISP dataset and five in the DSADS dataset. So there are nine feature extractors, each for a sensor in the REALDISP dataset. The DSADS dataset's sensor positions can roughly find their corresponding partners in the REALDISP dataset (see the mapping in Table 6), and the sensor data will pass through the corresponding sensor feature extractors to extract features. Our method will generate the remaining incomplete sensors' features as described. Table 6 shows the detailed experiment setting.

Table 6. Cross Dataset Experiment Setting

Dataset	Role	Subjects	Classes	Sensor Positions
REALDISP	Source	1~17	0: Walking, 1: Running, 2: Jump up, 3: Rowing, 4: Cycling	0: right lower arm, 1: right upper arm, 2: back, 3: left upper arm, 4: left lower arm, 5: right calf, 6: right thigh, 7: left thigh, 8: left calf
DSADS	Target	1~8	0: Walking in a parking lot, Walking (4km/h) on a treadmill in flat, 1: Running, 2: Jumping, 3: Rowing, 4: Cycling on an exercise bike horizontally	2: torso, 0: right arm, 4: left arm, 6: right leg, 7: left leg

We compare our method with the baselines mentioned above. The results are shown in Table 7. As we only select several common classes of the two datasets, the number of samples is much smaller than the whole dataset. The training of adversarial learning will be unstable. Also, the selected classes are more distinguishable, making the training easily overfitting to the training data. Due to the discrepancy of the complete sensor number of the two datasets, i.e., nine and five, the decreasing trend is not very obvious with the increase of the miss rate. Note that the accuracy is much higher than before since there are only five classes. We can see that our method still outperforms the other baselines in most miss rates. In addition, our approach is well combined with the traditional data augmentation method to improve the performance further.

5.6 Ablation Study

For the ablation study, we mainly test the performance of the two augmentation strategies. The accuracy and weighted F1 score results under different miss rates on the REALDISP dataset and DSADS dataset are shown in

Table 7. Average results of cross-validation in cross dataset situation. *(Values in the cells represent "mean(std)"; "**Bold**" format for best performance in the column)*

Miss rate		0.0	0.2	0.4	0.6	Average
Available	Acc	0.769	0.747	0.706	**0.790**	0.753
	F1	0.716	0.718	0.701	**0.781**	0.729
DataAug	Acc	0.861	0.718	0.751	0.780	0.777
	F1	0.861	0.666	0.707	0.765	0.750
GAIN	Acc	0.800	0.608	0.652	0.680	0.685
	F1	0.792	0.573	0.598	0.676	0.660
DANN	Acc	0.791	0.601	0.650	0.656	0.675
	F1	0.736	0.584	0.654	0.656	0.657
DGER	Acc	0.816	0.699	0.790	0.688	0.748
	F1	0.820	0.654	0.776	0.617	0.717
AALH	Acc	0.853	0.825	0.818	0.784	0.820
	F1	0.843	0.801	0.814	0.781	0.810
AALH(aug)	Acc	**0.902**	**0.913**	**0.834**	0.770	**0.855**
	F1	**0.901**	**0.911**	**0.827**	0.766	**0.851**

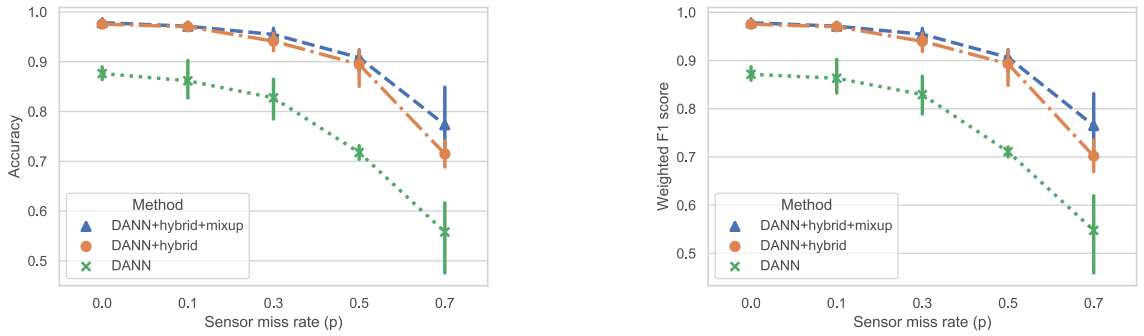
Figure 12 and Figure 13, respectively. Each figure shows three lines representing the DANN method, DANN plus hybrid missing strategy, and DANN plus hybrid missing strategy and domain mixup method.

REALDISP Dataset We conduct the ablation study under miss rates: 0.0, 0.1, 0.3, 0.5 and 0.7. We can find that the hybrid missing strategy contributes a lot to performance improvement. This simple strategy leads to a large performance gap between DANN and DANN plus hybrid missing. However, although the domain mixup method has better performance in average accuracy, the strategy may only be effective when the miss rate is larger than or equal to 0.5. Moreover, the improvement is about 4.1% when the miss rate is 0.7 in terms of average accuracy. This results from a large number of sensors in this dataset since the information is still enough to distinguish the activities when only one or two sensors are missed. In reality, a large miss rate that means less than four sensors are left draws our attention since this fits daily life situations where people carry several modern devices to recognize activity jointly. Thus the domain mixup strategy works well in practical cases.

DSADS Dataset We conduct the ablation study under miss rates: 0, 0.2, 0.4, and 0.6. We can find that the hybrid missing strategy lifts the performance and forms an obvious gap between DANN plus hybrid missing strategy and DANN baseline. For the domain mixup strategy, it is effective under all the miss rates, and the improvements are similar across different miss rates. This may result from the limited sensor number of the dataset. Thus, once some sensors are left, the performance can be explicitly improved with our multi-domain mixup strategy.

5.7 Parameter Study

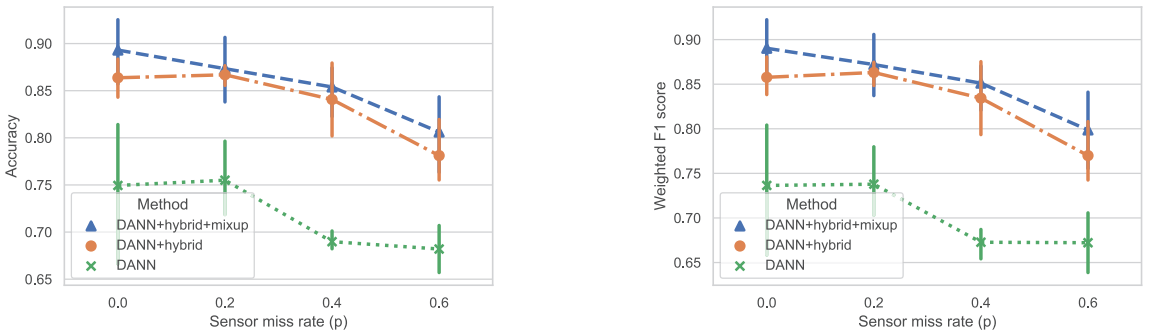
5.7.1 Impact of Hybrid Missing Rate Sets. One significant parameter in our hybrid missing strategy is how many missing rates we use to augment the representation space. In this section, we choose a different number of missing rates and different missing rate sets that can uniformly cover the whole range of 0 to 1 to the utmost extent.



(a) Accuracy comparison under different miss rates

(b) F1 score comparison under different miss rates

Fig. 12. Ablation study of two augmentation strategies on REALDISP dataset



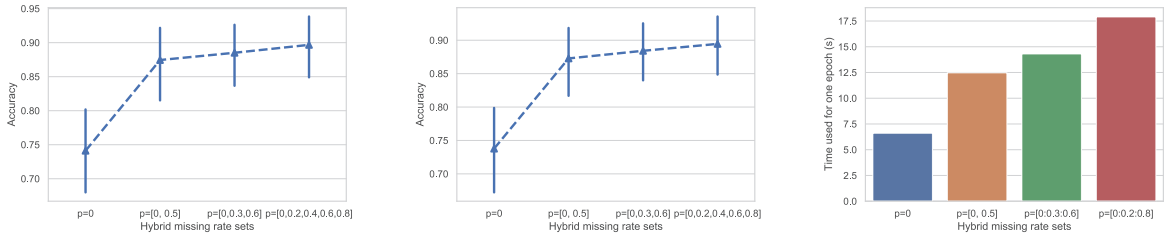
(a) Accuracy comparison under different miss rates

(b) F1 score comparison under different miss rates

Fig. 13. Ablation study of two augmentation strategies on DSADS dataset

REALDISP Dataset. The selective hybrid miss rate sets are: $\{0\}$, $\{0, 0.5\}$, $\{0, 0.3, 0.6\}$ and $\{0, 0.2, 0.4, 0.6, 0.8\}$, where $\{0\}$ means not using hybrid missing strategy. We report the accuracy, weighted F1 score, and time used for one epoch in Figure 14. We can find that with the increase of the number of miss rates, the performance linearly increases. The time used for training one epoch increases substantially if we choose to use additional missing rates. The reason is that in our mixup method, we need to traverse the batch and generate corresponding mixup samples for each sample in the batch. The time used still increases steadily with the number of miss rates increasing, but the difference is small between adjacent miss rate sets. Thus we choose the miss rate set as $\{0, 0.2, 0.4, 0.6, 0.8\}$ which is a good trade-off between performance and computation. Note that this is a one-time cost since we don't need the strategy in the test phase.

DSADS Dataset. The selective test miss rate sets are: $\{0\}$, $\{0, 0.5\}$, $\{0, 0.3, 0.6\}$ and $\{0, 0.2, 0.4, 0.6\}$. We report the accuracy, weighted F1 score, and time used for one epoch in Figure 15. We don't include 0.8 in the miss rate

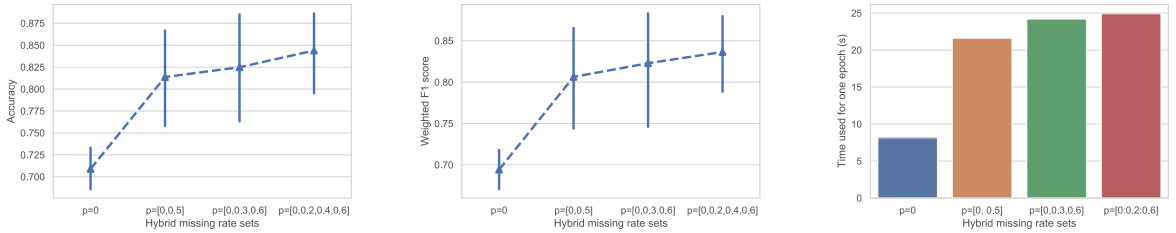


(a) Accuracy for different hybrid missing rate sets

(b) F1 score for different hybrid missing rate sets

(c) Time used for training one epoch for different hybrid missing rate sets

Fig. 14. Impact of hybrid missing rate sets for accuracy, F1 score, and time used for training one epoch on REALDISP dataset



(a) Accuracy for different hybrid missing rate sets

(b) F1 score for different hybrid missing rate sets

(c) Time used for training one epoch for different hybrid missing rate sets

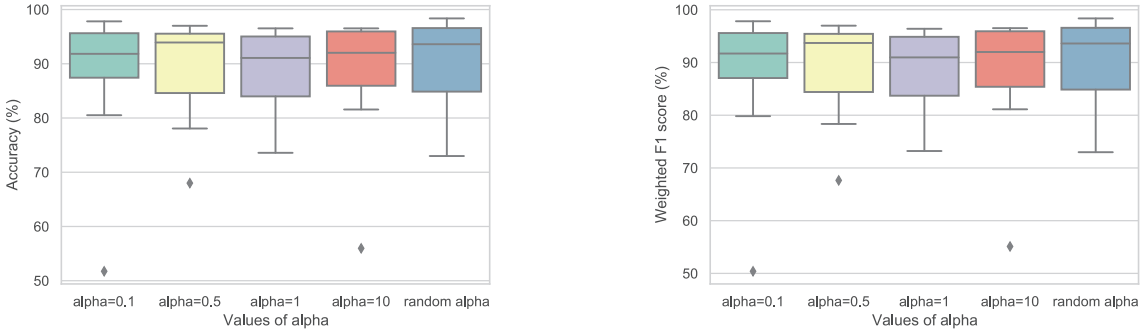
Fig. 15. Impact of hybrid missing rate sets for accuracy, F1 score, and time used for training one epoch on DSADS dataset

set because the total number of sensors in the DSADS dataset is only 5. Even though the sensors are complete, with a 0.8 miss rate, there is only one sensor left. If the 0.8 miss rate plays the role of the additional mask, it is too large to work since each sample should include at least one sensor's data such that the 0.8 miss rate will not increase diversity but introduce duplicate information. We can find that with the increase of the number of miss rates, the performance first significantly increases and then steadily increases. From the perspective of training time for one epoch, we find that time differences between different miss rate sets are relatively small. Therefore, for this dataset, according to our experiment, the miss rate set should be $\{0, 0.2, 0.4, 0.6\}$ to achieve better performance.

5.7.2 Impact of α. There is a parameter α that needs to be set in Dirichlet distribution to generate the combination weights of all the domains. We conduct a sensitivity analysis of this parameter.

REALDISP Dataset. The value set of α for the experiment is selected as $\{0.1, 0.5, 1, 10\}$ and a random number between 0 and 1 in each iteration, which is the method we used in previous experiments, is also selected for comparison. We show the performance of all the cross-validation cases in the format of a boxplot. The miss rate is the set of $\{0.1, 0.3, 0.5, 0.7\}$. The result is shown in Figure 16. We can find that the average results are similar and not sensitive to the value of α. However, the variation range of the performance is influenced by the value of α. It seems that when the value of α is too large or too small, meaning that the mixed sample is almost the

average of all the domains' samples or the original data, outliers occur, leading to bad performance in one or two specific cases. Thus the value of α needs to be tuned for different datasets. However, when we take a random α, we can prevent outlier cases, showing that setting α as a random value works for all the cases and can guarantee satisfactory performance.

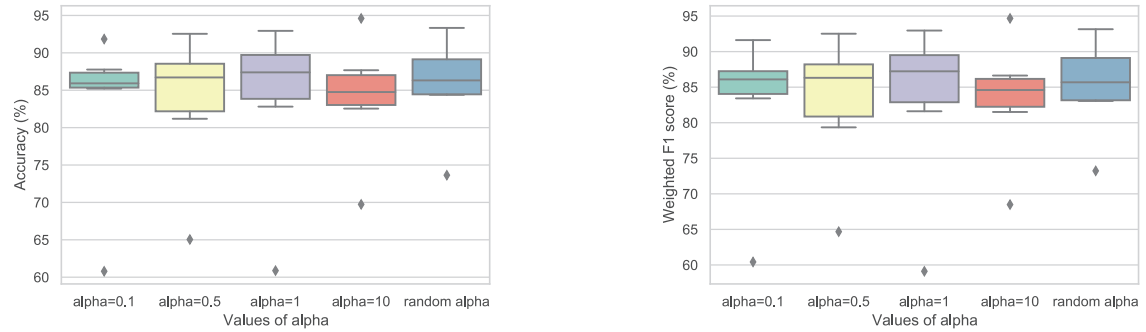


(a) Accuracy comparison under different α

(b) F1 score comparison under different α

Fig. 16. Sensitivity test for α in Dirichlet distribution on REALDISP dataset

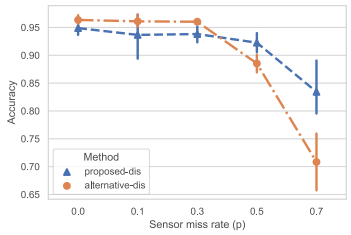
DSADS Dataset. The range of α is selected as $\{0.1, 0.5, 1, 10\}$ and the random number between 0 and 1. We show the performance of all the cross-validation cases in the format of a boxplot. The miss rate set is selected as the set of $\{0.2, 0.4, 0.6\}$. We can observe that the average accuracy and weighted F1 score fluctuate slightly with the variation of the value of α while the variation range changes apparently with different values of α. Although there are outliers in all cases, the result obtained by setting α as a random number has the minimum variation range, demonstrating our method's effectiveness in getting satisfactory performance under various cases.



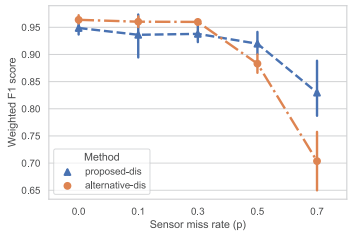
(a) Accuracy comparison under different α

(b) F1 score comparison under different α

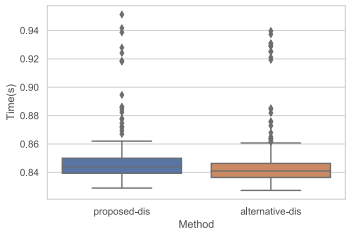
Fig. 17. Sensitivity test for α in Dirichlet distribution on DSADS dataset



(a) Accuracy comparison of the two formats of discriminators in REALDISP dataset

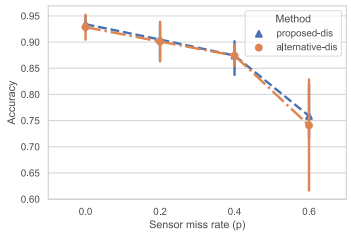


(b) F1 score comparison of the two formats of discriminators in REALDISP dataset

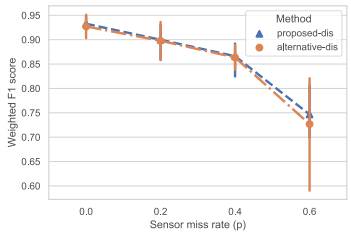


(c) Time used for training one iteration for two formats of discriminators in REALDISP dataset

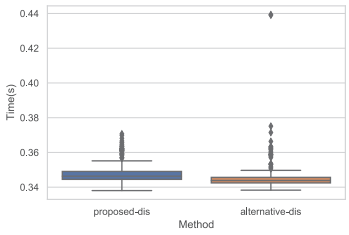
Fig. 18. Performance comparison of two formats of discriminators in REALDISP dataset.



(a) Accuracy comparison of the two formats of discriminators in DSADS dataset



(b) F1 score comparison of the two formats of discriminators in DSADS dataset



(c) Time used for training one iteration for two formats of discriminators in DSADS dataset

Fig. 19. Performance comparison of two formats of discriminators in DSADS dataset.

5.7.3 Impact of the Discriminator Format. The input to the discriminator is of the shape (B, T, F) where B is the batch size, T is the time steps in the sequence, and F is the feature dimension. As the discriminator comprises fully connected (FC) layers that only take 2D data as input, we choose to reshape the input to $(B \times T, F)$, making the discriminator process each time step independently, and we call this method 1. An alternative is to reshape the input to $(B, T \times F)$, which will increase the feature dimension by T, and we call this method 2. The current setup will include some more computational overhead. The number of multiplication operations calculated by method 1 is $T \times (F \times D_1 + D_1 \times D_2 + D_2 \times D_3)$, where D_1, D_2, and D_3 are the output neurons of the FC layers in the discriminator. And the number of multiplication operations calculated by method 2 is $T \times F \times D_1 + D_1 \times D_2 + D_2 \times D_3$. The additional operations of the current setup (Method 1) is $(T-1) \times D_1 \times D_2 + (T-1) \times D_2 \times D_3$. The discriminator has simple architecture where $D_1 = 32$, $D_2 = 16$, and D_3 is the number of source subjects, and the computational overhead is not too much. But the advantage of the current setup is that the discriminator parameters are fewer, making it easier to train. We can see from Figure 18 and Figure 19 that compared with the alternative, our current discriminator format has better performance, especially in larger miss rates, and the computational time for each iteration is similar to before.

6 DISCUSSION AND FUTURE WORK

In this section, we discuss the aspects from which our framework can be further improved in future work.

6.1 Unseen Sensors

Our setup assumes that the target partial set sensors must be included in the source domain. Although this is a restriction, we think the problem setup is still realistic for three reasons: First, the positions that frequently wear sensors are countable, e.g., wrist, back, left lower leg, etc., determined by the user habit and product format. Thus there is a significant chance that the source domain sensors can cover the target sensor. Second, there are abundant wearable sensor-based datasets that have collected data from different sensors. We can leverage these datasets to train the model with as many sensor positions as possible. Third, our method still outperforms the other baselines in the non-ideal sensor displacement and cross dataset situations. We leave the problem of the unseen sensor in the target domain as our future work, e.g., the source domains have sensors attached to the upper body while the target domains have sensors attached to the lower body. This is a difficult situation since the framework should be able to handle both various sensor combinations and new sensor positions. Based on our current model architecture, an additional branch for feature extraction can be set for the newly unseen sensor. But the source domain data may need more complicated augmentations to cover the distribution of the target domain.

6.2 User Input

As we can see from Fig. 3, the masks indicating the missing sensors need the specific information of which sensors are missed. This calls for the user input of the sensor position. We think this is not a heavy burden for the users in the current design as the available sensors are only a countable set. The users may select the sensor position when putting on them. In addition, the user will not frequently change the wearing position during activity since the product form determines the potential wearing positions. Lastly, we have found some papers [1, 27] that can predict the sensor positions from the raw sensor data. Our method can be combined with these works to relieve the user's burden.

6.3 Adaptive Missing Strategy

As we can see from our parameter study, the number of missing ratios has an impact on the final result. It is better to find an adaptive way to use our hybrid missing strategy and set the number of missing branches. There are some works [6] that additionally train a network fed with some intermediate network result and use this information to guide the hyperparameter setting. For our case, we may train such a network to output a mask that indicates which missing rates are valid. We leave this in our future work.

6.4 Advanced Model Architecture

In this paper, most of the modules in our framework are multilayer perceptron (MLP) which are composed of fully connected layers, and we didn't pay much attention to finding advanced model architecture for better performance and generalization. There is still opportunity to improve model performance by some advanced architectures such as attention layer [31], BERT [9], etc. Also, graph neural networks (GNN) have drawn more and more attention. [22] has tried to use GNN to recover missing representations with ground truth values in the training phase. There may be a chance to model the partial sensor sets on the human body as partial graphs and think about how to fuse multiple partial graphs and generalize the model to the unseen partial graph.

6.5 Different Label Space

In our problem setting, different subjects share the same label space. However, there is a more practical and challenging problem where the target subject has different activities compared to the source subjects. There are some recent works [16, 19] considering such heterogeneous domain generalization problems. We expect to handle such heterogeneous label space problems in the future.

7 CONCLUSION

In this paper, we investigate a novel and practical framework for HAR, which is to combine diverse datasets with heterogeneous subjects and sensor sets and derive a general model that can achieve high performance in new application scenarios with unseen subjects and sensor combinations. To mitigate the discrepancies of heterogeneous datasets, we develop two augmentation strategies, i.e., hybrid missing strategy and multi-domain mixup strategy, that significantly improve the robustness and generalization of the model. Experiments show that the proposed method can achieve an improvement of 10.3% in average accuracy on the REALDISP dataset and an improvement of 10.3% in average accuracy on the DSADS dataset, respectively, compared with the corresponding best baselines, demonstrating the superior performance of our framework.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous editors and reviewers for their valuable comments and helpful suggestions. This research is supported in part by the Key-Area Research and Development Program of Guangdong Province (No. 2020B0101390001), in part by the National Natural Science Foundation of China (No. 62002150), in part by the Hong Kong RGC under Contract C7016, Contract C6889, and Contract RIF R6021-20, and in part by HKUST under Contract R8015 and Contract L3016. We would like to thank the Turing AI Computing Cloud (TACC) [34] and HKUST iSING Lab for providing us computation resources on their platform.

REFERENCES

[1] Navid Amini, Majid Sarrafzadeh, Alireza Vahdatpour, and Wenyao Xu. 2011. Accelerometer-based on-body sensor localization for health and medical monitoring applications. *Pervasive and mobile computing* 7, 6 (2011), 746–760.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. 2012. Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. In *IWAAL*.

[3] Oresti Banos, Mate Attila Toth, Miguel Damas, Hector Pomares, and Ignacio Rojas. 2014. Dealing with the Effects of Sensor Displacement in Wearable Activity Recognition. *Sensors* 14, 6 (2014), 9995–10023. https://doi.org/10.3390/s140609995

[4] 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. https://doi.org/10.1093/comjnl/bxt075

[5] Andreas Bulling, Jamie A. Ward, and Hans Gellersen. 2012. Multimodal Recognition of Reading Activity in Transit Using Body-Worn Sensors. *ACM Trans. Appl. Percept.* 9, 1, Article 2 (March 2012), 21 pages. https://doi.org/10.1145/2134203.2134205

[6] Qi Cai, Yingwei Pan, Yu Wang, Jingen Liu, Ting Yao, and Tao Mei. 2020. Learning a Unified Sample Weighting Network for Object Detection. arXiv:2006.06568 [cs.CV]

[7] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I Jordan. 2018. Partial transfer learning with selective adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2724–2732.

[8] Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. 2020. A Systematic Study of Unsupervised Domain Adaptation for Robust Human-Activity Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1 (2020), 39:1–39:30. https://doi.org/10.1145/3380985

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[10] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. arXiv:1505.07818 [stat.ML]

[11] 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.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML]

[13] Natasha Jaques, Sara Taylor, Akane Sano, and Rosalind Picard. 2017. Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction. In *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*. 202–208. https://doi.org/10.1109/ACII.2017.8273601

[14] Jeya Vikranth Jeyakumar, Liangzhen Lai, Naveen Suda, and Mani Srivastava. 2019. SenseHAR: A Robust Virtual Activity Sensor for Smartphones and Wearables. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems* (New York, New York) *(SenSys '19)*. Association for Computing Machinery, New York, NY, USA, 15–28. https://doi.org/10.1145/3356250.3360032

[15] Paula Lago, Moe Matsuki, Kohei Adachi, and Sozo Inoue. 2021. *Using Additional Training Sensors to Improve Single-Sensor Complex Activity Recognition.* Association for Computing Machinery, New York, NY, USA, 18–22. https://doi-org.lib.ezproxy.ust.hk/10.1145/3460421.3480421

[16] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M. Hospedales. 2019. Episodic Training for Domain Generalization. arXiv:1902.00113 [cs.CV]

[17] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot. 2018. Domain Generalization with Adversarial Feature Learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5400–5409. https://doi.org/10.1109/CVPR.2018.00566

[18] Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. 2019. MisGAN: Learning from Incomplete Data with Generative Adversarial Networks. arXiv:1902.09599 [cs.LG]

[19] Yiying Li, Yongxin Yang, Wei Zhou, and Timothy M. Hospedales. 2019. Feature-Critic Networks for Heterogeneous Domain Generalization. arXiv:1901.11448 [cs.LG]

[20] Zhenpeng Li, Jianan Jiang, Yuhong Guo, Tiantian Tang, Chengxiang Zhuo, and Jieping Ye. 2020. Domain Adaptation with Incomplete Target Domains. arXiv:2012.01606 [cs.LG]

[21] Shengzhong Liu, Shuochao Yao, Yifei Huang, Dongxin Liu, Huajie Shao, Yiran Zhao, Jinyang Li, Tianshi Wang, Ruijie Wang, Chaoqi Yang, and Tarek Abdelzaher. 2020. Handling Missing Sensors in Topology-Aware IoT Applications with Gated Graph Neural Network. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (4 Sept. 2020). https://doi.org/10.1145/3411818 Publisher Copyright: © 2020 ACM. Copyright: Copyright 2020 Elsevier B.V., All rights reserved..

[22] Shengzhong Liu, Shuochao Yao, Yifei Huang, Dongxin Liu, Huajie Shao, Yiran Zhao, Jinyang Li, Tianshi Wang, Ruijie Wang, Chaoqi Yang, and Tarek Abdelzaher. 2020. Handling Missing Sensors in Topology-Aware IoT Applications with Gated Graph Neural Network. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 3, Article 90 (Sept. 2020), 31 pages. https://doi.org/10.1145/3411818

[23] Shengzhong Liu, Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Huajie Shao, and Tarek Abdelzaher. 2020. GIobalFusion: A Global Attentional Deep Learning Framework for Multisensor Information Fusion. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1, Article 19 (March 2020), 27 pages. https://doi.org/10.1145/3380999

[24] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. 2010. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of Machine Learning Research* 11, 80 (2010), 2287–2322. http://jmlr.org/papers/v11/mazumder10a.html

[25] Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. 2017. Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 5115–5124.

[26] Attila Reiss and Didier Stricker. 2012. Introducing a New Benchmarked Dataset for Activity Monitoring. In *2012 16th International Symposium on Wearable Computers*. 108–109. https://doi.org/10.1109/ISWC.2012.13

[27] Vu Ngoc Thanh Sang, Shiro Yano, and Toshiyuki Kondo. 2018. On-Body Sensor Positions Hierarchical Classification. *Sensors* 18, 11 (2018). https://doi.org/10.3390/s18113612

[28] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. 2018. Generalizing Across Domains via Cross-Gradient Training. arXiv:1804.10745 [cs.LG]

[29] Yang Shu, Zhangjie Cao, Chenyu Wang, Jianmin Wang, and Mingsheng Long. 2021. Open Domain Generalization with Domain-Augmented Meta-Learning. arXiv:2104.03620 [cs.CV]

[30] Yonatan Vaizman, Nadir Weibel, and Gert Lanckriet. 2018. Context Recognition In-the-Wild: Unified Model for Multi-Modal Sensors and Multi-Label Classification. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 168 (jan 2018), 22 pages. https://doi.org/10.1145/3161192

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[32] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. 2021. Time Series Data Augmentation for Deep Learning: A Survey. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence* (Aug 2021). https://doi.org/10.24963/ijcai.2021/631

[33] Sabine Wieluch and Friedhelm Schwenker. 2019. Dropout induced noise for co-creative gan systems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 0–0.

[34] Kaiqiang Xu, Xinchen Wan, Hao Wang, Zhenghang Ren, Xudong Liao, Decang Sun, Chaoliang Zeng, and Kai Chen. 2021. TACC: A Full-stack Cloud Computing Infrastructure for Machine Learning Tasks.

[35] Hongfei Xue, Wenjun Jiang, Chenglin Miao, Ye Yuan, Fenglong Ma, Xin Ma, Yijiang Wang, Shuochao Yao, Wenyao Xu, Aidong Zhang, and Lu Su. 2019. DeepFusion: A Deep Learning Framework for the Fusion of Heterogeneous Sensory Data. In *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing* (Catania, Italy) *(Mobihoc '19)*. Association for Computing Machinery, New York, NY, USA, 151–160. https://doi.org/10.1145/3323679.3326513

[36] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 351–360. https://doi.org/10.1145/3038912.3052577

[37] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing Data Imputation using Generative Adversarial Nets. arXiv:1806.02920 [cs.LG]

[38] Piero Zappi, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Troster. 2007. Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness. In *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. 281–286. https://doi.org/10.1109/ISSNIP.2007.4496857

[39] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. arXiv:1710.09412 [cs.LG]

[40] Mingmin Zhao, Shichao Yue, Dina Katabi, Tommi S Jaakkola, and Matt T Bianchi. 2017. Learning sleep stages from radio signals: A conditional adversarial architecture. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 4100–4109.

[41] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. 2020. Domain Generalization via Entropy Regularization. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/b98249b38337c5088bbc660d8f872d6a-Abstract.html