Online Domain Adaptive Classification for Mobile-to-Edge Computing

Forough Shirin Abkenar*, Leonardo Badia†, and Marco Levorato*
*Donald Bren School of Information and Computer Sciences, University of California at Irvine, United States
†Dept. of Information Engineering (DEI), University of Padova, Italy
Email: fshirina@uci.edu, leonardo.badia@unipd.it, levorato@uci.edu

***Abstract*—A key challenge of today's systems is the mismatch between the high computational demands of modern neural network models for data analysis and the severely limited resources of mobile devices. Existing solutions focus on model simplification and task offloading to compute-capable edge servers. The former often leads to performance degradation, whereas the latter requires the transfer of information-rich signals and is subject to the impairments of wireless channels. To address these issues, a framework that establishes a novel form of collaboration between mobile devices and edge servers is proposed herein. The core idea is to deploy lightweight models on mobile devices that are intelligently updated to match the current, and local, distribution of the samples being observed. The framework develops the temporal patterns of the samples to determine the optimal model update policy, as well as channel resources allocated to the mobile users. The performance of the proposed framework is evaluated via extensive experiments with both synthetic and real-world datasets.**

***Index Terms*—Online classification, Local domain adaptation, Optimal resource usage, Edge Computing.**

1. Introduction

The most challenging emerging applications, such as vehicular autonomy, mobile healthcare, and augmented reality, necessitate the real-time analysis of information-rich signals [1]. The analysis algorithms typically take the form of deep neural networks (DNNs) trained on general datasets. As a result, state-of-the-art models often have a large number of parameters and thus, high demands in terms of computing power, memory, and energy reservoir [2, 3]. These needs clash with the severely limited resources of mobile devices (MDs) [4]. To address these issues, the research community primarily focused on two approaches: (i) model simplification [5–7]; and (ii) edge offloading [8–10]. The former approach reduces the size of the neural models by using techniques such as pruning, quantization, and knowledge distillation, and often leads to a non-negligible performance degradation [11]. The latter approach leads to a considerable increase in channel resources, while also exposing computing tasks to delay uncertainty due to channel impairments [12, 13].

In contrast with this approach, we propose an innovative solution that stems from the assumption that in some application scenarios, MDs may be observing a series of samples spanning only a local subset of the entire dataset, that is, a "local" domain. In principle, if the input sample distribution is known a priori, this may enable the deployment of simpler classifiers focused on the specific feature region, whose complexity could be maintained within the capacity of an MD. However, as the distribution is often not known beforehand and may shift over time, a focused domain-specific classifier needs to be continuously adapted, and this part of the solution goes beyond the capabilities of the MDs.

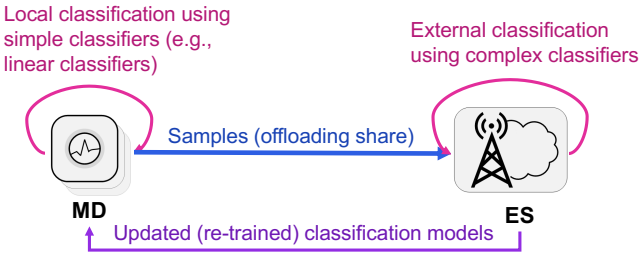

Figure 1: General architecture of the proposed framework.

At the same time, the edge server (ES) has sufficient resources to execute more refined classifiers and perform domain-specific retraining. However, the continuous transmission of data from all the MDs would consume a prohibitively large amount of resources, while also being unnecessarily redundant since it would not exploit model simplifications provided by domain adaptation [14].

This mismatch of resources implies that we are simultaneously facing two challenges: (i) offloading data to the ES is inconvenient as it may clog a constrained communication channel and misuse a powerful computational resource, whereas a simpler classifier (albeit tailored to the specific domain) would suffice for most of the samples [15]; (ii) the limitations of a local classifier available at an MD prevent it from reaching high accuracy as well as performing a proper domain adaptation [16] in the first place.

To solve these two issues at once, we propose a framework to provide adaptive online retraining capabilities to the MDs through a collaborative two-tier mobile-edge architecture. This results in a novel approach to online domain adaptive classification where the two existing separate

challenges are reworked to use each one of them to solve the other. Fig. 1 shows the general architecture of the proposed framework. The key idea is that instead of exploiting the ES just for data offloading, we take advantage of its superior computational capabilities to *train* the low-complexity local classifiers at the MDs. This allows the MDs to receive a proper setting for their very simple classifiers based on the exchange of distribution parameters periodically estimated, rather than offloading the data in its entirety.

We remark that offloading of raw data cannot be totally avoided, as it is still required for critical samples whose accuracy is beyond the capabilities of the local classification. Still, the exchanges with the ES (i.e., the offloading ratio) should be kept to a minimum, which avoids congestion on the limited capacity channel between the ES and the MDs. Moreover, this solves the problem of the MD seeing only a specific domain and not the entirety of the data and thereby requiring a tailored parametrization of the classifier.

Notice that the proposed framework differs from existing domain adaptation methods, which mostly focus on offline training of domain-specific classifiers of equal complexity [17], and instead, provides optimized lightweight online DA capabilities to resource-constrained devices.

To sum up, this paper presents the following contributions. First, we introduce the general architecture of the proposed algorithm. Moreover, we address two different challenges: a quantification of its effectiveness involving the dimensioning of the tradeoff between accuracy and amount of allowed data offloading, and subsequently a dynamic assessment of the performance under domain drift. We are able to prove that the proposed scheme is effective and versatile, as it can achieve very high accuracy with limited use of the system resources. At the same time, the critical point shifts to the dynamic adaptation in the presence of a dataset drift. It is shown that the proposed architecture is robust to rapidly changing domains, which triggers an interesting tradeoff between performance and resource consumption.

We perform numerical validation in different scenarios: a synthetic dataset explained in the following and a real-world stress detection usecase based on the UNITE dataset [18]. Both evaluations prove the suitability of the proposed approach under different conditions; for example, the synthetic dataset is balanced, whereas the real-world dataset is heavily unbalanced as it is of anomaly detection. However, our proposal is found to be effective in both cases, thereby establishing a general validity of the proposed technique and hinting at its better implementability in the Internet of Things (IoT) scenarios, which is considered to be a possible next step for future investigations.

The rest of this paper is organized as follows. In Section 2 we review the preliminaries for our investigation, describing the proposed architecture, the related work on domain adaptation and how our original proposal inherently differs from all of them, and the datasets used. Section 3 sets a first analytical evaluation of the proposed architecture as an offline optimization for a general (static) scenario split into local domains, and the first set of results is shown. In Section 4, we further extend it as an online dynamic investigation when the domains drift, and we discuss the resulting performance. Finally, Section 5 concludes the paper.

2. Background

2.1. Proposed Architecture

The severely limited resources of MDs prevent them from using neural network models, if not through the ES. Also, the MDs are not trained to take advantage of model simplifications allowed by domain adaptation [12].

The situation can be represented by the following metaphor. MDs are like local fishermen that are short-sighted and cannot catch fish in front of them, yet all they can do is ask a nearby battleship (the ES) for assistance. However, not knowing the specifics of the domain, the battleship is just able to launch a torpedo, which is surely able to kill the fish but with an unnecessary waste of resources. It would be much more convenient to adopt the old approach of "teaching the MD how to fish" and let them do the work by themselves.

Inspired by this reasoning, in [19] we proposed a novel way to exploit the two-tier edge computing-enabled IoT architecture, where the MD uses the ES mainly to primarily offload *domain parameters* instead of data. This allows for significantly decreasing communication exchanges since a compact domain representation (e.g., statistical parameters of the distribution of the observed data) can be sent to the ES instead of the actual data. Accordingly, the ES can generate synthetic data very similar to the real one using Metropolis-Hastings Algorithm [20]. At the same time, this solution takes proper advantage of the local classification domains (LCDs) [21] and can actually improve accuracy even with simpler local classifiers used at the MD.

We remark that, although this approach might lead to better accuracy, data offloading is actually not prohibited and can be used, albeit to a limited extent. Indeed, some samples may still be offloaded to the ES for more accurate classification [14]. Therefore, the classification can be performed either locally at the MD or externally at the ES. In particular, we assume the ES owns a predefined predictor that perfectly matches the ground-truth [21]. Also, the ES is responsible for providing the best classification model for the MD at every time-frame.

Accordingly, in [19] we found that, while LCDs accurately classify most of the samples, they are still imprecise to obtain perfect accuracy, and it is convenient to offload a small share of the data. Clearly, this approach must be controlled through a proper policy, since allowing for unlimited offloading would still achieve perfect accuracy but it contradicts the premise of having the MD which is carrying most of the classification burden. Thus, in [19] we investigated a policy, called MaxAcc Offloading, to maximize the resulting accuracy under a given limitation assigned to the share of data that can be offloaded to the ES, and with different approaches to choosing them. We found that an effective policy is to first sort the data in descending order of prediction loss and those with the highest loss are offloaded to

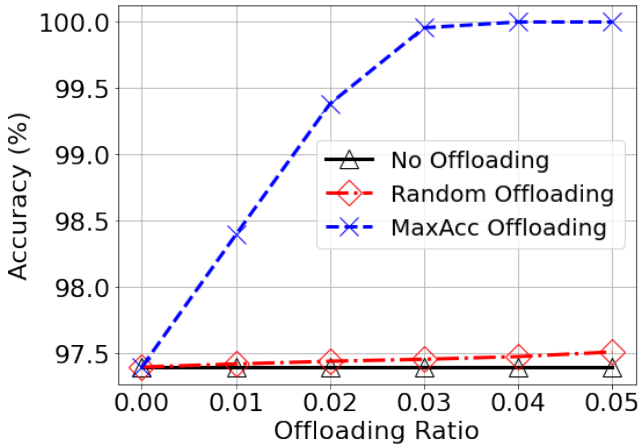

Figure 2: An example of the superiority of proposed MaxAcc Offloading policy in [19] on a synthetic dataset.

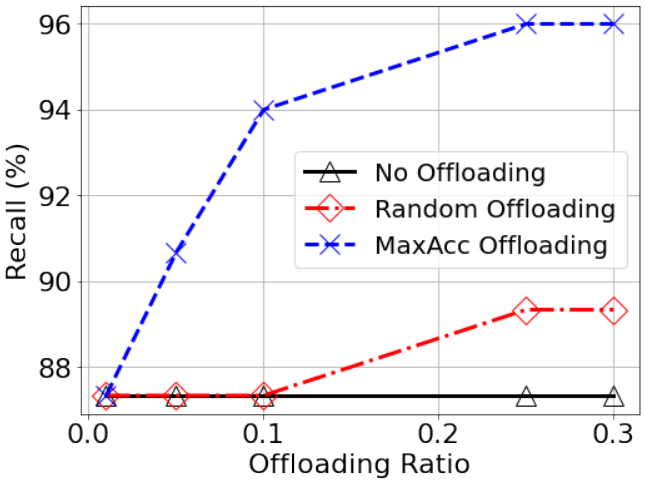

Figure 3: An example of the superiority of proposed MaxAcc Offloading policy in [19] on a real dataset.

the ES, which achieves near-perfect accuracy under extremely limited data offloading ratios.

To show the superiority of the proposed MaxAcc Offloading over the existing policies, we have compared its performance with two benchmark policies, namely: (*i*) No Offloading, where the model is updated every LCD, yet no sample is offloaded to the ES; and (*ii*) Random Offloading, where the samples are randomly chosen to be offloaded to the ES. Simulations have been performed over a synthetic dataset, including $N = 10^5$ samples each with two features, and the real dataset UNITE [18]. Notably, the number of samples labeled 1 and 0 is balanced for the synthetic dataset. Hence, we use accuracy to measure the performance of the policies. However, these labels are not balanced for the real dataset (that is, the number of samples with label 0 is significantly higher than the number of samples with label 1). Therefore, we use recall [22] instead of accuracy for the real dataset that is given by the number of true positives over the sum of true positives and false negatives. The results obtained for the synthetic dataset (Fig. 2) and the real dataset (Fig. 3) w.r.t different offloading ratio indicate that the MaxAcc Offloading outperforms No Offloading and Random Offloading in terms of both accuracy and recall.

It is worth mentioning that [19] is a preliminary for the current paper. In [19], we formed LCDs and defined novel policies for offloading purposes. In the current paper, we use a similar approach but comprehensively investigate the impact of domain drift and accordingly, perform domain adaptations for the datasets so that the accuracy of classifications is high. Thus, the following points are still open for the present investigation. First of all, instead of arbitrarily setting a given share for offloading, we investigate the tradeoff between this value and the resulting accuracy, which offers a more general perspective. Also, even though the proposed scheme works extremely well under static conditions, we still need to investigate what happens in case of domain drift and whether the proposed methodology still works. For example, it is important to quantify how often one should re-train the local classifiers, which would set the bottleneck on the MD-ES channel since the offload of actual data is found to be extremely contained. Finally, we also evaluate the performance of synthetic theoretical datasets as well as real data from a concrete case study, as will be explained next.

2.2. Domain Adaptation

The performance of traditional machine learning (ML) methods is negatively affected by a phenomenon, called dataset shift (drift). Concept shift and covariate shift are two major types of dataset shift. Under the concept shift [23], the input (independent variable) is mismatched with its predicted label (target variable). Changing environment is the main driver of such a drift [24]. In covariate shift [25], the distribution of the inputs varies over time due to different reasons such as rotation and scaling of an image [26], environment change and physical condition/emotion [27], verity of the population in data collection [28], etc. Domain adaptation [2, 29, 30], a subcategory of transfer learning [31], and active learning [32] are promising methodologies to cope with such challenges arising from domain shift. Both methods aim to learn a well-performing model for data distribution.

Domain adaptation trains an ML model on the samples from a source domain different from the target domain [33], while active learning trains the model on the most informative samples in the same domain [34]. For instance, the authors in [26] proposed the importance weighting (IW) method that aims to give larger weights to the training samples with the highest similarity to the corresponding samples in the test set. To this end, the Kullback-Leibler divergence is employed to estimate the importance weights. In a similar work [28], the authors proposed transductive training learning where the optimal model is obtained by minimizing the average product of error and the density ratio, which is defined in terms of the ratio between the target set and

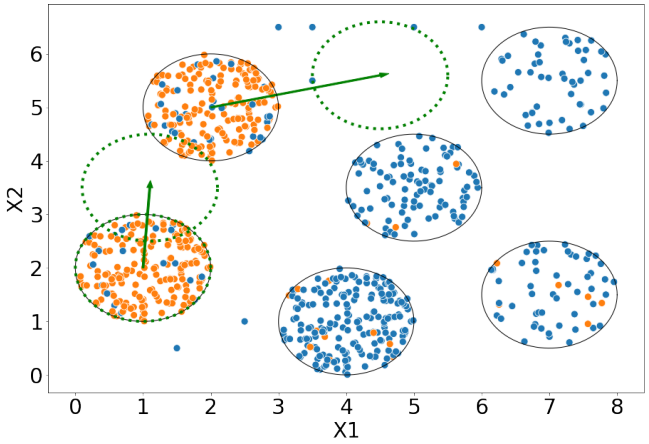

Figure 4: General online domain adaptation scenario

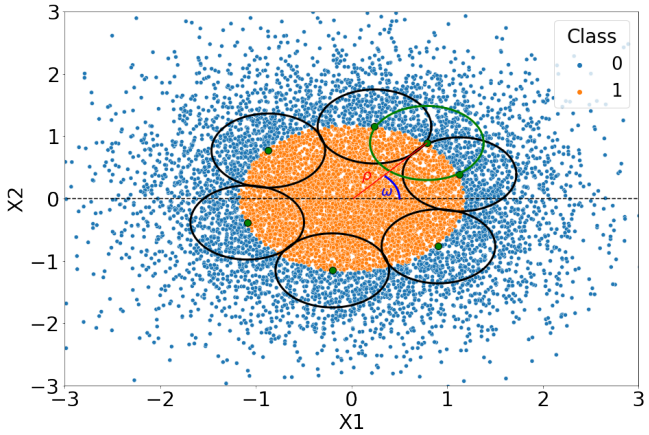

Figure 5: Specific example of local circle domains for synthetic data

the training set. Self-supervised out-of-distribution (SOoD) detection was introduced in [35] to alleviate both concepts shift and covariate shift. The method improves the predictive performance of the system by minimizing images' predictive entropy of unlabeled entries. This results in detaching in-distribution examples from OoDs on the target data domain. Active learning-based strategies was proposed in [36] to detect concept drift, in which case the current classifier is replaced with a new one. The proposed joint domain adaptation based on adversarial dynamic parameter learning (ADPL) in [37] employs two discriminators to keep a balance between the marginal distribution and the conditional distribution alignment (concept drift). It also considers the effects of the distance between both distributions (covariate shift).

Yet, all of these domain adaptation and active learning methods require to deploy sophisticated analytical models for the real-time processing of the information-rich input data. This often results in extremely complex DNNs, whose execution requires considerable memory and computing resources, which is a bottleneck for the MDs in IoT systems.

2.3. Online Domain Adaptation

The online domain adaptation proposed in this paper requires some preliminaries in the offline mode. Under the assumptions of the proposed online domain adaptation, different local adaptation domains (LADs) are formed, which include a group of samples following the same probability distribution. These predefined LADs are pre-trained by the ES. Notably, both the formation and the pre-training of the LADs are fulfilled offline. In the next step, for the online domain adaptation, a local classification domain (LCD) is formed to encompass the samples in the dataset. The LCD preferably starts its movement from one of the well-trained LADs in the dataset and smoothly moves around samples (shifts toward other LADs) following different movement patterns with various shift speeds.

Fig. 4 shows an example of the samples, labeled for classes 0 and 1, diffused in a dataset. Due to the high sporadic of the samples in the dataset, the LADs are formed where the density of the samples is high, i.e., where the samples follow the same distribution, and the model is pre-trained for each LAD. The green dotted circles in Fig. 4 illustrate the LCD in the dataset with the movement direction shown by green arrows. The movement is started from one particular LAD (centered in $(X_1, X_2) = (1, 2)$ in this particular example, and follows different movement patterns with various shift speeds to reach other LADs for more precise classifications. According to the efficiency of the models of two LADs that the LCD is moving between, the LAD model that results in the highest accuracy is chosen as the classification model for the LCD. Besides, the best offloading ratio, β, is set to offload the samples to the ES so that the accuracy of classification improves, while minimizing resource usage in the system.

2.3.1. Circular Example. Fig. 5 shows a specific example of the synthetic circular-based dataset, consisting of $N{=}10^4$ samples of synthetic data, each with $n{=}2$ features following a Gaussian distribution with zero mean and unit standard deviation, using the `make_gaussian_quantiles` function predefined in Python [38]. As seen in Fig. 5, the general dataset, including balanced samples with labels 0 (blue dots) and 1 (orange dots), is divided into n_c number of LADs each including balanced data as well. For each LCD, the models are pre-trained. Notably, the number of LADs plays an important role in improving classification accuracy. Similar to Fig. 4, the green dotted circle in Fig. 5 shows the moving LCD that starts from one LAD and follows an angular movement with either constant or random angle by keeping the same distance from the center (i.e., the average) of all data. Depending on the distribution of the samples, the moving LCD decides to use one of the adjacent LADs' models. Apart from the optimal number of LADs, the offloading ratio (β) is another parameter that must be chosen optimally in a way that the accuracy is maximized, while minimal resources are used for the offloading purposes. In the following section, we will discuss the corresponding optimization problem.

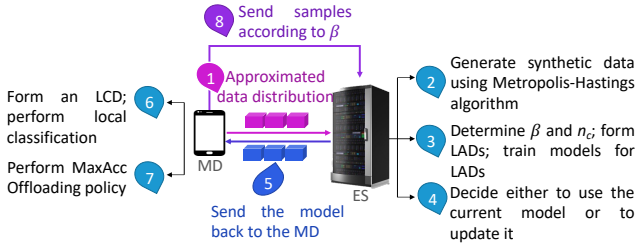

Figure 6: Specific architecture of the proposed framework including one MD-ES pair.

3. Online Adaptive Classification Framework

We consider a two-tier edge computing-enabled IoT system containing an edge layer that consists of an ES, and an IoT layer comprising M MDs. A reliable wireless medium interconnects the ES with the MDs, whereby the ES submits the training data for the local classification at the MDs, and also, the MDs offload samples to the ES. Due to the limited capacity of the communication channel, it must be used thriftily. Thus, the domain training data and the offloaded samples should be kept to a minimum. Fig. 6 shows the specific architecture of the proposed framework for one MD-ES pair. Without loss of generality, the architecture can be expanded to M MD-ES pairs.

It is assumed that an MD acquires N samples represented by $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, each of them being a vector of n real values, i.e., $\mathbf{x}_j \in \mathcal{X} \subseteq \mathbb{R}^n$ for all j. The objective of this architecture is to perform a binary classification, by which each $\mathbf{x}_j$ is assigned to either a positive or a negative class. We let $\delta : \mathcal{X} \mapsto \{0, 1\}$ be the function at the ES that returns the ground-truth label $y = \delta(\mathbf{x})$, according to which the set of all possible samples $\mathcal{X}$ can be partitioned into two disjoint subsets $\mathcal{X}_1$ and $\mathcal{X}_0 = \mathcal{X} \backslash \mathcal{X}_1$ corresponding to the positive and negative classes, respectively, where $\delta(\mathbf{x}) = k$ if and only if $\mathbf{x} \in \mathcal{X}_k$, $k \in \{0, 1\}$.

A simple classifier $\tilde{\delta}_\Theta$, parameterized by Θ, is deployed at the MD to approximate δ, where $\tilde{y} = \tilde{\delta}_\Theta$ is the predicted label by the MD. As the complexity of the classifier decreases, the approximation of the overall dataset deteriorates in terms of accuracy. Thus, the execution of the local classification $\tilde{\delta}_\Theta$ at the MD can achieve a poorer performance compared to δ. On the other hand, the external classification, i.e., offloading the data to the ES, boosts accuracy, yet uses infrastructure-level resources such as bandwidth and server time. Hence, a trade-off between local and external classifications needs to be addressed, so that accuracy is maximized, while reducing resource usage.

The proposed online adaptive classification framework aims to maximize accuracy by jointly employing local and external classifications, both with limited usage. In light of that, we first need to explore the values chosen for the system parameters, i.e.: (i) the fraction β of data offloaded to the ES; and (ii) the number n_c of LADs in the local online domain-constrained classification. Thereafter, online optimization is performed to adaptively classify the samples in real-time

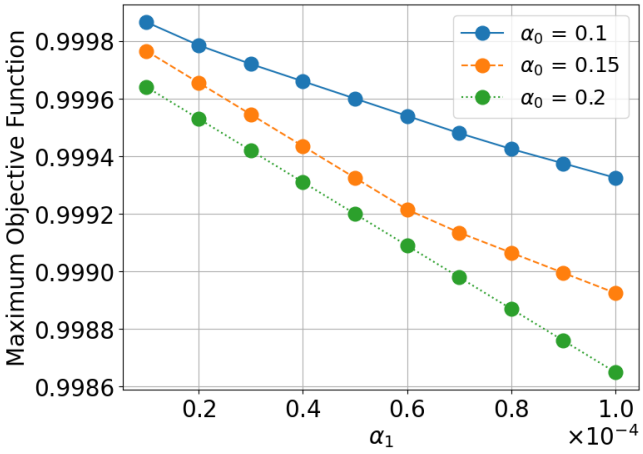

Figure 7: Maximum value of the objective function (2) for offline optimization.

using the predefined LADs derived from offline optimization in a way that the classification accuracy is boosted while minimizing resource usage. The rest of this section develops offline optimization to achieve the optimal values of β and n_c, followed by online optimization for the real-time domain adaptation applied to both synthetic and real datasets.

3.1. Static Optimization

The objective of this section is to optimize the overall accuracy of the synthetic dataset shown in Fig. 5 with respect to n_c and β so that the accuracy is maximized while minimizing the resource usage in the forms of β and n_c. Defining Υ as the accuracy of predictions, i.e., $Pr[\tilde{y} == y]$, that is a function of both β and n_c, the corresponding objective function (i.e., the reward) is given as

$$\max_{\beta, n_c} \Upsilon \tag{1}$$
$$s.t. \quad \beta \geqslant 0 \tag{C1}$$
$$n_c > 0 \tag{C2}$$

Since Υ is a function of both β and n_c, we use the method of Lagrange multipliers to solve (1). Therefore, we have

$$r = \Upsilon - \alpha_0 \beta - \alpha_1 n_c, \tag{2}$$

where α_0 and α_1 are shadow prices (Lagrange multipliers) of the optimization [39]. The idea is that we would like to optimize the accuracy Υ but without using extremely high values for β and n_c. So our specific choice is motivated by the fact that, ideally, it would be extremely easy to get high accuracy by simply letting β or n_c to grow indefinitely, and instead, we want to prioritize the choices that reach near-optimal accuracy with reasonably low values of β and n_c (as we will see, this is indeed possible).

To optimize the reward of (2), we implemented a binary classification for $N = 10^4$ samples, normally distributed.

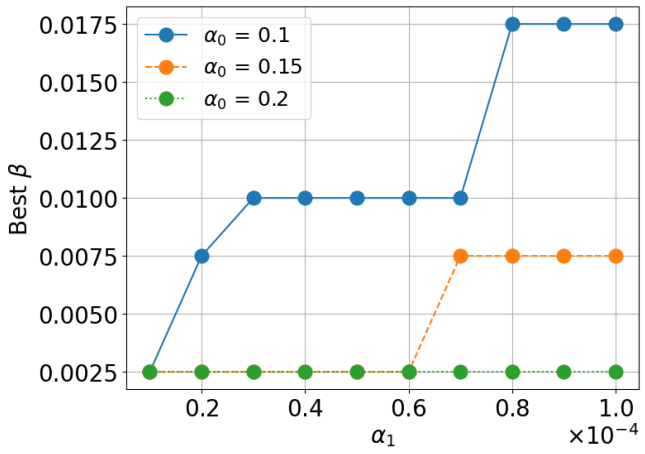

Figure 8: The best offloading ratio, β, for offline optimization.

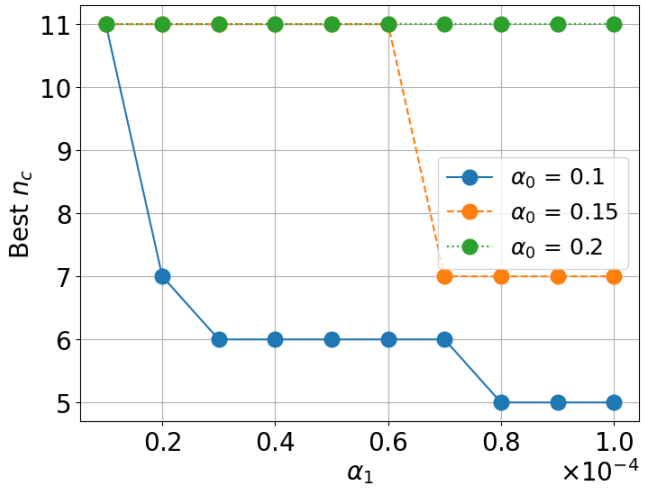

Figure 9: The best number of adaptation domains, n_c, for offline optimization.

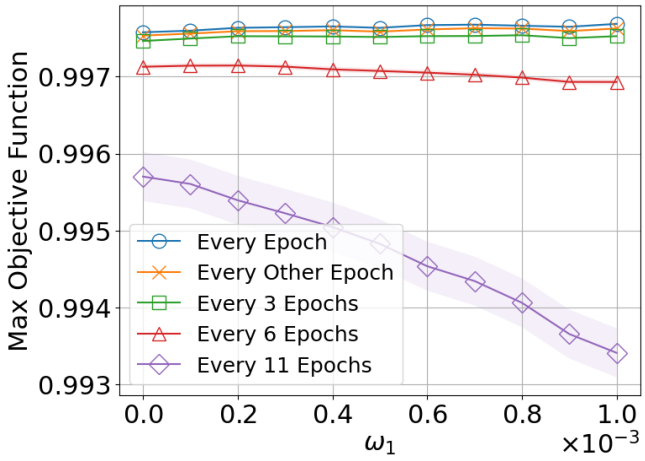

Figure 10: The effect of domain adaptation's delay on the maximum value of the objective function (3).

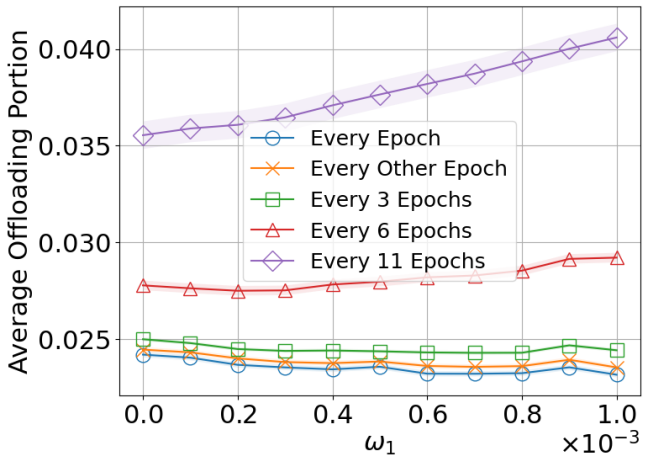

Figure 11: The effect of domain adaptation's delay on the best offloading ratio, β.

We simulated $\beta \in [0.25, 5] \times 10^{-2}$ stepped by 0.0025, $n_c \in \{k, k = 3, 4, \ldots, 12\}$, $\alpha_0 \in \{0.1, 0.15, 0.2\}$, $\alpha_1 \in \{k \times 10^{-5} : k = 1, 2, \ldots, 10\}$. Figs. 7 – 9 show the corresponding results. Fig. 7 illustrates that all values for α_0 provide high accuracy for the classification. However, increasing α_0 subsequently increases the impact of resource usage in the system, and thus, the reward defined in (2) decreases. Therefore, 0.1 is a suitable value for α_0. On the other hand, Fig. 7 shows that a lower α_1 improves the overall reward. Hence, α_1 is set to 10^{-5}.

Relying on the coefficients obtained above, Figs. 8 and 9 show that $\beta = 0.02$ and $n_c = 10$ are good parametric choices that can achieve the highest value for (2).

4. Dynamic Evaluation of Domain Drift

We now want to track the impact of domain adaptation in the system in real-time and decide how many samples need to be offloaded to converge the maximum accuracy in the system. The proposed method aims to minimize resource usage in the system while maximizing classification accuracy. Notably, in the dynamic approach, we set $n_c = 10$ and change only β, since the latter can be dynamically adapted, but n_c cannot. Thus, following the same approach as (1) in the previous section, the reward function is defined as

$$r = \Upsilon - \alpha_0 \beta \tag{3}$$

In the following, both synthetic and real-world datasets are studied via simulation to display the effectiveness of the proposed framework in improving system performance in terms of both accuracy and resource usage.

4.1. Results for Synthetic Dataset

Considering the synthetic dataset provided in Section 2.3.1, we initialize $n_c = 10$, $\alpha_0 = 0.1$, $\beta = 0.02$, and train the model for all 10 LADs. Then, a moving LCD is formed

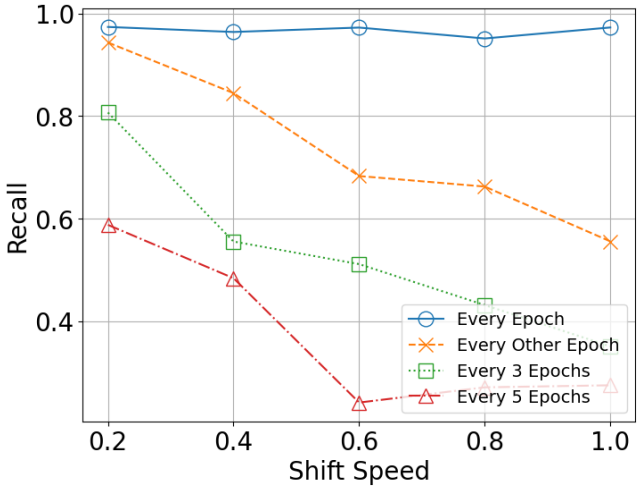

Figure 12: Recall without offloading the samples to the ES

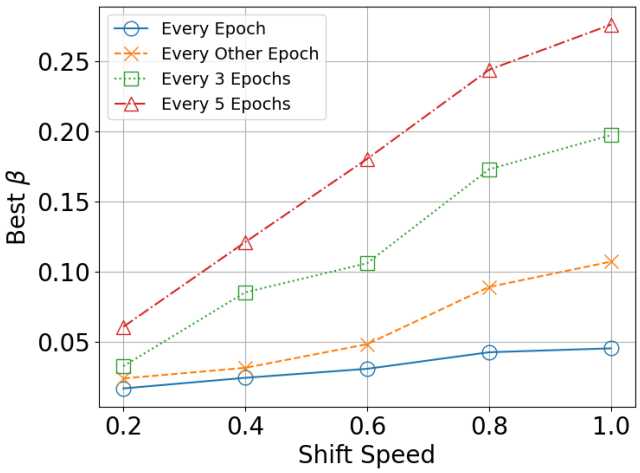

Figure 13: The best offloading ratio, β, for the UNITE dataset

starting from one predefined LAD and moves around samples with a rotation by an angle $\omega = \omega_0 + \omega_1$, where ω_0 is a constant angle set to $\pi/300$ in the following simulations, and ω_1 is a random angle following the Gaussian distribution with a mean of zero and a variance of σ^2, $\mathcal{N}(0, \sigma^2)$, where $\sigma \in \{10^{-4}k\pi, k = 0, 1, 2, \ldots, 20\}$. To show the impact of real-time domain adaptation, we compare a persistent update where the model is updated in every epoch with other scenarios, where the model is updated less frequently, i.e., every 2, 3, 6, 11 epochs.

Fig. 10 shows the corresponding values for (3) with a confidence interval of 95%, shown in highlights over the diagrams. The results imply that by widening the range of the randomness for ω_1, the LCD is more likely to move back and forth and with a faster shift speed. This increases the inefficiency of the current model for the classification of the samples. Thus, the model might need to be updated. However, the delay in updating the model decreases the probability of choosing the most adaptive model. As such, more samples

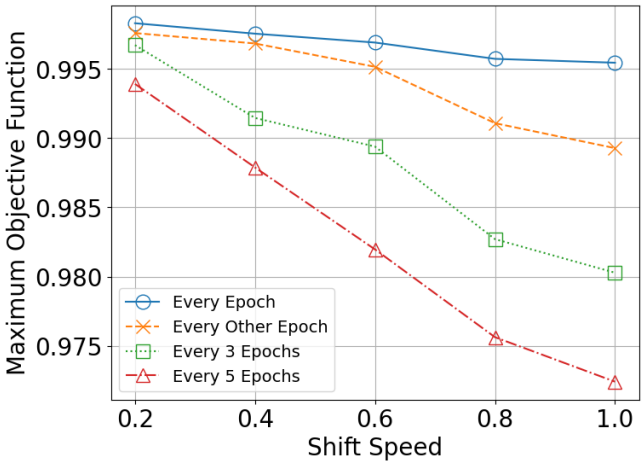

Figure 14: Maximum value of the objective function (4) for UNITE dataset

must be offloaded to the ES for classification. Thus, as seen in Fig. 11, resource usage increases, which subsequently results in the degradation of the reward value in (3). Moreover, the results indicate that any delay in domain adaptation (updating the model) leads to decreasing the classification accuracy, so that the persistent policy of updating at every epoch yields the most accurate classification.

4.2. Results for Real-World Dataset

We apply the proposed method to the UNITE dataset, including stressful and non-stressful samples, each with 13 features that are BPM (beats per minute, heart rate), IBI (inter-beat interval, average time interval between two successive heartbeats, called NN intervals), SDNN (standard deviation of NN intervals), SDSD (standard deviation of successive differences between adjacent NNs), RMSD (root mean square of successive differences between the adjacent NNs), pNN20 (the proportion of successive NNs greater than 20ms), pNN50 (the proportion of successive NNs greater than 50ms), MAD (median absolute deviation of NN intervals), SD1 and SD2 (standard deviations of the corresponding Poincaré plot), S (area of ellipse described by SD1 and SD2), and BR (Breathing Rate) [18].

Unlike the synthetic dataset where the data is mostly balanced in terms of the number of samples assigned to the considered labels in the dataset, the data in the real-world dataset is usually unbalanced. Therefore, rather than accuracy in (2), we use recall [22], denoted as Ξ, and defined as $TP/(TP + FN)$, where TP and FN stand for true positive and false negative, respectively. Therefore, the reward function for the UNITE dataset is expressed as

$$r = \Xi - \alpha_0 \beta \,. \tag{4}$$

We fulfill the simulations for the samples whose collection time starts at 12:00 am. The LCD's radius includes samples within five minutes and it is shifted with speeds of 0.2, 0.4,

0.6, 0.8, and 1.0, corresponding to the lowest to the fastest speeds. Also, we consider two different scenarios for the speed of updating the model: (*i*) persistent updating of the model, where the model is updated every epoch; and (*ii*) sporadic updating of the model, where the model is updated with different frequencies, i.e., every other epoch, every 3 epochs, and every 5 epochs.

Fig. 12 shows the recall value Ξ for different scenarios without offloading samples to the ES. As seen in the figure, persistent updating of the model leads to high recall, which decreases if less frequent updating is used. As illustrated by Fig. 13, this also leads to an increase in resource usage for the system. As a result, Fig. 14 shows that the reward provided in (4) decreases by reducing the speed of adaptation.

Furthermore, the results obtained in Fig. 12 indicate that by increasing the shift speed, the recall decreases as well. Indeed, when the shift speed is low, the distribution of data does not change significantly, and thus, the current model provides good accuracy to the samples. On the other hand, increasing the shift speed leads to a momentous change in the distribution of the samples. As a result, the current model is not able to classify the samples accurately and a higher share is required to be offloaded to the ES. Hence, as seen in Fig. 13, the offloading ratio increases. Subsequently, the reward value defined in (4) decreases for faster shifts.

5. Conclusions

In this paper, we proposed an online domain adaptive classification framework to perform a classification task leveraging the collaboration between an MD and the ES in a setting in which the distribution of the observed samples drifts over time. The proposed framework determines model updates and fine-grained offloading decisions.

Simulation results implemented on both synthetic and real datasets reveal that the proposed framework can improve the accuracy of simple classifiers deployed at MDs while reducing resource usage. According to the results obtained from experiments, it is important to frequently update the model, possibly at every epoch, to get a satisfactory performance. However, this comes at a system cost, especially in terms of latency to receive the updated model.

In future work, a novel deep reinforcement learning (DRL) model can be developed to infer the speed of data shift and evaluate the best update frequency of the model [40], so as to avoid imposing further delays derived from the updating procedure.

Acknowledgment

This work was supported by the Intel Corporation and the NSF grant (MLWiNS-2003237 and CCF-2140154) and Cisco.

References

[1] D. Stichling and B. Kleinjohann, "CV-SDF - a model for real-time computer vision applications," in *Proc. IEEE WACV*, 2002, pp. 325–329.

[2] S. Zhou, L. Wang, S. Zhang, Z. Wang, and W. Zhu, "Active gradual domain adaptation: Dataset and approach," *IEEE Trans. Multimedia*, vol. 24, pp. 1210–1220, 2022.

[3] T. Haubner, A. Brendel, and W. Kellermann, "End-to-end deep learning-based adaptation control for frequency-domain adaptive system identification," in *Proc. ICASSP*, 2022, pp. 766–770.

[4] I. Burago and M. Levorato, "Randomized edge-assisted on-sensor information selection for bandwidth-constrained systems," in *Proc. Asilomar Conf. Sign. Syst. Comput.*, 2018, pp. 1462–1469.

[5] Y. Yan and Q. Pei, "A robust deep-neural-network-based compressed model for mobile device assisted by edge server," *IEEE Access*, vol. 7, pp. 179 104–117, 2019.

[6] R. El Shawi, Y. Sherif, M. Al-Mallah, and S. Sakr, "Interpretability in healthcare a comparative study of local machine learning interpretability techniques," in *Proc. IEEE CBMS*, 2019, pp. 275–280.

[7] A. Messalas, C. Aridas, and Y. Kanellopoulos, "Evaluating MASHAP as a faster alternative to LIME for model-agnostic machine learning interpretability," in *Proc. IEEE BigData*, 2020, pp. 5777–5779.

[8] J. A. Neto, J. C. B. Fonseca, and K. Gama, "Towards online learning and concept drift for offloading complex event processing in the edge," in *Proc. IEEE/ACM SEC*, 2020, pp. 167–169.

[9] H. Flores, "Edge intelligence enabled by multi-device systems," in *Proc. IEEE PerCom Wkshps*, 2020.

[10] A. V. Guglielmi, M. Levorato, and L. Badia, "A Bayesian game theoretic approach to task offloading in edge and cloud computing," in *Proc. IEEE ICC Wkshps*, 2018.

[11] Z. Ding, J. Zhao, and J. Sun, "Model channel pruning method based on squeeze-and-excitation mechanism and upper quartile truncation," in *Proc. ISCSIC*, 2021, pp. 114–118.

[12] I. Burago and M. Levorato, "Cloud-assisted on-sensor observation classification in latency-impeded IoT systems," in *Proc. IEEE ISIT*, 2019, pp. 1462–1466.

[13] F. Shirin Abkenar, P. Ramezani, S. Iranmanesh, S. Murali, D. Chulerttiyawong, X. Wan, A. Jamalipour, and R. Raad, "A survey on mobility of edge computing networks in iot: State-of-the-art, architectures, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2329–2365, 2022.

[14] J. Qiu, R. Wang, A. Chakrabarti, R. Guérin, and C. Lu, "Adaptive edge offloading for image classification under rate limit," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 3886–3897, 2022.

[15] E. A. Tuli, J. M. Lee, and D.-S. Kim, "Federated learning-based computation offloading for low-bandwidth edge internet of things," in *Proc. APCC*, 2022, pp. 377–379.

[16] Z. Deng, K. Zhou, D. Li, J. He, Y.-Z. Song, and T. Xiang, "Dynamic instance domain adaptation," *IEEE Trans. Image Process.*, vol. 31, pp. 4585–4597, 2022.

[17] J. Xu, D. Vázquez, K. Mikolajczyk, and A. M. López, "Hierarchical online domain adaptation of deformable

part-based models," in *Proc. IEEE ICRA*, 2016, pp. 5536–5541.

[18] A. Tazarv, S. Labbaf, S. M. Reich, N. Dutt, A. M. Rahmani, and M. Levorato, "Personalized stress monitoring using wearable sensors in everyday settings," in *Proc. IEEE EMBC*, 2021, pp. 7332–7335.

[19] F. Shirin Abkenar, L. Badia, and M. Levorato, "Selective data offloading in edge computing for two-tier classification with local domain partitions," in *Proc. IEEE PerCom Wkshps*, 2023.

[20] M. Kenyeres and J. Kenyeres, "Performance analysis of generalized metropolis-hastings algorithm over mobile wireless sensor networks," in *Proc. Cybernetics & Informatics (K&I)*, 2020.

[21] I. Burago, M. Levorato, and S. Singh, "Semantic compression for edge-assisted systems," in *Proc. ITA*, 2017.

[22] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "Svms modeling for highly imbalanced classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 281–288, 2009.

[23] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, 2019.

[24] N. L. A. Ghani, I. A. Aziz, and M. Mehat, "Concept drift detection on unlabeled data streams: A systematic literature review," in *Proc. IEEE ICBDA*, 2020, pp. 61–65.

[25] G. Dharani, N. G. Nair, P. Satpathy, and J. Christopher, "Covariate shift: A review and analysis on classifiers," in *Proc. GCAT*, 2019.

[26] A. Hassan and A. Shaukat, "Covariate shift approach for invariant texture classification," in *Proc. IEEE MLSP*, 2013.

[27] M. Yamada, M. Sugiyama, and T. Matsui, "Covariate shift adaptation for semi-supervised speaker identification," in *Proc. IEEE ICASSP*, 2009, pp. 1661–1664.

[28] J. Pavez, C. Valle, and H. Allende, "A covariate shift method using approximated density ratios," in *Proc. ICPRS*, 2016.

[29] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 770–787, 2010.

[30] Z. Zhou, F. Wang, J. Yu, J. Ren, Z. Wang, and W. Gong, "Target-oriented semi-supervised domain adaptation for WiFi-based HAR," in *Proc. IEEE Infocom*, 2022, pp. 420–429.

[31] H. Daumé and D. Marcu, "Domain adaptation for statistical classifiers," *J. Artif. Intell. Res.*, vol. 26, no. 1, p. 101–126, 2006.

[32] M. M. Crawford, D. Tuia, and H. L. Yang, "Active learning: Any value for classification of remotely sensed data?" *Proc. IEEE*, vol. 101, no. 3, pp. 593–608, 2013.

[33] N. Jiang, Y. Deng, O. Simeone, and A. Nallanathan, "Online supervised learning for traffic load prediction in framed-ALOHA networks," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1778–1782, 2019.

[34] H. Zhang, W. Liu, and Q. Liu, "Reinforcement online active learning ensemble for drifting imbalanced data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3971–3983, Aug. 2020.

[35] B. Bozorgtabar, G. Vray, D. Mahapatra, and J.-P. Thiran, "Sood: Self-supervised out-of-distribution detection under domain shift for multi-class colorectal cancer tissue types," in *Proc. IEEE/CVF ICCVW*, 2021, pp. 3317–3326.

[36] B. Krawczyk, B. Pfahringer, and M. Woźniak, "Combining active learning with concept drift detection for data stream mining," in *Proc. IEEE BigData*, 2018, pp. 2239–2244.

[37] Y. Yuan, Y. Li, Z. Zhu, R. Li, and X. Gu, "Joint domain adaptation based on adversarial dynamic parameter learning," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 5, no. 4, pp. 714–723, 2021.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learning Res.*, vol. 12, pp. 2825–2830, 2011.

[39] L. Badia, S. Merlin, A. Zanella, and M. Zorzi, "Pricing VoWLAN services through a micro-economic framework," *IEEE Wireless Commun.*, vol. 13, no. 1, pp. 6–13, 2006.

[40] A. Chowdhury, S. A. Raut, and H. S. Narman, "DA-DRLS: Drift adaptive deep reinforcement learning based scheduling for IoT resource management," *J. Netw. Comp. Appl.*, vol. 138, pp. 51–65, 2019.