Towards Recognizing Food Types for Unseen Subjects

JIEXIONG GUAN, Computer Science Department, William & Mary, USA
JUNJIE WANG, Computer Science Department, William & Mary, USA
WEI NIU, Computer Science Department, William & Mary, USA
ZHEN PENG, Computer Science Department, William & Mary, USA
SHUANGQUAN WANG, Computer Science Department, Salisbury University, USA
ZHENMING LIU, Computer Science Department, William & Mary, USA
GANG ZHOU, Computer Science Department, William & Mary, USA
BIN REN, Computer Science Department, William & Mary, USA

Recognizing food types through sensor signals for unseen users remains remarkably challenging, despite extensive recent studies. The efficacy of prior machine learning techniques is dwarfed by giant variations of data collected from multiple participants, partly because users have varied chewing habits and wear sensor devices in various manners. This work treats the problem as an instance of the domain adaptation problem, where each user represents a domain. We develop the first multi-source domain adaptation (MSDA) method for food-typing recognition, which consists of three major components: stratified normalization, a multi-source domain adaptor, and adaptive ensemble learning. New techniques are developed for each component. Using a real-world dataset comprised of 15 participants, we demonstrate that our method achieves 1.33× to 2.13× improvement in accuracy compared with nine state-of-the-art MSDA baselines. Additionally, we perform an in-depth ablation study to examine the behavior of each component and confirm their efficacy.

Additional Key Words and Phrases: Food Type Recognition, Multi-Source Domain Adaptation, Stratified Normalization, Adaptive Ensemble

Authors' addresses: Jiexiong Guan, jguan@wm.edu, Computer Science Department, William & Mary, Williamsburg, Virginia, USA, 23187-8795; Junjie Wang, jwang51@wm.edu, Computer Science Department, William & Mary, Williamsburg, Virginia, USA, 23187-8795; Wei Niu, wniu@wm.edu, Computer Science Department, William & Mary, Williamsburg, Virginia, USA, 23187-8795; Zhen Peng, zpeng01@wm.edu, Computer Science Department, William & Mary, Williamsburg, Virginia, USA, 23187-8795; Shuangquan Wang, spwang@salisbury.edu, Computer Science Department, Salisbury University, Salisbury, Maryland, USA, 21801-6862; Zhenming Liu, zliu20@wm.edu, Computer Science Department, William & Mary, Williamsburg, Virginia, USA, 23187-8795; Gang Zhou, gzhou@cs.wm.edu, Computer Science Department, William & Mary, Williamsburg, Virginia, USA, 23187-8795; Bin Ren, bren@cs.wm.edu, Computer Science Department, William & Mary, Williamsburg, Virginia, USA, 23187-8795.

ACM 2637-8051/2024/9-ART
https://doi.org/10.1145/3696424

1 INTRODUCTION

The escalating global prevalence of obesity poses a significant public health risk, contributing to an alarming number of premature deaths each year in both developed and rapidly developing countries. In the United States alone, approximately 670,000 deaths annually are attributed to nutrition- and obesity-related diseases, including heart disease, cancer, and diabetes. Moreover, maintaining a well-structured dietary pattern is crucial for the health status of every individual. To address this health crisis, there has been a push for the development of innovative sensor-based technologies and machine learning models aimed at monitoring food intake and analyzing collected data. Among the various solutions, machine learning models using motion and acoustic sensors, often integrated into wearable devices like smartwatches, earphones, and glasses, have shown promise [47]. Notable advancements include the application of random forests [36] and neural networks [55] to model food typing.

However, a critical challenge emerges when models are trained on a collective dataset from various users—these models struggle to generalize effectively across different sets of users. Table 1 illustrates the diminished performance when predicting food types for unseen users, highlighting the limitation of existing machine learning models. This generalization issue becomes a significant barrier to large-scale deployment, as users expect technologies to work seamlessly across diverse individuals. The variations in sensor locations and chewing habits among different users hinder machine learning models from extracting robust user-oblivious signals, a problem commonly referred to as the domain adaptation (DA) problem [5].

Table 1. **Prior works' low generalization performance.** We referenced the original paper and re-implemented their works to present seen and unseen users' accuracy scores. The original implementations have data from the same user in training and testing sets and achieved high accuracy. The accuracy scores are degraded to a great extent when the setting is changed to predict the unseen user, i.e., split data from one user as the testing set exclusively.

	# Users	# Food types	Model type	Acc. of predicting	
				Seen users	Unseen users
Mirtchouk et~al. [36]	6	40	Random forest	82.7%	29%
Wang et~al. [55]	15	11	Two-layer Perceptron	82.3%	23%

In this work, we specifically focus on the multi-source domain adaptation problem (MSDA) [34, 51, 61], where labeled training data belongs to multiple domains, exacerbating the challenges associated with domain divergence. Existing domain adaptation methods fall short when applied to the task of predicting food types for unseen users. To address this, we propose a comprehensive pipeline that integrates a diverse set of techniques to combat multiple interrelated subproblems (see Figure 4). Our contributions can be categorized into three main techniques:

Domain-invariant features: These features refer to data representations that remain consistent across different domains. While end-to-end neural networks aim to automatically extract features unaffected by distribution variations, such approaches prove ineffective in our setting. CoDATs [57], for example, work on long-range muscle motions (e.g., sitting down, walking), making it unclear how an end-to-end neural network can efficiently extract food-type signals from fine-grained muscle movements (e.g., chewing different types of foods) across diverse domains. To mitigate domain shifts, we propose to apply an "anti-wisdom" approach, leveraging handcrafted features to trade domain expertise and manual labor for a simplification in fitting the target function. Additionally, we introduce stratified normalization, inspired by stratified sampling in statistics, to control feature variations across domains. These techniques, relying on different principles, work synergistically to enhance signal extraction.

Source-source domain adaptation: This technique aims to align models trained on different source domains to control dissimilarity, facilitating effective generalization across diverse sources. While existing works [45, 56,

57, 60] primarily concentrate on adapting multiple source domains to the target domain, they often overlook the inherent divergence among the source domains themselves. It is essential to recognize that the multiple source domains not only differ from the unseen target domain but also exhibit variations among each other to varying extents. Therefore, the complexity of domain divergence is growing with the number of source domains. Consequently, mitigating source-source domain divergence becomes crucial when training a reliable classifier for the target domain, otherwise, the model will struggle to learn from multiple domains with different distributions. In our work, we renovate a multi-branch neural network where each branch independently adapts one source domain to the target. This adaptation process incorporates a consensus regularizer [33] to guide all branches, encouraging them to learn common features and effectively reduce source-source domain divergence.

Adaptive ensemble weight: This technique addresses the challenge of static ensemble weights, which can be suboptimal, leading to accuracy degradation when incorporating data or models from irrelevant domains. Inspired by a theoretical work [34], we introduce a two-stage adaptive ensemble method that dynamically adjusts ensemble weights for different users. Notably, our solution employs source-source similarities to filter out useless or harmful models prone to mispredicting food types for unseen users.

While domain adaptation challenges are prevalent in many machine learning applications, existing techniques are often tailored to specific domains and lack generalizability. This limitation becomes more pronounced in our context, where adapting to multiple domains becomes increasingly challenging with a growing number of users. Previous approaches are frequently tested on a limited number of domains (e.g., up to four domains [7, 16, 18, 26, 30, 40, 44, 54]) or synthetic data [39], rendering them less suitable for our multi-domain scenario. In contrast, our solution stands out as a "cocktail" flavor, offering a pipeline in which each stage is either robust or effective to adapt various numbers of domains. The first stage employs hand-crafted features and stratified normalization for each domain independently, ensuring effectiveness regardless of the number of domains. The second stage employs a multi-branch structured neural network with consensus regularization to control domain similarities. In the final stage, our adaptive ensemble scheme further enhances robustness across different domain scales.

In summary, our contributions are as follows:

- A Multi-Source Domain Adaptation (MSDA) pipeline with renovated algorithm components is proposed to address the generalization issue in the food type recognition task.
- A new set of techniques and principles is introduced, incorporating consensus regularizer, stratified normalization, and domain knowledge-guided feature extraction, to address the more severe domain divergence problem caused by the growing number of domains.
- A two-stage adaptive ensemble method is designed to automatically assign weights to relevant domains and prune off irrelevant ones. This method is robust to parameter settings and further improves accuracy.
- Extensive empirical evaluation is conducted. We experimentally verified the importance of hand-crafted features in the food typing task with multiple domains. Besides, the evaluation shows that our method achieved 1.33× to 2.13× higher accuracy than other baselines.

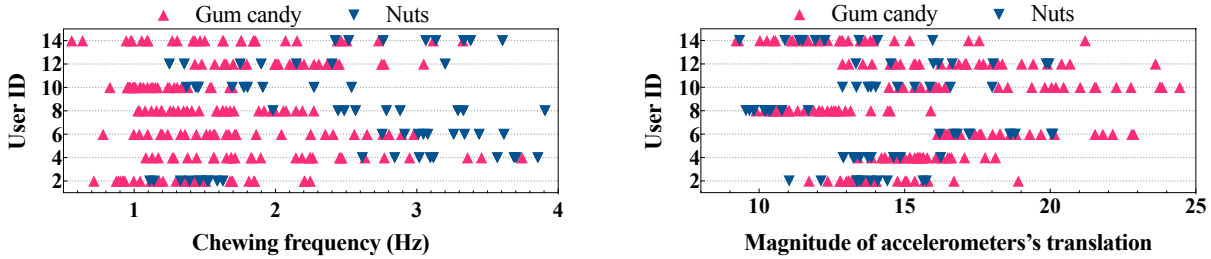
The rest of the paper is organized as follows: Section 2 explains the research effort closely related to this work. Section 3 presents the challenges of performing domain adaptation on food typing. Section 4 demonstrates the overall solutions. Section 5 describes experiments to evaluate our methods. Finally, a conclusion remark is given in Section 6.

2 RELATED WORK

Domain adaptation. The main challenge of domain adaptation was to reduce the domain discrepancy between different domains, which was approached from multiple perspectives: **1) Data manipulation and feature engineering.** Several existing works selected a subset of training samples or assigned weights to them based on the distance of one training sample to the test set [21, 31, 41]. Similarly, Nikolaidis et al. [37] iteratively selected

subset training samples with high confidence scores and fine-tuned the classifier with the selected data and predicted labels. An et al. [4] used labeled target samples to fine-tune specific layers of a neural net that produced user-specific features. In contrast, our approach prunes and assigns weights to the trained sub-models where the information of the dataset had been learned so that labeled data were not wasted. TCA [38] and CORAL [49] learned matrix mappings to align the features of different domains. Instance Normalization [15, 28] and AdaBN [29] designed domain-adapted normalization layers to transform intermediate feature maps in a neural net. These methods were not straightforward to integrate into our framework but were more suitable for other tasks or training methodologies. **2) Neural network innovation.** Maximum mean discrepancy (MMD) [48] measured the discrepancy between two domains and was applied to train various neural nets to reduce distribution shift [17, 32, 53, 64, 65]. Inspired by MMD-based solutions, various neural nets coupled with domain discrepancy measurement functions were proposed, including Deep-CORAL [50] and GAN-based solutions [13, 52, 59]. These approaches could be seamlessly extended to multi-source domain adaptation [51, 61] by combining multiple source domains into one; however, they were susceptible to accuracy degradation [14, 29, 57] because the learning procedure was interfered with by quadratically increased domain divergences [42]. Our method instead learned data from different domains through multi-branch model training [9, 43]. Finally, Luo et al. [33] proved that the disagreement between multiple sources was the upper bound for classification errors, so optimizing the consensus regularizer led to better prediction performance [27, 39, 64]. **3) Ensemble learning.** It was proven that the target distribution could be represented as a weighted combination of source distributions [34]. Accordingly, many existing efforts trained one model or multiple sub-models and late-fused the prediction confidence scores with uniform weights [46, 64] or fine-tuned weights based on various metrics. Peng et al. [39] assigned source-only accuracy weights to sub-models. Xu et al. [58] calculated a perplexity score during the adversary training procedure as a weight. Guo et al. [19] designed a point-to-set metric based on Mahalanobis distance to re-weight domain experts. Zhao et al. [62] re-weighted trained distilled source classifiers using Wasserstein distance. Our method updates the mask weights (ω_m) and the similarity weights (ω_s) based on the entropy of ℓ_1 distances between domain-specific models and MMD metrics. Another category of ensemble schemes was to update weights during training. In contrast, our method updates weights after training without interfering with the learning procedure.

Domain adaptation and Food type recognition. Recognizing food types through sensor signals achieved promising results in recent years. Oliver Amft's team achieved 80 $\sim$ 100% accuracy in classifying four food types using earbud-embedded microphone sensors [2]. Later, they produced two prototypes that achieved 80% and 86.6% accuracy, respectively, in classifying 19 food types [1, 3]. Yin et al. [6] proposed a prototype for recognizing seven types of food using two microphones embedded in a neckband. The microphone could also be placed near the mouth to classify six types of food [20]. Besides microphones, a smart utensil containing an array of LEDs could recognize twenty food types [22]. An intraoral sensor placed in the mouth while eating classified nine food types based on temperature and jawbone movement [8]. However, the current state-of-the-art is the work combining microphones with other sensor types. Samantha's team identified 40 different types of food with an accuracy of 82.7% [36], combining a microphone-embedded earbud, Google Glass, and two smartwatches. Although these food type recognition methods achieved acceptable accuracy, none considered the domain adaptation problem. Therefore, their recognition accuracy could significantly decrease when the application environment or scenario changed (see Table 1). Although prior works have applied domain adaptation to sensor signals in other tasks, they are not suitable for the food-type recognition task for various reasons. For example, Zheng et al. [63] generated fake labels for the target domain based on MMD [48] to recognize daily behaviors utilizing sensors scattered in an apartment. Mathur et al. [35] studied the domain adaptation problem caused by different sensor deployment locations. Jiang et al. [23] adopt an adversary training approach to recognize human activities for single subject on WiFi signals. These methods are not effective in recognizing food types without incorporating domain knowledge.



(a) Feature distribution of chewing speed

(b) Feature distribution of chewing force

Fig. 1. **Per-user feature distribution.** The left figure illustrates the feature distribution of chewing frequency, while the right figure depicts chewing force. Two classes are visualized with different markers and colors. Generally, users chew gum more slowly and with more force than nuts.

3 PROBLEM SETUP, MOTIVATION, AND CHALLENGES.

This section describes the problem definition, reviews standard techniques, and performs preliminary experiments to motivate our solutions.

Problem setup. Figure 1 illustrates the chewing habits of different users for two types of foods: gum candy and nuts. Users generally chew nuts faster and with less force than gum candy due to the properties of the foods—gum is chewy, while nuts are crispy. However, the distributions among different users vary significantly, leading to potential misclassification of food types. For instance, user 6 and user 8 exhibit completely different chewing forces, with the minimum chewing force of user 6 being greater than the maximum chewing force of user 8. This indicates distinct marginal distributions. Consequently, using a model trained on data from user 8 to predict data from user 6 could result in all instances being misclassified as gum candy, which requires a stronger chewing force. Similarly, user 2 and user 12 chew gum candy and nuts at a similar frequency, leading to similar conditional distributions of chewing speed. However, this differs from other users, who chew nuts at a higher frequency. These distribution divergences contribute to poor generalizability to unseen users, as shown in Table 1. Additionally, determining which labeled users are similar to a new, unlabeled user is challenging. To address the domain divergence issues in recognizing food types for unseen users, we first formalize this challenge as a multi-source domain adaptation (MSDA) problem. We then analyze the performance of prior solutions to motivate our designs.

In an MSDA scenario, there exist n source domains and a target domain T, corresponding to different individuals. We observe features and labels ($\mathbf{x}$'s and y's) from source domains and only features ($\mathbf{x}$'s) from the target. Our goal is to build a classifier for predicting labels in the target domain using the available labeled data from n users and unlabeled data from the target user. Let s_i be the number of observations from domain i and $S_i = \{(\mathbf{x}_i^j, y_i^j)\}_{j\in[s_i]}$ be the set of observations, each of which is independent and identically distributed (i.i.d.) sampled from the distribution $\mathcal{D}_i$. Similarly, we assume that the data $(y_T, \mathbf{x}_T)$'s are sampled from distribution $\mathcal{D}_T$ (note that y_T' are the ground truth and not observed). Let also $X_i = \{\mathbf{x}_i^j\}_{j\in[s_i]}$ $(i \in [n])$ be the set of features, and $X_T = \{\mathbf{x}_T^j\}_{j\in[t]}$ be the features of the target, where t is the total number of (unlabeled) observations from the target domain. Finally, let $\mathcal{D}_i(X)$ and $\mathcal{D}_T(X)$ be the feature distribution in domains i and T, respectively.

When $y_i = y_T$, meaning each domain has the same set of labels, the problem is defined as the closed set MSDA. If this condition does not hold, but for at least one y_i, $y_i \cap y_T \subset y_T$, the problem is defined as open set MSDA [61]. We describe and evaluate our method primarily using the closed set MSDA setting, similar to prior approaches [27, 39, 45, 56, 57, 60]. To test whether our method can adapt to the more challenging scenario where the target domain can only learn partial labels from each source domain, we also evaluate our method under the open set MSDA configuration in Section 5.4.

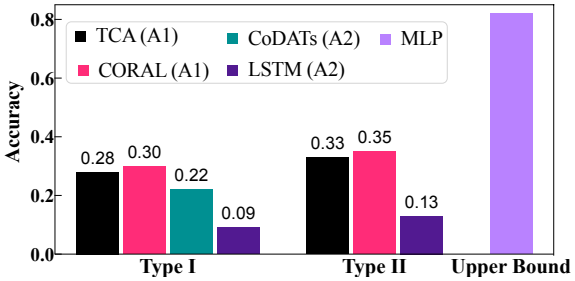


Fig. 2. **Prior solutions' accuracy.** Comparing existing domain adaptation algorithms to the upper bound (where labels from the target domain are available). We consider two variants of each existing solution: Type I variants use data from all domains to train one single model, whereas Type II variants train a model from each domain and run ensemble learning algorithms over multiple model predictions.

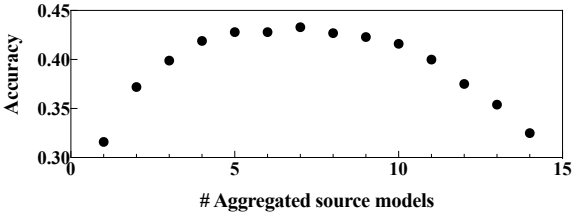


Fig. 3. **Ablation study of Multi-source domain adaptor.** A simple experiment for the breakdown of "more data is better," we use the standard ensemble learning method to build a forecast based on a linear combination of source models' forecasts. x-axis is the number of used source domains, and y-axis is the achieved accuracy. Each point represents the performance of the best-aggregated models subject to the source number constraint. One can see that the performance of the consolidated model first improves, then degrades as the number of sources increases.

Prior Solutions and Performance. Domain adaptation appears widely in many areas. Different downstream applications possess different distribution structures, so generic domain adaptation building blocks may not always be effective. To motivate our work, we review standard techniques and perform preliminary experiments to highlight their inefficacy. As aforementioned in Section 2, existing works develop or use techniques from one or more of the following categories: **A1. Data manipulation and feature engineering. A2. Neural network innovation. A3. Ensemble learning.** For example, CORAL [49] and TCA [38] use A1, DANN [14] uses A2, Schweikert et al [46] use A1 and A3, CoDATS [57] uses A2 and A3. Note that A1 and A2 usually do not appear together because there is a strong belief that properly designed neural network models can automatically learn representations from raw data and do not need heavy feature engineering. Therefore, no work simultaneously uses A1-A3.

Figure 2 showcases the results of our preliminary experiments on these techniques. The *Upper bound* refers to a setting, in which labels in the target domain are accessible. Colored bars depict the performance of existing MSDA algorithms. Rigorous tuning efforts were applied to these algorithms, exploring two variants: *Type I*, utilizing data from all sources to train a single model, and *Type II*, training a model for each source and generating forecasts for the target through a linear combination of source models. The upper bound achieves over 80% accuracy, while all MSDA techniques fall below 35%. This performance gap underscores the need for substantial advancements in techniques, and intriguingly, Type II variants, employing ensemble learning, tend to outperform their Type I counterparts—an observation we will revisit and leverage in our algorithm design.

Reasoning about the performance. A salient challenge we face here is that the *interactions* between sources and target, and between domains grow *quadratically* in the number of users and the source-source and source-target *divergences* are *uniformly high*. Specifically, a model tuned for a specific target requires us to control the divergences between each source and the target, and between sources, the latter of which is quadratic in the number of users. When divergences between sources are weak, source-source interaction can be suppressed in a model. All existing techniques in Figure2 focus on the source-target interaction and ignore the source-source interaction so they have reasonable performance only when the divergences between each pair of source domains are moderate, and quickly deteriorate when divergences are significant.

Compounding a large source number and large divergences further amplify the weakness of existing techniques/solutions: *(i)* Use A1 (or A1 & A3) without A2. TCA [38] and CORAL [49] focus on designing specialized

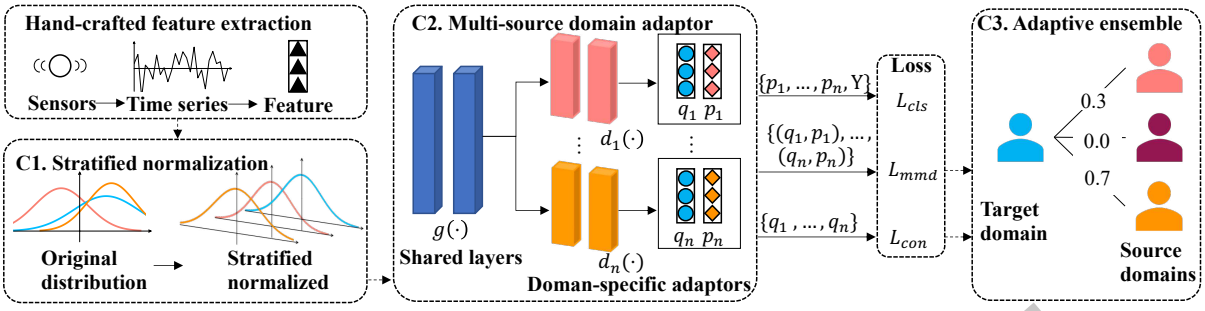


Fig. 4. **Overview of our solution.**

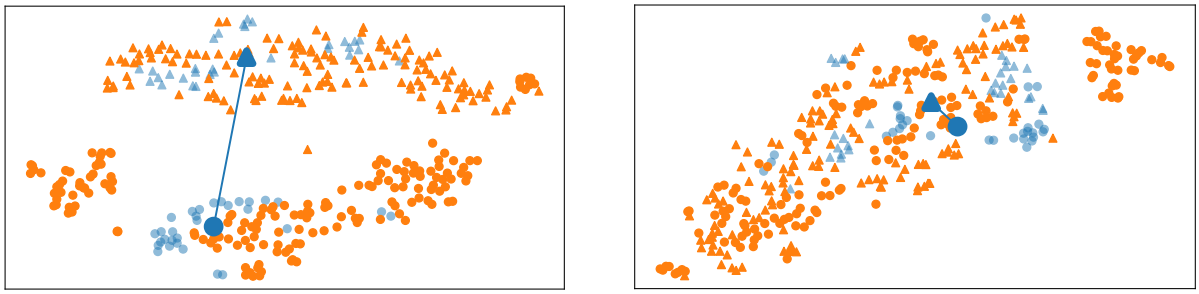
procedures to transform features $\mathbf{x}$'s from different domains and pipe the transformed $\mathbf{x}$ with a standard neural net, which is usually sub-optimal because neural network architecture/loss functions are not optimized toward the structure of the MSDA problem. *(ii)* Use A2 (or A2 & A3) without A1. While deep architectures (e.g., CoDATS [57]) are more flexible in learning feature representations, they cannot be fully automated to perform representation learning from the raw data effectively. We observe that deep architectures' inability to use raw features directly is a generic problem for wearable ML problems and is *not* tied to a specific downstream prediction task (in our case food type prediction). Section 4.6 elaborates further our observations. *(iii)* Problems of A3. Ensemble learning assumes that each weak learner delivers sufficiently "orthogonal" and useful predictions. This assumption also breaks. Specifically, we notice that sometimes having more ensembles in fact can *harm* (see Figure 3). This result shows that increasing the number of ensembles first improves the prediction capability, and then degrades it [14, 29, 57]. This highlights a delicate interaction among ensembles and the challenges in weighting (and pruning) them.

4 OUR APPROACH

This section explains our solution. Our key observation is that we need to innovate a broad set of techniques across *all* A1-A3 (feature engineering/normalization, model architecture, and ensemble weighting) and integrate them to collectively tackle the source-source and source-target divergence problems. Changes in one component (e.g., feature engineering) can result in complex interactions with other components, so it is important to design a "pipelined" system consisting of loosely interacting components. Each component in the pipeline addresses a specific ML subproblem and can be implemented using one or more techniques. This pipeline articulates and restricts the search space, defining the possible ways to integrate different techniques. By doing so, we can allocate most of our computational resources to explore combinations of more promising techniques and limit the resources spent on tuning less effective ones. We first provide an overview, and then describe each component in detail.

4.1 Overview

Figure 4 provides a visual representation of our pipeline. Initially, raw time-series instances undergo processing in the **feature extractor**, generating 65-dimensional hand-crafted features. This process yields target features $\tilde{\mathbf{x}}_T^j = h(\mathbf{x}_T^j)$ and source features $\tilde{\mathbf{x}}_i^j = h(\mathbf{x}_i^j)$, with $i \in [n]$. Our domain adaptation algorithm is structured around three key components (C1-C3): **C1. Stratified Normalization:** This component, vital for Multi-Source Domain Adaptation (MSDA), normalizes features from diverse domains to a consistent scale. This step is particularly crucial for datasets exhibiting significant shifts in marginal distributions across domains.**C2. Multi-Source Domain Adaptor:** Comprising a shared layer $g(\cdot)$ and a set of n classifiers $\{d_1(\cdot), \ldots, d_n(\cdot)\}$, this component



(a) Vanilla normalization (b) Stratified normalization

Fig. 5. **Visualization (t-SNE) of normalized data.** Points in • and △ are observations from two different users. Different colors represent different classes. Centroids of blue points from different users are highlighted (by bolder marks). (a) uses vanilla normalization, whereas (b) uses stratified normalization (S-Norm). Both $\Pr[\tilde{X}_i]$ (points in the same shape) and $\Pr[\tilde{X}_i \mid Y_i]$ (i.e., the cloud of points in the same color from the same user) get much closer for two users using S-Norm.

manages the delicate balance between model robustness and diversity. The shared layer, $g(\cdot)$, extracts robust features across domains, minimizing divergence. Each classifier, $d_i(\cdot)$, is individually optimized to learn labels from its respective source domain. Specifically, each branch $d_i(\cdot)$ outputs p_i for the source domain, feeding it into cross-entropy loss $\mathcal{L}_{cls}$ to train the classifier. Simultaneously, q_i is produced from the target domain, and the pairs (q_i, p_i) are utilized in the maximum mean discrepancy loss $\mathcal{L}_{mmd}$ to reduce source-target divergence. Independence among the branches ensures diverse predictions, crucial for effective downstream ensembling. Furthermore, p_i outputs contribute to the consensus regularization $\mathcal{L}_{con}$ to mitigate source-source divergence. **C3. Adaptive Ensemble Learner:** Treating each output from d_i as an ensemble, this component determines suitable ensemble weights by leveraging $\mathcal{L}_{mmd}$ and $\mathcal{L}_{con}$. These weights are dynamically adjusted, assigning greater significance to sources more akin to the target.

4.2 Hand-crafted feature extraction

We follow the approach [55] to construct a total collection of 65 features optimized for building food-type recognition models. See Table 2. We let $h(\cdot)$ be the feature engineering procedure so that $h(\mathbf{x}) \in \mathbf{R}^{65}$. Recall that $\tilde{\mathbf{x}} = h(\mathbf{x})$, and we also let $\tilde{X}_i = \{h(\mathbf{x}_i^j)_j\}$ ($i \in [n] \cup \{T\}$, and $j \in [s_i] \cup \{T\}$). Our pipeline critically relies on hand-crafted features, which are more robust for food-typing tasks and departs from a recent "fashionable" trend that aims to use a neural network to learn features automatically [57]. See also Section 4.6.

4.3 Stratified normalization

Machine learning algorithms often assume that the data in training and test sets are from the same distribution, which is severely violated in our setting. First, each user could wear devices in slightly different ways. Second, people have different chewing habits. For example, when user i eats faster than user j, i's chewing time will be shorter, but his or her chewing force will be stronger. Therefore, $\mathcal{D}_i$ and $\mathcal{D}_j$ could be drastically different.

Traditional normalization re-scales the input features across sources to have uniform standard deviations and means, which is ineffective in our setting. Figure 5a shows data collected from two domains, i and j (users). After normalizing the training data, the data still shift between $\mathcal{D}_i$ and $\mathcal{D}_j$. The problem will become more pronounced when the number of sources grows.

To address this challenge, we introduce a simple yet effective domain adaptation technique named stratified normalization (S-Norm). S-Norm draws inspiration from stratified sampling, a method developed for sampling from multiple subpopulations. It performs normalization independently for each $\tilde{X} = \{\tilde{X}_1, ..., \tilde{X}_n, \tilde{X}_T\}$. S-Norm

serves two primary purposes: *(i)* aligning features from different domains to the same scale, ensuring $\Pr(\tilde{X})$ is well-aligned (see Figure 5b). *(ii)* enhancing the alignment and ease of learning of conditional distributions $\Pr[\tilde{x}_i|y_i]$ for $i \in [n] \cup \{T\}$ across different domains. For instance, users often chew nuts faster than ice-creams, making it simpler to extract this signal under stratified normalization.

4.4 Multi-source domain adaptation

The multi-source domain adaptation takes re-scaled features as input and outputs a total number of n predictions (ensembles). It possesses a branching structure, consisting of a "root" component $g(\cdot)$ and a total number of n branches $d_i(\cdot)$ ($i \in [n]$). Each $d_i(\cdot)$'s and $g(\cdot)$ has the linear-ReLu-linear structure. All the training data from different domains first flow into $g(\cdot)$ simultaneously, and afterward, they are branched out to different $d_i(\cdot)$'s. A $d_i(\cdot)$ consumes labeled data from source domain i and unlabeled data from the target domain. Intuitively, $g(\cdot)$ aims to extract features that are robust across all domains, whereas $d_i(\cdot)$ aims to train an augmented model for $\Pr[y_i \mid \tilde{\mathbf{x}}_i]$ that approximates $\Pr[y_T \mid \tilde{\mathbf{x}}_T]$, i.e., learning the link function for the target based on link function for the source i as well as unlabeled target data.

Next, we explain how we implement this idea. We view the domain adaptor as sending $\tilde{X}_i$ and $\tilde{X}_T$ to an embedded space. We apply two techniques to learn $g(\cdot)$ and $d_i(\cdot)$'s.

- *Technique 1. Properly construct embedded space.* Intuitively, we aim to make sure after we apply $g_i(\cdot)$ to $s(\tilde{\mathbf{x}}_i)$'s and $g(\tilde{\mathbf{x}}_T)$'s, these two "clouds" have similar distribution in the embedded space.
- *Technique 2. Shrinking towards the mean.* For a fixed target T, we hope to set $d_i(g(\tilde{\mathbf{x}}_T^j))$'s for different i to be "similar" so that the total "function complexity" across all the models we learned becomes smaller, which improves bias-variance tradeoff.

We also note that both techniques provide methods for measuring similarities (or distances) between pairs of models trained from different source domains, as well as between a source domain and the target domain. The distance measures derived from these techniques will be used in the ensemble learning component, which operates outside the deep learning loop (see Section 4.5).

Construction of embedded space. Our goal is to bring the following two sets of points closer to the embedded space:

$$\{d_i(g(\tilde{\mathbf{x}}_i)) : \tilde{\mathbf{x}}_i \sim \mathcal{D}_i(X)\} \quad \text{and} \quad \{d_i(g((\tilde{\mathbf{x}}_T)) : \tilde{\mathbf{x}}_T \sim \mathcal{D}_T(X)\}.$$

We use the maximum mean discrepancy to measure the statistical distance.

Definition 4.1. *Let $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_n\}$. The **maximum mean discrepancy** (MMD) between P and Q is*

$$\mathcal{L}_{mmd}(P, Q) = \left\| \frac{1}{s} \sum_{p \in P} \phi(p) - \frac{1}{t} \sum_{q \in Q} \phi(q) \right\|_{\mathcal{H}}^2,$$

where $\mathcal{H}$ is the reproducing kernel Hilbert space (RKHS), and the $\phi(\cdot)$ denotes a feature map to map the inputs to the $\mathcal{H}$, which is achieved by a kernel $k(P, Q) = \langle \phi(P), \phi(Q) \rangle$

Our algorithm uses the Gaussian RBF kernel: $k(u, v) = \exp(-\lambda \|u - v\|^2)$ for $\mathcal{H}$. Also, let $p_i = \{d_i(g(\tilde{\mathbf{x}}_i))\}_{i \in n}$ be the feature representations for domain i at branch i, and $q_i = \{d_i(g(\tilde{\mathbf{x}}_T)\}_{i \in n}$ be the feature representations for the target at branch i. We let

$$\mathcal{L}_{mmd} = \frac{1}{n} \sum_{i \in [n]} \mathcal{L}_{mmd}(p_i, q_i).$$

L_{mmd} aggregates the distance of each pair source-target domain. By minimizing the $\mathcal{L}_{mmd}$, each domain-specific adaptor $d_i(\cdot)$ would be able to map the $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_T$ into similar representations.

Shrink towards the mean. We impose global constraints over $d_i(\cdot)$'s. We want that $d_i(\cdot)$'s to shrink towards the same function to achieve improved bias-variance tradeoffs. Specifically, if models shrink towards the same one, data would be sufficient to train a model, but the model will not be expressive enough to have reasonable forecasting power. If we do not shrink at all, there are way too many models to be learned to reduce bias and increase variance.

We leverage the **consensus regularization** [33] based on L1 distance to achieve this target.

Definition 4.2. *The $\mathcal{L}_{con}$ measures the L1 distance between the outputs of each pair of domain-specific adaptors, which can be formulated as:*

$$\mathcal{L}_{con} = \frac{1}{n \times (n-1)} \sum_{j=1}^{n-1} \sum_{i=j+1}^{n} \sum_{\tilde{\mathbf{x}}_T \in \mathcal{D}_T(X)} |d_i(g(\tilde{\mathbf{x}}_T)) - d_j(g(\tilde{\mathbf{x}}_T))|.$$

The effectiveness of consensus regularization in alleviating over-fitting is analyzed in Section 5.3.2. In an alternative view, the consensus regularization reduces the domain divergence of each pair of the source-source domain, and the MMD reduces the domain divergence of each pair of the source-target domain.

We use $\mathcal{L}_{mmd}$ and $\mathcal{L}_{con}$ developed above together with the standard cross-entropy loss $\mathcal{L}_{cls}$ to construct the final loss function. Recall that

$$\mathcal{L}_{cls} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{s_i} J(C(d_i(g(\tilde{\mathbf{x}}_i^j))), y_i^j),$$

where $C(\cdot)$ is the Softmax function, and $J(\cdot, \cdot)$ is the cross-entropy loss function. Our final goal is to minimize the following loss function:

$$\mathcal{L} = \lambda \mathcal{L}_{mmd} + (1 - \lambda)\mathcal{L}_{con} + \mathcal{L}_{cls},$$

where λ is a constant value to balance two loss functions and we set it to 0.5. We remark that specific details of our cost function can be tweaked. For example, $\mathcal{L}_{mmd}$ can be replaced by CORAL [49, 50] or GAN-based loss, whereas ℓ_2-loss can replace $\mathcal{L}_{con}$. Our experiments find that making these minor changes does not result in additional performance gain.

4.5 Adaptive ensemble learning

4.5.1 Ensemble learning. This section explains our ensemble learning procedure. Our consolidated forecast is a linear combination of n ensembles

$$\tilde{g}(\tilde{\mathbf{x}}_T) = \sum_{i=1}^{n} \omega_i d_i(g(\tilde{\mathbf{x}}_T)), \tag{1}$$

where ω_i's are uniform weights, i.e., $\omega_i = \frac{1}{n}$, for $i \in n$, or adaptive weights. Uniform weighted consolidated prediction is the default option for ensemble learning. We develop an adaptive weight assignment technique to achieve better prediction consolidation. Note also that we use a standard convention to represent the output of classifiers, i.e., $d_i(\mathbf{x})$ outputs a probability measure in $\mathbf{R}^m$, where m is the number of categories and the j-th coordinate/component in the output represents the probability that the output is in category j (estimated by d_i). Therefore, $\tilde{g}(\cdot) \in \mathbf{R}^m$. In evaluation, we assume that $\tilde{g}(\cdot)$ picks up the category with the highest probability estimate.

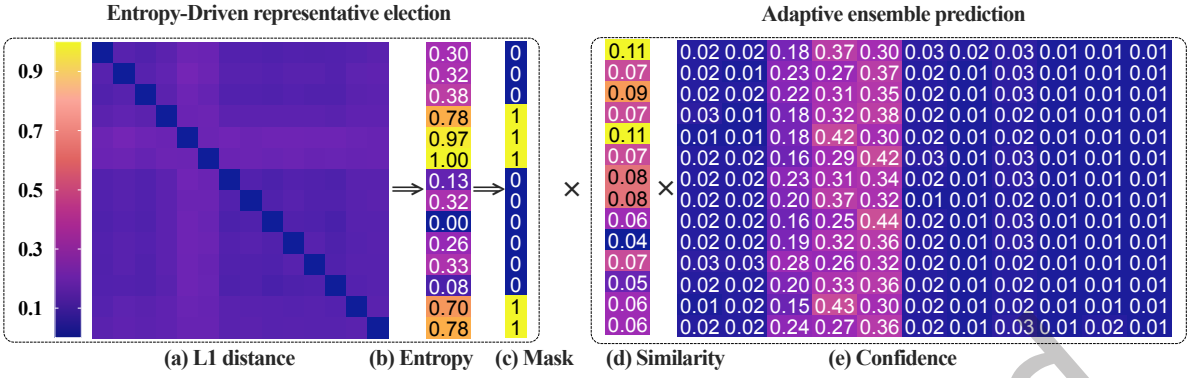


Fig. 6. **The adaptive ensemble learning procedure. (a) Distance Matrix** A**:** A is symmetric. Each element $A_{i,j}$ represents the L1 distance between $d_i(g(\tilde{\mathbf{x}}_T))$ and $d_j(g(\tilde{\mathbf{x}}_T))$. Note that when $i = j$, $A_{i,j} = 0$. **(b) Entropy Vector:** Each element is the entropy of the corresponding row of A. The displayed values are min-max standardized. **(c) Mask Vector** ω_m**:** Element 1's represent the corresponding domain-adaptors as representatives. 0's represent redundancy. **(d) Similarity Vector** ω_s**:** ω_s is transformed from the outputs of $\mathcal{L}_{MMD}$ to measure the similarity of each source domain to the target domain. **(e) Confidence Matrix:** Each row is produced by the $C(d_i(g(\tilde{\mathbf{x}}_t)))$ to represent the prediction confidence over the labels.

4.5.2 Adaptive Weight Assignment. Source domains have different approximations of the target domain. Assigning a higher weight ω_i to a domain-specific model i benefits consolidated forecasting. Therefore, our goal is to learn ω_i *without* labels from the target domain.

Design intuition. Our algorithm for determining ω_i's need to *(i)* utilize the observation that having more ensembles is not always better, and *(ii)* ensure that ω_i's dynamically change according to the target; using static target-oblivious weights is ineffective.

When labels are available, estimating ω_i is a simple regression problem. Here, we build our solution by unwinding key intuitions of solving a (possibly over-parametrized) linear regression and "re-implement" these intuitions in the no-label setting by using information-theoretic tools. Let us first briefly review the linear regression problem. Recall that the (ordinary least squares) OLS coefficient estimator is $(XX^{\mathrm{T}})^{-1}X^{\mathrm{T}}y$, which consists of a feature-feature interaction (source-source interaction in our setting) component $(XX^{\mathrm{T}})^{-1}$, and a feature-response (source-target) interaction component $X^{\mathrm{T}}y$. Commonly used regularizations often "shrink" XX^{T} to control variance-bias tradeoffs. For example, ridge regression shrinks XX^{T} towards identity, whereas principal component regression (PCR) shrinks XX^{T} towards low-rank matrices.

We mimic the linear regression and design two subroutines to capture source-source interactions and source-target interactions. The component using source-source interactions prunes away ineffective models, resembling pruning away inconsequential subspaces in PCR, whereas the component using source-target interactions further fine-tunes model weights.

1. Source-source interaction: entropy-driven representative election. Here, our goal is to identify a subset of orthogonal signals that resemble variable selections in PCR. Recall that PCR selects a subset of orthogonal vectors as regressors. We aim to generalize the notion of orthogonality but the models $d_i(\cdot)$'s are non-linear so standard PCR techniques do not work. Instead, we re-utilize the consensus regularization measures introduced in Section 4.4. Recall that the consensus regularization defines the distance between each pair of $d_i(g(\tilde{\mathbf{x}}_T))$'s, which outputs a $n \times n$ symmetric similarity matrix A (Figure 6 (a)), where

$$A_{i,j} = \sum_{\tilde{\mathbf{x}}_T \in \mathcal{D}_T(X)} |d_i(g(\tilde{\mathbf{x}}_T) - d_j(g(\tilde{\mathbf{x}}_T))|.$$

We prefer a model i whose distances to other models are uniform (i.e., generalizing orthogonality), and we use entropy to measure the orthogonality. Specifically, let the entropy associated with model i be $\sum_{j\in[n]} -A_{i,j}\log A_{i,j}$, which is maximized when $A_{i,j}$ are the same for different j's. We then choose the models with the largest entropy (either using the top-k rule or through thresholding after min-max standardization). A mask vector $\omega_m \in \mathbf{R}^n$ is generated to indicate whether each individual model is selected. See Figure 6 (b) and (c). Both top-k and thresholding rules require a pre-defined hyper-parameter that controls the aggressiveness of the pruning operation. Section 5.3.3 presents a parameter sensitivity analysis to show the robustness of our method.

2. Source-target interaction: adaptive ensemble prediction. We next explain how we use source-target interaction. The MMD distance measures the distances $d_i(g(x_i))$ and $d_i(g(x_T))$ in the RKHS. I.e., the distance between the i-th source domain to the target domain. We leverage the MMD distance to generate the similarity vector ω_s to assign more weights to the more similar domain-specific adaptor (Figure 6 (d)). Compounding with the ω_m, we obtain the weight vector $\omega = \omega_m \cdot \omega_s$. Then the confidence matrix (Figure 6 (e)) is retrieved to forecast the label for $\tilde{\mathbf{x}}_T$ according to Equation 1.

We remark that *(i)* our procedure depends on the target's features (both ω_m and ω_s) so the final weights dynamically adapt to the structure of the target features. *(ii)* Existing adaptive ensemble methods require an additional parameter that is updated during the training phase [27], whereas ours does not interfere with the training procedure. Instead, it is a post-training method that directly adjusts the ω_i based on the loss function.

4.6 Discussions

Our algorithm incorporates a function $h(\cdot)$ crafted by domain experts to transform features, utilizing manually built features. In contrast, existing works [57, 64] often leverage deep learning to autonomously learn $h(\cdot)$. This design decision is not arbitrary; rather, it stems from a discerned universal phenomenon prevalent in classification problems involving the analysis of fine-grained muscle movements. Our conjecture posits that deep learning models face challenges in learning semantically meaningful intermediate features crucial for accurate responses in our domain. This conjecture is substantiated by comparing our problem with vision/NLP problems. In vision/NLP tasks, deep learning models excel at identifying interpretable local patterns in lower layers, which are then utilized for making predictions. For instance, convolutional layers in vision models extract local texture patterns, while NLP models discern words/tokens with similar meanings. However, in our setting, deep learning proves less effective in learning useful 'local' features from raw time-series sensor data [11].

To validate our conjecture, we design additional experiments where we task neural networks with predicting a set of seemingly 'simpler' tasks using raw data. If our conjecture held true, deep learning models would struggle to predict these tasks. We define the 'simple tasks" as manually built features, including simple statistics such as the number of chews or the duration of each chew. We select two deep-learning models to predict the 65-dimensional hard-crafted features. The first model comprises two LSTM layers with a dropout rate of 0.5, followed by a fully

Table 2. **We examine the ability of DL models to learn all 65 features.** Left/right gyroscope and accelerometer sensors are abbreviated to LG, RG, LA, and RA. Features 1-7 are statistics of chewing speed and duration; 8-9 are the magnitude of translation and rotation; 10-23 are the number of mean-crossing, entropy/energy of frequency spectrum, maximum frequency component, and statistics of spectrum component, 24-27, 38-51, and 52-65 extract the same features as 10-23 but on different sensors.

Target features	1-7	8	9	10-23	24-37	38-51	52-65	1-65
Sensors	LG, RG	LA, RA	LG, RG	LA	RA	LG	RG	All
LSTM Avg. R^2	0.197	0.003	0.002	0.261	0.374	0.418	0.349	0.372
CoDATs Avg. R^2	0.263	-0.038	-0.004	0.6231	0.6138	0.5549	0.5324	0.482

connected layer for prediction. The second model, CoDATs, mirrors the original implementation but adjusts the last fully connected layer based on the number of features. Both models employ mean squared error (MSE) loss and the Adam optimizer [24] with a learning rate of 0.5e-3. We split the dataset into 2100 training samples and 608 test samples. We fix each time-series sample to a length of 1024 by padding zeros or truncating and use a batch size of 128. We use R^2 metric to measure the regression performance.

Table 2 illustrates that deep learning models struggled to accurately predict hand-crafted features, even with meticulous parameter tuning. While it might be feasible to fine-tune a neural network for predicting specific features, using a single neural network architecture to predict a substantial portion of hand-crafted features appears challenging. This underscores a fundamental difference between our problem and vision/NLP problems, where a single architecture can typically extract a diverse set of 'local features' such as various textures or the meanings of many words."

Conclusion: The features $\tilde{\mathbf{x}}_i$'s and $\tilde{\mathbf{x}}_T$ originate from vastly different distributions, presenting a formidable challenge even for simple responses, such as the 65 extracted features. In light of this, mastering effective transfer learning in the food typing task remains a complex endeavor. Our investigation strongly suggests that, given the current landscape, relying on manually-built features proves to be a more efficacious approach. This observation aligns with recent empirical findings [11], further emphasizing the ongoing difficulty in leveraging automated methods to bridge the gap between diverse feature distributions.

5 EVALUATION

5.1 Evaluation Methodology

This section evaluates our proposed domain adaptation method for the task of food typing. Specifically, we show that our algorithm outperforms state-of-the-art baselines significantly. We also perform extensive experiments including ablation studies to analyze the roles and efficacy of different components in our pipeline. Moreover, we perform open set MSDA evaluations to test the extensibility of our methods.

In our experimentation, we employ a rigorous evaluation technique known as LOOCV, which stands for Leave One User Out Cross-Validation. LOOCV is a specialized form of cross-validation where the model is trained on all users except one, and the excluded user serves as the target domain for validation. This process is iteratively repeated until each user has been left out exactly once. LOOCV provides a robust assessment of the model's generalization performance, especially in scenarios where user-specific characteristics play a significant role.

Table 3. **Data distribution of** `food-15`. The dataset includes 15 users and spans 11 food categories, comprising a total of 2708 samples. Individual users contributed samples ranging from 144 to 197, and each food type has 63-544 samples. Notably, User10 lacks samples for vegetables, and User11 has no samples for gum.

	Nuts	**Gum**	**Dry Fruit**	**Fruits**	**Pretzel**	**Corn/Fry**	**Cookie**	**Vegetable**	**Bread**	**Meat**	**Cream**	**Total**
User1	30	10	30	38	10	30	10	10	10	10	8	**196**
User2	30	10	29	30	10	21	10	10	10	10	1	**171**
User3	30	10	30	40	10	30	10	10	10	10	4	**194**
User4	30	10	30	39	10	28	10	10	10	9	8	**194**
User5	29	10	29	38	6	24	10	9	10	9	4	**178**
User6	30	10	30	40	10	27	9	9	8	10	4	**187**
User7	30	10	30	30	10	29	10	9	10	10	2	**180**
User8	30	10	30	40	10	27	10	10	10	10	8	**195**
User9	29	10	30	36	9	29	10	10	10	10	3	**186**
User10	29	10	30	36	10	30	10	0	10	10	6	**181**
User11	10	0	27	39	10	28	10	10	10	10	2	**156**
User12	28	8	27	38	9	21	10	10	10	1	1	**163**
User13	30	6	18	23	9	18	10	9	10	10	1	**144**
User14	29	10	30	37	10	28	10	10	9	10	3	**186**
User15	30	10	30	40	10	30	10	10	9	10	8	**197**
Total	**424**	**134**	**430**	**544**	**143**	**400**	**149**	**136**	**146**	**139**	**63**	**2708**

5.1.1 Dataset. We use a standard benchmark human chewing datasets introduced in [55], namely `food-15`. Comprising data from 15 participants, each engaging with up to 20 distinct types of food, this dataset captures chewing activities via gyroscope and accelerometer sensors. For consistency and improved interpretability, we adopt the categorization scheme proposed in [55], condensing the 20 food types into $m = 11$ categories. The data summary is detailed in Table 3. This categorization proves advantageous for two primary reasons. Firstly, from a clinical perspective, predicting food categories is often more meaningful than predicting individual types. Secondly, the variation in participants' dietary habits, such as one individual consuming almonds while another opts for peanuts. Both almonds and peanuts, fall under the "Nuts" category. Predicting unseen labels (food types) for the target users is a non-scope in this work.

Table 4. **Comparison of our method (`Ours`) and nine state-of-the-art baselines.** A1.Data represents manipulating data to align to different domains without innovating model architecture/loss function, and A2.Model represents a method that has innovated model architecture/loss function, see Section 3. Domain knowledge-guided feature extraction means a baseline using hand-crafted features. Data-driven feature extraction methods use neural nets to learn representations on raw time series data.

Work	Feature extraction	Domain-invariant feature transform	Adaptation target	Ensemble scheme
Single-source domain adaptation				
1.CORAL [49]	Domain knowledge	A1. Data	Source-Target	Source-combined
2.TCA [38]	Domain knowledge	A1. Data	Source-Target	Source-combined
3.DANN [14]	Domain knowledge	A2. Model	Source-Target	Uniform weight
Multi-source domain adaptation				
4.DARN [56]	Domain knowledge	A2. Model	Source-Target	Dynamic weight
5.MDMN [27]	Domain knowledge	A2. Model	Source-Target, Source-Source	Dynamic weight
6.M3SDA [39]	Domain knowledge	A2. Model	Source-Target, Source-Source	Model accuracy weight
7.MDAN [60]	Domain knowledge	A2. Model	Source-Target	Dynamic weight
8.MuLANN [45]	Domain knowledge	A2. Model	Source-Target	Source-combined
9.CoDATs [57]	Data-driven	A2. Model	Source-Target	Source-combined
Ours	Domain knowledge	A1. Data, A2. Model	Source-Target, Source-Source	Dynamic weight

5.1.2 Baseline Methods. We examine a wide range of baselines, including a domain expert model without domain adaptation [55], marked as No-Ada, three single-source domain adaptation methods: CORAL [49], TCA [38], DANN [14], and six multi-source domain adaptation methods: DARN [56], MDMN [27], M3SDA [39], MDAN [60], MuLANN [45], CoDATs [57]. Section 2 reviews these baselines. Table 4 also compares their key characteristics against our algorithm. CoDATs [57] distinguishes itself by utilizing raw time series datasets from sensors. In contrast, the remaining baselines were originally devised for recognition problems other than food type, such as vision and natural language processing. It is not obvious how we can effectively pipe an architecture for time series with these solutions. Thus, we feed the 65-dimensional hand-crafted features to these baselines. To maintain consistency in our experiments, we employ the same set of neural network hyperparameters (e.g., number of hidden nodes, number of layers) for both baselines and our proposed method. This practice enables us to control the impact of model complexity, mitigating concerns of overfitting or underfitting. Adapting single-source domain adaptation methods to the multi-source setting introduces additional challenges. For No-Ada, CORAL, and TCA, we merge all source data to create a large joint source dataset as the training data. In the case of DANN, we adhere to its single-domain scheme by training individual models on each source-target pair and

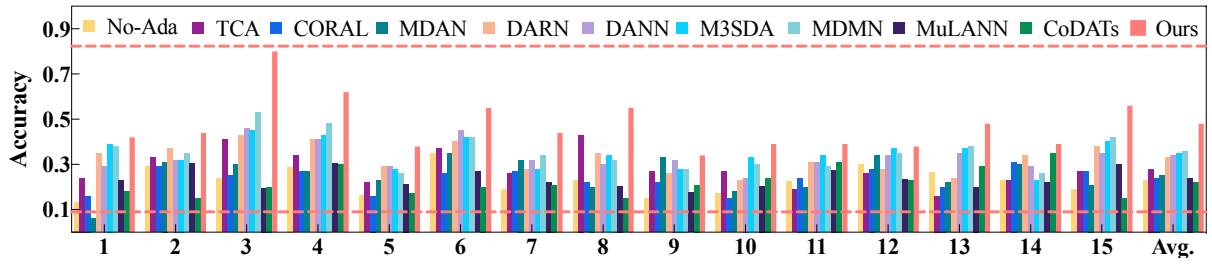


Fig. 7. **Evaluating accuracy of our method (`Ours`) and nine state-of-the-art baselines.** Here, the upper bound (MLP model trained with labeled target domain data) is 82.3%, and the lower bound (LSTM model learns source-combined data without domain knowledge) is 9% on average, drawn as two horizontal dashed red lines.

subsequently ensembling the models with uniform weights. This approach aligns with methodologies applied in prior works [39, 56, 60].

5.1.3 Lower and upper bounds. To put the numbers into context, we also present lower and upper bounds of domain adaptation performance for this dataset. The lower bound is defined as the performance of a "strawman" model, in which all data from different sources are used to train one single model, and the model is used to predict data from an unseen target, a.k.a., a "no adaptation" solution. This is the default approach for ML modeling. The upper bound is constructed as training a model *with the knowledge* of the target's labels (i.e., a "target only" model). The upper bound represents the behavior of typical ML models in an (excessively) ideal world, whereas the lower bound represents the performance of a model one would expect from a typical practitioner. The gap between the lower and upper bounds reflects the performance surprise and is a good indicator of the "difficulties" of our domain adaptation problem.

The lower bound trains an LSTM model on source-combined data without domain knowledge. The upper bound approximation uses the MLP model from Wang et al. [55] because it outperforms other model families.

5.2 Overall accuracy

Figure 7 compares the average accuracy of our method and baseline methods on each target domain of the food-15 dataset, respectively. Our method outperforms baseline methods in *each* target domain with a 1.33× to 2.13× accuracy improvement. We also note while in general domain adaptation in food-type prediction problems is remarkably difficult, performance on certain targets (e.g., participant 3) is quite close to the upper bound (a promising sign). It remains an interesting open problem to understand when a participant is easy to predict.

The two methods that do not utilize source adaptation techniques, No-ada and LSTM, perform 22.8% and 38.5% worse than our approach, respectively, confirming the effectiveness of domain adaptation in predicting food types for unseen users. The LSTM learns features from raw time-series data, while No-Ada relies on domain expert features, highlighting the robustness of manually extracted features in handling domain shifts. Moreover, compared with CoDATS [57] (CNN on raw time series), our method extracts features with expert knowledge and is 25.2% better on average, confirming the intuition of preferring not to use raw data (Section 4.6). TCA [38] and CORAL [49] use only standard neural net (NN), and are 19.2% and 23.8% worse than us, respectively. Therefore, engineering NN is essential. MDMN [27] and M3SDA [39] originally designed for computer visions, optimize source-source and source-target divergence simultaneously and are the best baselines, also confirming the importance of reducing source-source divergences (Section 3). They are nevertheless 11.8% and 12.7% worse than us, respectively. They do not have stratified normalization or adaptive ensemble learning. The cost

functions and architectures in our NN are also different. We are 13.9% and 23.7% better than DANN [14] and MuLANN [45], which do not directly address source-source divergence. We are also 14.7% and 22% better than DARN [56] and MDAN [60], respectively, which couple ensemble-weight learning with deep learning (i.e., ensemble weights are part of the network, which could potentially impact DL training in an adversarial manner), whereas we take a two-staged approach (i.e., learning the ensemble weights after training the multi-source domain adaptor).

To delve deeper into the performance of the proposed food typing methods in the face of domain divergence challenges, we analyze the confusion matrix as depicted in Fig. 8. Notably, certain classes exhibit a high degree of ease in being classified into each other. For instance, classes such as 'Nuts' and 'Dry Fruit' or 'meat' and 'bread' seem to be easily confused, as evidenced by the relatively high numbers in the corresponding off-diagonal elements. This suggests a potential similarity or overlap in the features that the model uses for classification between these pairs of classes. Recognizing food types through chewing behavior across different users remains a challenging task. However, the presented work represents a significant stride toward a promising solution.

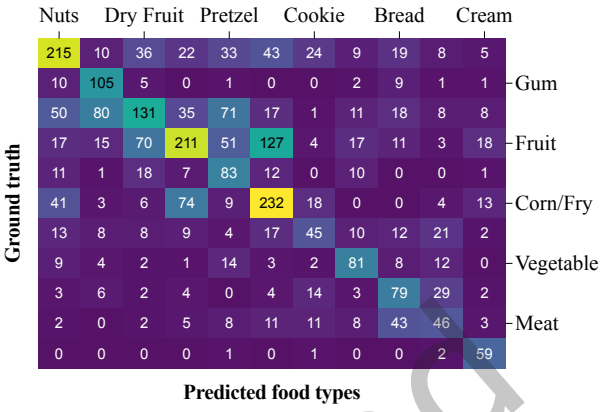


Fig. 8. **Sum of the confusion matrix over the 15 users.** Each row in the matrix represents the true class, and each column represents the predicted class, the prediction and ground truth labels are annotated. The diagonal elements represent the correctly classified instances for each class, while off-diagonal elements indicate misclassifications.

Our average accuracy is 47.5%, insufficiently close to the "productization" level. Note also that there are a total number of 11 classes, so a "null" model has a 9% accuracy. The problem we face appears to resemble model development for ImageNet [10], which requires multi-year effort to engineer a fully effective model (the most accurate model three years after ImageNet's inception is only slightly over 60%. [25]). We remark that strawman's (lower bound) performance is 9%, the best baseline is 35.7% (after substantial hyper-parameter tuning), and ours is 47.5%, which represents a 5.28 multiplicative improvement. Our "delta" with the strawman is 1.44 times stronger than the delta of the best baselines and the strawman.

5.3 Ablation Study

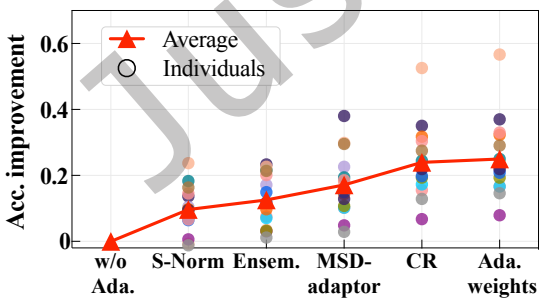
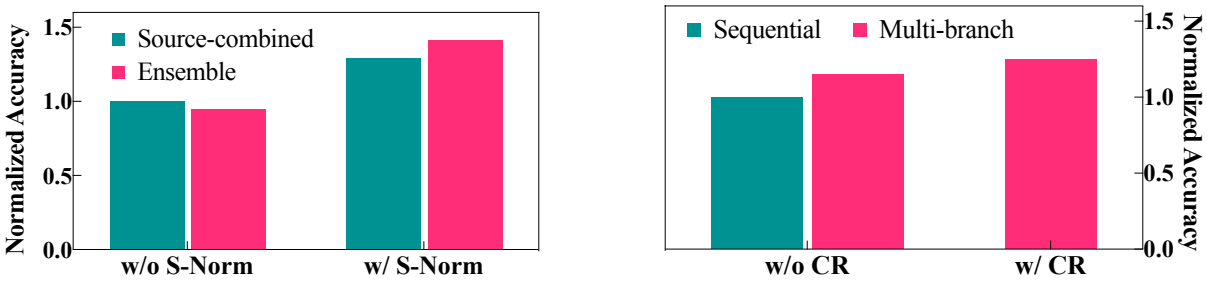


Fig. 9. **Ablation study of our methods.**

Three components are vital to our model, including stratified normalization, architectures with multi-source domain adaptation (including consensus regularizer), and adaptive ensembling. This section performs ablation studies to examine each component's effect. Note that the maximum mean discrepancy (MMD) technique handles source-target interaction and is widely deployed in modern DA algorithms so we do not specifically examine this component for conciseness.

Overview. Figure 9 presents a bird's eye view of model performance after progressively adding each of the components. Specifically, the red line represents the performance evolution on average after adding the components,



(a) **Ensemble and S-Norm.** (b) **Multi-branch and Consensus regularize (CR).**

Fig. 10. **Normalized accuracy comparison with and without S-Norm, Ensemble, MSD-Adaptor, and CR.**

whereas dots in different colors represent the performance of individual users. After adding `S-Norm`, the model improves by 9.66% on average. Then we add uniform ensemble learning (`Ensem.`), which multi-source domain adaptation techniques assume, the model further improves by 2.8%. Next we add multi-source domain adaptation (`MSD-adaptor`), which consists of two steps, including first adding all techniques except for the consensus regularization (`CR`), and then adding consensus regularization. We can see that the gross performance improvement is 11.55% whereas the consensus regularization on its own contributes 6.85% improvement. Finally, we add adaptive weights (`Ada.weights`) to replace uniform weights in ensemble learning. We can see that while the average performance improvement is moderate, they are more powerful for some targets (and never result in worse performance). The next part further examines/interprets each individual technique in detail.

5.3.1 Interplay between Stratified Normalization and Ensemble Learning. Ensemble learning is the de facto design to address the MSDA problem [51]. To study the interplay between `S-Norm` and ensemble learning, we compare the following settings: uniform weight ensemble learning with or without S-Norm, and combining data from multiple domains to train one model, a.k.a., a "source-combined" model, with or without S-Norm. Figure 10a shows the comparison results. It proves that `S-Norm` substantially improves model accuracy by 1.3× even without being integrated with ensemble learning. Integrating S-Norm with ensemble learning further enhances the accuracy by 1.41×. It is interesting that applying ensemble learning only without S-Norm results in an accuracy that is slightly worse than the non-ensemble version.

5.3.2 Multi-source Domain Adaptor and Consensus Regularizer. **All other techniques without consensus regularizer.** A substantial body of prior works has demonstrated the efficacy of multi-branch architecture (i.e., hard parameter sharing layers reduce risks of overfitting [9, 43]) and the MMD [48, 53] cost function. Thus, it may not be surprising that these techniques continue to work in our problem. The more worthwhile point is that these techniques are *additive*, i.e., they can be integrated seamlessly with other innovations in our pipeline. To study the effectiveness of a multi-branch structure, we compare it with a sequential MLP model. As Figure 10b shows, the multi-branch structure model achieves higher accuracy than the sequential model. Most importantly, a multi-branch model facilitates the integration of other domain adaptation designs, i.e. consensus regularize.

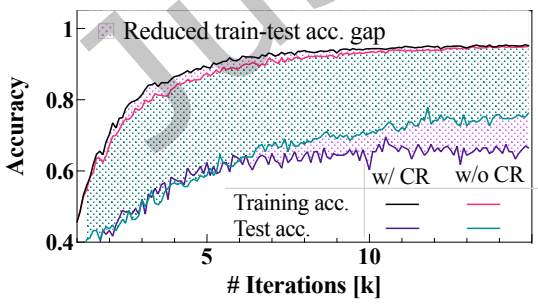


Fig. 11. **The efficacy of CR in alleviating overfitting.**

Consensus regularization. We further perform convergence analysis for training and test sets to inspect the impact

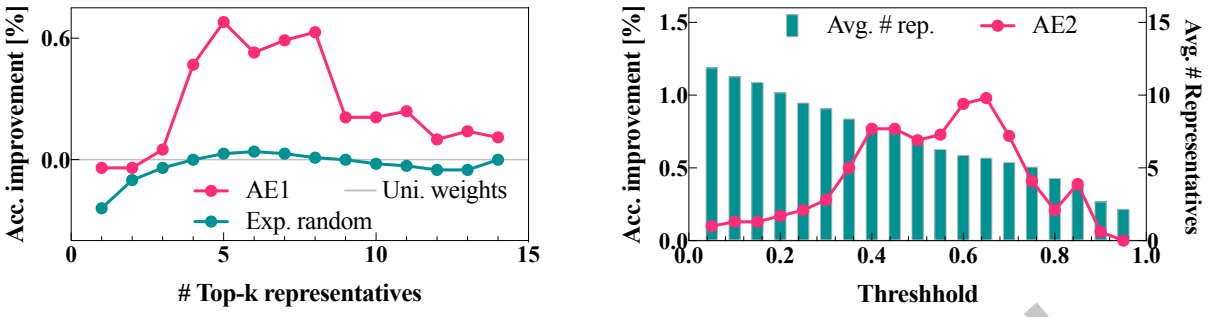


Fig. 12. **Parameter sensitivity analysis of adaptive weight selection in ensemble learning.** (a) AE1 selects top-k representatives that have the largest entropy value, where $k \in \{1, \ldots, n\}$. (b) AE2 selects the representatives through thresholding. The domain-specific adaptors that have a larger entropy value than the threshold will be elected as representatives. The left and right axes show the interaction between the threshold and the number of representatives. We iteratively set the threshold parameter from 0.05 to 0.95 with a step of 0.05 to test the accuracy. Both methods are robust to hyper-parameter settings and have accuracy over the default ensemble method that assigns uniform weights to all domain-specific adaptors.

of consensus regularizers. See Figure 11 for the training and test accuracy using models with and without consensus regularizer. One can see that *(i)* the consensus regularizer slows down the training process but eventually coincides with the model without the regularizer. *(ii)* the test performance continues to improve over time even when the training errors stall. These observations resemble behaviors of AdaBoost [12], and it is an interesting open problem to understand how they are connected. In addition, the reduction in the training-test gap highlights the consensus regularizer's efficacy in alleviating the overfitting problem.

5.3.3 Adaptive Weights Assignment in Ensemble Learning. The ensemble learning module includes an optimization that adaptively assigns a weight to each domain-specific adaptor. Each adaptive weight consists of a 0-1 mask vector ω_m that decides if this domain-specific model is included in the ensemble and a fine-tuning vector ω_s that measures the similarity between a source domain and the target domain in the embedded space. As aforementioned (in Section 4.5.2), our method offers two ways to determine the value of ω_m: either a top-k strategy (`AE1`, AE is short for Adaptive Ensemble) or a threshold-based strategy (`AE2`).

Figure 12a shows that our top-k strategy (i.e., `AE1`) consistently outperforms the default uniform weight assignment and random selection in accuracy by an average value of 0.28% and 0.6%, respectively, for each eligible parameter setting. Figure 12b studies the interaction between threshold value selection (in `AE2`) and the number of representatives (i.e., selected domain-specific models). As Figure 12 shows, a higher threshold results in more aggressive pruning, i.e. fewer representatives participate in predicting. This result also shows that selecting a middle-range value of the threshold achieves the optimal improvement (1%) over the uniform weight assignment. Users can select either `AE1` or `AE2` for their application (and dataset) based on a similar empirical study.

5.4 Open Set Multi-source Domain Adaptation

Prior experiments assumed an ideal condition where each domain consumes the same type of food (except for user 10 and user 11, refer to Table 3). This setting is commonly used by the research community [51, 61]. However, such a condition may not always exist in the real world. In this section, we address a more challenging scenario where each source domain only provides partial food types. This is formally defined as the open set MSDA problem [61], where $y_i \cap y_T \subset y_T$. This setting is more challenging because the total amount of labeled data is reduced, and the target domain can only learn a few food types from one specific user and other food types from

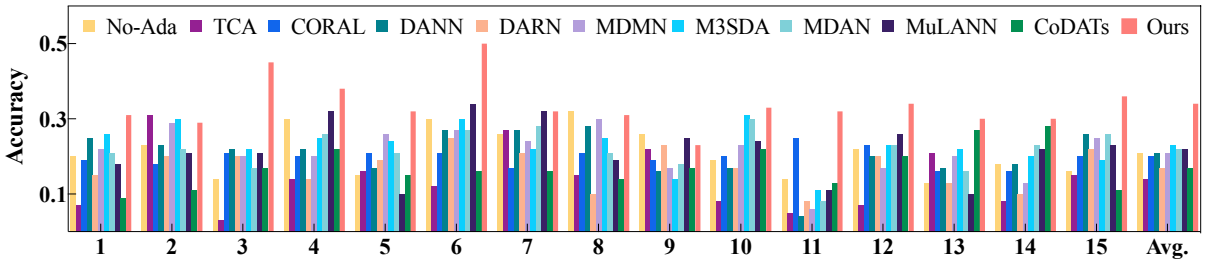


Fig. 13. **Evaluating the accuracy of our method (`Ours`) and baselines under the open set configuration.** Each user provides approximately 50% of the labels.

different users. Such a situation could occur in real-world applications when extending the model to recognize more food types, as new data may need to be provided by different users.

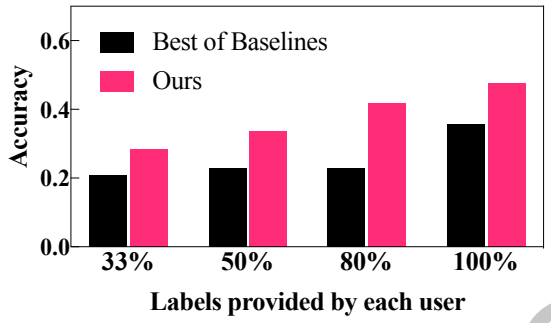


Fig. 14. **Breakdown study of the open set problem.** Use different proportions of labeled data to compare our method with the best-performing baseline method.

To validate our methods in this setting, we modify the dataset so that each user provides partial labels. For example, user 1 provides labels with odd IDs, and user 2 provides labels with even IDs, meaning the model can only learn approximately 50% of the food types from each user. Figure 13 compares the average accuracy of our model with baseline methods under this setting. Our method outperforms the baseline methods with a 1.39× to 2.40× improvement, indicating that our approach is extensible to incorporate more food types. Figure 14 presents a detailed analysis by varying the percentage of food types provided by each source domain. Our method surpasses the best baseline by a range of 1.37× to 1.82×. Although our method experiences accuracy losses, these are attributable to the reduced data size under the open set configuration.

6 CONCLUSION AND FUTURE WORK

This work develops the first multi-source domain adaptation (MSDA) method for food typing recognition, which consists of a pipeline with three main components. First, the stratified normalization aligns the conditional and marginal distributions of features to adapt to different domains, improving accuracy by 9.66% compared with a no-adaptation baseline. Second, a multi-source domain adaptor is trained on the domain-aligned features to learn a generalizable classifier for recognizing food types, incorporating a consensus regularizer and the maximum mean discrepancy. This component further increases accuracy by 11.55%. Finally, the adaptive ensemble weight selection prunes irrelevant sub-models of the multi-source domain adaptor and fine-tunes the weights for ensembling, contributing an additional 0.68%-1% accuracy improvement. Our evaluation empirically validates the importance of the consensus regularizer and domain knowledge in providing generalizable forecasting through sensor signals. We compare our method with nine state-of-the-art baselines to evaluate accuracy improvements in both closed set and open set MSDA problems, demonstrating that our method achieves 1.33× to 2.13× and 1.39× to 2.40× accuracy improvements, respectively.

Based on the current study, our future work includes: 1) improving the model to achieve higher accuracy and recognize a greater variety of food types in both closed set and open set MSDA problems, 2) extending our method to more challenging problems, such as zero-shot MSDA, where target data are not available during training.

REFERENCES

[1] Oliver Amft. 2010. A wearable earpad sensor for chewing monitoring. In *SENSORS, 2010 IEEE*. IEEE, 222–227.

[2] Oliver Amft, Mathias Stäger, Paul Lukowicz, and Gerhard Tröster. 2005. Analysis of chewing sounds for dietary monitoring. In *International Conference on Ubiquitous Computing*. Springer, 56–72.

[3] Oliver Amft and Gerhard Troster. 2009. On-body sensing solutions for automatic dietary monitoring. *IEEE pervasive computing* 8, 2 (2009), 62–70.

[4] Sizhe An, Ganapati Bhat, Suat Gumussoy, and Umit Ogras. 2023. Transfer Learning for Human Activity Recognition Using Representational Analysis of Neural Networks. *ACM Trans. Comput. Healthcare* 4, 1, Article 5 (mar 2023), 21 pages. https://doi.org/10.1145/3563948

[5] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning* 79, 1 (2010), 151–175.

[6] Yin Bi, Mingsong Lv, Chen Song, Wenyao Xu, Nan Guan, and Wang Yi. 2015. Autodietary: A wearable acoustic sensor system for food intake recognition in daily life. *IEEE Sensors Journal* 16, 3 (2015), 806–816.

[7] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683* (2012).

[8] Keum San Chun, Sarnab Bhattacharya, Caroline Dolbear, Jordon Kashanchi, and Edison Thomaz. 2020. Intraoral temperature and inertial sensing in automated dietary assessment: a feasibility study. In *Proceedings of the 2020 International Symposium on Wearable Computers*. 27–31.

[9] Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796* (2020).

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[11] Shereen Elsayed, Daniela Thyssens, Ahmed Rashed, Hadi Samer Jomaa, and Lars Schmidt-Thieme. 2021. Do we really need deep learning models for time series forecasting? *arXiv preprint arXiv:2101.02118* (2021).

[12] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[13] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*. PMLR, 1180–1189.

[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.

[15] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2414–2423.

[16] Boqing Gong, Kristen Grauman, and Fei Sha. 2013. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *International conference on machine learning*. PMLR, 222–230.

[17] Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. 2012. Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*. Citeseer, 1205–1213.

[18] Gregory Griffin, Alex Holub, and Pietro Perona. 2007. Caltech-256 object category dataset. (2007).

[19] Jiang Guo, Darsh J Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. *arXiv preprint arXiv:1809.02256* (2018).

[20] Simone Hantke, Felix Weninger, Richard Kurle, Fabien Ringeval, Anton Batliner, Amr El-Desoky Mousa, and Björn Schuller. 2016. I hear you eat and speak: Automatic recognition of eating condition and food type, use-cases, and impact on asr performance. *PloS one* 11, 5 (2016), e0154486.

[21] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. 2006. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems* 19 (2006), 601–608.

[22] Qianyi Huang, Zhice Yang, and Qian Zhang. 2018. Smart-U: smart utensils know what you eat. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1439–1447.

[23] Wenjun Jiang, Chenglin Miao, Fenglong Ma, Shuochao Yao, Yaqing Wang, Ye Yuan, Hongfei Xue, Chen Song, Xin Ma, Dimitrios Koutsonikolas, et al. 2018. Towards environment independent device free human activity recognition. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 289–304.

[24] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (may 2017), 84–90. https://doi.org/10.1145/3065386

[26] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. 2017. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*. 5542–5550.

[27] Yitong Li, Michael Murias, Samantha Major, Geraldine Dawson, and David E Carlson. 2018. Extracting relationships by multi-domain matching. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 6799–6810.

[28] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. 2017. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036* (2017).
[29] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. 2018. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition* 80 (2018), 109–117.
[30] Chuang Lin, Sicheng Zhao, Lei Meng, and Tat-Seng Chua. 2020. Multi-source domain adaptation for visual sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 2661–2668.
[31] Miaofeng Liu, Yan Song, Hongbin Zou, and Tong Zhang. 2019. Reinforced training data selection for domain adaptation. In *Proceedings of the 57th annual meeting of the association for computational linguistics*. 1957–1968.
[32] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*. PMLR, 97–105.
[33] Ping Luo, Fuzhen Zhuang, Hui Xiong, Yuhong Xiong, and Qing He. 2008. Transfer learning from multiple source domains via consensus regularization. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 103–112.
[34] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation with multiple sources. (2009).
[35] Akhil Mathur, Anton Isopoussu, Nadia Berthouze, Nicholas D Lane, and Fahim Kawsar. 2019. Unsupervised domain adaptation for robust sensory systems. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*. 505–509.
[36] Mark Mirtchouk, Christopher Merck, and Samantha Kleinberg. 2016. Automated estimation of food type and amount consumed from body-worn audio and motion sensors. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 451–462.
[37] Konstantinos Nikolaidis, Stein Kristiansen, Thomas Plagemann, Vera Goebel, Knut Liestøl, Mohan Kankanhalli, Gunn-Marit Traaen, Britt Øverland, Harriet Akre, Lars Aakerøy, and Sigurd Steinshamn. 2022. My Health Sensor, My Classifier – Adapting a Trained Classifier to Unlabeled End-User Data. *ACM Trans. Comput. Healthcare* 3, 4, Article 48 (nov 2022), 24 pages. https://doi.org/10.1145/3559767
[38] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2010. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks* 22, 2 (2010), 199–210.
[39] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. 2019. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1406–1415.
[40] Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. (2012).
[41] Alberto Poncelas, Gideon Maillette de Buy Wenniger, and Andy Way. 2019. Transductive data-selection algorithms for fine-tuning neural machine translation. *arXiv preprint arXiv:1908.09532* (2019).
[42] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2018. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910* (2018).
[43] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).
[44] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. 2010. Adapting visual category models to new domains. In *European conference on computer vision*. Springer, 213–226.
[45] Alice Schoenauer-Sebag, Louise Heinrich, Marc Schoenauer, Michele Sebag, Lani F Wu, and Steve J Altschuler. 2019. Multi-domain adversarial learning. *arXiv preprint arXiv:1903.09239* (2019).
[46] Gabriele Schweikert, Gunnar Rätsch, Christian Widmer, and Bernhard Schölkopf. 2008. An empirical analysis of domain adaptation algorithms for genomic sequence analysis. *Advances in neural information processing systems* 21 (2008).
[47] Nur Asmiza Selamat and Sawal Hamid Md Ali. 2020. Automatic food intake monitoring based on chewing activity: A survey. *IEEE Access* 8 (2020), 48846–48869.
[48] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. 2007. A Hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*. Springer, 13–31.
[49] Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of Frustratingly Easy Domain Adaptation. In *AAAI*.
[50] Baochen Sun and Kate Saenko. 2016. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. In *ECCV 2016 Workshops*.
[51] Shiliang Sun, Honglei Shi, and Yuanbin Wu. 2015. A survey of multi-source domain adaptation. *Information Fusion* 24 (2015), 84–92.
[52] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7167–7176.
[53] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* (2014).
[54] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5018–5027.
[55] Shuangquan Wang, Gang Zhou, Jiexiong Guan, Yongsen Ma, Zhenming Liu, Bin Ren, Hongyang Zhao, Amanda Watson, and Woosub Jung. 2021. Inferring food types through sensing and characterizing mastication dynamics. *Smart Health* 20 (2021), 100191.
[56] Junfeng Wen, Russell Greiner, and Dale Schuurmans. 2020. Domain aggregation networks for multi-source domain adaptation. In *International Conference on Machine Learning*. PMLR, 10214–10224.

[57] Garrett Wilson, Janardhan Rao Doppa, and Diane J Cook. 2020. Multi-source deep domain adaptation with weak supervision for time-series sensor data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1768–1778.
[58] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. 2018. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3964–3973.
[59] Chaohui Yu, Jindong Wang, Yiqiang Chen, and Meiyu Huang. 2019. Transfer Learning with Dynamic Adversarial Adaptation Network. In *2019 IEEE International Conference on Data Mining (ICDM)*. 778–786. https://doi.org/10.1109/ICDM.2019.00088
[60] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. 2018. Adversarial multiple source domain adaptation. *Advances in neural information processing systems* 31 (2018), 8559–8570.
[61] Sicheng Zhao, Bo Li, Pengfei Xu, and Kurt Keutzer. 2020. Multi-source domain adaptation in the deep learning era: A systematic survey. *arXiv preprint arXiv:2002.12169* (2020).
[62] Sicheng Zhao, Guangzhi Wang, Shanghang Zhang, Yang Gu, Yaxian Li, Zhichao Song, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. 2020. Multi-source distilling domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 12975–12983.
[63] Vincent Wenchen Zheng, Derek Hao Hu, and Qiang Yang. 2009. Cross-domain activity recognition. In *Proceedings of the 11th international conference on Ubiquitous computing*. 61–70.
[64] Yongchun Zhu, Fuzhen Zhuang, and Deqing Wang. 2019. Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5989–5996.
[65] Yongchun Zhu, Fuzhen Zhuang, Jindong Wang, Guolin Ke, Jingwu Chen, Jiang Bian, Hui Xiong, and Qing He. 2020. Deep subdomain adaptation network for image classification. *IEEE transactions on neural networks and learning systems* 32, 4 (2020), 1713–1722.