Self-Train: Self-Supervised On-Device Training for Post-Deployment Adaptation

Jinhao Liu
Computer Science and Engineering
University of California San Diego
La Jolla, California, United States
jil037@ucsd.edu

Xiaofan Yu
Computer Science and Engineering
University of California San Diego
La Jolla, California, United States
x1yu@ucsd.edu

Tajana Rosing
Computer Science and Engineering
University of California San Diego
La Jolla, California, United States
tajana@ucsd.edu

***Abstract*—Recent years have witnessed a significant increase in deploying lightweight machine learning (ML) on embedded systems. The list of applications range from self-driving vehicles to smart environmental monitoring. However, the performance of ML models after the deployment degrades because of potential drifting of the device or the environment. In this paper, we propose *Self-Train*, a self-supervised on-device training method for ML models to adapt to post-deployment drifting without labels. *Self-Train* employs offline contrastive feature learning and online drift detection with self-supervised adaptation. Experiments on images and real-world sensor datasets demonstrate consistent accuracy improvements over state-of-the-art online unsupervised methods with 2.45x at maximum, while maintaining lower execution time with a maximum of 32.7x speedup.**

***Index Terms*—Self-Supervised Learning, On-Device Adaptation, Lightweight Training**

I. Introduction

In recent years, the number of Internet-of-Things deployments has been increasing exponentially with an expectation of 26.4 billion connections by 2026 [1]. The latest advances of lightweight Machine Learning (ML) algorithms and more powerful embedded hardware have enabled on-device intelligence, including a variety of applications ranging from self-driving vehicles [2] to smart environmental monitoring [3]. A typical deployment starts with offline training on a pre-collected dataset and performs online inference with streaming sensor input. While superb performances are observed in multiple scenarios [4], [5], only a few works have identified the potential drifting issue when migrating from offline (prior to deployment) to online (post deployment). Specifically, drifting could result from sensor aging [6] or non-stationary environment with dynamically changing data distribution that is different from the pre-collected dataset [7]. Drifting occurs frequently in practice and degrades the performance of ML models dramatically, leading to accuracy as low as 22.64% without adaptation [8].

The aforementioned drifting problem is best characterized by concept drift [9]–[11] which is a well-known issue in machine learning. Concept drift describes unpredictable changes in the probability distribution of streaming data over time (more details in Section III-A). Previous works have attacked concept drift with transfer learning, which considers the training set and the test set as two stationary distributions, named the source and target respectively, and builds a mapping between the source and target such that a model trained on the source can be adapted to give accurate predictions on target samples [12]. Nevertheless, the edge computing scenario brings forth a unique set of challenges that past adaptive methods fail to fully address. First, edge systems place significant constraint on computational power that limits the feasibility of complex models like deep neural networks. In addition, the following two aspects are often overlooked.

- **Label Acquisition Cost.** In the post-deployment scenarios of embedded systems, it is often hard and time-consuming to acquire true labels of drifted samples. However, supervision is crucial to produce category-level alignment for accurate drift adaptation for concept drift [13]. Waiting for true labels to adapt the model accurately can lead to significant delay in prediction or cost expensive human efforts. Existing methods often employ semi-supervised approaches using a few ground truth labels as anchors to map other unlabeled samples [7], which alleviate but fail to address the essential problem.
- **Online Adaptability.** Most existing transfer learning methods use offline adaptation, which expects all target data to be available before the model can be trained, then generate predictions [14]. In our scenario data arrives in streams and the model is expected to make accurate prediction for each sample in real time. Offline adaptation would significantly delay the prediction and perform poorly in real-time adaptation upon the possible non-stationary shift and class imbalance.

To address these issues, we propose *Self-Train*, a self-supervised online model adaptation method for on-device real-time adaptation. We assume a small labeled class-balanced dataset is given before the deployment, and unlabeled streaming data with non-stationary distribution comes in a sample-by-sample manner after the deployment. *Self-Train* employs contrastive learning to extract features that best maintain inter-class separation, then performs online drift detection based on the feature-space distances and triggers re-training appropriately. Contrastive learning is a self-supervised approach to learn representation that minimizes the distance between similar samples (positive pairs) and maximizes the distance

between dissimilar ones (negative pairs), achieving state-of-the-art performances in both offline and online training [15]–[17]. Recent works have reported that contrastive learning is able to learn more robust representations as compared to end-to-end cross-entropy loss [18].

Specifically, *Self-Train* consists of a lightweight neural network with a contrastive feature extraction layer and a softmax classifier layer; and a small training memory. The proposed algorithm works in the following manner: first, a labeled source dataset is used to train the model offline and initialize the memory. Once deployed, the model is exposed to the unlabeled streaming data with potential drift. An online drift detection module is designed by comparing the feature-space distances between the new sample and the memory samples, after which memory update is performed to replace the oldest sample according to pseudo-labels. Finally, *Self-Train* triggers re-training once we accumulate a substantial amount of new drifted samples.

To summarize, the contributions of the paper are as follows:

(1) Originating from the concept drift problem, we spot and formally define the *prior- vs. post*-deployment drifting problem that is common for pervasive ML deployment on small embedded systems.
(2) We propose *Self-Train* to perform post-deployment on-device adaptation for non-stationary and unlabeled streaming samples. *Self-Train* employs a contrastive learning-based feature extractor, an online drift detector using feature-space distances and memory-based retraining. In contrast to existing works [7], [19], [20], *Self-Train* excels at real-time online adaptability without the help of external labels, which suits the practical deployment in the wild.
(3) We conduct experiments on real-world sensor and artificial image datasets. Our experiments on the real-world sensor dataset shows a maximum accuracy improvement of 2.45x over the second-best baseline method. We also evaluate the execution time and energy consumption of *Self-Train* and baseline methods on Raspberry Pi 4, the latest edge computing platform. Our method decreases execution time by at most 32.7x, while saving 93% of energy.

The rest of the paper is organized as follows. Section II reviews related literature in concept drift. Section III defines the target problem and introduces the proposed design of *Self-Train*. Section IV reports the experimental results on real-world sensor and artificial image datasets. The whole paper concludes in Section V.

II. Related Works

We review existing literature addressing concept drift from the following angles.

A. Semi-Supervised vs. Unsupervised Learning

Most previous works for online adaptation employed semi-supervised learning where a small set of labeled samples is provided. [7] reduces the number of required labeled samples by adapting only on drifted samples. To adapt to drifting gas sensors, [21], [22] use transfer samples (i.e., measurements taken with several hand-crafted reference gas mixtures) as anchors to infer the overall direction and extent of the drift.

Compared to semi-supervised methods, unsupervised methods assume a tighter constraint with no labeled samples available after deployment. Statistical mapping [23] and subspace projection [24] have been adopted for sample-wise mapping, which however are not suitable for adapting distribution drifting. Alternatively, [25] optimizes subspace projection to preserve both intra-class affinity and inter-class separation in the learned projection. Deep learning models such as deep belief networks (DBN) [26] and stacked denoising autoencoders [27] can learn domain-invariant features that are robust to drift, thus have been applied to adapting gas sensor drift [21], [26]. [20] utilizes optimal transportation to build a mapping between the probability distribution of the source and target samples. [13], [28] rely on high-confidence pseudo-labels to supervise the adaptation process.

In contrary to previous works, our design assumes no label in adaptation and uses self-supervised contrastive learning, which learns more robust representations against drifting compared to using cross-entropy loss [18]. Moreover, our design conserves computational power by extracting the features directly without reconstruction.

B. Offline vs. Online Adaptation

Traditional transfer learning focuses on the offline scenario where all samples are assumed to be available during adaptation [25]. Offline methods typically are built around complex models thus are too computationally-intensive to run on edge devices. In practice, the online adaptive learning methods are required to generate prediction immediately after a sample arrives, and updates the model incrementally whenever drift is detected [14]. [7] used covariance matrix to detect drift and perform incremental light-weight update to the model with the drifted samples. [19] utilized a least-square support vector machine and updated the model by replacing the nearest support vector with each drifted sample. Nevertheless, the above online methods are optimized for computational cost while sacrificing performances. Our design aims at accurate adaptation subject to limited on-device computational resources.

C. Contrastive Learning

Several recent works applied contrastive learning to domain adaptation. Shen *et al.* utilized contrastive pre-training similar to SimCLR on both the unlabeled source and target dataset to learn transferable features, and fine-tune the model on labeled source data [29]. Qian *et al.* investigated the feasibility of 4 state-of-the-art contrastive learning models in small-scale Human Activity Recognition (HAR) tasks and designed augmentation techniques specific to the time-series sensor data. Wang *et al.* presented an augmentation-free contrastive method for unsupervised domain adaptation (UDA) by forming positive pairs between the target and source samples belonging to the same class [30]. Compared to previous contrastive

TABLE I
LIST OF IMPORTANT NOTATIONS IN PROBLEM FORMULATION.

Symbol	Meaning
D_0	Labeled offline dataset
D_s	Unlabeled streaming Dataset
M	Training memory
C_k	Representation of samples in class k
$\vec{C}_k$	The centroid of class k's representation
R_k	Average distance to class center in source samples
d_i	distance to class center for the i^{th} sample
θ_d	Threshold for drift detection
θ_c	Threshold for pseudo-label confidence check
p	Probablity distribution of input samples and output labels

UDA methods, *Self-Train* adapts to the target incrementally, therefore does not depend on availability of target data for meaningful adaptation. In addition, *Self-Train* does not rely on augmentation, thus reduce the computational time which is critical for edge systems.

III. METHOD

In this section, we first introduce the background of our post-deployment drifting problem in Section III-A, then we provide thorough details of *Self-Train* in Section III-B.

A. Background

Offline and online datasets. In our drifting problem, we assume two datasets are available. The offline dataset $D_0 = (X_0, Y_0)$ is a small labeled dataset which is collected prior to deployment and used to perform initial training on the model. After deployment, the edge device gathers a continuous stream of drifting data samples in a time sequence. These samples can be sensor measurements, images, or any information collected from the environment. Note that our model does not require the timestamp or delay between each sample to be present in the collected data. We define this data stream as the unlabeled dataset $D_s = X_s$.

Drifting Distribution. Data drift can be defined as a change in the joint probability distribution $p(x, y)$ between two datasets, namely source and target: $p_{source}(x, y) \neq p_{target}(x, y)$, where x is the input sample and y is the output label. If we apply the conditional probability equation, $p(x, y) = p(x)p(y|x)$, we then have two separate factors that contribute to data drift:

$$p_{source}(x) \neq p_{target}(x) \tag{1a}$$

$$p_{source}(y|x) \neq p_{target}(y|x) \tag{1b}$$

The former is defined as covariate shift, which considers the change between two sample distributions. Covariate drift is a well-studied problem in traditional unsupervised transfer learning [23], [31], [32] to build a robust model on unseen test data. However, it ignores the latter factor of the change in conditional probability, which has a direct influence on a model's predicting accuracy.

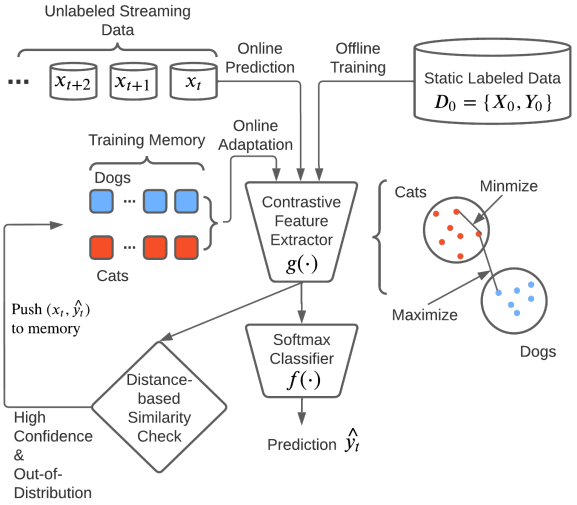

Fig. 1. Visualization of *Self-Train*.

Concept drift considers both of these conditions to be an indicator of drift, emphasizing on changes in $p(y|x)$. Here $p(y|x)$ is defined as the concept [14]. For example, a sample x that belongs to class A before the shift happens, may shift to class B. This will not cause a change in the sample distribution $p(x)$, but leads to a change in $p(y|x)$. While concept drift provides a more comprehensive definition of the drift, it is hard to detect unsupervisedly without the ground truth label y, as the formula suggests. We offer an alternative method to detect drift via weak supervision provided by comparing the distance between the contrastive representation of drifted samples to source samples.

B. Design of Self-Train

The complete model of *Self-Train* is visualized in Figure 1, which works in the following manner:

(1) *Self-Train* trains a neural network consisting of a contrastive feature extraction layer g and a softmax predictive layer f trained on an initial pre-drift dataset D_0 with labels Y_0.
(2) We calculate pseudo-label confidence threshold by taking the average distance between each source sample and the sample mean of the same class.
(3) We apply the model to an unlabeled streaming dataset D_s to generate contrastive representation and prediction for each sample as soon as they arrive. Samples within the confidence threshold are pushed into the training memory.
(4) We update the model once a significant number of drifting samples have been accumulated in the memory.

To ensure responsive adaptation in class-imbalanced streaming data, we propose a training memory with a same-class replacement strategy so that the model can utilize historical data, when a substantial drift is detected but samples from some classes are not available. The training memory M is setup as a dictionary with class label $y_i \in Y_0$ as the key. $M[y_i]$

is a fixed size queue that holds training samples which have label y_i. During the initial training, M is initialized with the offline data $D_0 = (X_0, Y_0)$. To insert a new selected drifted sample x and its predicted pseudo-label $\hat{y}$, the front of the queue $M[\hat{y}]$, which is the oldest sample, is popped, and the new sample x is inserted at the end of the queue, keeping the size of each queue constant. In this way, we maintain the class-wise balance in M among all seen classes.

The three major components of *Self-Train* (i.e., contrastive representation learning, drift detection and self-supervised retraining) are introduced in more details in the following lines.

Contrastive Representation Learning. The first layer of our network is trained on D_0 to generate low-dimensional representations that best discriminate data from different classes by training on the contrastive loss (Equation 2). Contrastive loss maximizes the distance between samples of different classes while minimizing the distance between samples of the same class [33]. Such scheme effectively pulls samples from the same class into tight clusters, making it easier to detect drifted samples. Figure 2 visualizes the effect of contrastive learning on the gas sensor array dataset with each color represents one class. It can be observed that the feature extractor trained on pre-drift data successfully extracts inter-class separability and intra-class affinity, which can be maintained post-drift.

$$L_{con} = \frac{\sum_{x_i, x_j \in D, i \neq j} Y \cdot d_{i,j}^2 + (1 - Y) \cdot \max(m - d_{i,j}, 0)^2}{|D|(|D| - 1)} \tag{2}$$

where:

- $(x_i, y_i), (x_j, y_j) \in D$
- $Y = 1$ when $y_i = y_j, Y = 0$ when $y_i \neq y_j$
- $d_{i,j} = ||x_i - x_j||$
- m: defines a radius around x_i such that only dissimilar samples within are included in the loss

The loss is normalized by the divisor $|D|(|D| - 1)$ so that its value is not effected by batch size. The normalization step is important because we prefer large batch size during the initial training phase so that the contrastive feauture is representative of the entire source distribution; whereas during adaptation, we prefer smaller batch size so that we can make responsive model update with a smaller number of drifted samples.

In the context of drift-correction, this approach has two advantages: first, it helps prevent drifted samples from being classified to another class by increasing the representation's discriminitive power between classes; second, grouping samples from the same class into a tight cluster makes it easier to detect new samples that drifted from the original distribution.

After we finish training the representation, we freeze the parameters of the representation layer and attach a softmax layer with cross-entropy loss to handle classification. The model is trained again on the labeled D_0 until convergence.

From Classification Confidence to Drift Detection. Previous works in supervised adaptation have shown that an effective strategy to update the model is selecting the samples most distinct from the source distribution [7]. This strategy presents a pitfall in unsupervised learning where we need to rely on pseudo-labels instead of ground truth labels, because the hardest samples often decrease the credibility of pseudo-labels, which plays a critical role in online adaptation [19].

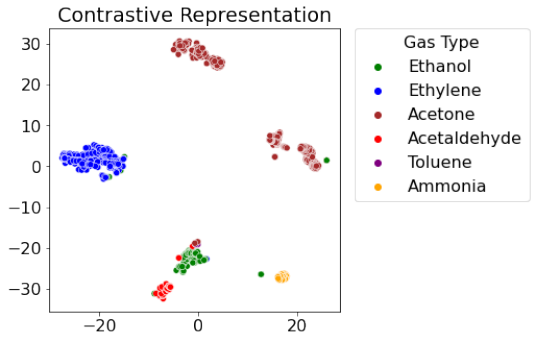

Fig. 2. t-SNE plots of learned features after the contrastive learning-based feature extractor on the gas sensor array dataset [8], [34]. Each color represents one gas type.

To solve this dilemma, we propose a semi-hard sample selection strategy which selects out-of-distribution samples that generate high-confidence pseudo labels by exploiting the inter-class separability of our contrastive representations.

To derive the threshold for classification confidence and drift detection, we develop an effective metric that allows us to compare streaming samples against the source distribution by exploiting the intra-class affinity and inter-class separability of contrastive representations. We first define a baseline of comparison as the mean point in contrastive representation of source samples for each class k. Let $C_k = g(\{x_i | y_i = k, (x_i, y_i) \in D_0\})$ be the contrastive representation of source samples in class k, then the mean point can be given by $\bar{C}_k = \frac{\Sigma C_k}{|C_k|}$. Then, we build a unit sphere around each $\bar{C}_k$ with radius $R_k = \frac{\Sigma ||\bar{C}_k - C_k||}{|C_k|}$. If an incoming sample is sufficiently close to the unit sphere, we can conclude that it is in-distribution and have high classification confidence.

When the initial model is applied to the unlabeled streaming dataset D_s, for each incoming sample $x_i \in D_s$, we generate its contrastive representation $c_i = g(x_i)$ and predicted label $\hat{y}_i = f(c_i)$. We define $d_i = ||\bar{C}_{\hat{y}_i} - c_i||$ as the distance from the current sample to the pre-drift class center. To allow more fine-grained tuning of the model, we define two thresholds, θ_d and θ_c to be applied as percentage of the threshold $\bar{C}_k$:

- When $d_i > \theta_d \cdot R_k$, we conjecture that the d_i is sufficiently distant from the pre-drift samples for x_i to be considered out-of-distribution.
- When $d_i < \theta_c \cdot R_k$, we conjecture that d_i is small enough for x_i to be considered to have high classification confidence.

We define semi-hard samples as ones that satisfy both constraints. θ_d, θ_c allow us to control the uncertainty of the pseudo-labels used for adaptation. With a large θ_c, we intro-

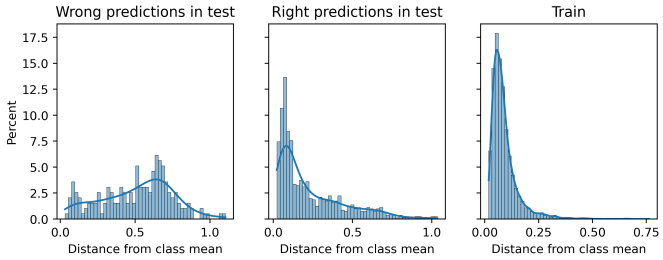

Fig. 3. Distribution of distance to the class center d_i in training and perturbed testing samples with respect to classification correctness.

duce harder samples that allows our model to adapt to the drift faster but may also introduce errors; while θ_d acts as a lower bound that prevents the model from meaningless adaptation on in-distribution samples thus saving computation cost.

To show the effectiveness of the proposed semi-hard samples selection, Figure 3 demonstrates the relationship between classification accuracy and distance to c_i using perturbed CIFAR10-C dataset [35]. It can be observed that the correctly predicted samples have similar d_i (distance to class center) as training samples with a long tail to the right, while most incorrectly predicted samples have a greater d_i than training samples.

An efficient online adaptive model should be reactive, meaning that it is only updated when drift is detected. Compared with a blind adaptation strategy [14], which updates the model on fixed intervals, our design self-corrects sooner and saves computation time by updating the model only when needed.

Self-supervised Model Re-Training. In this section, we will introduce our sample replacement strategy to compose training sets for model updates. The training set can be composed in two ways, either with a mix of old and new samples, or entirely with newly selected ones. While in an offline scenario, one can hand-pick equal number of latest samples from each class to form a balanced training set. Such a strategy is unrealistic for handling streaming data, where the arriving samples can be highly imbalanced. For example, we can have a long series of drifted samples from class A without seeing any from class B. Failure to adapt to drifted class A's samples while waiting for class B's samples can be detrimental to maintaining high accuracy.

To make the model adaptation more reactive to drift in unbalanced streaming data, we propose a replacement strategy where the number of samples in each class remains constant, and each selected new sample with pseudo-label $\hat{y}_i$ replaces the oldest sample in the class corresponding to $\hat{y}_i$ in a First-in-First-out fashion. We initiate the model update when the number of newly arrived samples reach a preset proportion p of the size of the original training set such that we can make meaningful update to the model. Intuitively, the smaller we set p, the more reactive our model can deal with drift, but more computationally inefficient as we need to run more re-training steps. Hence it is important to tune the value of p to balance sensitivity and computation efficiency.

IV. Results

A. Experiment Setup

We implement *Self-Train* with Python 3.7 and tensorflow v2.1.0 [36]. Important parameters settings are reported in Table II. We also experiment *Self-Train* on a Raspberry Pi (RPi) 4B platform with 4GB RAM [37] and measure the execution time and energy consumption using the Hioki 3334 powermeter [38]. Our implementation is available on GitHub.[1]

TABLE II
List of Important Parameter Settings

Parameters	Datasets		
	Gas	***MNIST***	***CIFAR10-C***
$\eta^{a}_{contrastive}$	.0003	.0005	.001
$\eta^{b}_{classifier}$	.001	.001	.001
$batch\ size$	80	32	64
θ_d	0.1	0.6	0.2
θ_c	5.5	5	4

[a]Learning rate of contrastive layer.

[b]Learning rate of classifier layer.

Datasets. We evaluate on three datasets including Internet-of-Things sensing and image classification tasks under perturbances.

(1) **Gas Sensor Array Drift Dataset** exhibits class-imbalanced distribution as well as continuous, non-linear and random drift through time [8], [34]. It features a series of 128-dimensional numerical sensor measurement samples, each belonging to one of the six types of gas. The dataset is divided into 10 batches in temporal order spanning a period of 36 months; In our setup, we use batch 1 as the labeled offline training data, and treat the following batches as an unlabeled online data stream for adaptation.

(2) **MNIST** digit dataset [39] is used to test the performance on small-scale image classification task. A small subset of the original training data is used as the offline dataset to reduce training time both during the initial training and model updates. A drifted dataset is artificially created by applying gaussian noise to the streaming images in MNIST. We generate gaussian noise with a mean of 1 and a standard deviation of 0.3.

(3) **CIFAR10-C** [35] is a state-of-the-art robustness benchmark with 19 types of common corruptions and pertubations. We use this dataset to test the model's performance on a relatively harder image classification task and the model's ability to adapt to different types of drift.

Machine Learning Models. As outlined in the Section III, our model has two components. *Contrastive Representation Layer*: For the gas sensor dataset, we use two fully connected

[1]https://github.com/jinbest17/self-supervised-online-adaptation

layers with 100 and 80 neurons respectively. For image classification tasks, a simple CNN with two convolution layers and 32 and 64 filters respectively. The model is trained with a the ReLU activation function, the Adam optimizer, and the contrastive loss metric. *Classification Layer*: We use a dense softmax layer with the number of neurons equal to the number of classes to handle classification.

Baselines. For the gas sensor dataset, we compare our results with state-of-the-art online unsupervised adaptation methods, unsupervised particle adaptive classifier **(uPAC)** based on least square support vector machine [19] and Optimal Transport Adaptation **(OT)** [20]. We acknowledge there are promising offline methods available, but they are not designed for streaming data and real-time adaptation, therefore can utilize a deeper network structure to retrieve more comprehensive information about the drift. For the image classification task, we only employ OT as our baseline because they share a similar neural network-based structure.

B. Results

Gas Sensor Array Dataset. Figure 4 shows the online prediction accuracy achieved by our proposed method and the baselines on the chemical sensor drift dataset. Results for *Self-Train* are averaged for six runs to control outliers. We run each baseline once since their results are deterministic with optimized parameters. Our model shows consistent improvements over the baselines in all batches. Compared with the second-best accuracy for each batch, *Self-Train* provides a maximum improvement of 2.45x (batch 4). Compared with OT, uPAC sees slower accuracy degradation at first. This could be due to uPAC using a more fine-grained and informed model update strategy in which a new sample will always update the closest and oldest sample. Whereas OT's adaptation accuracy depends on the assumption that the Wasserstein-Distance between the target distribution and the source distribution is small, this is often not the case in a real world dynamic environment. Therefore, the performance of OT degrades drastically when applied to complex drifting behaviors such as the chemical gas sensor dataset.

MNIST Dataset. Figure 5 compares our model with the OT baseline and a non-adapting classifier with the same network architecture as our *Self-Train* model. Comparing the average accuracy across the 3200 noised test samples, our method improves the baseline OT by 1.07x, and the non-adapting classifier by 1.3x. To observe the performance of online adaptation, accuracy is also calculated after each 100-sample interval in the streaming data. While OT is able to take advantage of the domain adaptation power provided by optimal transportation initially, its model update method uses all pseudo-labels from the drifted data without estimating its correctness, thus the predictive power decreases overtime due to training with wrong pseudo-labels. Whereas our method selectively updates on high-confidence pseudo-labels and is therefore able to maintain high prediction accuracy.

CIFAR10-C Dataset. Figure 6 compares our model with the OT baseline and a non-adapting classifier for 18 image corruption tasks. Because CIFAR10-C is harder to classify using the same network architecture than MNIST, we show adaptation results on 8000 test samples to ensure the models have sufficient samples to compensate for the lower initial accuracy. Our model demonstrates consistent improvements over the baselines, improving OT by at most 3.20x and at least 2.60x for each task, and the non-adapting classifier by at most 1.42x and at least 1.19x. OT's disadvantage of not selecting accurate pseudo-labels becomes more apparent in the CIFAR10-C tasks due to a lower classification accuracy around 0.4 to 0.6 of the non-adapting classifier. As a result, OT descends to a trivial model as adaptation continues.

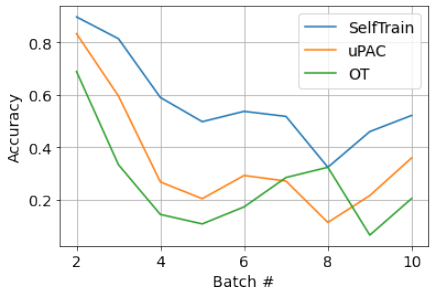

Fig. 4. Prediction accuracy on each online batch of the gas sensor array drift dataset.

Fig. 5. Accuracy on the noised MNIST dataset as more samples come in.

Execution Time and Energy Consumption. Figure 7 demonstrates the measured execution time and energy consumption on the RPi 4B platform. Compared with the baselines, *Self-Train* consumes the least time to execute and energy. On the gas sensor dataset, *Self-Train* runs 9.75x and 32.7x faster than OT and uPAC respectively, while saving 90% and 97% of energy compared to OT and uPAC. On the MNIST data, *Self-Train* runs 11.9x faster than OT and consequently saves 92% energy. On the CIFAR-10-C data, *Self-Train* runs 9.01x faster than OT and consequently saves 89% energy. The increase in efficiency can be attributed to two factors. First, our model perform drift detection on contrastive representation, and its dimensionality reduction property decreases the computation cost for distance-based metrics. Second, our drift detection strategy focus on drifted samples that are significantly different from the pre-drift distribution, thus reducing the number of samples needed to perform meaningful adaptation. We opt not to test uPAC on the

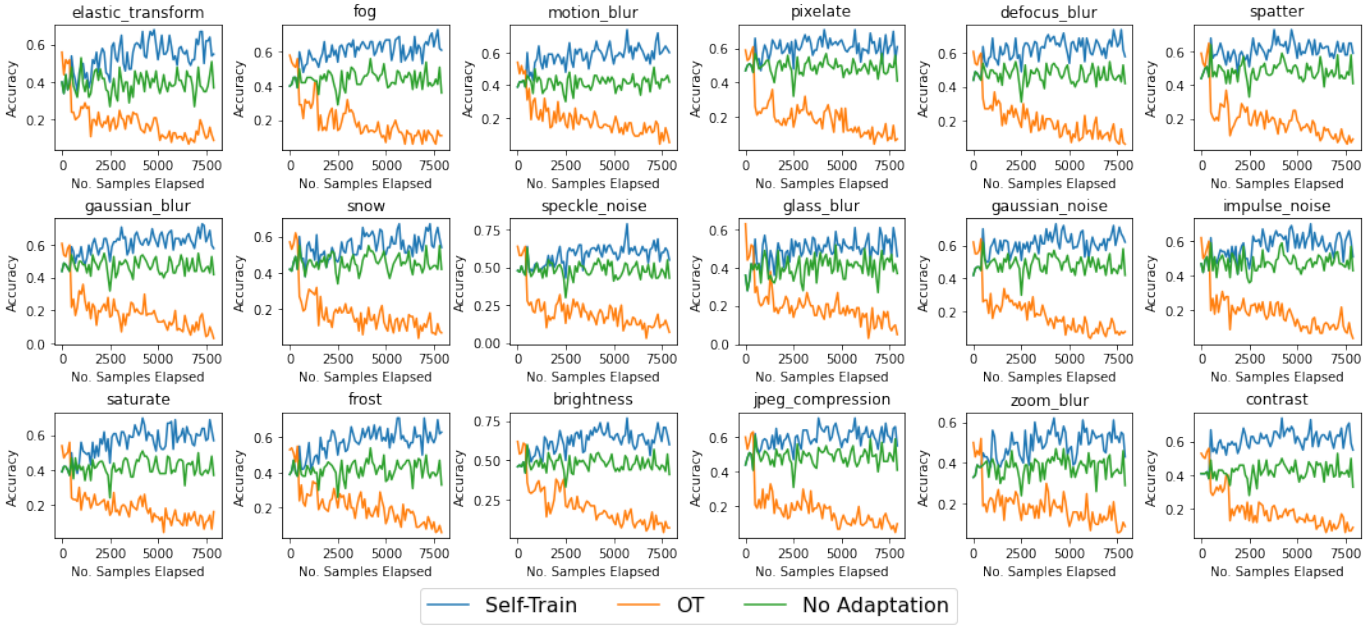

Fig. 6. Accuracy results on CIFAR10-C as more samples come in.

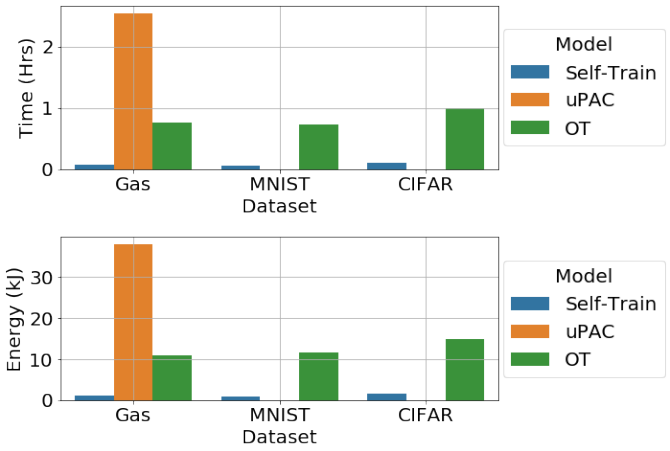

Fig. 7. Execution time and energy consumption of *Self-Train* and baselines

image classification tasks because it is not designed for high-dimensional data, and cannot finish running within reasonable amount of time.

V. Conclusion

In this paper, we propose an online method for concept drift adaptation with minimum prediction delay in dynamically changing environments while eliminating label acquisition cost by utilizing the supervisory power of the high-confidence pseudo-labels generated by the model itself. As an on-device system, our *Self-Train* model also places focus on computation cost by selectively updating to drifted samples and opting for minimal network complexity. We test our method on both real world drifting and images with artificial corruptions and perturbations. Our model shows consistent accuracy improvements with 2.45x at maximum, while maintaining lower execution time with a maximum of 32.7x speedup.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. This work was supported by National Science Foundation under Grants #2003279, #1826967, #2100237, #2112167, and #2112665.

References

[1] Ericsson, "Ericsson mobility report," Jun. 2021. [Online]. Available: https://www.ericsson.com/4a03c2/assets/local/reports-papers/mobility-report/documents/2021/june-2021-ericsson-mobility-report.pdf

[2] I. Ahmad and K. Pothuganti, "Design & implementation of real time autonomous car by using image processing & iot," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2020, pp. 107–113.

[3] N. A. Jasim, H. TH, and S. A. Rikabi, "Design and implementation of smart city applications based on the internet of things." *International Journal of Interactive Mobile Technologies*, vol. 15, no. 13, 2021.

[4] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for ai-enabled iot devices: A review," *Sensors*, vol. 20, no. 9, p. 2533, 2020.

[5] Z. Lv, L. Qiao, and S. Verma, "Ai-enabled iot-edge data analytics for connected living," *ACM Transactions on Internet Technology*, vol. 21, no. 4, pp. 1–20, 2021.

[6] P. Nelli, G. Faglia, G. Sberveglieri, E. Cereda, G. Gabetta, A. Dieguez, A. Romano-Rodriguez, and J. Morante, "The aging effect on sno2–au thin film sensors: electrical and structural characterization," *Thin Solid Films*, vol. 371, no. 1-2, pp. 249–253, 2000.

[7] S. Lee and S. Nirjon, "Learning in the wild: When, how, and what to learn for on-device dataset adaptation," in *Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, 2020, pp. 34–40.

[8] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, and R. Huerta, "Chemical gas sensor drift compensation using classifier ensembles," *Sensors and Actuators B: Chemical*, vol. 166-167, p. 320–329, May 2012.

[9] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[10] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," *Big data analysis: new algorithms for a new society*, pp. 91–114, 2016.

[11] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[12] C. Yang, Y.-m. Cheung, J. Ding, and K. C. Tan, "Concept drift-tolerant transfer learning in dynamic environments," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[13] C. Chen, W. Xie, T. Xu, W. Huang, Y. Rong, X. Ding, Y. Huang, and J. Huang, "Progressive feature alignment for unsupervised domain adaptation," *CoRR*, vol. abs/1811.08585, 2018. [Online]. Available: http://arxiv.org/abs/1811.08585

[14] I. Khamassi, M. Sayed Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Systems*, vol. 9, 03 2018.

[15] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020.

[16] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020. [Online]. Available: https://arxiv.org/abs/2002.05709

[17] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[18] H. Cha, J. Lee, and J. Shin, "Co2l: Contrastive continual learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9516–9525.

[19] Q. Huang, D. Yang, L. Jiang, H. Zhang, H. Liu, and K. Kotani, "A novel unsupervised adaptive learning method for long-term electromyography (emg) pattern recognition," *Sensors*, vol. 17, no. 6, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/6/1370

[20] D. Kim, J. Kwon, B. Jeon, and Y.-L. Park, "Adaptive calibration of soft sensors using optimal transportation transfer learning for mass production and long-term usage," *Advanced Intelligent Systems*, vol. 2, no. 6, p. 1900178, 2020.

[21] K. Yan and D. Zhang, "Correcting instrumental variation and time-varying drift: A transfer learning approach with autoencoders," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 9, p. 2012–2022, 2016.

[22] ——, "Calibration transfer and drift compensation of e-noses via coupled task learning," *Sensors and Actuators B: Chemical*, vol. 225, pp. 288–297, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925400515306407

[23] S. Inoue and X. Pan, "Supervised and unsupervised transfer learning for activity recognition from simple in-home sensors," *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2016.

[24] L. Zhang, Y. Liu, Z. He, J. Liu, P. Deng, and X. Zhou, "Anti-drift in e-nose: A subspace projection approach with drift reduction," *Sensors and Actuators B: Chemical*, vol. 253, pp. 407–417, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925400517311759

[25] Z. Yi, W. Shang, T. Xu, and X. Wu, "Neighborhood preserving and weighted subspace learning method for drift compensation in gas sensor," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2021.

[26] Q. Liu, X. Hu, M. Ye, X. Cheng, and F. Li, "Gas recognition under sensor drift by using deep learning," *International Journal of Intelligent Systems*, vol. 30, no. 8, p. 907–922, 2015.

[27] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, ser. UTLW'11, 2011, p. 17–37.

[28] Z. Fu, X. He, E. Wang, J. Huo, J. Huang, and D. Wu, "Personalized human activity recognition based on integrated wearable sensor and transfer learning," *Sensors*, vol. 21, no. 3, p. 885, 2021.

[29] K. Shen, R. Jones, A. Kumar, S. M. Xie, J. Z. HaoChen, T. Ma, and P. Liang, "Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation," 2022. [Online]. Available: https://arxiv.org/abs/2204.00570

[30] R. Wang, Z. Wu, Z. Weng, J. Chen, G.-J. Qi, and Y.-G. Jiang, "Cross-domain contrastive learning for unsupervised domain adaptation," 2021. [Online]. Available: https://arxiv.org/abs/2106.05528

[31] B. Sun and K. Saenko, "Deep CORAL: correlation alignment for deep domain adaptation," *CoRR*, vol. abs/1607.01719, 2016. [Online]. Available: http://arxiv.org/abs/1607.01719

[32] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2066–2073.

[33] R. Hadsell, S. Chopra, and Y. Lecun, "Dimensionality reduction by learning an invariant mapping," 02 2006, pp. 1735 – 1742.

[34] E. Martinelli, G. Magna, S. De Vito, R. Di Fuccio, G. Di Francia, A. Vergara, and C. Di Natale, "An adaptive classification model based on the artificial immune system for chemical sensor drift mitigation," *Sensors and Actuators B: Chemical*, vol. 177, pp. 1017–1026, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925400512013032

[35] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *Proceedings of the International Conference on Learning Representations*, 2019.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[37] "Raspberry Pi 4B," https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, [Online].

[38] "Hioki3334 Powermeter," https://www.hioki.com/en/products/detail/?product_key=5812, 2022.

[39] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.