Adapting to Unknown Conditions in Learning-based Mobile Sensing

Taesik Gong, Yeonsu Kim, Ryuhaerang Choi, Jinwoo Shin, and Sung-Ju Lee

Abstract—Many applications utilize sensors on mobile devices and apply deep learning for diverse applications. However, they have rarely enjoyed mainstream adoption due to many different *individual conditions* users encounter. Individual conditions are characterized by users' unique behaviors and different devices they carry, which collectively make sensor inputs different. It is impractical to train countless individual conditions beforehand and we thus argue meta-learning is a great approach in solving this problem. We present *MetaSense* that leverages "seen" conditions in training data to adapt to an "unseen" condition (i.e., the target user). Specifically, we design a meta-learning framework that learns "how to adapt" to the target via iterative training sessions of adaptation. MetaSense requires very few training examples from the target (e.g., one or two) and thus requires minimal user effort. In addition, we propose a *similar condition detector* (SCD) that identifies when the unseen condition has similar characteristics to seen conditions and leverages this hint to further improve the accuracy. Our evaluation with 10 different datasets shows that MetaSense improves the accuracy of state-of-the-art transfer learning and meta learning methods by 15% and 11%, respectively. Furthermore, our SCD achieves additional accuracy improvement (e.g., 15% for human activity recognition).

Index Terms—Mobile computing, mobile sensing, machine learning, meta learning, few-shot learning

1 Introduction

With the utilization and coupling between deep learning and various sensors in mobile devices, mobile sensing applications are services have become boundless. Recent mobile sensing applications include human activity recognition [1], [2], [3], [4], acoustic context recognition [5], [6], device-free authentication [7], [8], sign language recognition [9], emotional status recognition [10], [11], and even Parkinson's disease detection [12]. These mobile sensing applications show the potential of benefits from context-based services, enabled by a single device, e.g., a smartphone.

Although these mobile sensing applications have great potential for enriching our daily lives, most fail to enjoy mainstream adoption and remain as merely research prototypes due to the critical challenge when deployed to real users: performance degradation caused by different *individual conditions* users have. We refer to an *individual condition* as a combination of all user-specific dependencies that affect sensor readings (e.g., the user's behavior pattern and the specific device the user has). For example, in human activity recognition with smartphone motion sensors, users have different characteristics in their behaviors for the same activity; someone walks slowly while others fast with their own walk stride. In addition, their mobile devices have unique specifications such as weight, shape, sensor sampling rates, errors and biases, to name a few. These dependencies, as they make sensor readings dissimilar, significantly degrade the performance of mobile sensing [13], [14], [15]. Considering the possible number of different user behaviors and devices, and even the numerous combinations between them, it is an important research question to overcome individual conditions for mobile sensing to be practically available for wider deployment.

As deep learning performs well under trained conditions, a naïve approach to solve this problem is to train a model on all possible individual conditions beforehand. However, in order to train numerous parameters without overfitting, it often requires more than thousands of training instances [16], [17], [18]. This approach is infeasible as it requires the tedious and costly work of data collection and labeling process of every user. Sensor calibration or extracting condition-independent features have been considered as an alternative [14], [19], [20], [21], [22], [23], [24], [25], [26]. For example, one can use the squared sum of the each x, y, and z axis of accelerometer values to make it an orientation-independent feature with motions sensors [19], [23]. This approach, however, is limited to a specific sensor type (e.g., accelerometers), a specific dependency (e.g., phone orientation), or a specific application (e.g., human activity recognition). As it requires a new tailored method when the sensor type, dependency, and application vary, this cannot be a general solution for diverse types of mobile sensing applications. We are motivated by the aforementioned challenges and the limitations of existing approaches, and seek to answer the following important question: "How to overcome the individual condition problem in mobile sensing with minimal user efforts?"

We present *MetaSense*, a framework that is capable of adapting to unknown individual conditions with very few

- *Taesik Gong is with School of Computing, KAIST, Republic of Korea. E-mail: taesik.gong@kaist.ac.kr*
- *Yeonsu Kim is with School of Computing, KAIST, Republic of Korea. E-mail: yeonsu.kim@kaist.ac.kr*
- *Ryuhaerang Choi is with School of Computing, KAIST, Republic of Korea. E-mail: fkdfkd98@kaist.ac.kr*
- *Jinwoo Shin is with Graduate School of AI and School of Electrical Engineering, KAIST, Republic of Korea. E-mail: jinwoos@kaist.ac.kr*
- *Sung-Ju Lee is with School of Electrical Engineering, KAIST, Republic of Korea. E-mail: profsj@kaist.ac.kr*

Manuscript received April 19, 2005; revised August 26, 2015.

training examples (e.g., one or two) from the target user's condition, i.e., *few-shot adaptation*. To handle many individual conditions, we design a *meta learning* framework for mobile sensing. Specifically, from available (seen) training dataset, MetaSense first generates multiple episodic *tasks* via our unique task generation strategies. Each task is specially designed to teach the sensing model how to adapt to new conditions. With these tasks, MetaSense trains the model in a way that its parameters are adaptive to condition changes. Once the model is trained, it has the ability to adapt its parameters to the target users' condition given only few training examples from the target before using the model. MetaSense has several advantages over exiting solutions: (i) While achieving high accuracy with adaptation, MetaSense significantly reduces users' burden of data collection as it requires only few training examples just once before using the model. (ii) MetaSense requires less adaptation time than conventional training of deep neural networks and thus is suited for resource-constrained mobile devices. (iii) MetaSense can be applied to any deep learning models, any type of sensors, or any type of applications, and thus it is a general model-agnostic and condition-agnostic solution.

Existing meta learning algorithms for few-shot learning [27], [28], [29], [30] train the models with a large corpus of image data. In mobile sensing however, there are limited number of available datasets and aforementioned numerous individual conditions; therefore, meta learning in mobile sensing requires a different and sophisticated training method compared to existing methods. The unique contribution of MetaSense, beyond adopting meta learning to mobile sensing, is that we design our unique task generation strategies to maximally utilize the limited data for training mobile sensing models. We believe MetaSense is the first realization of meta learning into practical mobile sensing and thus bridges the gap between research and practice.

While our first version of MetaSense [31] has shown good accuracy in mobile sensing when deployed in untrained individual conditions, here, we take a step further and investigate the situation where certain conditions in the source dataset are similar to a target condition. Different mobile sensing applications have their unique sensing characteristics and thus a different level of heterogeneity among conditions. Some sensing applications have a less degree of heterogeneity among conditions. For example, ambient scene detection [32] and song identification [33] do not depend on users' behaviors. When certain conditions in the source are very similar to the target condition, this can be an opportunity for MetaSense to improve the performance for the target condition. To that end, we measure the impact of utilizing similar conditions and propose *similar condition detector* that identifies when similar conditions exist in the source and leverages the hint to further enhance the accuracy (§4).

We evaluate MetaSense with two datasets collected in the wild: (i) human activity recognition via motions sensors, and (ii) speech recognition via microphones, which are collected under realistic settings considering different users' behaviors, device models and types, sensor positions and orientations, and so on. We also evaluate MetaSense with eight different public datasets, including activity detection, stress detection, and vision datasets to understand the performance and the generalizability of our approach for other domains. Our evaluation of MetaSense with six baselines including the state-of-the-art transfer learning [34] and meta learning algorithms [27], [30] indicates that MetaSense not only outperforms existing solutions in terms of accuracy but also requires significantly less adaptation time. In particular, MetaSense outperforms the accuracy of transfer learning by 15% and meta learning by 11% on average, thanks to our unique task generation strategies. Moreover, our *similar condition detector* further improves the performance especially when only very few data is given (e.g., 15% additional improvement for the activity recognition dataset).

We summarize our key contributions as follows: (i) We present *MetaSense*, a meta-learning based adaptation framework for deep mobile sensing. To the best of our knowledge, MetaSense is the first attempt to adopt meta learning for the individual condition problem in deep mobile sensing. (ii) We propose three *task generation strategies* to address limited available data, which are keys to being effective in mobile sensing. (iii) We propose *similar condition detector* that determines whether similar conditions to the target condition exist in the source dataset and utilizes the hint for improving the performance.

2 Background and Motivation

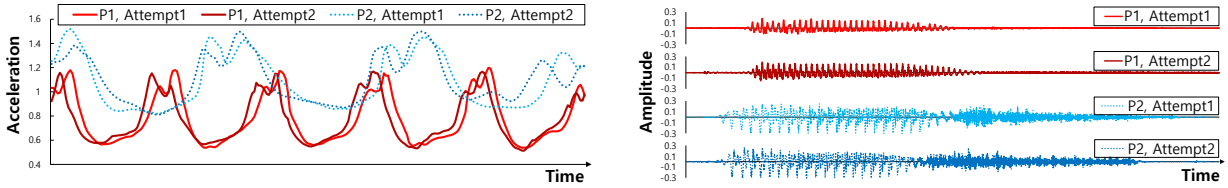
We illustrate why models should adapt to individual conditions in mobile sensing. We investigate what are the factors that degrade in-the-wild performance of mobile sensing applications and demonstrate the problem through two case studies; human activity recognition and speech recognition.

2.1 Why Conditions Matter

While recent studies have shown the potential of a variety of mobile sensing applications powered by deep learning [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], they must overcome the challenge of diverse *individual conditions* for wider adoption. Mobile sensing applications get input from the sensors in smart devices for their services, e.g., Inertial Measurement Unit (IMU) for motions and microphone for audio. The sensed values, however, are highly dependent on various conditions. We summarize the following two major categories where the individual conditions come from.

User dependency: Humans have different physical conditions and behaviors that make them unique between each other. In human activity recognition (HAR) for instance, users have dissimilar patterns of "walking" in term of the speed and stride, which could be confused with someone's "running". In addition, some people prefer to put their phone in their pocket, while others hold in hand, and each smartphone position makes different sensor readings even with the same device. Since users' behaviors are unbounded and cannot be easily characterized in advance, user dependency is one of the major obstacles for mobile sensing to overcome.

Device dependency: Users have their own devices that have a different shape, weight, sensor specification, and so on, which make the model get different sensor values. Especially for IMU sensors, different devices have different



(a) Magnitude of acceleration from the activity "jumping". (b) Audio waves from the word "yes".

Fig. 1: Comparison of raw signals between and within users P1 and P2. Attempt 1 and 2 are specified for each user.

sensor biases, errors, and sampling rates [15]. In addition, software heterogeneity (e.g., different versions of OS) makes sensor readings different [35]. With the recent spread of wearable devices, some users might run the sensing application in their wearable devices instead of smartphones. As the number of unique Android devices has already exceeded over 24,000 in 2015 [36], it seems infeasible to collect data from all possible devices in advance to train and make the model work effectively for every device.

Previous studies have shown that the user and device dependencies degrade the performance of mobile sensing [13], [15]. While there have been attempts to resolve the dependencies, most focus on isolated dependencies, e.g., user dependency [19], [37], [38], [39], [40], device position/orientation [20], [21], [41], and hardware/software heterogeneity [26], [35]. However, mobile sensing when deployed in practice, typically faces all of the dependencies.

2.2 Case Study: Activity & Speech Recognition

To understand how the individual condition affects deep mobile sensing performance, we collected two datasets, i.e., activity and speech recognition with ten different users (P1–P10) and devices (seven smartphones and three smartwatches). Activity recognition has nine activities and speech recognition has 14 keywords. Note that there are no duplicate devices or users, and data collection is performed without specific restrictions to allow and encourage different behaviors of users. The resulting dataset contains ten individual conditions from ten users. The details of the dataset and preprocessing are described in §5.1.1.

Figure 1 compares the raw signal within and between users P1 and P2. Figure 1a illustrates the square root of the squared sum of x, y, and z-axis accelerations for the "jumping" activity of users P1 and P2. We specify two different instances of the jumping activity as Attempt1 and Attempt2 in order to compare within-condition variability to cross-conditions variability. Similarly, Figure 1b shows the raw audio waves from the keyword "yes". The top two graphs show two different instances of P1 while the bottom two graphs show those of P2. As shown in both figures, while two different attempts from the same user appear similar, different conditions make significantly different sensor readings even for the same class (i.e., "jumping" and "yes"). This result clearly suggests that a model trained on some conditions could perform poorly when faced with a new condition (§5.2).

3 Meta-Learned Adaptation

We present MetaSense by starting with an overview and the meta learning scheme MetaSense uses (§3.1). We then detail our task generation algorithms to catalyze the effect of meta learning (§3.2), followed by the parameter update algorithm that makes the model adaptive to untrained conditions (§3.3). With the generated model, we explain how MetaSense can adapt rapidly to a new/unseen user with a few labeled data (§3.4).

3.1 Overview: Meta-Learned Adaptation

We consider a practical scenario where a model developer has a *source dataset* collected under several *individual conditions*, e.g., activity recognition data from multiple users measured with their own devices. Under the scenario, the goal of MetaSense is to adapt to a new/unseen user's condition when only a few target user's data samples are available. We denote a labeled data instance for each class as a *shot*. Note that we assume a few shots (e.g., one or two) are given from the target user and the original source dataset does not contain any data samples from the target user. Namely, a model developer first trains a base model with the source dataset, and the model further adapts to the target user's conditions using very small gradient steps of a few labeled data samples.

To handle adaptation with only a few shots, we design a meta learning framework, also known as learning to learn, to train the model. Meta learning [27], [28], [29], [30] generally aims to learn a new task or environment rapidly, by learning how to learn. As an analogy, one can easily learn how to ride an ice skate if she already knows how to ride a roller skate. Our intuition behind using meta learning is that deep mobile sensing systems could be deployed to numerous unknown individual conditions, which could be resolved by learning how to adapt to unknown conditions. Hence, the meta-objective of MetaSense is learning effective parameters that has an ability to adapt to an unseen condition.

Figure 2 shows an overview of MetaSense. MetaSense trains the deep sensing model through two steps, i.e., *base-model training* and *adaptation*. Specifically, MetaSense makes the base model learn how to adapt to a new condition with only a few shots and gradient steps. The base model is trained on a set of *tasks*, where each task is generated from the source dataset. Individual task mimics a situation where the model performs under a new untrained condition. After training, the base model has the knowledge of how to adapt to a new condition with a few shots. In the adaptation step, a target user provides a few shots to the model and the model adapts its parameters with a small number of gradient steps (e.g., 10). When the *similar condition detector* determines if there are similar source conditions that would help the adaptation, MetaSense fetches extra shots from

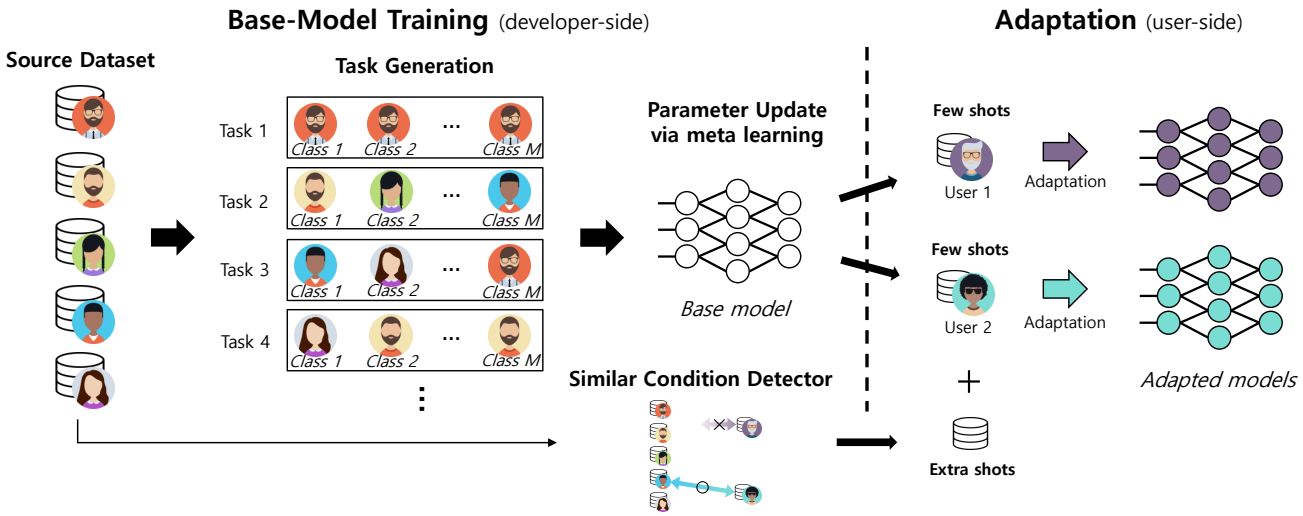


Fig. 2: MetaSense overview.

the condition, which we detail in §4. After the adaptation process, the model is ready for the target user's conditions.

Algorithm 1 MetaSense Base-Model Training

Input: Source dataset $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$, learning rate hyperparameters α, β
Output: Trained parameters θ

1: $\theta \leftarrow$ random initialization
2: **while** not finished **do**
3: $\quad \mathcal{T} \leftarrow \text{GENERATETASK}(\mathcal{D})$ ▷ Details in §3.2
4: $\quad$ **for** $\mathcal{T}_i \in \mathcal{T}$ **do**
5: $\quad\quad S_{\mathcal{T}_i} \leftarrow K$ support samples from $\mathcal{T}_i$
6: $\quad\quad Q_{\mathcal{T}_i} \leftarrow K$ query samples from $\mathcal{T}_i$ where $S_{\mathcal{T}_i} \cap Q_{\mathcal{T}_i} = \emptyset$
7: $\quad\quad$ Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with $S_{\mathcal{T}_i}$ via Equation (2)
8: $\quad\quad \theta'_{\mathcal{T}_i} = \theta - \alpha \cdot \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ ▷ Get $\mathcal{T}_i$-specific parameters
9: $\quad\quad$ Evaluate $\mathcal{L}_{\mathcal{T}_i}(f_{\theta'_{\mathcal{T}_i}})$ with $Q_{\mathcal{T}_i}$
10: $\quad g_\theta = \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_{\mathcal{T}_i}})$
11: $\quad \theta \leftarrow \theta - \beta \cdot \text{ADAM}(g_\theta)$ ▷ Update θ

Algorithm 1 outlines our base-model training method, where we provide further details in §3.2 and §3.3. We refer to $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ as the source dataset, $\boldsymbol{x}_i$ is an input vector, and $\boldsymbol{y}_i$ is a one-hot label vector where $\boldsymbol{y}_i \in \mathcal{Y}$, $\mathcal{Y}$ is the set of all labels, and $|\mathcal{Y}| = M$. The source dataset has been collected from multiple conditions. Base-model training algorithm outputs trained parameters θ for a deep sensing model $f_\theta(\boldsymbol{x})$. With $\mathcal{D}$, MetaSense generates a set of tasks $\mathcal{T}$ that are designed to boost the effect of meta-learning objective, e.g., learning to adapt to the target condition (line 3). We explain the details of task generation in §3.2. From each generated $\mathcal{T}_i \in \mathcal{T}$, MetaSense samples a *support set* and a *query set* without overlap between them (line 5–6). For each task, MetaSense computes temporal parameters $\theta'_{\mathcal{T}_i}$ via stochastic gradient descent (SGD) with the support set and evaluates the loss function $\mathcal{L}_{\mathcal{T}_i}$ with the temporal parameters and the query set (line 7–9). The final parameters are updated by a meta objective via Adam optimizer [42], which minimizes the sum of each task-specific loss (line 11). We detail this parameter update process in §3.3. Through this meta learning process, the trained parameters learn an effective way to adapt to the unseen task, i.e., the target condition.

3.2 Task Generation

Unlike existing meta learning methods where tasks are randomly generated by sampling from a large available dataset, how to efficiently and effectively leverage the limited source dataset is the unique challenge in applying meta learning to mobile sensing. We view each task as each individual condition in mobile sensing. Thus, the goal of our task generation is to generate diverse and realistic individual conditions given the source dataset so that MetaSense can teach the base model how to adapt to possible condition changes via various tasks.

MetaSense generates a set of tasks $\mathcal{T}$ from $\mathcal{D}$. Each task $\mathcal{T}_i \in \mathcal{T}$ has a set of data samples, $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_{N_{\mathcal{T}_i}}, \boldsymbol{y}_{N_{\mathcal{T}_i}})\}$. We assume the model developer has a source dataset $\mathcal{D}$ from C individual conditions (e.g., a user wearing a smartwatch on the left wrist, a user placing a smartphone in a pocket, etc.). We let a subset of dataset $\mathcal{D}_c \subset \mathcal{D}$, an *individual condition dataset (ICD)* that represents a dataset from a specific condition such that

$$\bigcup_{c=1}^{C} \mathcal{D}_c = \mathcal{D} \text{ and } \mathcal{D}_c \cap \mathcal{D}_d = \emptyset, \tag{1}$$

where $\cup$ is the set of all elements in the collection, $c \neq d$, and $1 \leq c, d \leq C$. We devise three strategies of generating tasks to maximize the effect of meta learning for few-shot adaptation to individual conditions as follows.

Per-condition tasks: To mimic a situation where the base model meets an unseen target condition when deployed to the target user, we generate each per-condition task from a sampled data by each ICD. Specifically, we generate $\mathcal{T}_i$ where $\mathcal{T}_i \subset \mathcal{D}_{c=i}$ for all $1 \leq i \leq C$ as illustrated in Figure 3. With these tasks, the base model experiences adapting to *real* individual conditions. As the number of per-condition tasks is equal to the number of collected conditions C, the number of conditions the model developer has affects the performance of base model training. We found that the

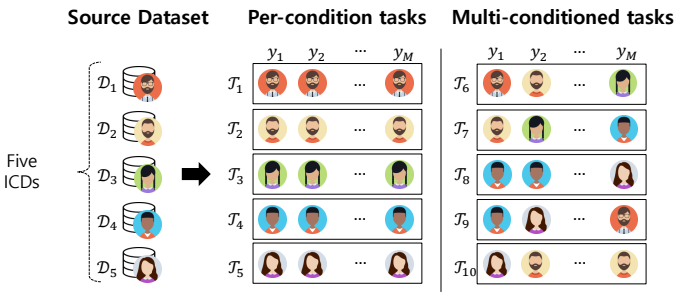


Fig. 3: An illustration of per-condition tasks and multi-conditioned tasks generated from five individual conditioned datasets (ICDs).

more conditions the base model is trained on, the better the performance. This is because the base model has more opportunities in advance to experience diverse conditions before being deployed to the target user.

Multi-conditioned tasks: We generate multi-conditioned tasks in addition to per-condition tasks. A multi-conditioned task is made of samples from *randomly* selected ICDs. Our intuition for generating multi-conditioned tasks is to intentionally provide the base model with artificial conditions so that it can be (i) trained on more diverse tasks beyond per-conditions tasks while (ii) avoiding possible overfitting to the per-conditions tasks. We generate C number of multi-conditioned tasks for each iteration of training in order to take advantage of the randomness for generalization. Specifically, for each class label $\boldsymbol{y}_j \in \mathcal{Y}$, we randomly choose an ICD $\mathcal{D}_c$ and sample data from the dataset for the label as illustrated in Figure 3. That means for all $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ in a multi-conditioned task if $\boldsymbol{y}_i = \boldsymbol{y}_j$ then $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}_c$. Each multi-conditioned task is thus generated from at most $|\mathcal{Y}| = M$ ICDs. This way, we can generate more realistic tasks compared to entirely random sampling from $\mathcal{D}$, as it keeps input distribution within each class.

Homogeneous task generation: We generate the above tasks with keeping labels consistent across tasks. This is contrary to existing meta learning approaches [27], [28], [29], [30] where the labels for each task are mixed randomly, i.e., $\mathcal{Y}_{\mathcal{T}_i} \neq \mathcal{Y}_{\mathcal{T}_j}$ where $\mathcal{T}_i \neq \mathcal{T}_j$. Since the objective of most meta learning studies in machine learning is focused on adapting to arbitrary tasks that might not have the same labels, this label mixing strategy would be natural for them. On the other hand, in our problem, the source and the target datasets have different distribution but have the same label space, i.e., $\mathcal{Y}_{\mathcal{T}_i} = \mathcal{Y}_{\mathcal{T}_j}$ where $\mathcal{T}_i \neq \mathcal{T}_j$. We found keeping labels consistent is effective as it leverages the common knowledge on the same label set $\mathcal{Y}$ across tasks.

Since the source dataset is collected from multiple conditions, it is common to have imbalanced numbers of data instances among $\mathcal{D}_c$. When generating per-condition and multi-conditioned tasks, we sample a batch of data uniformly across conditions (i.e., giving the same weight to all conditions and accordingly the generated tasks) in order to avoid being biased to some conditions that have a higher number of samples than others.

The generated task set $\mathcal{T}$ via the above three strategies has $2C$ tasks. The base model iterates each task to update the parameters as explained next. We evaluate the effectiveness of our unique task generation strategies in §5.3.

3.3 Parameter Updates

With the generated tasks in §3.2, we train the parameters of the base model via meta learning. Specifically, MetaSense employs model-agnostic meta learning (MAML) [27] for updating the parameters. MAML is applicable to any deep neural networks that use gradient descent (model-agnostic) and requires only a few gradient steps to update the model. The assumption of MAML is that there exist initial parameters that are transferable to a new task with only a few shots. MAML trains the initial parameters in a way that the trained parameters are adaptive to change of tasks. Our intuition behind adopting MAML is that for deep sensing models there exist effective initial parameters that are transferable between individual conditions, so that the parameters can be adapted to the target condition within a few gradient steps.

From each task $\mathcal{T}_i$, MetaSense samples a *support set* $S_{\mathcal{T}_i}$ and a *query set* $Q_{\mathcal{T}_i}$ that have K shots, respectively. K should be a small number (e.g., 5) to simulate a few shots from a target user. Support sets are used for training the task-specific parameters $\theta'_{\mathcal{T}_i}$, which simulates adapting parameters to a target condition. Query sets are used for evaluating the task-specific parameters and eventually updating the parameters θ of our interest (i.e., the base model). We ensure support sets and query sets have no overlapping data. We target a multi-class classification problem and use cross-entropy loss to evaluate the per-task loss, i.e.,

$$\mathcal{L}_{\mathcal{T}_i}(f_\theta) = \sum_{(\boldsymbol{x}_j, \boldsymbol{y}_j) \in S_{\mathcal{T}_i}} \boldsymbol{y}_j \log f_\theta(\boldsymbol{x}_j) + (1 - \boldsymbol{y}_j) \log f_\theta(1 - \boldsymbol{x}_j). \tag{2}$$

We then get $\mathcal{T}_i$-specific parameters $\theta'_{\mathcal{T}_i}$ with a few gradient descent steps (e.g., 5 steps):

$$\theta'_{\mathcal{T}_i} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta), \tag{3}$$

where the task learning rate α is a hyperparameter: it is usually set as a higher number (e.g., 0.1) than traditional learning rate to enforce fast adaptation [27].

With the task-specific parameters, we define a meta-objective function as follows:

$$\arg\min_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_{\mathcal{T}_i}}) \quad \text{where } \theta'_{\mathcal{T}_i} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta). \tag{4}$$

The meta-objective is finding parameters θ that minimize the sum of task losses. Note that each task loss $\mathcal{L}_{\mathcal{T}_i}$ is evaluated by the task-specific parameters $\theta'_{\mathcal{T}_i}$ that are calculated by a few gradient descent steps with a query set that has a few shots (Eq. (3)). This enforces θ to be sensitive to task changes so that it becomes effective within a few gradient steps. Note that the tasks generated by MetaSense reflect individual conditions. The meta-objective is thus interpreted as minimizing the sum of task-specific losses, so that the optimal parameters of θ become an *effective initialization* of the model such that with the parameters the model can rapidly adapt to a new condition after several gradient steps with a few shots.

The last step is updating the parameters θ by minimizing the meta objective with Adam optimizer [42]:

$$\theta \leftarrow \theta - \beta \cdot \text{ADAM}(g_\theta) \quad \text{where } g_\theta = \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_{\mathcal{T}_i}}), \quad (5)$$

and the meta learning rate β is a hyperparameter. Note that the trained base model f_θ has initial parameters $\theta_0 = \theta$ that are experienced through meta learning with multiple tasks that simulate encountering unseen conditions in the real world. The base model is now prepared for adaptation with a few shots from a target user.

3.4 Adaptation

After the base-model training is performed by the developer, it could be deployed to real users. The base model is adapted for the target user with a few shots (e.g., 1 or 2 samples per class) once at the beginning of the sensing application. We denote $\mathcal{U} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_L, \boldsymbol{y}_L)\}$ as the target user's dataset that has $\frac{L}{M}$ shots. In the adaptation, we assume the target user's dataset has no identical conditions to the source dataset that the base model is trained with, i.e., $\mathcal{D} \cap \mathcal{U} = \emptyset$.

Let the base model be f_{θ_0}, where θ_0 is the initial parameters that are trained through meta learning. After i gradient descent steps with the few shots, the parameters become:

$$\theta_i = \theta_{i-1} - \alpha \nabla_\theta \mathcal{L}_{\mathcal{U}}(f_{\theta_{i-1}}). \quad (6)$$

Note that since the base model experienced a set of tasks through Eq. (2)–(5), the trained parameters can effectively adapt to the target condition with the meta-learned knowledge. While requiring only a few shots from the target user, another advantage of this parameter update algorithm is that it takes only a few gradients steps, which significantly reduces the training time on the resource-constraint mobile devices. We evaluate the time taken to adapt the model compared to other deep neural networks baselines in §5.4.

4 Similar Condition Detector

For cases where certain conditions in the source dataset are similar to the target condition, we aim to improve the classification accuracy by reusing similar source samples as additional shots for the adaptation step. To this end, we design the *similar condition detector* (SCD) that identifies whether the source dataset includes a similar condition to the target and fetches additional shots that help adapt to the target.

4.1 Motivation

Considering numerous combinations of users and devices, MetaSense learns how to adapt to a target condition via meta learning, instead of only learning directly from the available dataset. The assumption that a target condition is different from source conditions, however, might not hold in some cases. There are situations where the source dataset includes a condition that is (highly) similar to the target condition. For instance, some applications do not involve user dependency, such as ambient scene detection [32], song identification [33], earthquake detection [43], etc. As they depend only on devices, there is a good chance of having duplicate conditions between source and target conditions.

TABLE 1: Accuracy comparison between traditional DNNs (Src+Tgt) and MetaSense with and without similar conditions.

	W/o similar cond.		W/ similar cond.	
	Src+Tgt	MetaSense	Src+Tgt	MetaSense
Activity	30.26	**67.28**	**86.20**	74.76
Speech	48.57	**63.93**	63.92	**68.21**

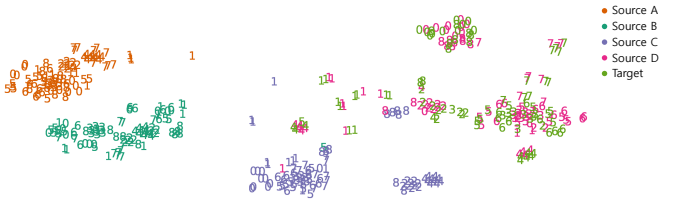


Fig. 4: T-SNE visualization of source conditions and the target condition under the with-similar-condition case.

On the contrary, some applications do not entail device dependency, such as app recommendation by users' usage pattern [44]. Combined with an increasing number of available source data (open datasets) and crowdsourcing platforms [45], [46], MetaSense should utilize cases when source and target conditions could be similar.

For cases where the target condition is included in the source conditions, the meta learning objective of MetaSense might not perform better than a traditional learning objective via a standard supervised learning scheme. Table 1 shows an example of this situation for the activity and speech recognition data, given 1-shot from the target condition (details are in §5). Src+Tgt is a traditional supervised learning trained with both the source and the target data. As shown, MetaSense outperforms Src+Tgt in the "without-similar" condition, demonstrating the ability of MetaSense to adapt to the target condition with very few training examples. However, in the "with-similar" case, we observe that MetaSense often perform worse than Src+Tgt. This is because MetaSense fundamentally regards conditions as independent tasks and does not assume condition similarity between the source and the target. Accordingly, the adaptation step depends solely on the few shots given from the target. Therefore, it is essential for MetaSense to manage such scenarios to further enhance its performance across various mobile sensing applications.

When there exists a level of similarity between source and target conditions, our goal for the performance of MetaSense is to be as comparable as the traditional supervised learning. To this end, we propose *similar condition detector* (SCD) to identify whether similar conditions to target condition exist in source conditions, and utilize them as additional shots for the adaptation step.

4.2 Design of the Similar Condition Detector

Recent studies have demonstrated that the learned representations through backpropagation in neural networks could be utilized as distinctive features among different distributions of data [29], [30], [47]. Our key intuition of designing the similar condition detector (SCD) is that if a certain condition in the source dataset is similar to the target condition, the learned representations (for simplicity, we refer it as *features*) of the source condition have a closer

distance to the target condition, compared with the other source conditions.

Figure 4 shows T-SNE [48] visualization when a similar condition to the target is included in the source.[1] The numbers refer to class labels (0-8) of our activity recognition dataset (§2.2). We sampled Source D and Target from the same user and device, while Source A, B, and D are from different combinations. It clearly illustrates the proximity of samples between similar conditions.

The main goal of SCD is to determine a criteria with which the detector decides whether there is a source condition that is similar the target condition. A naïve approach would be to simply select a source condition that has the closest distance to the target and fetch additional shots from the condition. However, choosing the condition with the closest distance does not necessarily improve the accuracy, as the closest distance still might not be close enough. Therefore, a desirable SCD should only select source conditions that would improve the accuracy. We hence follow a statistical approach to detect similar conditions, which is further divided into building distance profiles (§4.2.1), detecting similar conditions (§4.2.2), and fetching additional shots (§4.2.3).

4.2.1 Distance Profile

From the source dataset, we generate a *distance profile* that is composed of the distribution of intra/inter-condition feature distance. A distance profile represents the feature distance across conditions in the source dataset, which is later utilized to detect matched conditions for the target condition. When the training of the base model is complete, a distance profile is generated with the base model before the adaptation to the target.

For each condition c in a source dataset, we define a *prototype* $\mathcal{M}_c$ as a set of mean feature vectors per each class:

$$\mathcal{M}_c = \{\boldsymbol{m}_1, \boldsymbol{m}_2, ..., \boldsymbol{m}_{|\mathcal{Y}|}\}, \tag{7}$$

where $1 \leq c \leq C$ and $\boldsymbol{m}_i = \frac{1}{|K|}\sum_{(\boldsymbol{x}_k, \boldsymbol{y}_k)\in\mathcal{D}_c} \mathcal{F}(x_k)$. Here, $\boldsymbol{y}_k = \texttt{OneHot}(i)$, K is the number of samples in $\mathcal{D}_c$ satisfying $\boldsymbol{y}_k = \texttt{OneHot}(i)$, $\mathcal{F}(\cdot)$ is the learned features, i.e., the intermediate output of the model before fully-connected layers, and $\texttt{OneHot}(\cdot)$ is one-hot representation.

We define the distance between a shot s and a prototype $\mathcal{M}_c$ for condition c as follows:

$$\texttt{Dist}(s, \mathcal{M}_c) = \frac{1}{|\mathcal{Y}|}\sum_{l=1}^{|\mathcal{Y}|} \left\| \mathcal{F}(x_s^l) - \boldsymbol{m}_l \right\|_2, \tag{8}$$

such that $(\boldsymbol{x}_s^l, \boldsymbol{y}_s^l) \in s$ and $\boldsymbol{y}_s^l = \texttt{OneHot}(l)$. This distance is the average of class-wise distances between a shot and a prototype. We use class-wise distance because even within a condition, each class has different learned representation, as observed in Figure 4.

With the distance metric, we create a distance profile from the distributions of the distances between all the shots and the prototype pairs. Specifically, we create a distance profile, P, a C by C matrix of which element is the mean and the standard deviation of the distance distribution as follows:

$$P_{c_i,c_j} = \{\texttt{Mean}(\texttt{Dist}_{s\in S_{c_i}}(s, \mathcal{M}_{c_j})), \texttt{Stdev}(\texttt{Dist}_{s\in S_{c_i}}(s_{c_i}, \mathcal{M}_{c_j}))\}, \tag{9}$$

where P_{c_i,c_j} is the c_i^{th}-row and c_j^{th}-column element of P, S_{c_i} refers to all shots in $\mathcal{D}_{c_i}$ and $1 \leq c_i, c_j \leq C$.

S_c: Shots in condition c $\mathcal{M}_c$: Prototype of condition c

	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_6$	$\mathcal{M}_7$	$\mathcal{M}_8$	$\mathcal{M}_9$
S_0	136, 17	303, 17	290, 17	290, 17	242, 17	252, 17	319, 17	239, 16	312, 16	304, 16
S_1	275, 2	89, 5	212, 3	234, 3	267, 2	229, 3	210, 3	266, 2	251, 3	254, 2
S_2	281, 6	236, 10	136, 12	216, 9	252, 8	218, 9	201, 10	271, 8	236, 10	229, 8
S_3	272, 4	247, 5	205, 5	115, 7	229, 5	211, 6	243, 5	256, 4	263, 7	230, 4
S_4	224, 9	281, 7	244, 7	231, 7	121, 10	208, 9	287, 6	227, 8	312, 6	254, 6
S_5	243, 17	252, 19	218, 19	220, 19	215, 18	115, 19	265, 19	228, 18	286, 17	222, 17
S_6	309, 9	232, 13	199, 14	251, 12	293, 11	262, 12	130, 15	305, 10	250, 11	266, 11
S_7	241, 14	299, 12	283, 13	278, 11	248, 12	243, 14	318, 12	153, 16	326, 10	279, 12
S_8	285, 3	254, 4	214, 5	254, 5	301, 3	265, 4	230, 5	298, 4	88, 7	275, 4
S_9	300, 8	280, 10	235, 10	245, 11	266, 10	227, 13	272, 9	271, 11	299, 7	142, 11

----- similar condition

	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_6$	$\mathcal{M}_7$	$\mathcal{M}_8$	$\mathcal{M}_9$
$S_{\texttt{tgt}}$	297	277	234	241	259	223	273	265	299	150

Fig. 5: An example of distance profile and comparison with the distance to the target condition.

The upper matrix in Figure 5 illustrates an example of distance profile for our activity recognition dataset (§5.1.1). The distance profile has the distance distribution (mean and standard deviation) of all shots in the source with respect to the source conditions. Note that each condition has a distinctive feature distance relationship with other conditions and has the shortest distance with itself.

4.2.2 Detection Algorithm

Given the shots from the target condition, we calculate the distance between the target condition and the source conditions in the same way as in Equation (8). For detecting a similar condition, we leverage the observation that when a source condition has a similar distribution to the target, the distance patterns of the samples in that condition with respect to the prototypes are comparable to that of the target condition (Figure 5).

We determine a condition c_i as a similar condition when (i) the index of shortest distance is the same as that of $S_{\texttt{tgt}}$ and (ii) it satisfies the following equation for all c_j:

$$\left| \texttt{Mean}(\texttt{Dist}_{s\in S_{\texttt{tgt}}}(s, \mathcal{M}_{c_j})) - \texttt{Mean}_{c_i,c_j} \right| \leq \gamma \cdot \texttt{Stdev}_{c_i,c_j}, \tag{10}$$

where $P_{c_i,c_j} = \{\texttt{Mean}_{c_i,c_j}, \texttt{Stdev}_{c_i,c_j}\}$, $S_{\texttt{tgt}}$ is the collection of the target shots, and γ is a hyperparameter. Figure 5 shows that S_9 is detected as a similar condition to the target following the detection algorithm. There could be multiple choices for the hyperparameter γ and we chose 2.58, which covered 99% of the distribution and performed well in our experiment. To tune γ, one can use a hyperparameter searching algorithm such as Bayesian Optimization [49]. However, finding the optimal γ value without test data is infeasible. One possible way for finding an appropriate γ with only a source dataset is by selecting the best value through cross-validation within the source dataset. We think this is viable because we observed that the characteristics of a dataset remain the same within the dataset, even if the target condition changes. We leave it as future work.

1. T-SNE is a dimension reduction algorithm widely used when visualizing multi-dimensional data in a way that similar points gather closer than dissimilar ones.

TABLE 2: Settings for our data collection.

User	Device	Type	IMU rate	OS
P1	Samsung Galaxy J7	Phone	100Hz	7.0.0
P2	Google Nexus5	Phone	200Hz	6.0.1
P3	Essential Phone	Phone	400Hz	7.1.1
P4	Google Pixel2	Phone	400Hz	8.1.0
P5	HUAWEI P20	Phone	500Hz	8.1.0
P6	Samsung Galaxy S9	Phone	500Hz	8.0.0
P7	LG G5	Phone	200Hz	6.0.1
P8	LG Urbane	Watch	200Hz	Wear 2.23.0
P9	LG G Style	Watch	100Hz	Wear 2.6
P10	ASUS Zenwatch3	Watch	100Hz	Wear 2.23.0

4.2.3 Fetching Additional Shots

After detecting similar conditions, we use the available shots from the selected conditions and combine them with the target shots. We use the augmented shots for the adaption stage (Equation (6)).

5 Evaluation

We evaluate MetaSense to answer the following questions: (i) How well does MetaSense perform against existing approaches? (ii) How effective are MetaSense's task generation strategies? (iii) How rapidly can MetaSense adapt to the target? (iv) How well does MetaSense perform on different datasets? (v) What is the performance impact of our similar condition detector?

5.1 Settings

5.1.1 Data Collection

We detail the data collection and preprocessing of our datasets. The goal of our data collection was to evaluate MetaSense with real-world datasets collected under individual conditions. Specifically, we collected two most widely used types of sensors, IMU and audio. We recruited ten users (aged 21-29; mean 24.6, and three females) and conducted IRB-approved data collection experiments. Each user performed activity recognition (for IMU) and speech recognition (of audio) tasks. We randomly distributed ten different Android devices (seven smartphones and three smartwatches) for each user as listed in Table 2. We believe this dataset is the first dataset collected under individual conditions from two common sensors (IMU and audio).

Our activity recognition task was composed of nine activities that are commonly used in the literature [15], [50]. Specifically, they were "walking", "running", "stair down", "stair up", "lying", "standing", "stretching", "sitting", and "jumping". Participants performed each task for around 2–5 minutes with the duration varying based on the intensity of the activity. Note that we let the participants hold their device freely (e.g., in the pocket, on hand, or on wrist) for each activity to assure conditions are individual and natural. We did not give explicit guidelines to the activities, so that participants performed the activities according to their personal interpretation. We recorded each x, y, and z-axis of accelerometer and gyroscope values at the maximum sampling rate. We divide the data with 256-length window and use it to train the model.

The second task is speech recognition. We chose 14 words considering IoT applications [51]: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", "forward", "backward", "follow", and "learn" were used. Each participant held the device in their preferred fashion and uttered each word 30 times in an office room. We did not control their behaviors so that they had different individual conditions, such as speech loudness, speed, and the distance between the device and the user, etc. We recorded each utterance of a word for 2 seconds with 16 kHz sampling rate.

5.1.2 Baselines

We compare MetaSense to six baselines, which are widely used approaches for handling various untrained conditions: traditional DNNs, the state-of-the-art transfer learning and few-shot learning approaches. We not only aim to compare the performance of MetaSense to these baselines, but also inspect how these baselines perform under condition changes. Specifically, we have the following baselines.

Src: For Src (source only), we use only the source dataset for training the deep neural network and there is no adaptation to the target user. Src is a widely used method in resolving the diversity of inputs, i.e., training on as many data as possible that are collected from diverse conditions.

Tgt: Tgt (target only) trains the model with only the few shots from the target user.

Src+Tgt: Src+Tgt (source plus target) uses both the source dataset and the target users' few shots for training the deep neural network. Compared to Src, this baseline leverages the target user's data for adaptation while utilizing a relatively large amount of source data to learn general representations.

TrC: Transfer Convolutional (TrC) [34] is the state-of-the-art transfer learning in adapting to a target user's activity recognition with motion sensors with a few data samples. Specifically, TrC first trains the model with the source dataset. When TrC adapts the model, it freezes the CNN layers' parameter and fine-tunes only the following fully connected layers with a few shots.

PN: Prototypical Network (PN) [30] is one of the state-of-the-art few-shot learning algorithms based on meta learning. Given a few training data, PN generates prototypes in embedding space and each prototype is the representative of each class. In inference, PN uses the Euclidean distance metric to classify the closest prototype (i.e., class).

MAML: Another popular few-shot learning baseline is MAML [27], which is adopted in our parameter update stage §3.3. The performance difference between PN and MAML would indicate which method is more effective in deep mobile sensing, while the comparison between the original MAML and MetaSense would highlight the impact of our task generation strategies.

5.1.3 Implementation

To ensure a fair evaluation, we used the same model architecture and hyperparameters, e.g., learning rates, for all DNN-based baselines and MetaSense. We designed them with convolutional neural networks (CNN) followed by fully-connected layers. CNN is a widely used architecture not only in vision but also in activity and speech recognition with mobile sensors [34], [52], [53]. Specifically, the model architecture was composed of three to five convolutional layers, followed by three fully-connected layers. We used

Rectified Linear Unit (ReLU) for activation function. We used two regularization techniques, i.e., L_2-regularization and batch normalization to prevent overfitting. We trained the model with Adam optimizer [42]. We used five gradient descent steps for training the base model (Eq. (2)) with $K = 5$ and ten steps for adaptation (Eq. (6)). We implemented MetaSense using the PyTorch framework [54] and trained the model in a server equipped with eight NVIDIA TITAN Xp GPUs and 256 GB memory with Intel Xeon E5-2697 2.30 GHz processors.

5.2 Result

We trained the base model in a leave-one-user-out manner. Specifically, we used the others' data as the source dataset for each target user. Namely, there are ten evaluations in total, and for each user we have a source dataset with nine ICDs of the other users. We report the average accuracy for the untrained/target user among the ten scenarios. We focus on 1, 2, 5, and 10-shot cases that are frequently used in few-shot learning evaluations [27], [28], [30]. We used early stopping on the validation set and evaluate the accuracy on the test set.

Figure 6 reports the accuracy of the baselines and MetaSense for activity recognition (Figure 6a) and speech recognition (Figure 6b). The error bar is standard deviation (stdev for short) across users and thus high stdev indicates the method has high variance among users, i.e., low stdev suggests the method shows stable performance across users.

In general, as the number of shots increases, the accuracy also increases except for Src as Src does not use the target data. In most cases, Tgt performs better than Src, which means the learned representations from multiple other conditions would not generalize to a new condition. This again highlights the importance of adaptation for deep mobile sensing. Tgt, however, does not achieve higher performance than MetaSense, in particular when the number of data is small due to overfitting. In all cases, MetaSense outperforms the baselines, which shows the effectiveness of our approach when dealing with new unseen/target conditions. In activity recognition, MetaSense improves the accuracy of Src from 27.6% to 67.2% with only one shot, where the improvement is 15% higher compared to TrC. Furthermore, MetaSense outperforms the few-shot learning baselines thanks to our task generation strategies, which we dissect in §5.3.

Figure 7 illustrates the receiver operating characteristic (ROC) curves of the baselines and MetaSense for the activity and speech recognition datasets. We also specify the area under the curve (AUC) for each method, where AUC=0.5 means a random classifier while AUC=1 is a perfect classifier. Similarly, MetaSense shows its effectiveness over the baselines without depending on a single false-positive-rate/true-positive-rate threshold. We found similar patterns in other experiments, and we thus focus on the accuracy metric in the following experiments.

5.3 Effect of Task Generation

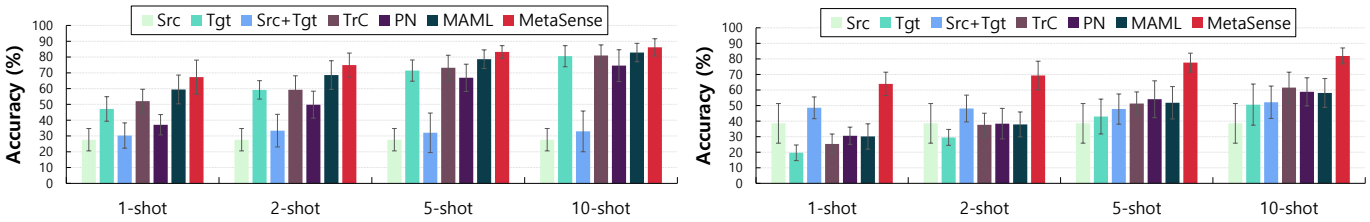
We now examine the effectiveness of our task generation methods described in §3.2. We evaluated the accuracy of MetaSense while gradually adding each of our task generation method. As a baseline, we used random task generation from the source dataset, which is widely used in recent meta learning approaches [27], [28], [29], [30]. We implement the random task generation as described in §3.2. More specifically, tasks are generated from the instances sampled randomly from the source data regardless of conditions. We then use the same number of random tasks as the per-conditioned tasks. We use the activity and speech recognition datasets and report the accuracy for 1, 2, 5, and 10-shot cases.

Figure 8 reports the accuracy gain of our task generation methods. Random refers to the random task generation. The accuracy improvement escalates as each of our task generation strategies is added. The result shows that our per-condition (Per) and multi-conditioned (Multi) tasks are effective than random sampling. This means those tasks can teach more plausible conditions to the base model than the randomly generated tasks. Furthermore, generating homogeneous tasks (Homo) helps to accumulate the common knowledge learned from the tasks that has the same label set $\mathcal{Y}$ that would improve the performance when faced with a target task that also has $\mathcal{Y}$. In summary, the results demonstrate the importance of task generation algorithms to teach the base model, and our task generation methods effectively utilize the given source dataset (18% gap on average, 33% in the extreme case compared to Random) so that they catalyze the efficacy of meta learning for resolving the condition problem.

5.4 Adaptation Overhead

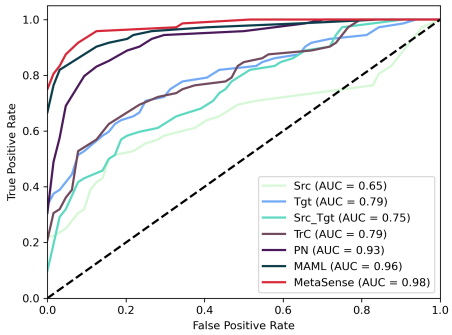
It is important to note that all the baselines and MetaSense requires different adaptation overhead. In this section, we demonstrate that MetaSense is also computationally efficient in the overhead of adaptation, i.e., the training time required to adapt to the target, which is crucial to ensure high quality mobile user experience. We investigate how many training epochs are required for each method to converge to its best performance (with respect to validation). In the experiments, we compare only Tgt, Src+Tgt, TrC, and MetaSense because Src and PN do not require the adaptation step while MAML has the same adaptation overhead as MetaSense. We report the accuracy averaged among 10 users in the 5-shot cases, where the overall trends for other shot cases are similar.

Figure 9 plots the accuracy changes for the target as training for adaptation proceeds. Note that while Tgt, TrC and MetaSense require only the target user's data for adaptation, Src+Tgt trains with the entire data composed of the source and the target datasets. Therefore, each epoch of Src+Tgt requires about ten times more time than others with our datasets. For Tgt, TrC, and MetaSense, the required time for one epoch is the same. Both activity recognition (Figure 9a) and speech recognition (Figure 9b) show that MetaSense entails significantly less adaptation overhead compared to other approaches while achieving the highest accuracy. TrC requires fewer epochs to converge compared to Tgt as TrC already learned the representations through the source dataset and fine-tunes its parameters to the target via transfer learning. MetaSense maximally leverages the

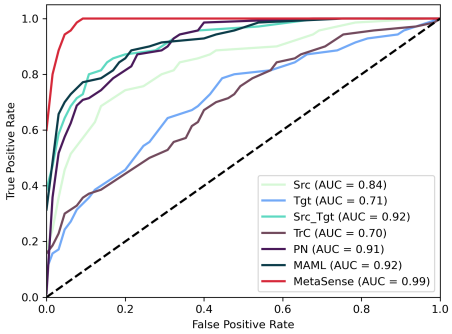


(a) Activity recognition. (b) Speech recognition.

Fig. 6: Average accuracy with 1, 2, 5, and 10-shots.



(a) Activity recognition.



(b) Speech recognition.

Fig. 7: ROC curves for the activity and speech recognition datasets given 5 shots from the target.

source data via meta learning so that it has the fastest convergence. As we use only ten gradient steps for adaptation as described in §5.1.3, MetaSense converges with only ten gradient steps. A different number of gradient steps could be used, e.g., more steps for achieving higher accuracy or fewer steps for minimizing the training overhead.

5.5 Other Datasets

We used additional four mobile sensing datasets to investigate MetaSense's generalizability to other sensing datasets. We also used four vision datasets to understand whether the method of MetaSense could translate onto another domain.

5.5.1 HHAR

Heterogeneity Human Activity Recognition (HHAR) dataset [15] was collected with nine users for six human activities. Each user was equipped with eight smartphones around the waist and four smartwatches in the arms and logged accelerometer and gyroscope values for each activity. This dataset has user and device-model dependency but does not include various device positions as each mobile device is located at specific positions. We used the 256-length window with 50% overlapping between two consecutive windows [15]. After eliminating duplicate device models and conditions with less than 10 shots, we have six users and four different devices that result in a total of 24 conditions. We evaluated each 24 conditions with 15 (5×3) ICDs, ensuring no overlap in either the target device or the user. We report the average accuracy of the 24 conditions.

5.5.2 DSA

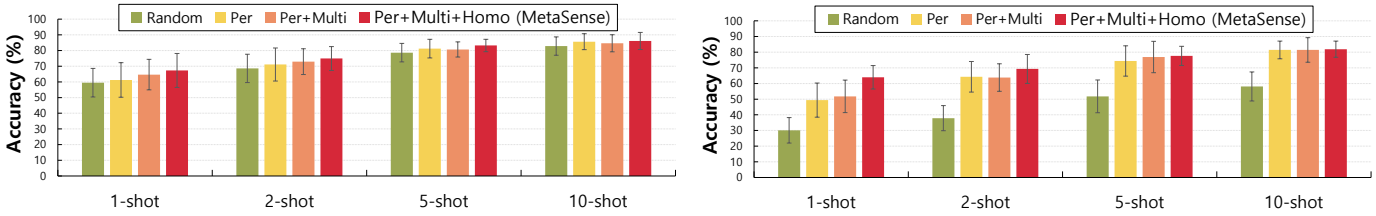
Daily and Sports Activities (DSA) dataset [50] was collected with eight users for 19 daily and sports activities. Each user was equipped with the same five sensor units, with each unit composed of an accelerometer, a gyroscope, and a magnetometer, on five different positions: torso, right arm, left arm, right leg, and left leg. This dataset therefore has user and sensor-position dependencies. We use the 125-length window [50]. There are a total of 40 conditions, and similar to HHAR, we evaluate each of 40 conditions with 28 (7×4) exclusive ICDs. We report the average accuracy of the 40 conditions.

5.5.3 WESAD

Wearable Stress and Affect Detection (WESAD) [55] dataset was collected with 15 subjects for stress and affect detection. Each user was equipped with the same wrist- and chest-worn devices that include the following sensing modalities: blood volume pulse, electrocardiogram (ECG), electrodermal activity (EDA), electromyogram (EMG), respiration, body temperature, and three-axis acceleration. Furthermore, three different affective states (neutral, stress, amusement) and self-reports of the subjects are included in the dataset. This dataset has user dependency but does not contain device-model and sensor-position dependencies as the same wearable devices were used. We used all sensor modalities for our evaluation. We followed the lowest sampling rate (4 Hz for EDA and temperature sensors) and down-sampled the other sensor values for the consistency of the model among datasets. We used the 8-length window for training. There are 15 conditions, and we report the average accuracy of them. The evaluation of MetaSense with this dataset highlights the effectiveness in stress and affect detection.

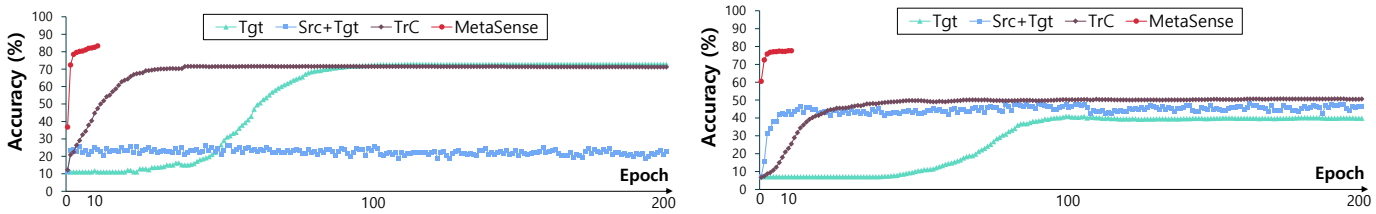
5.5.4 ExtraSensory

ExtraSensory [56] dataset was collected with 60 participants in the free-living environment for seven days. Each subject used their personal phone (34 were iOS users and 26 were Android users), logged sensor data and self-reported labels describing their activity context. As there was no constraint on activities the participants needed to perform, the distribution of activities are different among users. We thus



(a) Activity recognition. (b) Speech recognition.

Fig. 8: Accuracy with and without our task generation strategies.



(a) Activity recognition. (b) Speech recognition.

Fig. 9: Target accuracy changes over epochs.

Fig. 10: Four vision datasets used in our experiment.

used the three most common activities (lying-down, sitting, and walking) and removed the other labels. Besides, users could selectively provide a subset of sensor information. We used the data with all sensor modalities (accelerometer, gyroscope, magnetometer, audio, and location) and removed data with any missing sensor information. We used the 8-length window with 50% overlapping between two consecutive windows. We selected users with greater than or equal to 20 shots (the minimum for the following evaluations) of the three classes with full sensor information, which leaves us with 13 users. We report the average of the 13 individual conditions.

5.5.5 Vision Datasets

Although MetaSense is designed for mobile sensing, we want to test the performance of MetaSense in another domain and evaluate whether our approach could be translated int other domains. We therefore experimented our framework with vision datasets, which is the most active domain of machine learning research. Specifically, we used four vision datasets as shown in Figure 10: MNIST [57], MNIST with different background and colors (MNIST-M) [47], Synthetic numbers (SYNNUM) [47], and Street-View House Number data set (SVHN) [58], which are used in the unsupervised domain adaptation problem [47]. As these four datasets have the same class set (digits) and have different distributions, we think they are suitable for our experiment. We selected one dataset as the target and used the remaining as the source dataset. We report the average accuracy of the four targets.

5.5.6 Results

Figures 11, 12, 13, and 14 show the accuracy of the baselines and MetaSense with the HHAR, DSA, WESAD, and

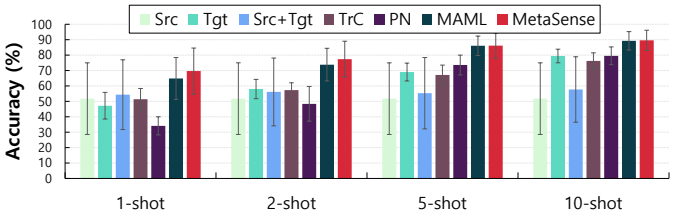


Fig. 11: Accuracy of the baselines and MetaSense on the HHAR dataset.

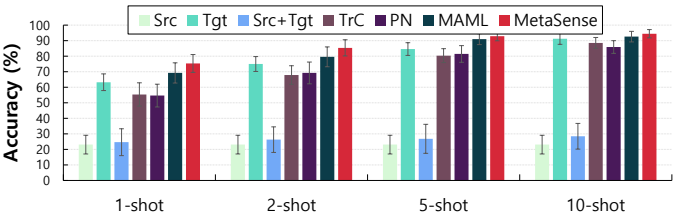


Fig. 12: Accuracy of the baselines and MetaSense on the DSA dataset.

ExtraSensory datasets, respectively. The results indicate that the effectiveness of MetaSense generalizes to other sensing datasets. The baselines show different trends between different datasets. For instance, the higher accuracy of Src and Src+Tgt in the HHAR dataset than in the DSA dataset means HHAR has more similar distributions among the conditions. On the other hand, Src and Src+Tgt perform poorly in the DSA dataset due to the severe differences between conditions. MetaSense nevertheless shows robust performance due to its flexibility in learning and adaptivity to new conditions.

Figure 15 shows the result for the vision datasets. Compared to the mobile sensing datasets, Src and Src+Tgt outperform Tgt and other methods. We interpret that this is because the knowledge learned from vision datasets (i.e., ten digits) is more transferable to different datasets with the same classes, while mobile sensing datasets are more individual than vision datasets, and thus the target data is more important. Still, MetaSense outperforms transfer learning and meta-learning based adaptation, which shows its effectiveness in the vision domain compared to other

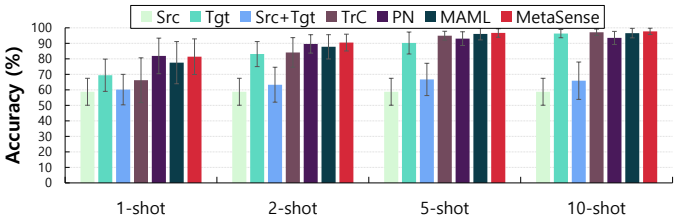


Fig. 13: Accuracy of the baselines and MetaSense on the WESAD dataset.

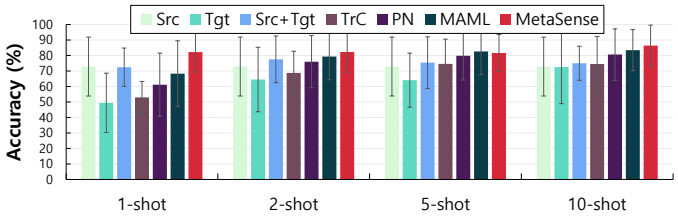


Fig. 14: Accuracy of the baselines and MetaSense on the ExtraSensory dataset.

adaptation methods.

5.6 Impact of Similar Condition Detector

We now evaluate the impact of gathering additional shots from SCD. We evaluate the accuracy of MetaSense under two settings, i.e., with- and without-similar conditions in the source dataset to understand the impact of SCD in both cases. We used all aforementioned datasets for the evaluation. To separately evaluate the with- and without-similar cases within the same dataset, we additionally generated situations where similar conditions are included in the source dataset. Specifically, we divided the data in the target condition into halves, incorporated one half into the source dataset, and the other half is used for few-shots of the target. By doing so, a source dataset includes a similar condition to the target condition. We also evaluate the impact of having partially identical conditions in the source dataset in §5.6.3.

5.6.1 Accuracy with Similar Conditions

We investigate the effect of SCD when similar conditions are included in the source. Figure 16 shows the accuracy of the baselines and MetaSense among four datasets. Tgt is excluded as it does not use source dataset in the training. We compare MetaSense without SCD to MetaSense with "naïve SCD" and SCD, where the naïve SCD is a baseline that always selects the condition with the closest distance to the target without considering the similarity. For instance, in Figure 5, naïve SCD chooses S_9 as the distance from its prototype to the target is the shortest (150). In our with-similar-conditions settings, naïve SCD can be seen as an effective upper bound for SCD as it always assumes there exists a similar condition and fetches shots from it.

As expected, if similar conditions are included in the source data, the accuracy of Src and Src+Tgt is drastically increased compared with results without similar conditions. It is interesting to note that the performance of transfer learning (TrC) and meta learning (PN, MAML) baselines are far below that of Src and Src+Tgt as they do not leverage similar conditions in the source and the target. Similarly, MetaSense *without* SCD suffers from the same

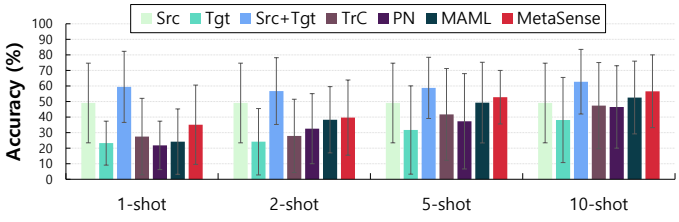


Fig. 15: Accuracy of the baselines and MetaSense on the vision datasets.

problem. Meanwhile, with SCD, MetaSense shows a significant improvement. The accuracy improvement is prominent especially when the number of given shots is few (e.g., 15% increase for activity recognition in the 1-shot case), by utilizing additionally fetched shots from similar conditions. We also observe that with the help of SCD, the classification accuracy of MetaSense (w/ SCD) is comparable to that of naïve SCD.

5.6.2 Accuracy without Similar Conditions

When a source dataset does not have similar conditions from the target, a desirable SCD should not degrade the performance resulting from wrong selections. We thus evaluate the accuracy impact of SCD in scenarios without similar conditions in the source dataset. Figure 17 shows the result. We focus on MetaSense without SCD, with naïve SCD, and with SCD as the accuracy of other methods are the same as previous results (Figures 6, 11, and 12). We only present the 5-shot case but the trend is similar across different number of shots.

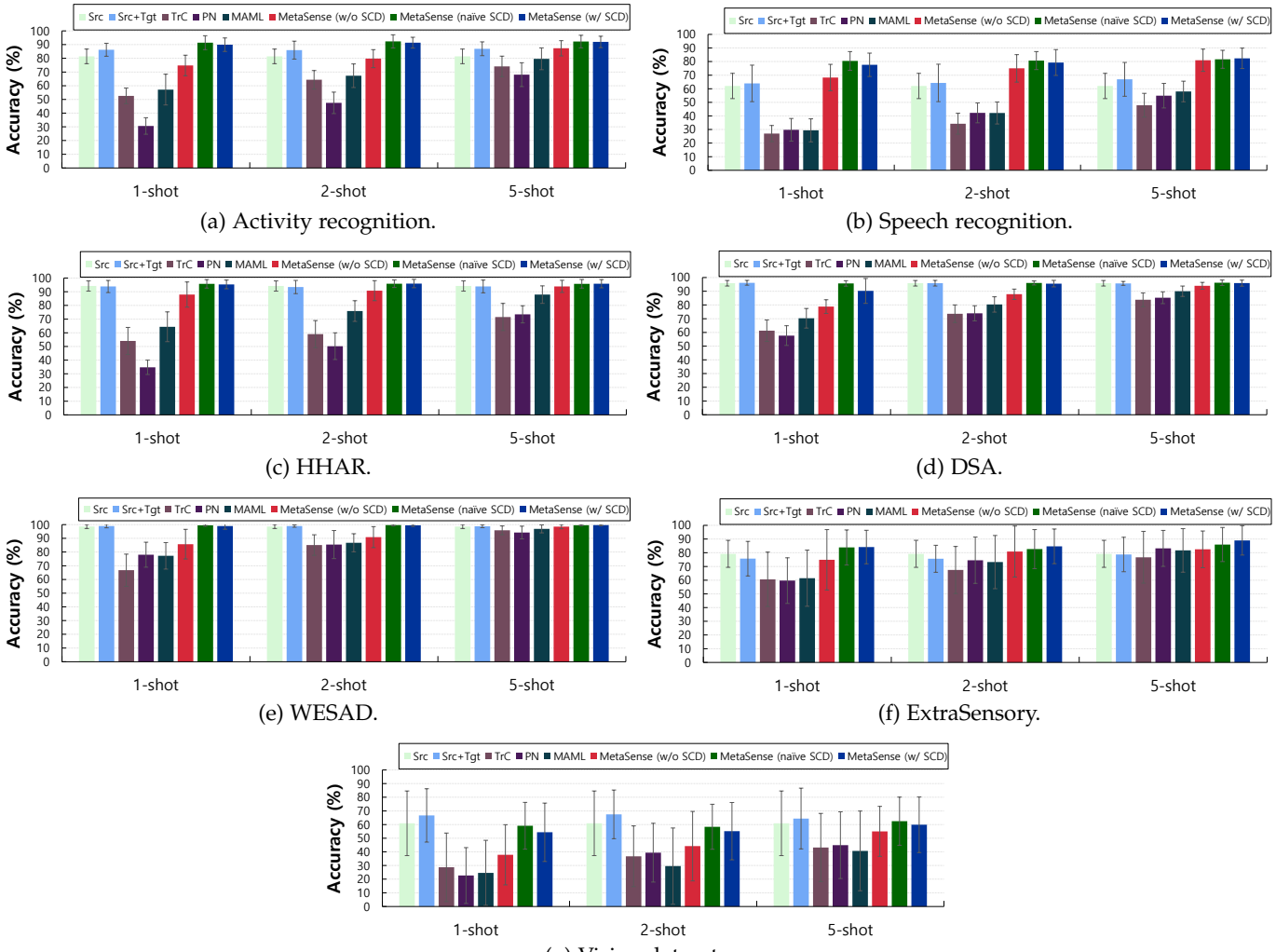
Compared with MetaSense without SCD, naïve SCD shows an accuracy degradation (except for the ExtraSensory and Vision datasets) as it always gets additional shots from the closest condition without considering whether it is similar to the target. On the other hand, MetaSense with SCD shows nearly identical performance to MetaSense without SCD by effectively rejecting wrong selections that are closest conditions but not similar to the target.

The result of the vision datasets is in line with the findings of the previous result in Figure 15. In both with- and without-similar conditions cases, naïve SCD is better in the vision experiment because of utilizing transferable knowledge from other conditions. We found that the ExtraSensory dataset could leverage the common knowledge from other conditions, and thus there is no degradation in naïve SCD, possibly due to fewer classes (three) compared to the other datasets.

In summary, SCD enhances the accuracy of MetaSense when similar conditions exist in the source dataset and does not degrade the accuracy when there is no similar conditions. We believe our SCD framework, combined with MetaSense, further brings mobile sensing applications closer to wide deployment in real settings.

5.6.3 Impact of Training with the User, Device, or Position

In the previous experiments for the with-similar case, the source data includes the target condition. This inclusion simplifies the analysis in that we can separately evaluate under with- and without-similar conditions. However, learning with the data collected from the identical condition



(a) Activity recognition.

(b) Speech recognition.

(c) HHAR.

(d) DSA.

(e) WESAD.

(f) ExtraSensory.

(g) Vision datasets.

Fig. 16: Accuracy with similar conditions included in the source dataset.

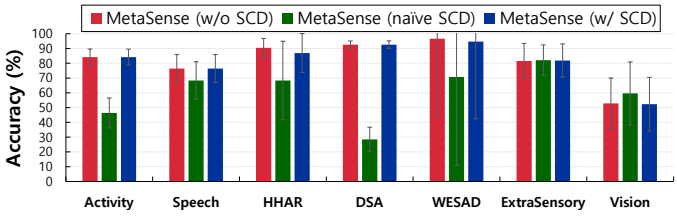


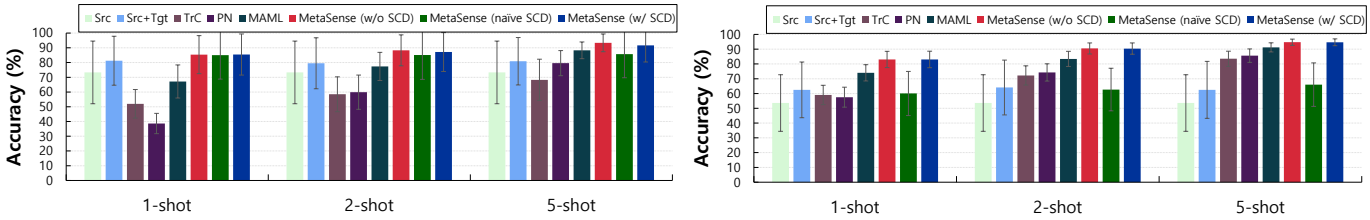
Fig. 17: Accuracy without similar conditions in the source dataset given 5 shots from the target.

to the target might always improve accuracy. We thus investigate the impact of SCD when the source data includes "partially identical" conditions to the target. For instance, the source data includes the same device as the target user has, but collected from a different user.

We use HHAR and DSA datasets for this experiment as they were collected from all combinations (i.e., from all users × devices or users × positions) and thus suitable for our purpose. Specifically, for the HHAR dataset, we select a target among "six users × four devices" combinations, and use the rest 23 as the source data. For the DSA dataset, we select a target among "eight users × five positions" combinations and use the rest 39 as the source data. This way, we can include partial conditions (e.g., training with the target user and the device, but not exactly the same pair) as opposed to the previous experiments with no overlap between the source and the target §5.5.

Figure 18a shows the result for HHAR and Figure 18b shows the result for DSA. The result shows that with partially similar conditions, the overall trend is somewhere between those without similar conditions (Figure 17) and with similar conditions (Figure 16). We observed that SCD did not detect anything similar to the target condition in most cases, and thus the performance between MetaSense w/o SCD and w/ SCD is similar. This is also in line with our motivation and previous findings that a combination of user, device, and position makes a unique individual condition.

Interestingly, we found that SCD detected a similar condition in some cases, but it did not necessarily result in performance improvement or decline. Especially for the HHAR with 1-shot case, we found that the most similar condition to the target selected from the naïve SCD does not harm the performance, while with more shots given, the accuracy of naïve SCD decreases compared to MetaSense. This implies that with more shots, directly adapting to the target data is more effective than relying on the most similar condition from the source, and our SCD algorithm can effectively catch this situation.



(a) HHAR. (b) DSA.

Fig. 18: Accuracy when (i) the same user or device is included (HHAR) and (ii) the same user or position is included in the source data.

6 Related Work

We summarize prior approaches that tackle the challenge of diverse dependencies in mobile sensing.

6.1 Synthetic Training

One category to mitigate the dependency problem is to train with synthetic training examples generated from the source dataset [35], [59]. Mathur et al. [35] proposed building a deep model with synthetic data made of multiple devices to mitigate hardware/software heterogeneities of smart devices. However, this solution is focused only on the device dependency. CrossSense [59] proposed a roaming model for large-scale cross-site WiFi sensing. It leverages a large amount of source data for generating synthetic data that mimic unseen instances or users from the target site. However, it requires thousands of samples from the target site to train the roaming model, while MetaSense requires only a few shots.

6.2 Utilizing Unlabeled Target Data

Another line of research utilizes unlabeled data from a target condition [52], [53], [60], [61], [62]. This approach employs transfer learning (or domain adaptation); with labeled data from the source and unlabeled data from the target condition, it trains an adaptive model for the target condition. The advantage of this approach is that target users need not label their data. Although the approach does not require labeled target data, it needs a large amount of target data compared to our few-shot learning scheme. Furthermore, these approaches have been limited to specific individual conditions, e.g., changes of sensor positions on the body [61], [62] and changes of users with the same device [60]. It is uncertain whether such unsupervised approach would be accurate under a complex combination of multiple dependencies where the input distribution is different from the source dataset; a study showed that the performance for HAR under individual conditions is only marginally improved or often dropped with the unlabeled target data [53].

7 Discussion

We discuss the limitations of MetaSense and suggest future research.

7.1 Long-term Behavior Changes

Our current design of MetaSense requires users to provide a few shots only at the initial adaptation step. After the adaptation, the model is adapted to the target user's condition. However, user behaviors could change with time (e.g., walking slowly when one gets ill) and this could affect the model performance. To handle such a scenario, one can periodically adapt the model parameters for fast adaptation. Since the model is already adapted for that user at the initial step, the model would require even fewer data to adapt to the behavior changes. We remark that a recent meta learning scheme [63] that continuously adapts to non-stationary environments could be a promising direction to explore for adapting to long-term behavior changes.

7.2 Number of Classes and User Effort

MetaSense aims to minimize the users' labeling burden via the concept of few-shot learning. However, as a *shot* means one labeled instance for each class, the labeling cost increases when the number of classes increases. This is an inherent issue in classification problems where the number of classes is inversely proportional to the classification accuracy given the same amount of training data per class. For less user burden, adaptation with few shots from partial classes could be considered. For instance, a recent study [64] proposed generating data for missing classes via a generative adversarial network (GAN), which could further mitigate user effort in conjunction with MetaSense.

7.3 Other Dependencies

We considered a typical practical scenario in mobile sensing where there exist different user behaviors and different devices (accordingly sensor positions and orientations). We realize in real deployments there could be other unexpected dependency problems such as environmental changes that we have not considered. However, our approach could be employed in other dependency problems. For instance, activity recognition with Wi-Fi signals faces the challenge of environment and user dependency [52], [59]. In situations where combinations of dependencies make input distributions significantly heterogeneous, we believe the insights and methods from MetaSense could be applied.

8 Conclusion

We investigated the problem of individual conditions in mobile sensing and how deep learning models perform under such situations. Inspired by the recent successes of meta learning in the machine learning community, we proposed MetaSense, a few-shot adaptation system that learns to learn for deep mobile sensing as a solution to this problem. MetaSense leverages intelligently generated tasks, parameter updates via meta learning, and similar condition detection for resolving individual conditions in mobile sensing. In essence, MetaSense is model-agnostic,

i.e., applicable to any deep learning models, and condition-agnostic, i.e., its coverage is not limited to a specific type of sensors and applications. Our evaluation with multiple real-world datasets showed that MetaSense outperforms other approaches in both accuracy and adaptation time with very few training examples. We believe MetaSense is a step towards mainstream adoption of mobile sensing for practical impact. The proposed meta learning approach and the insights from our study could be applied in innovative mobile sensing applications so that everyday users could deploy them without being limited by operating conditions.

ACKNOWLEDGMENTS

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No.NRF-2020R1A2C1004062) and by Microsoft Research.

REFERENCES

[1] C. A. Ronao and S.-B. Cho, "Deep convolutional neural networks for human activity recognition with smartphone sensors," in *International Conference on Neural Information Processing*. Springer, 2015, pp. 46–53.

[2] ——, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert systems with applications*, vol. 59, pp. 235–244, 2016.

[3] V. Radu, C. Tong, S. Bhattacharya, N. D. Lane, C. Mascolo, M. K. Marina, and F. Kawsar, "Multimodal deep learning for activity and context recognition," *Proceedings of IMWUT*, vol. 1, no. 4, p. 157, 2018.

[4] A. Soro, G. Brunner, S. Tanner, and R. Wattenhofer, "Recognition and repetition counting for complex physical exercises with deep learning," *Sensors*, vol. 19, no. 3, p. 714, 2019.

[5] N. D. Lane, P. Georgiev, and L. Qendro, "Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *Proceedings of UbiComp*. ACM, 2015, pp. 283–294.

[6] G. Laput, K. Ahuja, M. Goel, and C. Harrison, "Ubicoustics: Plug-and-play acoustic activity recognition," in *Proceedings of UIST*. ACM, 2018, pp. 213–224.

[7] B. Zhou, J. Lohokare, R. Gao, and F. Ye, "Echoprint: Two-factor authentication using acoustics and vision on smartphones," in *Proceedings of MobiCom*. ACM, 2018, pp. 321–336.

[8] J. Chauhan, J. Rajasegaran, S. Seneviratne, A. Misra, A. Seneviratne, and Y. Lee, "Performance characterization of deep learning models for breathing-based authentication on resource-constrained devices," *Proceedings of IMWUT*, vol. 2, no. 4, p. 158, 2018.

[9] Q. Dai, J. Hou, P. Yang, X. Li, F. Wang, and X. Zhang, "The sound of silence: end-to-end sign language recognition using smartwatch," in *Proceedings of MobiCom*. ACM, 2017, pp. 462–464.

[10] J. Lu, C. Shang, C. Yue, R. Morillo, S. Ware, J. Kamath, A. Bamis, A. Russell, B. Wang, and J. Bi, "Joint modeling of heterogeneous sensing data for depression assessment via multi-task learning," *Proceedings of IMWUT*, vol. 2, no. 1, p. 21, 2018.

[11] A. Mehrotra and M. Musolesi, "Using autoencoders to automatically extract mobility features for predicting depressive states," *Proceedings of IMWUT*, vol. 2, no. 3, p. 127, 2018.

[12] H. Zhang, C. Song, A. Wang, C. Xu, D. Li, and W. Xu, "Pdvocal: Towards privacy-preserving parkinson's disease detection using non-speech body sounds," in *Proceedings of MobiCom*, 2019, pp. 1–16.

[13] G. M. Weiss and J. W. Lockhart, "The impact of personalization on smartphone-based activity recognition," in *AAAI Workshop on Activity Context Representation: Techniques and Languages*, 2012, pp. 98–104.

[14] Y. E. Ustev, O. Durmaz Incel, and C. Ersoy, "User, device and orientation independent human activity recognition on mobile phones: Challenges and a proposal," in *Proceedings of UbiComp Adjunct*. ACM, 2013, pp. 1427–1436.

[15] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen, "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition," in *Proceedings of SenSys*. ACM, 2015, pp. 127–140.

[16] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of CVPR*, 2014, pp. 1701–1708.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of CVPR*, 2017, pp. 4700–4708.

[19] P. Siirtola and J. Röning, "Recognizing human activities user-independently on smartphones based on accelerometer data," *IJIMAI*, vol. 1, no. 5, pp. 38–45, 2012.

[20] R. Yang and B. Wang, "Pacp: a position-independent activity recognition method using smartphone sensors," *Information*, vol. 7, no. 4, p. 72, 2016.

[21] H. Guo, L. Chen, G. Chen, and M. Lv, "Smartphone-based activity recognition independent of device orientation and placement," *International Journal of Communication Systems*, vol. 29, no. 16, pp. 2403–2415, 2016.

[22] A. Grammenos, C. Mascolo, and J. Crowcroft, "You are sensing, but are you biased?: A user unaided sensor calibration approach for mobile sensing," *Proceedings of IMWUT*, vol. 2, no. 1, p. 11, 2018.

[23] P. Siirtola and J. Röning, "Ready-to-use activity recognition for smartphones," in *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*. IEEE, 2013, pp. 59–64.

[24] F. Gu, A. Kealy, K. Khoshelham, and J. Shang, "User-independent motion state recognition using smartphone sensors," *Sensors*, vol. 15, no. 12, pp. 30 636–30 652, 2015.

[25] M. Shoaib, S. Bosch, O. Incel, H. Scholten, and P. Havinga, "A survey of online activity recognition using mobile phones," *Sensors*, vol. 15, no. 1, pp. 2059–2085, 2015.

[26] J. Saha, C. Chowdhury, and S. Biswas, "Device independent activity monitoring using smart handhelds," in *Cloud Computing, Data Science & Engineering-Confluence, 2017 7th International Conference on*. IEEE, 2017, pp. 406–411.

[27] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of ICML*. JMLR. org, 2017, pp. 1126–1135.

[28] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proceedings of ICML*, 2016, pp. 1842–1850.

[29] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015.

[30] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017, pp. 4077–4087.

[31] T. Gong, Y. Kim, J. Shin, and S.-J. Lee, "MetaSense: few-shot adaptation to untrained conditions in deep mobile sensing," in *Proceedings of SenSys*. ACM, 2019, pp. 110–123.

[32] M. Rossi, S. Feese, O. Amft, N. Braune, S. Martis, and G. Tröster, "Ambientsense: A real-time ambient sound recognition system for smartphones," in *PerCom Workshops*. IEEE, 2013, pp. 230–235.

[33] S. Chang, J. Lee, S. K. Choe, and K. Lee, "Audio cover song identification using convolutional neural network," *arXiv preprint arXiv:1712.00166*, 2017.

[34] S. A. Rokni, M. Nourollahi, and H. Ghasemzadeh, "Personalized human activity recognition using convolutional neural networks," in *AAAI*, 2018.

[35] A. Mathur, T. Zhang, S. Bhattacharya, P. Veličković, L. Joffe, N. D. Lane, F. Kawsar, and P. Lió, "Using deep data augmentation training to address software and hardware heterogeneities in wearable and smartphone sensing devices," in *Proceedings of IPSN*. IEEE Press, 2018, pp. 200–211.

[36] L. Mirani, "There are now more than 24,000 different android devices," Aug 2015. [Online]. Available: https://qz.com/472767/there-are-now-more-than-24000-different-android-devices/

[37] R. Saeedi, S. Norgaard, and A. H. Gebremedhin, "A closed-loop deep learning architecture for robust activity recognition using wearable sensors," in *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2017, pp. 473–479.

[38] P. Siirtola and J. Röning, "Reducing uncertainty in user-independent activity recognition-a sensor fusion-based approach." in *ICPRAM*, 2016, pp. 611–619.

[39] H. Koskimäki and P. Siirtola, "Adaptive model fusion for wearable sensors based human activity recognition," in *FUSION*. IEEE, 2016, pp. 1709–1713.

[40] P. Siirtola, H. Koskimäki, and J. Röning, "From user-independent to personal human activity recognition models using smartphone sensors," *Proc ESANN'16*, pp. 471–476, 2016.

[41] Z. Wang, D. Wu, R. Gravina, G. Fortino, Y. Jiang, and K. Tang, "Kernel fusion based extreme learning machine for cross-location activity recognition," *Information Fusion*, vol. 37, pp. 1–9, 2017.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[43] Q. Kong, R. M. Allen, L. Schreier, and Y.-W. Kwon, "Myshake: A smartphone seismic network for earthquake early warning and beyond," *Science advances*, vol. 2, no. 2, p. e1501055, 2016.

[44] Z. Shen, K. Yang, W. Du, X. Zhao, and J. Zou, "Deepapp: a deep reinforcement learning framework for mobile application usage prediction," in *Proceedings of SenSys*. ACM, 2019, pp. 153–165.

[45] T. Yan, M. Marzilli, R. Holmes, D. Ganesan, and M. Corner, "mcrowd: a platform for mobile crowdsourcing," in *Proceedings of SenSys*, 2009, pp. 347–348.

[46] S. S. Kanhere, "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2013, pp. 19–26.

[47] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, p. 2096–2030, Jan. 2016.

[48] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[49] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2951–2959. [Online]. Available: http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf

[50] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, vol. 43, no. 10, pp. 3605–3620, 2010.

[51] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[52] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas *et al.*, "Towards environment independent device free human activity recognition," in *Proceedings of MobiCom*. ACM, 2018, pp. 289–304.

[53] M. A. A. H. Khan, N. Roy, and A. Misra, "Scaling human activity recognition via deep learning-based domain adaptation," in *PerCom*. IEEE, 2018, pp. 1–9.

[54] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS*, 2017.

[55] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, and K. Van Laerhoven, "Introducing wesad, a multimodal dataset for wearable stress and affect detection," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, 2018, pp. 400–408.

[56] Y. Vaizman, K. Ellis, G. Lanckriet, and N. Weibel, "Extrasensory app: Data collection in-the-wild with rich user interface to self-report behavior," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–12. [Online]. Available: https://doi.org/10.1145/3173574.3174128

[57] Y. LeCun and C. Cortes, "MNIST handwritten digit database," –, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[58] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[59] J. Zhang, Z. Tang, M. Li, D. Fang, P. Nurmi, and Z. Wang, "Crosssense: Towards cross-site and large-scale wifi sensing," in *Proceedings of MobiCom*. ACM, 2018, pp. 305–320.

[60] R. Fallahzadeh and H. Ghasemzadeh, "Personalization without user interruption: Boosting activity recognition in new subjects using unlabeled data," in *Proceedings of the 8th International Conference on Cyber-Physical Systems*. ACM, 2017, pp. 293–302.

[61] J. Wang, V. W. Zheng, Y. Chen, and M. Huang, "Deep transfer learning for cross-domain activity recognition," in *Proceedings of the 3rd International Conference on Crowd Science and Engineering*. ACM, 2018, p. 16.

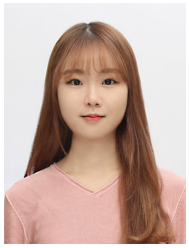
[62] Y. Chen, J. Wang, M. Huang, and H. Yu, "Cross-position activity recognition with stratified transfer learning," *arXiv preprint arXiv:1806.09776*, 2018.

[63] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," in *ICLR*, 2018.

[64] N. Suzuki, Y. Watanabe, and A. Nakazawa, "Gan-based style transformation to improve gesture-recognition accuracy," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 4, Dec. 2020. [Online]. Available: https://doi.org/10.1145/3432199

Taesik Gong received his Bachelors of Science in Computer Science from Yonsei University in 2016 (Summa Cum Laude). He received his Master of Science in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 2017. He is currently pursuing his Doctor of Philosophy (Ph.D.) in Computer Science in KAIST. His research interests are in mobile computing, ubiquitous sensing, and applied machine learning.

Yeonsu Kim received her Bachelors of Science in Computer Science from Korea Advanced Institute of Science and Technology (KAIST). She is currently pursuing her M.S. degree in Computer Science in KAIST. Her research interests include mobile computing, human-computer interaction, and machine learning.



Ryuhaerang Choi received her Bachelors of Science in Computer Science from Inha University in 2019 (Summa Cum Laude). She is currently pursuing her Master of Science (M.S.) in Computer Science from Korea Advanced Institute of Science and Technology (KAIST). Her research interests are ubiquitous sensing, mobile health and mobile human-computer interaction.

Jinwoo Shin is currently an associate professor (jointly affiliated) in the Graduate School of AI and the School of Electrical Engineering at KAIST. He is also a KAIST endowed chair professor. He obtained the Ph.D. degree (in Math) from Massachusetts Institute of Technology in 2010 with George M. Sprowls Award (for best MIT CS PhD theses). He was a postdoctoral researcher at Algorithms & Randomness Center, Georgia Institute of Technology in 2010-2012 and Business Analytics and Mathematical Sciences Department, IBM T. J. Watson Research in 2012-2013. After he joined KAIST in Fall 2013, he started to work on machine learning: topics include graphical models, distributed optimization, uncertainty estimation, etc. He received the Rising Star Award in 2015 from the Association for Computing Machinery (ACM) Special Interest Group for the computer systems performance evaluation community (SIGMETRICS).

Sung-Ju Lee is a Professor and KAIST Endowed Chair Professor at KAIST. He received his Ph.D. in computer science from the University of California, Los Angeles in 2000, and spent 15 years in the industry in Silicon Valley before joining KAIST. His research interests include computer networks, mobile computing, network security, and HCI. He is the winner of the HP CEO Innovation Award, the Best Paper Award at IEEE ICDCS 2016, and the Test-of-Time Paper Award at ACM WINTECH 2016. He is an IEEE Fellow and ACM Distinguished Scientist.