AI in 5G: The Case of Online Distributed Transfer Learning over Edge Networks

Yulan Yuan[1], Lei Jiao[2], Konglin Zhu[1,3], Xiaojun Lin[4], Lin Zhang[1]
[1]Beijing University of Posts and Telecommunications, China [2]University of Oregon, USA
[3]Purple Mountain Laboratories, China [4]Purdue University, USA

***Abstract*—Transfer learning does not train from scratch but leverages existing models to help train the new model of better accuracy. Unfortunately, realizing transfer learning in distributed cloud-edge networks faces critical challenges such as online training, uncertain network environments, time-coupled control decisions, and the balance between resource consumption and model accuracy. We formulate distributed transfer learning as a non-linear mixed-integer program of long-term cost optimization. We design polynomial-time online algorithms by exploiting the real-time trade-off between preserving previous decisions and applying new decisions, based on primal-dual one-shot solutions for each single time slot. While orchestrating model placement, data dispatching, and inference aggregation, our approach produces new models via combining the existing offline models and the online models being trained using weights adaptively updated based on inference upon data samples that dynamically arrive. Our approach provably incurs the number of inference mistakes no greater than a constant times that of the single best model in hindsight, and achieves a constant competitive ratio for the total cost. Evaluations have confirmed the superior performance of our approach compared to alternatives on real-world traces.**

I. Introduction

The 5G mobile communication networks are shaping the new paradigm of how users can explore and utilize Artificial Intelligence (AI). 5G networks consist of centralized gigantic data centers (referred to as "clouds") in the core [1], [2], and distributed cellular base stations often strengthened by co-located micro data centers or computing servers (referred to as "edges") as in Multi-access Edge Computing (MEC). AI models can be trained in the cloud using abundant data and dispatched to edges to serve users' inference requests [3], [4].

One fundamental problem of AI in 5G is that, as time goes, AI models often have varying or even decaying accuracies. First, the underlying data distribution, i.e., the relation between data's features and labels, drifts due to the non-stationary nature of the data and the environment [5], [6]. Under such "concept drifts", AI models which capture the relation between existing data's features and labels may not capture that for the new data. Second, the underlying data distribution may differ across communities or areas and AI models trained by different data may capture different relations between data's features and labels. As users move, their requests, which used to be served by the models at a previous edge, may not be properly resolved by the models at the new edge [7].

This work was supported in part by the Hunan Province Science and Technology Project Funds under Grant 2018TP1036, in part by the Key Laboratory of Modern Measurement & Control Technology, Ministry of Education, Beijing Information Science & Technology University under Grant KF20211123202, and in part by the U.S. National Science Foundation under Grants CNS-2047719 and CNS-2113893. Corresponding authors are Lei Jiao and Konglin Zhu.

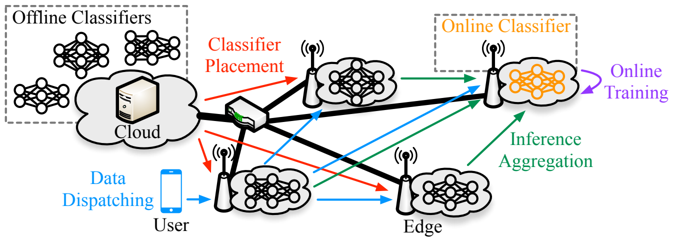

Fig. 1: Distributed Transfer Learning in Edge Networks

Transfer learning [8], [9] seems a promising solution to this problem. As existing AI models become less accurate, rather than dropping them and building new models from scratch, one can leverage existing models to help build the new models, i.e., transferring "knowledge" from existing models (i.e., *offline classifiers*) and models being trained (i.e., *online classifiers*) to the new model which is a combination of them, especially when the new data alone may be insufficient for training or when the underlying data distribution may possess periodical patterns. Yet, this approach faces critical challenges in 5G.

First, it is non-trivial to design and realize transfer learning in a distributed manner. While it is desired to keep data samples within the local edge networks without sending them to the remote cloud due to privacy and performance (e.g., traffic localization), extracting knowledge from each of the potentially heterogeneous offline classifiers, which may reside across different edges, to train the online classifier upon data samples that dynamically arrive requires to make a comprehensive set of control decisions, such as classifier placement, data sample dispatching, inference aggregation, and training parameter update, as exhibited in Fig. 1. Besides, distributed transfer learning also needs to ensure the quality or accuracy of the new classifiers being produced. It is challenging to make and navigate the trade-offs of these intertwined decisions.

Second, it is not an easy task to make distributed transfer learning cost-efficient in an online manner. Transfer learning needs to operate continuously as needed in the uncertain cloud-edge environment, where the operational cost of edges, the delay between edges, and the available capacity of each edge can vary unpredictably [7], [10]. We thus need to control the system on the fly to pursue the long-term optimization without observing any future inputs. This is particularly hard due to time-coupled decisions [11], [12] caused by selecting the edge to download the offline classifier from the cloud or host the online classifier to be created. For instance, hosting a classifier on a local edge now will save "start-up" cost of downloading the classifier and re-instantiating the edge environment if it is also needed here in the next time slot, but will waste the operational cost if this classifier turns out to be unwanted (e.g., if there is another cheaper edge for it) in the next time slot.

Existing research falls insufficient for addressing the aforementioned challenges. Some [7], [13]–[17] studied transfer learning's performance, efficiency, and accuracy, but have never considered resource or cost overhead, not to mention in distributed cloud-edge environments or in an online manner. Others [18]–[24] focused on resource utilization, job scheduling, and various optimizations for cloud and edge networks and related applications, but none of them has investigated distributed transfer learning. To the best of our knowledge, this work is the first to combine these two lines of research.

In this paper, we model and formulate distributed transfer learning over edge networks as a long-term cost optimization problem, which captures the operational cost of hosting offline and online classifiers, the start-up cost of downloading offline classifiers and instantiating local environments, the delay of dispatching data samples and aggregating inference results, as well as the number of mistakes incurred by applying the combined classifier to conducting inference upon data samples. Our problem is a non-linear mixed-integer program, which is NP-hard, makes no assumption on the dynamism of all the inputs, and allows arbitrary offline classifiers.

We then design a set of polynomial-time online algorithms to solve this problem. We firstly design a primal-dual-based Algorithm 1 to solve the one-shot problem to place offline classifiers along with data dispatching and inference aggregation, assuming all other control decisions have been made. We afterwards design Algorithm 2 to balance the new decisions from Algorithm 1 against the most recent previous decisions of the offline classifier placement, assuming the online classifier placement is given. Our rationale is to determine whether to switch to the new placement through a real-time comparison between the current start-up cost of switching to this new placement against the cumulative non-start-up cost of continuing to stay at the previous placement. Based on a similar idea, we also design Algorithm 3 to determine the online classifier placement with the cost associated to the decisions from Algorithm 2. Having a weight for each classifier, we further design Algorithm 4 which, at each time slot, invokes Algorithm 3 to set up the distributed transfer learning, decreases the weights of those classifiers with poor accuracy in terms of inference mistakes upon data samples that arrive dynamically, and updates the online classifier based on the loss it incurs. The final combined classifier per data sample is hence the weighted sum of existing offline classifiers and the online classifier being trained upon that data sample.

We have rigorously proved that our approach can guarantee an upper bound on the number of the mistakes made by the combined classifiers' inference results compared to the ground truth, which is in terms of the logarithm of the number of the offline classifiers, against the number of the mistakes made by the single best classifier in hindsight. Our approach also leads to a parameterized-constant competitive ratio for the total cost against the offline optimum which assumes all the inputs over the entire time horizon are to be observed all at once in prior.

We finally conduct extensive evaluations using real-world data traces. We utilize London's underground stations [25] with their dynamic passenger traffic to mimic users' inference requests, the geographical distance [26] to estimate the delay between edges, and the wholesale electricity price [27] to serve as the unit operational cost, respectively. We also use the text classification dataset `20Newsgroups` [28] with 8843 data samples for transfer learning. We observe multiple results: (i) our approach saves up to 60.6% total cost in the long run, compared to existing algorithms, and performs best by achieving up to 63.1% cost reduction when varying the weight of different components in the total cost; (ii) our approach also guarantees model quality in terms of lower mistake rates compared to state-of-the-art transfer learning approaches; and (iii) our algorithms execute fast on an commodity computer, finishing within 4.7 seconds per 15-minute-long single time slot, showing high efficiency and scalability.

II. Models and Problem Formulation

A. System Models

Edge Computing Networks: We study the system over a series of consecutive time slots $\mathcal{T} = \{1, ..., T\}$, corresponding to our decision-making frequency. We consider a set of geographically distributed edges $\mathcal{I}$, where an "edge" refers to a cellular base station or a WiFi access point equipped with a micro data center or a server cluster. The edges are connected to one another, and also to a common cloud via backhaul networks. For $i, j \in \mathcal{I}$ and $t \in \mathcal{T}$, we use d_{ij}^t to denote the delay between the edge i and the edge j at the time slot t, and use D_i^t to denote the available capacity of the edge i at the time slot t. We also consider that this edge environment provides a virtual machine (VM) or a container to host and run each offline or online classifier, as elaborated below.

Offline and Online Classifiers: The cloud maintains a set $\mathcal{K}$ of pre-trained "offline" classifiers that are to be downloaded to the edges to serve users with ultra-low latency and used to train a single "online" classifier being updated continuously to accommodate any concept drift. For $k \in \mathcal{K}$, $i \in \mathcal{I}$, and $t \in \mathcal{T}$, we use a_{ki}^t to denote the operational cost (e.g., electricity consumption) of hosting offline classifier k on edge i at t, use b_i^t to denote the operational cost of hosting the online classifier on edge i at t, use c_{ki} to denote the "start-up" cost of offline classifier k on edge i, including the cost (e.g., traffic or bandwidth consumption) of downloading offline classifier k from the cloud to edge i and the cost (e.g., lead time, system oscillation) of booting and preparing the VM or container on edge i, and use c_i to denote the "start-up" cost of the online classifier on edge i, which only includes booting and preparing the VM or container on edge i. The online classifier is directly

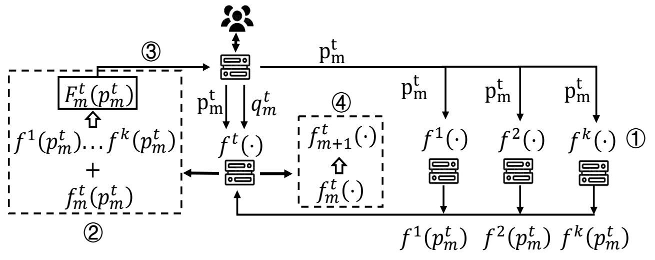

Fig. 2: Distributed Transfer Learning per Data Sample

created on edge using the training data samples per time slot, rather than being downloaded from the cloud. We also use $f^k(\cdot)$ to denote the "decision function" of the offline classifier k. Besides, we use $f_m^t(\cdot)$ to denote the decision function of the online classifier that is trained at the time slot t for the data sample m. Note that we write $f_m^t(\cdot)$ instead of $f^t(\cdot)$, because the single online classifier is being updated per data sample m during transfer learning, further elaborated as below.

Data Samples: We use $\mathcal{M}^t = \{1, ..., M_t\}$ to denote the data samples that arrive at the system at the time slot t from users. Each single data sample $m \in \mathcal{M}^t$ is represented as (p_m^t, q_m^t), where p_m^t refers to its feature values and q_m^t refers to its ground-truth label. Without loss of generality, we assume $q_m^t \in \{-1, 1\}, \forall m, \forall t$. We emphasize that q_m^t is only observable right after we conduct the inference for m using our offline and online classifiers. We also note that any data sample m may arrive at one edge but be dispatched to a different edge to do the inference. We use d_{mi}^t to represent the delay between the edge where the data sample m arrives and the edge i at t.

Distributed Transfer Learning: At the time slot t, as the data sample m arrives at the system, we design distributed transfer learning that works as follows, also shown in Fig. 2:

- *Step 1:* The data sample m with its feature value p_m^t is dispatched to every edge that has the offline classifiers or the online classifier. Note that it only needs to be dispatched to an edge once even if an edge hosts multiple classifiers. Receiving p_m^t, every offline classifier k computes $f^k(p_m^t)$ and the online classifier computes $f_m^t(p_m^t)$.
- *Step 2:* The decisions from the offline classifiers are sent to the edge that maintains the online classifier with all the weights of all the classifiers to compute the inferred label as $F_m^t(p_m^t) = sign(\sum_i \sum_k z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))$ [29], where $sign(\cdot)$ returns 1 for a positive value, -1 for a negative value, and 0 for 0; z_{kim}^t is the weight for the offline classifier k on the edge i for p_m^t; and w_{im}^t is the weight for the online classifier on the edge i for p_m^t.
- *Step 3:* The inferred label $F_m^t(p_m^t)$ is then sent to the edge where the data sample m arrives originally, and is further sent back to the user.
- *Step 4:* The ground-truth label q_m^t arrives at that same edge, and is dispatched to the edge that has the online classifier and all the weights. There, the weight for each classifier is updated and the decision function itself of the online classifier is also updated, i.e., $f_m^t(\cdot)$ is updated to $f_{m+1}^t(\cdot)$ using the received decision results of offline classifiers (see details in Section IV-B).

Our distributed transfer learning then proceeds to the next data sample $m+1$ at the time slot t. Note that, in the above process, we "transfer knowledge" from the existing offline decision functions $f^k(\cdot), \forall k$ to the new decision function $F_m^t(\cdot)$ which is a combination of $f^k(\cdot), \forall k$ and the online decision function $f_m^t(\cdot)$ being trained upon each data sample.

Control Decisions: We concentrate on making the following control decisions in this paper. We use $x_{ki}^t \in \{1, 0\}$ to denote whether or not the offline classifier k is downloaded from the cloud and hosted on the edge i at the time slot t. We use $y_i^t \in \{1, 0\}$ to denote whether or not the online classifier is trained and hosted on the edge i at the time slot t. We $u_{mi}^t \in \{1, 0\}$ to denote whether or not to transfer data sample m from the edge where it arrives to the edge i at t, and use $v_{ij}^t \in \{1, 0\}$ to denote whether or not to transfer the decision results of offline classifiers from the edge i to the edge j at t. We also use $z_{kim}^t, w_{im}^t \in [0, 1]$ to denote the weight for the offline classifier k on the edge i and the weight for the online classifier on edge i, respectively, for the data sample m at the time slot t, also described in the above.

Cost of Transfer Learning: The cost of distributed transfer learning at any individual time slot t consists of multiple components: (1) the operational cost of hosting classifiers on edges: $\sum_i \sum_k a_{ki}^t x_{ki}^t + \sum_i b_i^t y_i^t$; (2) the start-up cost of downloading classifiers from the cloud to edges and preparing the VMs or containers on edges: $\sum_i \sum_k c_{ki} \left[x_{ki}^t - x_{ki}^{t-1}\right]^+ + \sum_i c_i \left[y_i^t - y_i^{t-1}\right]^+$, where $[\cdot]^+ = \max\{\cdot, 0\}$; (3) the performance overhead incurred by running distributed transfer learning across edges, including the delay of dispatching data samples $\sum_i \sum_m d_{mi}^t u_{mi}^t$, the delay of transmitting decisions of offline classifiers $\sum_i \sum_j d_{ij}^t v_{ij}^t$, and the delay of transmitting the inferred label and the ground-truth label $2 \cdot \sum_i \sum_m d_{mi}^t y_i^t$. Note that the inferred label (or the ground-truth label) is only sent from (or to) the edge i that has the online classifier.

Mistakes of Transfer Learning: We consider the number of "mistakes" to measure the quality or accuracy of transfer learning [13], i.e., the number of occurrences where the inferred label does not match the ground-truth label. We denote the number of mistakes for all data samples of any single time slot t as $\sum_m \mathbb{I}\{sign[q_m^t \cdot (\sum_i \sum_k z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))] < 0\}$, where $\mathbb{I}\{\cdot\} = 1$ if the inequality condition contained is true and $\mathbb{I}\{\cdot\} = 0$ if not.

B. Problem Formulation, Challenges, and Goal

Problem Formulation: We minimize the sum of (i) the long-term total cost of transfer learning and (ii) the long-term total number of mistakes of transfer learning over time:

$$\begin{aligned}
\text{Min} \quad & C_1 = \textstyle\sum_t \sum_i \sum_k \left(a_{ki}^t x_{ki}^t + c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+\right) \\
& + \textstyle\sum_t \sum_i \left(b_i^t y_i^t + c_i[y_i^t - y_i^{t-1}]^+\right) \\
& + \textstyle\sum_t \sum_i \sum_m \left(d_{mi}^t \left(u_{mi}^t + 2y_i^t\right)\right) + \sum_t \sum_i \sum_j d_{ij}^t v_{ij}^t \\
& + \textstyle\sum_t \sum_m \mathbb{I}\Big\{sign\Big[q_m^t \cdot \big(\sum_i \sum_k z_{kim}^t sign(f^k(p_m^t)) \\
& + \textstyle\sum_i w_{im}^t sign(f_m^t(p_m^t))\big)\Big] < 0\Big\} \qquad (1)
\end{aligned}$$

$$
\begin{aligned}
\text{s.t.}\quad & z_{kim}^t \le y_i^t, \forall k,i,t,m, && \text{(1a)}\\
& w_{im}^t \le y_i^t, \forall i,t,m, && \text{(1b)}\\
& \textstyle\sum_i y_i^t = 1, \forall t, && \text{(1c)}\\
& \textstyle\sum_i x_{ki}^t = 1, \forall t,k, && \text{(1d)}\\
& \textstyle\sum_i (\sum_k z_{kim}^t + w_{im}^t) = 1, \forall m,t, && \text{(1e)}\\
& \textstyle\sum_k x_{ki}^t + y_i^t \le D_i^t, \forall i,t, && \text{(1f)}\\
& \textstyle\sum_j v_{ij}^t \ge x_{ki}^t, \forall k,i,t, && \text{(1g)}\\
& v_{ij}^t \le y_j^t, \forall i,j,t, && \text{(1h)}\\
& u_{mi}^t \ge x_{ki}^t, \forall k,i,m,t, && \text{(1i)}\\
& u_{mi}^t \ge y_i^t, \forall i,m,t, && \text{(1j)}\\
\text{var.}\quad & x_{ki}^t, y_i^t, u_{mi}^t, v_{ij}^t \in \{0,1\}, z_{kim}^t, w_{im}^t \in [0,1].
\end{aligned}
$$

Constraints (1a) and (1b) ensure that only the edge that hosts the online classifier can maintain all the weights for all the classifiers. Constraints (1c) and (1d) ensure that the online classifier can only be hosted by a single edge, and every offline classifier can only be hosted by a single edge. Constraint (1e) states that all the weights are normalized and their sum is one. Constraint (1f) respects the capacity of each edge. Constraints (1g) and (1h) guarantee that the decision computed by every offline classifier is transmitted to the edge that hosts the online classifier. Constraints (1i) and (1j) guarantee that every data sample is dispatched to every edge that hosts the classifier(s).

Challenges: It is non-trivial to solve the above optimization problem due to three challenges. First, we want to solve the problem in an *online* manner. That is, as time goes, at any time slot, we want to make control decisions for that time slot while observing only the inputs for that single time slot and no inputs for all the future time slots. For example, for the start-up cost $c_{ki}\left[x_{ki}^t - x_{ki}^{t-1}\right]^+$, we need to make $\boldsymbol{x}^{t-1}$ at $t-1$; however, at $t-1$, we have not made the decision of $\boldsymbol{x}^t$, without which it is difficult to make a good decision of $\boldsymbol{x}^{t-1}$ in order to optimize $c_{ki}\left[x_{ki}^t - x_{ki}^{t-1}\right]^+$. It is a similar case for $\boldsymbol{y}^{t-1}$ and its start-up cost $c_i\left[y_i^t - y_i^{t-1}\right]^+$. Second, the problem contains *nonlinear* terms, i.e., the number of the mistakes of transfer learning with $sign(\cdot)$ and $\mathbb{I}\{\cdot\}$ functions, which are intertwined with online training. While $f^k(p_m^t)$ can be observed as the offline classifiers are given and the data samples arrive, we need to determine how we should train or update the online classifier $f_m^t(\cdot)$ at t in our algorithms in addition to accommodating the nonlinearity. Third, the problem is *NP-hard*. The problem contains integer variables, and is actually NP-hard as it can be reduced to the existing uncapacitated facility location problem (if we only retain the variables $\boldsymbol{x}$ and $\boldsymbol{u}$ and the related terms in the formulation). The NP-hardness demands computationally efficient algorithms. It is not easy to achieve so in the offline setting, and it will be harder to do it in an online setting.

Goal: Our goal is to design polynomial-time approximation algorithms which make control decisions in an online manner and ensure that such decisions lead to a provable "competitive ratio". The competitive ratio r is a constant, which may contain parameters, to satisfy $C_1 \le rC_1^*$. Here, C_1 refers to the value of the objective function of the problem (1) evaluated with the solution produced by our online algorithms, and C_1^* refers to that evaluated with the optimal solution of (1) which were to be produced in the offline manner, when all the inputs were observed all at once before the start of the entire time horizon.

Algorithms Roadmap: First, to overcome intractability, we design a primal-dual approximation algorithm (i.e., Algorithm 1) to solve $\boldsymbol{x}^t$, $\boldsymbol{v}^t$ and $\boldsymbol{u}^t$ from the one-shot problem at any t, assuming $\boldsymbol{y}^t$ is given. Second, to overcome the challenge of being online, we design two algorithms (i.e., Algorithm 2, which invokes Algorithm 1, and Algorithm 3, which invokes Algorithm 2) to (re-)solve $\boldsymbol{x}^t$, $\boldsymbol{y}^t$, $\boldsymbol{v}^t$ and $\boldsymbol{u}^t$ at t, pursuing the dynamic trade-off between switching to a new decision and continuing to stay at the previous decision. Third, to accommodate nonlinearity and online training, we present the overall algorithm (i.e., Algorithm 4, which invokes Algorithm 3) to set the weights $\boldsymbol{z}^t$ and $\boldsymbol{w}^t$ of all classifiers given $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$ at t, and conduct online training by updating $f_m^t(\cdot)$ per data sample m. We elaborate these four algorithms and prove the performance guarantees in the next two sections.

III. Algorithm for One-shot Problem

In this section, we formulate the innermost problem of the offline classifier placement for each individual time slot, assuming all the other control decisions are pre-specified (and these decisions will be all made in the next section). We design a primal-dual algorithm, i.e., Algorithm 1, with a provable and guaranteed approximation ratio for this one-shot problem.

A. Innermost Problem

Consider C_1, i.e., the objective function of (1). If $\boldsymbol{y}^t$ is given, then at t, we can temporarily remove $\sum_m \mathbb{I}\{sign[q_m^t \cdot (\sum_i \sum_k z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))] < 0\}$, because as we will show, given $\boldsymbol{y}^t$ we will use Algorithm 4 in Section IV to determine $\boldsymbol{z}^t$ and $\boldsymbol{w}^t$ to satisfy Constraints (1a)$\sim$(1b). Also, if $\boldsymbol{y}^t$ is given, $\sum_i b_i^t y_i^t + \sum_i \sum_m 2d_{mi}^t y_i^t + \sum_i c_i\left[y_i^t - y_i^{t-1}\right]^+$ is known at t accordingly. Thus, we only need to focus on the following part of the problem (1):

$$
\begin{aligned}
\text{Min}\quad & C_2 = \textstyle\sum_{t,i,k}\left(a_{ki}^t x_{ki}^t + c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+\right)\\
& \quad + \textstyle\sum_{t,i,m} d_{mi}^t u_{mi}^t + \sum_{t,i} d_i^t v_i^t && \text{(2)}\\
\text{s.t.}\quad & \textstyle\sum_i x_{ki}^t = 1, \forall k,t, && \text{(2a)}\\
& \textstyle\sum_k x_{ki}^t \le Q_i^t, \forall i,t, && \text{(2b)}\\
& v_i^t \ge x_{ki}^t, \forall k,i,t, && \text{(2c)}\\
& u_{mi}^t \ge x_{ki}^t, \forall i,m,k,t, && \text{(2d)}\\
\text{var.}\quad & x_{ki}^t, v_i^t, u_{mi}^t \in \{0,1\}, && \text{(2g)}
\end{aligned}
$$

where $Q_i^t = D_i^t - y_i^t$. As $\boldsymbol{y}^t$ is given, we have replaced v_{ij}^t by v_i^t, d_{ij}^t by d_i^t, and $\sum_j v_{ij}^t \ge x_{ki}^t$ by $v_i^t \ge x_{ki}^t$ for simplification. This is because there is only one j where $y_j^t = 1$, and for all the other js, we have $y_j^t = 0$. So, for this specific j, we can set $v_{ij}^t = 1$, $\forall i$; for all the other js, we naturally have $v_{ij}^t = 0$, $\forall i$, due to (1h). v_{ij}^t is irrelevant to j now, but corresponds to v_i^t in a one-to-one manner; it is a similar case for d_{ij}^t and d_i^t.

To tackle the time-coupled term $\sum_{t,i,k} c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+$ in C_2 in an online manner, we will explore the real-time trade-off between keeping the "previous" decisions and applying the "new" decisions at each t, which will be discussed later in details. Now, in order to obtain such "new" decisions, we temporarily remove $\sum_{i,k} c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+$ in C_2 to construct the following one-shot problem at any individual t (where we have omitted the time index t to simplify the presentation). This is also what we call our "innermost problem":

$$\begin{aligned}
\text{Min}\quad & C_3 = \textstyle\sum_{i,k} a_{ki}x_{ki} + \sum_{i,m} d_{mi}u_{mi} + \sum_i d_i v_i && (3)\\
\text{s.t.}\quad & \textstyle\sum_i x_{ki} = 1, \forall k, && (3a)\\
& \textstyle\sum_k x_{ki} \le Q_i, \forall i, && (3b)\\
& v_i \ge x_{ki}, \forall k, i, && (3c)\\
& u_{mi} \ge x_{ki}, \forall i, m, k && (3d)\\
\text{var.}\quad & x_{ki}, d_i, u_{mi} \in \{0, 1\}. && (3e)
\end{aligned}$$

By relaxing the binary variables x_{ki}, d_i, u_{mi} into real domains and introducing dual variables $\lambda_k, \delta_i, \epsilon_{ki}, \phi_{kim}$ for (3a)$\sim$(3d), respectively, we write the Lagrange dual problem as

$$\begin{aligned}
\text{Max}\quad & D_3 = -\textstyle\sum_i Q_i\delta_i - \sum_k \lambda_k && (4)\\
\text{s.t.}\quad & a_{ik} + \delta_i + \lambda_k + \epsilon_{ki} + \textstyle\sum_m \phi_{kim} \ge 0, \forall i, k && (4a)\\
& \textstyle\sum_k \epsilon_{ki} \le d_i, \forall i, && (4b)\\
& \textstyle\sum_k \phi_{kim} \le d_{mi}, \forall m, i, && (4c)\\
\text{var.}\quad & \delta_i, \epsilon_{ki}, \phi_{kim} \ge 0, \lambda_k \in R. && (4d)
\end{aligned}$$

B. Primal-Dual Algorithm

We design Algorithm 1 to simultaneously construct integral feasible solutions to the primal problem (3) and feasible solutions to the dual problem (4). The idea of the primal-dual algorithm is to elevate the dual variable continuously until the dual constraint becomes tight (i.e., a constraint of the form of $ax \le b$ is considered tight when $ax = b$), and then the primal variable corresponding to that tight dual constraint can be set to a non-zero value in order to still satisfy the complementary slackness of the Karush-Kuhn-Tucker (KKT) conditions. Our Algorithm 1 is for each t, so t is omitted from the presentation.

We explain our Algorithm 1 following the above principle. By combing (4b) and (4c) with (4a), we can transform (4a) into $\sum_k(\lambda_k + \delta_i + a_{ik}) + d_i + \sum_m d_{mi} \ge 0$. Note that if this inequality is tight, then (4a) is tight. One sufficient condition to make this inequality hold is to make the following hold: $\lambda_k + \delta_i + a_{ik} + d_i/K + \sum_m d_{mi}/K \ge 0$, $\forall k$. Now, instead of increasing dual variables slowly, we can directly set the value of λ_k to $\lambda_k = -\min_{i\in\mathcal{I}}(\delta_i + a_{ik} + \frac{d_i}{K} + \frac{\sum_m d_{mi}}{K})$, which can make (4a) tight. Afterwards, the primal variable x_{ik}, which corresponds to (4a), can be set to 1; v_i and u_{mi} can be also set to 1 at this point. Based on what is stated above, we choose i as $i = argmin_{i\in\mathcal{I}}(a_{ki} + \delta_i + \frac{d_i}{K} + \frac{\sum_m d_{mi}}{K})$ for each k, as in Line 3 of our algorithm. Line 4 guarantees no violation of the constraint (3b), where ΔQ_i is defined as the remaining number of classifiers that edge i can host. We regard the dual variable δ_i as a reflection on the potential capability of edge i to host offline classifiers (i.e., the larger δ_i, the less likely i is to be selected). Thus, δ_i is increased for the selected edge i^* due to the decrease of ΔQ_{i^*} and remains intact otherwise. The update of δ_i is carefully designed for achieving low additive loss in approximation ratio, as in Line 8, where $\xi = \max_{i\in\mathcal{I}}\{Q_i\}$. Lines 9 and 10 update the dual variable λ_k and the primal variables x_{ki}, v_i, u_{mi}, respectively.

Algorithm 1: Offline Classifier One-Shot Placement

```
   Input: $a_{ki}$, $d_{mi}$, $d_i$, $y_i$, $Q_i = D_i - y_i$
1  Initialize: $\delta_i, \lambda_k, \epsilon_{ki}, \phi_{kim}, \Delta Q_i = 0$
2  for $k \in \mathcal{K}$ do
3      $i^+ = argmin_{i\in\mathcal{I}}(a_{ki} + \delta_i + \frac{d_i}{K} + \frac{\sum_m d_{mi}}{K})$;
4      while $\Delta Q_{i^+} + 1 > Q_{i^+}$ do
5          $\mathcal{I} = \mathcal{I} \backslash i^+$;
6          $i^+ = argmin_{i\in\mathcal{I}}(a_{ki} + \delta_i + \frac{d_i}{K} + \frac{\sum_m d_{mi}}{K})$;
7      $i^* = i^+$;
8      $\delta_{i^*} = \delta_{i^*}(1 + \frac{1}{Q_i}) + \frac{a_{i^*k} + d_{i^*}/K + \sum_m d_{mi^*}/K}{Q_i \xi}$;
9      $\lambda_k = -(\delta_{i^*} + a_{i^*k} + \frac{d_{i^*}}{K} + \frac{\sum_m d_{mi^*}}{K})$;
10     $x_{ki^*} = 1, u_{mi^*} = 1$;
11     $v_{i^*} = 1$ (i.e., $v_{i*j} = 1$ where $y_j = 1$);
   Output: $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}$
```

C. Performance Analysis

First, by Lemma 1, we demonstrate that Algorithm 1 is a polynomial-time algorithm with no violation of the constraints of the primal problem (3) and the dual problem (4). Second, by Theorem 1 on top of Lemma 1, we derive the approximation ratio r_1. We show $C_3 \le r_1 D_3 \le r_1 C_3^*$, where C_3 and D_3 refer to the objective function values of (3) and (4) evaluated with the feasible solutions returned by Algorithm 1; C_3^* refers to the optimal objective function value of the primal problem. Note that $D_3 \le C_3^*$ holds automatically due to duality.

Lemma 1. *Algorithm 1 returns feasible solutions to both the problem (3) and the problem (4) in the polynomial time.*

Proof. A solution is feasible for a problem if the solution satisfies the problem's constraints. For (3), (3a) is satisfied by Line 10. Lines 4$\sim$5 ensure no violation of the edge capacity limit, i.e., (3b). Once the edges to host offline classifiers are determined, (3c) and (3d) are also satisfied, following Line 10. Line 10 also guarantees (3e). For (4), this inequality $\sum_k(\lambda_k + \delta_i + a_{ik}) + d_i + \sum_m d_{mi} \ge 0$ is constructed based on (4a), (4b) and (4c), guaranteeing they are satisfied according to Lines 3$\sim$6. As for the time complexity of Algorithm 1, the *for* loop runs K times, and the *while* loop in the *for* loop runs at most I times according to its termination condition $\Delta Q_{i^+} + 1 > Q_{i^+}$. Thus, the total time complexity is $O(KI)$. □

Theorem 1. *Algorithm 1 is an r_1-approximation algorithm to the problem (3), i.e., $C_3 \le r_1 C_3^*$, where $r_1 = \frac{\xi}{\xi - 1}$.*

Proof. See Appendix A. □

IV. Algorithms of Long-Term Transfer Learning

In this section, we design Algorithms 2 and 3 that determine in real time the offline and online classifier placement with data dispatching and inference aggregation for each time

TABLE I: Summary of Notations

Notation	Definition
X^t_{SC}	$\sum_i c_{ki}[x^t_{ik} - x^{t-1}_{ik}]^+$
X^t_{-SC}	$\sum_{i,k} a^t_{ki} x^t_{ki} + \sum_{i,j} d^t_{ij} v^t_{ij} + \sum_{i,m} d^t_{mi} u^t_{mi}$
ΔX_{-SC}	$\sum_{\tau=\hat{t}}^{t-1} X^{\tau}_{-SC}$
Y^t_{SC}	$\sum_i c_i[y^t_i - y^{t-1}_i]^+$
Y^t_{-SC}	$X^t_{SC} + X^t_{-SC} + \sum_i b^t_i y^t_i + \sum_{i,m} 2d^t_{mi} y^t_i$
ΔY_{-SC}	$\sum_{\tau=\check{t}}^{t-1} Y^{\tau}_{-SC}$

slot. We also design Algorithm 4 for transfer learning upon each data sample as the classifier placement is determined dynamically. We theoretically analyze the number of mistakes of transfer learning and the competitive ratio for the total cost. We use some new notations in Table I to ease our presentation.

A. Online Algorithms for Classifier Placement

Our main rationale is to postpone changing the placement of the classifiers until "appropriate". That is, until the cumulative non-start-up cost (i.e., operational cost and delay of transfer learning incurred by continuing to host classifiers at previous locations) exceeds the current start-up cost (i.e., downloading cost and edge instantiation cost incurred by changing the classifier placement) times a constant which can be controlled.

We briefly explain our Algorithm 2.[1] In Line 2, ρ_2 is the controllable constant as aforementioned, and ΔX_{-SC} records the cumulative non-start-up cost from $\hat{t}$ to $t-1$, where $\hat{t}$ refers to the last time slot when the offline classifier placement changes before t. When the condition in Line 2 is satisfied, we adopt the decision returned by Algorithm 1; otherwise, we use the most recent decision as the current decision. We design Algorithm 3 in a similar spirit as for Algorithm 2. Specifically, we traverse I possible values of $\boldsymbol{y}^t$ in Lines 1∼2. Then, as in Line 4, only when ΔY_{-SC} exceeds $Y^t_{SC}(\widetilde{\boldsymbol{y}^t}, \boldsymbol{y}^{\check{t}})$ times ρ_1 will we adopt the new decision of $\boldsymbol{y}^t$, where $\check{t}$ indicates the last time slot of the online classifier placement change before t. Otherwise, in Line 10, we invoke Algorithm 2 given $\boldsymbol{y}^{\check{t}}$. We record all C^t_{-M} with the different $\boldsymbol{y}^t$, and find the minimum C^t_{-M} over all i with its corresponding decisions.

B. Online Algorithm for Transfer Learning

We propose our overall online transfer learning algorithm, i.e., Algorithm 4, to tie together every per-slot optimization of classifier placement and conduct the actual transfer learning process as data dynamically arrive. Our algorithm conducts online training in four steps: weights update, label inference, parameters update, and online classifier update. First, at each time slot, we invoke Algorithm 3 to find all the classifiers' placements in Line 4, and for the current data sample, we determine the weight for each classifier in Lines 7∼8. Then, for this data sample, we conduct the joint inference as a weighted sum of the static offline classifiers' results and the online classifier's result in Line 10. As receiving the ground-truth in Line 11, we next decrease the weights of those classifiers which misclassify instances so as to weaken their

[1]In Algorithm 2, we use symbols like $\widetilde{x}$ to refer to decisions obtained from Algorithm 1. Analogously, in Algorithm 3, we use symbols like $\widetilde{x}$ to represent decisions obtained from Algorithm 2. This should be clear from the context.

Algorithm 2: Conditional Offline Classifier Placement

```
   Input: $\boldsymbol{y}^t, \Delta X_{-SC}, \hat{t}$
1  given $\boldsymbol{y}^t$, get $\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{v}^t}, \widetilde{\boldsymbol{u}^t}$ by invoking Algorithm 1;
2  if $X^t_{SC}(\widetilde{\boldsymbol{x}^t}, \boldsymbol{x}^{\hat{t}}) \leq \frac{1}{\rho_2}\Delta X_{-SC}$ then
3      $\boldsymbol{x}^t = \widetilde{\boldsymbol{x}^t}$;
4      $\Delta X_{-SC} = X^t_{-SC}(\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{v}^t}, \widetilde{\boldsymbol{u}^t})$ ;
5      $\hat{t} = t$;
6  else
7      $\boldsymbol{x}^t = \boldsymbol{x}^{\hat{t}}$;
8      set $\boldsymbol{u}^t$ and $\boldsymbol{v}^t$ according to $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$;
9      $\Delta X_{-SC} = \Delta X_{-SC} + X^t_{-SC}(\boldsymbol{x}^{\hat{t}}, \boldsymbol{u}^t, \boldsymbol{v}^t)$;
   Output: $\boldsymbol{x}^t$, $\boldsymbol{v}^t$, $\boldsymbol{u}^t$, $\hat{t}$
```

Algorithm 3: Offline and Online Classifier Placement

```
   Input: $\Delta Y_{-SC}$, $\hat{t}$, $\check{t}$
1  for $i \in \mathcal{I}$ do
2      set $\widetilde{\boldsymbol{y}^t}$ as $\widetilde{y^t_i} = 1$, and $\widetilde{y^t_j} = 0$ for $j \neq i$;
3      given $\widetilde{\boldsymbol{y}^t}$, get $\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{v}^t}, \widetilde{\boldsymbol{u}^t}$ by invoking Algorithm 2;
4      if $Y^t_{SC}(\widetilde{\boldsymbol{y}^t}, \boldsymbol{y}^{\check{t}}) \leq \frac{1}{\rho_1}\Delta Y_{-SC}$ then
5          $\boldsymbol{y}^t = \widetilde{\boldsymbol{y}^t}$;
6          $\Delta Y_{-SC} = Y^t_{-SC}(\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{y}^t}, \widetilde{\boldsymbol{u}^t}, \widetilde{\boldsymbol{v}^t})$;
7          $\check{t} = t$;
8      else
9          $\boldsymbol{y}^t = \boldsymbol{y}^{\check{t}}$;
10         given $\boldsymbol{y}^t$, get $\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{v}^t}, \widetilde{\boldsymbol{u}^t}$ by invoking Algorithm
            2;
11         $\Delta Y_{-SC} = \Delta Y_{-SC} + Y^t_{-SC}(\widetilde{\boldsymbol{x}^t}, \boldsymbol{y}^t, \widetilde{\boldsymbol{u}^t}, \widetilde{\boldsymbol{v}^t})$;
12     $C^t_{-M} = Y^t_{-SC} + Y^t_{SC}$;
13 find the minimum $C^t_{-M}$ for all $i$ and its $\boldsymbol{x}^t, \boldsymbol{y}^t, \boldsymbol{u}^t, \boldsymbol{v}^t$;
   Output: $\boldsymbol{x}^t$, $\boldsymbol{y}^t$, $\boldsymbol{v}^t$, $\boldsymbol{u}^t$, $\hat{t}$, $\check{t}$
```

impact by updating the parameters used to determine the weights for the next data sample, as in Lines 13∼17. Finally, we update the online classifier itself based on its loss on the current data sample, as in Lines 19∼21. Here, we regard the data sample's feature p^t_m as a support vector and add it into the set of the support vectors of the online classifier: $f^t_{m+1} = f^t_m + \alpha^t_m q^t_m f^t_m(p^t_m, \cdot)$, where α^t_m is the coefficient for the support vector; $k(\cdot,\cdot)$ is the kernel function; and $\mathcal{F}$ is a constant trade-off value used to prevent the coefficient of the vector from being too large. Note that we allow offline classifiers of arbitrary and heterogenous types or formats, but without of loss of generality, we focus on training the online classifier as a Support Vector Machine in this paper. This is for concretizing our Algorithm 4, and does not impact our performance analysis and proofs.

C. Performance Analysis

We introduce some new notations to simplify our descriptions. We split C_1, the objective function of (1), as

Algorithm 4: Online Transfer Learning

```
Input: offline classifiers $f^{\mathcal{K}} = (f^1, f^2, ..., f^K)$,
       trade-off $\mathcal{F}$, and weight discount $\theta \in (0, 1)$
Initialize: $t = 1$, $\hat{t} = \check{t} = 0$, $f^0 = \emptyset$, $\zeta_{k1}^1 = \psi_1^1 = \frac{1}{K+1}$
for $t = 1, 2, ..., T$ do
    $f_0^t = f^{t-1}$;
    invoke Algorithm 3 to obtain $\boldsymbol{x}^t, \boldsymbol{y}^t$;
    for $m = 0, 1, ..., M^t$ do
        ▷ Weights update
        $z_{kmi}^t = \begin{cases} \zeta_{km}^t / (\sum_k \zeta_{km}^t + \psi_m^t), & y_i^t = 1 \\ 0, & y_i^t = 0 \end{cases}$
        $w_{mi}^t = \begin{cases} \psi_m^t / (\sum_k \zeta_{km}^t + \psi_m^t), & y_i^t = 1 \\ 0, & y_i^t = 0 \end{cases}$
        ▷ Label inference
        calculate inference:
          $\hat{q}_m^t = sign(\sum_{i,k} z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))$;
        receive ground-truth: $q_m^t \in \{-1, +1\}$;
        ▷ Parameters update
        for $k = 1, 2, ..., K$ do
            $\eta_{km}^t = \mathbb{I}\{sign[q_m^t \cdot sign(f^k(p_m^t))] < 0\}$;
            $\zeta_{k,m+1}^t = \zeta_{k,m}^t \theta^{\eta_{km}^t}$;
        $\gamma_m^t = \mathbb{I}\{sign[q_m^t \cdot sign(f_m^t(p_m^t))] < 0\}$;
        $\psi_{m+1}^t = \psi_m^t \theta^{\gamma_m^t}$;
        ▷ Online classifier update
        calculate loss: $l^t = [1 - q_m^t f_m^t(p_m^t)]^+$;
        if $l^t > 0$ then
            $f_{m+1}^t = f_m^t + \alpha_m^t q_m^t k(p_m^t, \cdot)$ where
              $\alpha_m^t = min\{\mathcal{F}, \frac{l^t}{k(p_m^t, p_m^t)}\}$;
```

$C_1 = \sum_t (C_M^t + C_{-M}^t)$, where $C_M^t = \sum_m \mathbb{I}\{sign[q_m^t \cdot (\sum_i \sum_k z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))] < 0\}$, and $C_{-M}^t = \sum_{i,k}(a_{ki}^t x_{ki}^t + c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+) + \sum_i (b_i^t y_i^t + c_i[y_i^t - y_i^{t-1}]^+) + \sum_i \sum_m (d_{mi}^t(u_{mi}^t + 2y_i^t)) + \sum_i \sum_j d_{ij}^t v_{ij}^t$. We also use C_M^{t*} and C_{-M}^{t*} to denote their optimal values. First, by Theorem 2, we exhibit that the total number of mistakes, i.e., $\sum_t C_M^t$, incurred by our transfer learning over time is no greater than a constant times the total number of mistakes incurred by the single best classifier (out of the offline classifiers and the online classifier), plus another constant. Second, by Theorem 3, we exhibit the competitive ratio of Algorithm 4. That is, we show $C_1 \leq rC_1^*$ and find r, where C_1 is the objective function value of the problem (1), evaluated with the solutions produced by Algorithm 4, and C_1^* is the optimal objective value. To do so, we derive $\sum_t C_M^t \leq r_2 \sum_t C_M^{t*}$ and $\sum_t C_{-M}^t \leq r_3 \sum_t C_{-M}^{t*}$, and then find r by $C_1 \leq r_2 \sum_t C_M^{t*} + r_3 \sum_t C_{-M}^{t*} \leq \max\{r_2, r_3\} C_1^* = rC_1^*$.

Theorem 2. *Algorithm 4 incurs the total number of mistakes as* $\sum_t C_M^t \leq \big(2 + 2\sqrt{2\ln(K+1)}\big) M_{min} + 2\ln(K+1)$*, where* $M_{min} = \min\{M^1, ..., M^k, ..., M^K, M^O\}$*,* $M^k = \sum_{t,m} \eta_{km}^t$*,* $M^O = \sum_{t,m} \gamma_m^t$*. Here,* η_{km}^t *and* γ_m^t *indicate whether the inference computed by the offline classifier* k *and the online classifier is wrong for the data sample* m *of the time slot* t*, respectively, as in Algorithm 4.*

Proof. See Appendix B. □

Theorem 3. *Algorithm 4 is an* r*-competitive online algorithm to the problem (1), i.e.,* $C_1 \leq rC_1^*$*, where* $r = \max\{r_2, r_3\}$*,* $r_2 = \frac{\ln(1/\theta) + \ln(K+1)}{1-\theta}$*, and* $r_3 = (1 + \frac{1}{\rho_1})(1 + \frac{1}{\rho_2} + D_{max}) r_1 \sigma$*. Here,* θ *is a constant in Algorithm 4;* ρ_1 *and* ρ_2 *are constants in Algorithms 2 and 3;* r_1 *is the approximation ratio of Algorithm 1, as in Theorem 1;* $D_{max} = \max\{\max_{i,k,t}\{\frac{b_i^t}{a_{k,i}^t K}\}, 2\}$*; and* $\sigma = \max_t\{\frac{\max_i\{\sum_k a_{ki} + \sum_j d_{ij} + \sum_m d_{mi}\}}{\min_i \sum_k a_{ki}}\}$*.*

Proof. See Appendix C. □

TABLE II: Sub-topics of Documents

Label of $+1$	Label of -1
comp.graphics	sci.crypt
comp.os.ms-windows.misc	sci.electronics
comp.sys.ibm.pc.hardware	sci.med
comp.sys.mac.hardware	sci.space
comp.windows.x	

V. Experimental Study

A. Experimental Settings

Transfer Learning Dataset and Classifiers: We use the text classification dataset `20Newsgroups` [28], which contains nearly 20,000 newsgroup documents with 61,188 unique words (i.e., features), associated to multiple topics. Each topic has several sub-topics, and each document has been labelled with one and only one sub-topic. We consider the 8843 documents associated to all the sub-topics of *comp* and *sci*, as shown in Table II. We treat all the sub-topics of *comp* as the label of $+1$, and all the sub-topics of *sci* as the label of -1. By matching one sub-topic from $+1$ with another sub-topic from -1, we have 20 pairs of sub-topics in total and for each of such pairs, we collect all their documents and train a Support Vector Machine (SVM) classifier. We select the classifier with *comp.windows.x* and *sci.space* as our online classifier (with a linear kernel function) to be trained during our experiments, and the rest 19 classifiers as our existing offline classifiers.

Edge Networks and Data Samples: We adopt the data of the 268 underground stations in London with dynamic passenger counts [25]. We choose the first 25 stations based on the total passenger count at each station, and envisage that each of such stations has an edge. We study the system for $T = 24$ hours and set the length of a single time slot as 15 minutes. We consider one document as one data sample. Based on the ratio of the passenger count at each edge in each time slot over the total passenger count across all edges and time slots, we spread the 8843 documents proportionally. We use the geographical distance [26] to estimate the network delay between the two edges. We set the dynamic operational cost as within $[2, 8]$ cents/kWh, following the wholesale electricity prices [27]. We vary the unit start-up cost as multiplied by a weight in order to

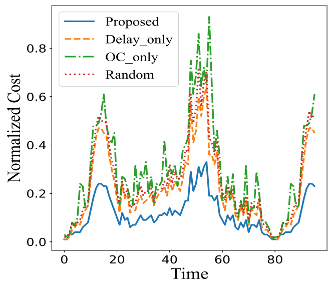

Fig. 3: Total Cost per Time Slot

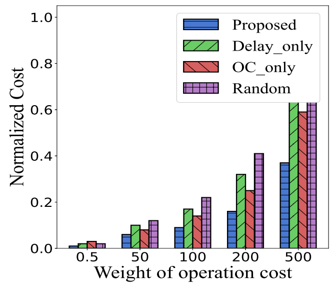

Fig. 4: Impact of Operational Cost

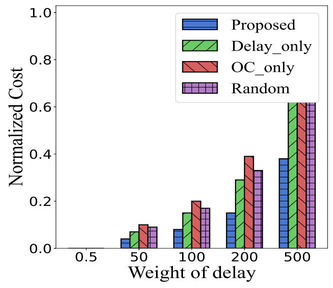

Fig. 5: Impact of Delay

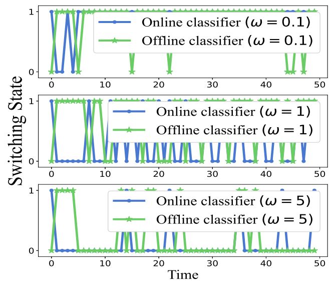

Fig. 6: Impact of Start-up Cost

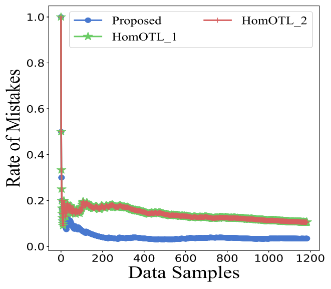

Fig. 7: Mistakes

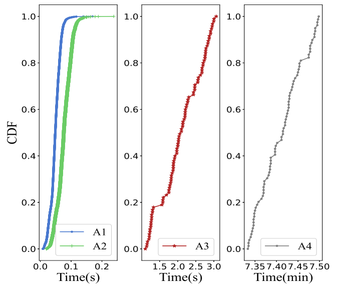

Fig. 8: Execution Time

demonstrate a spectrum of different results. We assume each edge can host 2$\sim$8 VMs or classifiers at most.

Algorithms and Implementation: We implement the following algorithms for comparison: (1) `Proposed` refers to our proposed online algorithms; (2) `Delay_only` chooses edges for classifiers only based on optimizing delay regardless of other costs, and directly downloads classifiers and/or preparing VMs as the one-shot optimum indicates (i.e., without postponing start-up); (3) `OC_only` chooses edges only based on optimizing operational cost regardless of others and without postponing state switching; (4) `Random` selects edges randomly without considering any cost-related optimization. We also implement (5) `HomOTL_1` and (6) `HomOTL_2`, which are state-of-the-art online transfer learning approaches [29]. Our implementation includes around 6,000 lines of Python codes, and we conduct all evaluations on a commodity laptop with an Intel Core i5 1.8-GHz CPU and 16-GB RAM.

B. Experimental Results

Total Cost: Fig. 3 visualizes the total cost of different algorithms per time slot over the entire time horizon. `Proposed` always produces the lowest total cost. Our approach achieves 47% less total cost than `Delay_only`, 60.6% less total cost than `OC_only`, and 53.8% less total cost than `Random`.

Impact of Operational Cost and Delay: Fig. 4 and Fig. 5 compares the total cost incurred by different algorithms as the weight of the operational cost and the weight of the delay varies, respectively. `Proposed` beats others no matter how these weights change. In Fig. 4, `OC_only` which only optimizes operational cost embodies more advantages compared to `Delay_only` and `Random`; yet, as operational cost becomes more important within the total cost, `Proposed` still outperforms it even under the largest weight value, by balancing the trade-off among the different cost components. The maximum cost reduction of `Proposed` is 51.3% compared to `Delay_only`, 37.3% compared to `OC_only`, and 63.1% compared to `Random`. In Fig. 5, `Proposed` still yields the maximum reduction of 48.3%, 61.5%, and 55.6% compared to `Delay_only`, `OC_only`, and `Random`, respectively.

Impact of Start-up Cost: Fig. 6 shows how the variation of the weight of the start-up cost, denoted as ω, influences the total number of the occurrences of state switching for the online classifier (i.e., preparing VMs) and the offline classifiers (i.e., downloading classifiers plus preparing VMs). In this figure, 1 means new and different decisions are applied to the current time slot; 0 means decisions of the previous time slot are applied to the current time slot. Our approach incurs more frequent state switching (i.e., there are more 1s than 0s) when ω is small because the state switching criterion can be satisfied easily, and leads to less frequent state switching as ω becomes larger, due to a stricter state switching criterion.

Mistakes of Transfer Learning: Fig. 7 presents the rate of the mistakes (i.e., the ratio of the number of incorrect inferences compared to ground truth over the total number of inferences). It shows `Proposed` is effective in transferring knowledge from existing classifiers, with an acceptable rate of mistakes lower than those of `HomOTL_1` and `HomOTL_2`.

Execution Time: Fig. 8 depicts the cumulative distribution of the execution time of each of our proposed algorithms. Algorithms 1$\sim$3 can be executed and finish within several seconds per 15-minute-long single time slot. For the time horizon of 24 hours, it takes no more than 7.5 minutes in total to finish running everything across all time slots, which includes the transfer learning process. Hence, our proposed algorithms are practically and computationally efficient.

VI. Related Work

Optimization of Transfer Learning: Daga et al. [7] designed a distributed transfer learning system for adapting to varying workload and data shift. Wu et al. [13] proposed online transfer learning for both homogeneous and heterogeneous environments. Yang et al. [14] focused on evaluating which source domain could be more suitable for transfer learning and the amount of knowledge transferred. Ding et al. [15] studied the minimization of the divergence among different sources by realizing cross-domain and cross-source knowledge transfer. Yang et al. [16] minimized domain discrepancy by promoting positive knowledge and decreasing the effect of unrelated instances. Yan et al. [17] introduced neural data servers to select the relevant transfer learning data.

These works focus on transfer learning, and almost all of them neglect resource usage and cost minimization from the systems perspective. The last work mentioned above is not typically for the cloud-edge and 5G environments.

Optimization over Edge Networks: Castellano et al. [18] explored optimal partitioning of shared resources in heterogeneous edge networks. Wang et al. [19] studied online resource allocation for edge computing in response to high dynamism of user mobility. Tu et al. [20] developed distributed learning optimization of the costs associated to device processing, offloading, and data disgarding. Meng et al. [21] optimized bandwidth and computing resource for deadline-restricted tasks. You et al. [22] explored dynamic resource provisioning in edge networks. Zhou et al. [23] proposed an online orchestration framework for cost-efficiency of cross-edge service functions. Han et al. [24] focused on minimizing the response time for latency-sensitive jobs in edge-cloud computing.

These works study cloud-edge systems and networks, but are unfortunately not about (distributed) transfer learning which has unique computing and and communication pattern. Thus, such existing research generally do not apply.

VII. Conclusion

Transfer learning is a useful and important technique, yet gets overlooked in the context of mobile communication networks. This paper aims to bridge this gap. We formulate a non-linear mixed-integer program via considering operational cost of edges, delay of networks, start-up cost of downloading classifiers and preparing local edge environments, and performance of transfer learning in terms of the mistakes of the combined classifiers. We design our online optimization algorithms and prove their theoretical guarantees. Using real-world data, we conduct extensive evaluations and have validated the practical efficacy and efficiency of our algorithms.

Appendix

A. *Proof of Theorem 1*

Let ΔP and ΔD denote the increment of the objective function in the problem (3) and (4), respectively, $\Delta D = -(\lambda_{k^*} + Q_{i^*}\Delta\delta_{i^*})$, where $\Delta\delta_{i^*} = \frac{\delta_{i^*}}{Q_i} + \frac{a_{i^*k}+d_{i^*}/K+\sum_m d_{mi^*}/K}{Q_i\xi}$ stands for the increment in δ_{i^*}. Thus, we have $\Delta D = -(\lambda_{k^*} + Q_{i^*}\Delta\delta_{i^*}) = -\lambda_{k^*} - Q_i(\frac{\delta_{i^*}}{Q_i} + \frac{a_{i^*k}+d_{i^*}/K+\sum_m d_{mi^*}/K}{Q_i\xi}) = (1-\frac{1}{\xi})(a_{i^*k^*} + \frac{d_i^*}{K} + \frac{\sum d_{mi^*}}{K}) = \frac{\xi-1}{\xi}\Delta P$. Due to $P^K = \sum_k(P^k - P^{k-1}) = \frac{\xi}{\xi-1}\sum_k(D^k - D^{k-1}) = \frac{\xi}{\xi-1}(D^K - D^0) = \frac{\xi}{\xi-1}D^K \le \frac{\xi}{\xi-1}C_3^{t*}$, followed by P^0 and D^0 are initialized with 0 and duality, we obtain $r_1 = \frac{\xi}{\xi-1}$.

B. *Proof of Theorem 2*

In order to simplify our proof, we introduce some new symbols. We use p_n to identify nth data sample, use $\omega_{k,n}$ to replace $\zeta_{k,m}^t$ and ψ_m^t, where $k \in \{1, ..., K, K+1\}$ (including the offline and online classifiers), use $P_{k,n} = \frac{\omega_{k,n}}{\sum_k \omega_{k,n}}$ to denote the normalized weight and $m_{k,n}$ to denote the mistakes of the classifier k, which are all updated as our Algorithm 4 shows.

Firstly, we prove that $\mathbb{I}\{q_n \cdot \hat{q}_n < 0\} = \mathbb{I}\{\sum_k P_{k,n}m_{k,n} > 0.5\}$. By assuming that there are only K_1 classifiers predict correctly (i.e., $sign(f^k(p_n)) = q_n$), we have $\hat{q}_n = sign(q_n(\sum_{k=1}^{K_1} P_{k,n} - \sum_{k=K_1+1}^{K+1} P_{k,n})$. Then, based on $\sum_k P_{k,n} = 1$, we obtain

$$q_n \cdot \hat{q}_n < 0 \iff \sum_{k=1}^{K1} P_{k,n} - \sum_{k=K_1+1}^{K+1} P_{k,n} < 0$$
$$\iff \sum_{k=K_1+1}^{K+1} P_{k,n} > 0.5 \iff \sum_k P_{k,n}m_{k,n} > 0.5.$$

Next, we have $\ln(\frac{\sum_k \omega_{k,n+1}}{\sum_k \omega_{k,n}}) = \ln(\sum_k P_{k,n}\theta^{m_{k,n}}) \le -(1-\theta)\sum_k P_{k,n}m_{k,n}$, thus $\ln(\frac{\sum_k \omega_{k,N}}{\sum_k \omega_{k,1}}) \le -(1-\theta)\sum_{k,n} P_{k,n}m_{k,n}$, and have $\ln(\frac{\sum_k \omega_{k,N}}{\sum_k \omega_{k,1}})$ lower bounded as $\ln(\frac{\sum_k \omega_{k,N}}{\sum_k \omega_{k,1}}) \ge \ln(\omega_{k,1}\theta^{\sum_n m_{n,k}}) = \ln(\frac{1}{K+1}) + \sum_n m_{k,n}\ln(\theta)$. Based on the above, we have $\sum_{k,n} P_{k,n}m_{k,n} \le \frac{\ln(1/\theta)\sum_n m_{k,n}+\ln(K+1)}{1-\theta} \le \frac{\ln(1/\theta)M_{min}+\ln(K+1)}{1-\theta}$. Finally, we upper bound the mistakes as $\sum_n \mathbb{I}\{q_n \cdot \hat{q}_n < 0\} \le 2\sum_{k,n} P_{k,n}m_{k,n} \le \frac{2\ln(1/\theta)M_{min}+2\ln(K+1)}{1-\theta}$. When we set $\theta = \sqrt{M_{min}}/(\sqrt{M_{min}} + \sqrt{\ln(K+1)})$, we further obtain $\big(2 + 2\sqrt{2\ln(K+1)}\big)M_{min} + 2\ln(K+1)$.

C. *Proof of Theorem 3*

Firstly, the start-up cost X_{SC}^t is no more than $\frac{1}{\rho_2}$ times ΔX_{-SC} within $[\hat{t}, t-1]$. Hence we have $\sum_t X_{SC}^t \le \frac{1}{\rho_2}\sum_t X_{-SC}^t$ as the worst case, i.e., the change of offline classifier placement always happens at each t. Similarly, we can obtain $\sum_t Y_{SC}^t \le \frac{1}{\rho_1}\sum_t Y_{-SC}^t$. Then, we have $\sum_t C_{-M}^t \le \sum_t Y_{SC}^t + \sum_t Y_{-SC}^t \le (1+\frac{1}{\rho_1})\sum_t Y_{-SC}^t$. Followed by the constraints of $\sum_i y_i^t = 1$ and $y_i^t \in \{0,1\}, \forall i,$ we have $\sum_{t,i}(b_i^t + \sum_m 2d_{mi}^t)y_i^t \le \max\{\max_{i,k,t}\{\frac{b_i^t}{a_{k,i}^t K}\}, 2\}\sum_t X_{-SC}^t$, and $\sum_t Y_{-SC}^t = \sum_t(X_{SC}^t + X_{-SC}^t) + \sum_{t,i}(b_i^t + \sum_m 2d_{mi}^t)y_i^t \le (1 + \frac{1}{\rho_2} + D_{max})\sum_t X_{-SC}^t$.

Next, we focus on $\frac{\max_{\boldsymbol{y}^t} X_{-SC}(Alg_2(\boldsymbol{y}^t))}{\min_{\boldsymbol{y}^t} X_{-SC}(Alg_2(\boldsymbol{y}^t))}$, where $Alg_2(\cdot)$ refers to Algorithm 2. We construct a new problem P_0 with $C_0 = \sum_{i,k} a_{ki}x_{ki} + \sum_{i,m} d_{mi}u_{mi} + \sum_{i,j} d_{ij}v_{ij}$ and the constraints of (1c)$\sim$(1d), (1f)$\sim$(1j), we can obtain $\frac{\max_{\boldsymbol{y}^t} X_{-SC}(Alg_2(\boldsymbol{y}^t))}{\min_{\boldsymbol{y}^t} X_{-SC}(Alg_2(\boldsymbol{y}^t))} = \frac{\text{Max}\, P_0}{\text{Min}\, P_0}$. Based on duality, we have $\frac{\text{Max}\, P_0}{\text{Min}\, P_0} \le \frac{P_1}{P_2} \le \frac{D_1}{D_2}$, where P_1 is the problems which maximizes C_0 with (1d)(1g)(1i), and $\sum_k x_{ki} \le Q_i, \forall i$, P_2 is minimization problem with the same constraints as P_1, and D_1 and D_2 are their dual problems. We introduce the dual variables $\bar{\lambda}_k, \bar{\delta}_i, \bar{\epsilon}_{kij}, \bar{\phi}_{kim}$ and $\widetilde{\lambda}_k, \widetilde{\delta}_i, \widetilde{\epsilon}_{kij}, \widetilde{\phi}_{kim}$ for P_1 and P_2, respectively. By choosing $\bar{\lambda}_k = a_{ki} + \frac{\sum_j d_{ij}}{K} + \frac{\sum_m d_{mi}}{K}, \bar{\delta}_i = 0, \bar{\epsilon}_{kij} = \frac{-d_{ij}}{K}, \bar{\phi}_{kim} = \frac{-d_{mi}}{K}$ and $\widetilde{\lambda}_k = -a_{ik}, \widetilde{\delta}_i = \widetilde{\epsilon}_{kij} = \widetilde{\phi}_{kim} = 0$, we obtain that $\frac{D_1}{D_2} \le \frac{\max_i\{\sum_k a_{ki}+\sum_j d_{ij}+\sum_m d_{mi}\}}{\min_i \sum_k a_{ki}}$, and define the ratio σ as $\max_t \frac{\max_i\{\sum_k a_{ki}+\sum_j d_{ij}+\sum_m d_{mi}\}}{\min_i \sum_k a_{ki}}$.

Based on the above, $\sum_t C_{-M}$ can be bounded as follows, $\sum_t C_{-M} = \sum_t Y_{SC} + \sum_t Y_{-SC} \le (1+\frac{1}{\rho_1})\sum_t Y_{-SC} \le (1+\frac{1}{\rho_1})(1+\frac{1}{\rho_2} + D_{max})\sum_t X_{-SC} \le (1+\frac{1}{\rho_1})(1+\frac{1}{\rho_2} + D_{max})\sigma\sum_t \min_{\boldsymbol{y}^t} X_{-SC}(Alg_2(\boldsymbol{y}^t)) \le r_3 \cdot \sum_t C_{-M}^*$, where $r_3 = (1+\frac{1}{\rho_1})(1+\frac{1}{\rho_2} + D_{max})\sigma r_1$. According to Theorem 2, we can obtain $\sum_t C_M \le M_{min}(\frac{2\ln(1/\theta)+2\ln(K+1)}{1-\theta}) = r_2 \cdot M_{min}$. Finally, we exhibit the competitive ratio r as follows, $C_1 = \sum_t(C_M + C_{-M}) \le r_2\sum_t C_M^{t*} + r_3\sum_t C_{-M}^{t*} \le \max\{r_2, r_3\}\, C_1^*$.

References

[1] A. Kiani, N. Ansari, and A. Khreishah, “Hierarchical capacity provisioning for fog computing,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 962–971, 2019.

[2] A. Alnoman and A. S. Anpalagan, “Computing-aware base station sleeping mechanism in h-cran-cloud-edge networks,” *IEEE Transactions on Cloud Computing*, 2019.

[3] A. Anand, G. De Veciana, and S. Shakkottai, “Joint scheduling of urllc and embb traffic in 5g wireless networks,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 477–490, 2020.

[4] J. Tang, B. Shim, and T. Q. Quek, “Service multiplexing and revenue maximization in sliced c-ran incorporated with urllc and multicast embb,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 881–895, 2019.

[5] Y. Sun, K. Tang, Z. Zhu, and X. Yao, “Concept drift adaptation by exploiting historical knowledge,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4822–4832, 2018.

[6] D. Brzezinski and J. Stefanowski, “Reacting to different types of concept drift: The accuracy updated ensemble algorithm,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2014.

[7] H. Daga, P. K. Nicholson, A. Gavrilovska, and D. Lugones, “Cartel: A system for collaborative transfer learning at the edge,” in *ACM SoCC*, 2019.

[8] T. Tommasi, F. Orabona, and B. Caputo, “Learning categories from few examples with multi model knowledge transfer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 928–941, 2013.

[9] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv, “Learn on source, refine on target: A model transfer learning framework with random forests,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1811–1824, 2016.

[10] H. McKay, N. Griffiths, P. Taylor, T. Damoulas, and Z. Xu, “Online transfer learning for concept drifting data streams.” in *BigMine @ ACM KDD*, 2019.

[11] S. Chen, L. Jiao, L. Wang, and F. Liu, “An online market mechanism for edge emergency demand response via cloudlet control,” in *IEEE INFOCOM*, 2019.

[12] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. Lau, “Moving big data to the cloud: An online cost-minimizing approach,” *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

[13] Q. Wu, H. Wu, X. Zhou, M. Tan, Y. Xu, Y. Yan, and T. Hao, “Online transfer learning with multiple homogeneous or heterogeneous sources,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 7, pp. 1494–1507, 2017.

[14] L. Yang, L. Jing, J. Yu, and M. K. Ng, “Learning transferred weights from co-occurrence data for heterogeneous transfer learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 11, pp. 2187–2200, 2016.

[15] Z. Ding, M. Shao, and Y. Fu, “Incomplete multisource transfer learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 2, pp. 310–323, 2018.

[16] 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.

[17] X. Yan, D. Acuna, and S. Fidler, “Neural data server: A large-scale search engine for transfer learning data,” in *IEEE/CVF CVPR*, 2020.

[18] G. Castellano, F. Esposito, and F. Risso, “A distributed orchestration algorithm for edge computing resources with guarantees,” in *IEEE INFOCOM*, 2019.

[19] L. Wang, L. Jiao, J. Li, J. Gedeon, and M. Mühlhäuser, “Moera: Mobility-agnostic online resource allocation for edge computing,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1843–1856, 2019.

[20] Y. Tu, Y. Ruan, S. Wagle, C. G. Brinton, and C. Joe-Wong, “Network-aware optimization of distributed learning for fog computing,” in *IEEE INFOCOM*, 2020.

[21] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, “Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing,” in *IEEE INFOCOM*, 2019.

[22] W. You, L. Jiao, S. Bhattacharya, and Y. Zhang, “Dynamic distributed edge resource provisioning via online learning across timescales,” in *IEEE SECON*, 2020.

[23] Z. Zhou, Q. Wu, and X. Chen, “Online orchestration of cross-edge service function chaining for cost-efficient edge computing,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.

[24] Z. Han, H. Tan, X.-Y. Li, S. H.-C. Jiang, Y. Li, and F. C. M. Lau, “Ondisc: Online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2472–2485, 2019.

[25] “London underground passenger counts data,” https://tfl.gov.uk/info-for/open-data-users/.

[26] “List of london underground stations,” https://wiki.openstreetmap.org/wiki/List_of_London_Underground_stations.

[27] “Hourly pricing,” https://hourlypricing.comed.com/live-prices/.

[28] “20newsgroups,” http://qwone.com/~jason/20Newsgroups/.

[29] P. Zhao, S. Hoi, J. Wang, and B. Li, “Online transfer learning,” *Artificial Intelligence*, vol. 216, pp. 76–102, 2014.