Hierarchical Inference with an Offload Queue

Srinivas Nomula* Parimal Parag* Ayalvadi Ganesh† Jaya Prakash Champati‡

IEEE INFOCOM 2026 - IEEE Conference on Computer Communications | 979-8-3315-4961-9/26/$31.00 ©2026 IEEE | DOI: 10.1109/INFOCOM59046.2026.11571740

***Abstract*—In a Hierarchical Inference (HI) system, an end device processes data samples using a local Deep Learning (DL) model. Only when the confidence of the local DL is less than a chosen threshold, it offloads the sample for remote DL inference on an edge server. By balancing the misclassifications and offloading costs, a HI system can improve accuracy and responsiveness and reduce bandwidth usage. To this end, prior work studied a HI learning problem for classification tasks, aiming to learn an optimal threshold on the confidence to minimize the cumulative costs. However, existing formulations considered (i) abstract offloading costs and (ii) assumed that the remote DL inference for an offloaded task is available by the end of each round. In contrast, considering that end devices are wirelessly connected to edge servers, we study a HI system with an offload queue where the offloading cost is implicitly modeled by the queue length. Also, we consider the practical setting, where the remote DL inference is obtained only after the task is served from the offload queue. For bounded but *unknown* arrival and service processes, we formulate the problem of average mismatch error minimization subject to queue stability. We propose a BOLD Hedge-Q policy based on Lyapunov optimization. The novelty of BOLD Hedge-Q lies in solving a delayed-feedback online learning problem — with sub-linear regret — that results from the Lyapunov drift-plus-penalty analysis. Given a penalty parameter $V \geq 1$, we show that under BOLD Hedge-Q, the queue is upper bounded by $V + \lambda_m$, and the mismatch error is within an additive factor of $O(1/V)$ from that of the optimal policy, where λ_m is the maximum number of arrivals in a round.**

***Index Terms*—deep learning, inference offloading, online learning, delayed feedback, Lyapunov stability.**

I. Introduction

Performing Deep Learning (DL) inference on end-user devices such as smartphones, drones and autonomous vehicles offers major advantages over a cloud-centric approach; processing data locally enhances privacy while reducing latency and bandwidth consumption. However, computational resource constraints of end devices necessitate the use of compact DL models (local DLs), which often suffer from lower accuracy, generalization capability, and robustness to noise [1]. In this context, Hierarchical Inference (HI) [2]–[9] has emerged as a promising technique for improving accuracy by leveraging a more accurate remote DL model at an edge server. In HI, inference is first performed on the local DL model for each data sample. If the local result is highly uncertain, the sample is offloaded to the remote DL server. HI offers two key advantages: 1) it balances the advantages of on-device inference with the high accuracy of remote inference, and 2) it supports flexible deployment, allowing off-the-shelf or proprietary models to be used at both the device and the server.

Prior work focused on classification tasks. Data samples are collected periodically by the end device, whose local DL model outputs a soft-max value for each class. The device may classify the sample locally or offload it for remote DL at a dedicated edge server. A loss or cost of 1 is incurred for each misclassification. The goal is to minimize the cost up to a given time horizon, T, subject to a constraint on the offloading cost. It is clear that the end device should classify samples locally into the class with the largest soft-max value. Moreover, if offloading costs are fixed, as assumed in [2]–[4], then samples whose confidence (largest soft-max value) falls below a threshold should be offloaded to the remote DL server (which is assumed to be better, or even error-free). The above references used labeled training data to compute the optimal threshold offline. Since inference-phase data distributions often deviate from training-phase distributions and offloading costs may vary, recent work [5]–[7] studied the *HI Learning (HIL)* problem of determining offloading thresholds online.

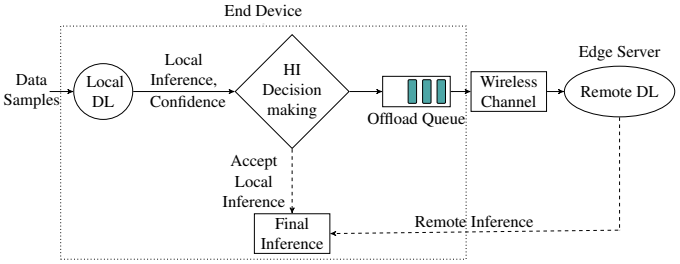


Fig. 1: A HI system with an offload queue at the end device.

Since HI is implemented in the inference phase, where ground-truth labels are unknown, prior work treated remote DL inference as a proxy for ground-truth. As a result, the misclassification loss is the *mismatch* between local and remote DL inference. Mismatch is unknown when the local inference is accepted, and needs to be estimated from samples offloaded to the remote DL. All prior work on HI assumes that mismatch is observed immediately upon offloading a sample. However, transmitting offloaded samples over a wireless channel and processing them at the edge server entail random delays, leading to delayed loss observation by the learning algorithm.

Authors* are with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, Karnataka 560012, IN. Author† is with the School of Mathematics, University of Bristol, Bristol BS8 1UG, UK. Author‡ is with the University of Victoria, Victoria BC V8P 5C2, CA.

This work is supported in part by EPSRC Grant EP/Y028732/1; in part by NSERC, Canada, Discovery Grant 54196-52150; in part by Qualcomm UR 6G India; in part by IBM-IISc Hybrid Cloud Lab (IIHCL) open research collaboration; in part by ANRF Grant CRG/2023/008854; in part by UKIERI Grant SPARC/2024/3927; and in part by Centre for Networked Intelligence (a Cisco CSR Initiative), IISc. All opinions are those of the authors and do not necessarily represent those of the funding agencies.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.

In this article, we study the HI system with an offload queue at the device to account for random transmission and processing delays before obtaining the remote DL inference; see Fig. 1. In our system, the offloading cost is determined by the queue length, in contrast to the abstract offloading costs considered in previous works. We consider a general discrete-time queuing model, where the *service time*, i.e., the duration for transmitting the sample, processing at the remote DL server and returning the inferred class, is generated from an arbitrary distribution. In contrast to previous works, which consider data samples arrive one per round or time slot, we consider a general arrival process, with the maximum number of arrivals per time slot bounded by λ_m. Our objective is to minimize the time-average expected mismatch error, subject to queue stability. We summarize our main contribution below.

A. Contributions

1) We study the HI system with an offload queue, and find an optimal fixed threshold policy θ_0 that minimizes mismatch error subject to queue stability.
2) We propose the BOLD Hedge-Q algorithm to solve the problem. The design of BOLD Hedge-Q is based on 1) the use of drift-plus-penalty analysis to decompose the infinite-horizon problem into solving a per-slot optimization problem in each time slot, and 2) the use of a reduction method for online learning with delayed feedback to solve the per-slot optimization problem.
3) For a finite horizon T, we prove that BOLD Hedge-Q incurs $O(\lambda_m((V/\lambda_m+1)\log|\mathcal{Z}|)^{\frac{1}{3}}T^{\frac{2}{3}})$ regret, where $\mathcal{Z}$ is the set of thresholds, $V \geqslant 1$ is a free parameter, and λ_m is the maximum number of arrivals in a round.
4) We further prove that, under BOLD Hedge-Q, the queue is upper bounded by $V+\lambda_m$, and the mismatch error is within an additive factor of $O(1/V)$ from the θ_0 policy.
5) Using simulation, we demonstrate that BOLD Hedge-Q achieves mismatch error close to zero and average queue length comparable to the Full-Offload policy when the arrival rate is below the service rate. In contrast, when the arrival rate exceeds the service rate, BOLD Hedge-Q maintains bounded queue length, albeit at the expense of increased mismatch error.

B. Related Work

Existing techniques for DL inference at the edge can be broadly classified into: *(i)* on-device inference [10], *(ii)* DNN-partitioning [11]–[13], *(iii)* inference load balancing [14]–[16], and *(iv)* Hierarchical Inference (HI) [5]. In [3], [9], HI was shown to achieve an accuracy-cost trade-off superior to the alternatives. As this relies on configuring the HI system with the optimal confidence threshold, the HIL problem has been gaining traction [5]–[7]. In [5], the authors considered fixed offloading cost and formulated HIL as a variant of the classical Prediction with Expert Advice (PEA) [17]. They proposed an algorithm and derived dataset-dependent regret bounds. The authors in [7] derived dataset-independent regret bounds and extended the analysis to a generalized setting where the offloading costs are random and potentially correlated. Both works assume that the offloading costs are revealed at the end of each round, even in rounds with no offload.

In [6], the authors studied the HI decision problem to minimize average mismatch error subject to a constraint on average offloading cost. They assumed that arrivals are Poisson, offloading costs are i.i.d., and are revealed only in rounds where there are offloads. Using a virtual queue for the offloading cost, the authors leveraged the Lyapunov optimization framework to transform the problem into a per-slot optimization problem with parameter V, which is solved using an online learning algorithm. However, their virtual queue stability proof is based on a strong assumption that, with high probability, their algorithm does not offload a data sample when the virtual queue size is large. Further, their regret bound depends on the inverse of the minimum value of the offloading cost. Furthermore, all the above works assume that the remote DL inference is observed immediately in the round where the sample is offloaded. In contrast, we study a HI system with general arrival and service processes. We consider a physical queue and ensure its stability by always accepting the local inference when the queue length exceeds V. We address the challenge of delays in observing the remote DL inference using a reduction method in online learning with feedback delays.

The problem we study is related to that of admission control in a queue [18]–[22], where every customer admitted into the queue yields a reward, while incurring a cost equal to their wait time in the queue. In contrast, in our problem, an inference task that is not offloaded may incur a mismatch error, but the expected loss is unknown and needs to be estimated from rounds in which inference tasks are offloaded.

Notation: We denote the set of first n positive integers by $[n]$, the set of nonnegative integers by $\mathbb{Z}_+$, the set of integers by $\mathbb{Z}$, the set of real numbers by $\mathbb{R}$, the set of functions from set A to set B by B^A, and the set of probability measures over a finite set B by $\mathcal{M}(B) \triangleq \{\alpha \in [0,1]^B : \sum_{b\in B}\alpha_b = 1\}$.

II. System Model

We assume a discrete-time model over time $t \in \mathbb{Z}_+$ and consider an edge AI system with an end device and an edge server. The end device and the server are connected via a wireless channel. At every time t, the data samples arrive at the device, and the task is to classify them into multiple classes. The device is equipped with a pre-trained compact DL model (local DL). The server hosts a larger, more accurate DL model (remote DL).

A. Task Arrivals and Service

Let M_t denote the number of classification tasks that arrived in time slot $t-1$ and are processed by local DL at the end of time slot $t-1$. We assume that the processing time of local DL is negligible. At the start of slot t, one needs to decide whether to offload these tasks to the remote server or not. We assume that $a : \Omega \to \{0,\dots,\lambda_m\}^{\mathbb{N}}$ be a random process with finite mean λ, variance σ_a^2, and maximum number of

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.

arrivals λ_m in each time slot t. The number of arrivals until time slot t is denoted $A_t \triangleq \sum_{s=1}^{t} a_s$. We denote the set of tasks that arrive in time slot $t-1$ as $\mathcal{A}_t \triangleq A_{t-1} + [a_t]$, We assume that the device maintains a queue of tasks to be offloaded with the task queue at time $t \in \mathbb{Z}_+$ denoted by Q_t. We denote the maximum number of tasks that can be served by the wireless channel from the queue in a time slot t by a random variable C_t, and assume that the random sequence $C : \Omega \to \mathbb{Z}_+^{\mathbb{N}}$ has a common finite and positive mean μ, variance σ_s^2, and maximum μ_m in each time slot t. We assume that the remote DL processing time is negligible[1], and thus the remote DL inference is available to the device once the task in the offload queue is served.

We consider that the statistics of the arrival and service processes are not known. Not all arriving tasks need to be offloaded, and hence it is possible that $\lambda > \mu$ in our setting.

B. Confidence Value and Mismatch Error

For the classification tasks arriving at the device, we denote the feature space by $\mathcal{X} \subseteq \mathbb{R}^d$ and the label set by $\mathcal{Y} \triangleq [c]$, where c is the number of classes. For each classification task $x \in \mathcal{X}$, a score $f_y(x) \in \mathbb{R}$ is provided for each output label $y \in \mathcal{Y}$ under typical DL classification models. As such, we denote the output of DL model as a map $f : \mathcal{X} \to \mathbb{R}^{\mathcal{Y}}$. We define a map $\hat{g} : \mathcal{X} \to \mathcal{M}(\mathcal{Y})$ that outputs a probability measure $\hat{g}(x) \in \mathcal{M}(\mathcal{Y})$ for all tasks $x \in \mathcal{X}$, defined in terms of DL output as $\hat{g}_y(x) \triangleq e^{f_y(x)} / \sum_{z \in \mathcal{Y}} e^{f_z(x)}$. For classification task $x \in \mathcal{X}$, the classification output $h(x) \in \mathcal{Y}$ is the one with the highest softmax value, and the confidence $Z(x) \in \mathcal{Z} \subseteq [0,1]$ is the corresponding softmax value, i.e.

$$h(x) \triangleq \arg\max_{y \in \mathcal{Y}} \hat{g}_y(x), \qquad Z(x) \triangleq \max_{y \in \mathcal{Y}} \hat{g}_y(x).$$

For the nth task, we denote the confidence value as $Z_n \triangleq Z(x_n)$. We assume that the confidence value sequence $Z : \Omega \to \mathcal{Z}^{\mathbb{N}}$ is *i.i.d.* with an unknown common distribution $F_Z : \mathcal{Z} \to [0,1]$ and probability mass function (PMF) $P_Z \in \mathcal{M}(\mathcal{Z})$. Further, we assume that the mismatch error sequence Φ is *i.i.d.* across all tasks from an unknown mismatch error probability $P_\Phi : \mathcal{X} \to [0,1]$.

Remark 1. Though we study the widely used maximum softmax value [23] as the confidence value for concrete instantiation, our policy and analytical results apply to any other confidence value computed on the score function.

Note that the DL models on devices are typically quantized, i.e., the weights and the score values are stored in 4, 8, or 16 bits to reduce the required memory storage [24]. Thus, $\mathcal{Z}$ is finite in practice. For example, an 8-bit quantized ResNet8 [25] has at most $|\mathcal{Z}| = 256$ unique maximum soft-max values. We denote the pre-trained local DL and remote DL classifiers as h_ℓ and h_r (both mappings in $\mathcal{Y}^{\mathcal{X}}$), respectively. A challenge in the edge AI system is that the ground-truth labels are not available in the inference phase. Therefore, as in [7], we assume that the remote DL inference is a proxy for the ground-truth label and use the mismatch between the local DL and remote DL as the cost of accepting the local inference. For a classification task $x \in \mathcal{X}$, we refer to this cost as the *mismatch error*, given by $\Phi(x) \triangleq \mathbb{1}_{\{h_\ell(x) \neq h_r(x)\}}$. For the nth task, we denote mismatch error as $\Phi_n \triangleq \Phi(x_n)$ and assume that the mismatch error sequence Φ is *i.i.d.* across all tasks from an unknown mismatch error probability $P_\Phi : \mathcal{X} \to [0,1]$. We note that Φ and Z are generally negatively correlated, but we may not know the exact relation due to the DL models' black-box nature.

[1]This assumption can be relaxed, though it introduces an additional term proportional to the remote DL processing time in the performance bounds.

C. Queue Evolution

Definition 1. Let $I_n \in \{0,1\}$ be the offloading indicator for task n. Then, the number of offloaded tasks in a slot t is denoted as

$$O_t \triangleq \sum_{n \in \mathcal{A}_t} I_n. \tag{1}$$

Definition 2 (Mismatch error)**.** Only a task that is not offloaded has a mismatch error, and hence mismatch error of task n is $\Phi_n \bar{I}_n$, and the aggregate mismatch error in a time-slot $t \in \mathbb{Z}_+$ is denoted by

$$E_t \triangleq \sum_{n \in \mathcal{A}_t} \Phi_n \bar{I}_n. \tag{2}$$

We define the mean mismatch error of classification tasks averaged over time horizon $[T]$ as $\bar{E}_T \triangleq \frac{1}{T} \sum_{t=1}^{T} \mathbb{E} E_t$.

Recall that C_t is the number of tasks that can be served by the remote server in slot t. We can write the task queue evolution for each time $t \in \mathbb{Z}_+$ as

$$Q_t = \max\left\{Q_{t-1} - C_t + O_t, 0\right\}. \tag{3}$$

We assume the initial task offload queue to be empty, i.e. $Q_0 \triangleq 0$. We denote the number of offloaded tasks and queue length, averaged over a time horizon $[T]$, as

$$\bar{O}_T \triangleq \frac{1}{T} \sum_{t=1}^{T} \mathbb{E} O_t, \qquad \bar{Q}_T \triangleq \frac{1}{T} \sum_{t=1}^{T} \mathbb{E} Q_t.$$

Definition 3. The task offloading queue evolving according to (3) is *strongly stable* if $\lim_{T \to \infty} \mathbb{E} \bar{Q}_T < \infty$.

We assume a first-come, first-served queue, and denote the set of tasks that have been served by the remote servers in time slot $t \in \mathbb{N}$ as $\mathcal{D}_t$. It follows that $Q_t = \sum_{s=1}^{t} (O_s - |\mathcal{D}_s|)$.

D. Class of Offload Policies

Offload decision for a task $n \in \mathcal{A}_t$ is based on the confidence value Z_n of the current task. A deterministic offload decision is a map from the collection $\{0,1\}^{\mathcal{Z}}$, and thus offload decision is an indicator that the confidence value belongs to some subset of the finite confidence value set $\mathcal{Z}$.

Definition 4 (Stochastic threshold policy)**.** We consider offload policies that are indicators to the subsets of type $[0,\theta] \cap \mathcal{Z}$ parametrized by threshold $\theta \in \mathcal{Z} \subseteq [0,1]$, such that offload indicator $I_n^\theta \triangleq \mathbb{1}_{\{Z_n \leqslant \theta\}}$. In particular, we consider *stochastic threshold policies* π such that for each task $n \in \mathcal{A}_t$, the offload indicator is $I_n \triangleq \mathbb{1}_{\{Z_n \leqslant \Theta_n\}}$ where Θ_n is a random

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.

variable with probability mass function (PMF) π_t defined for each threshold $\theta \in \mathcal{Z}$, as $\pi_t(\theta) \triangleq P\{\Theta_n = \theta\}$.

We focus on the following stochastic threshold policies.

Definition 5 (Stationary deterministic threshold policy)**.** A *stationary deterministic threshold policy* is a stochastic threshold policy π where PMF π_t has a unit mass on some $\theta \in \mathcal{Z}$ for all times $t \in \mathbb{N}$, and is denoted by a single threshold $\theta \in \mathcal{Z}$.

Definition 6 (Piecewise random threshold policy)**.** A *piecewise random threshold policy* is a stochastic threshold policy π where the threshold Θ_n is independent for each task n and identically distributed with PMF π_t for all tasks $n \in \mathcal{A}_t$.

Remark 2. We make two observations for this system. First, we observe that the mismatch error Φ_n cannot be observed when we accept the local inference $h_\ell(x_n)$. Second, we see that when a sample x_n is queued, mismatch error Φ_n can only be observed after a feedback delay which includes the queueing delay and the transmission delay.

Accordingly, we can define the available history under policy π at the local device at the end of time slot $t \in \mathbb{Z}_+$.

Definition 7 (History)**.** Let π be a policy that results in a sequence of task offloading decisions $(I_n : n \in \mathcal{A}_t)$ at each time $t \in \mathbb{N}$. We define the history $\mathcal{F}_t^\pi \triangleq \sigma(X_s^\pi, s \leqslant t)$ of the system under policy π until the end of time slot $t \in \mathbb{Z}_+$, where the state X_s^π at time s is defined as

$$(C_s, (I_n; n \in \mathcal{A}_s), (\Phi_n; n \in \mathcal{D}_s), \mathcal{A}_{s+1}, (Z_n; n \in \mathcal{A}_{s+1})).$$

A policy π is *causal* if the nth task offloading indicator I_n is $\mathcal{F}_{t-1}^\pi$ measurable for any $n \in \mathcal{A}_t$.

Remark 3. Note that $C_t, \mathcal{A}_t$, and $(I_n; n \in \mathcal{A}_t)$ are all $\mathcal{F}_t$ measurable, and therefore $O_t = \sum_{n \in \mathcal{A}_t} I_n$ is $\mathcal{F}_t$ measurable. Since $Q_t = \max\{Q_{t-1} - C_t + O_t, 0\}$, we can recursively show that Q_t is $\mathcal{F}_t$ measurable.

E. Problem Formulation

We are interested in finding an optimal stationary deterministic threshold offloading policy such that the average mismatch error of classification tasks is minimized while maintaining the stability of the task offloading queue.

Definition 8. We denote the set of stationary deterministic threshold policies that make the task offload queue strongly stable, as

$$\Upsilon \triangleq \left\{\theta \in \mathcal{Z} : \lim_{T\to\infty} \bar{Q}_T < \infty\right\}. \tag{4}$$

Remark 4. We note that the set of feasible thresholds Υ is non empty. We observe that when $\theta = 0$, no task is admitted to the queue in any slot, and hence $Q_t = 0$ for all time $t \in \mathbb{Z}_+$. It follows that $0 \in \Upsilon$.

Problem 1. *Find the optimal threshold for a stationary deterministic task offloading policy that minimizes the long-term average mismatch error of classification tasks while maintaining a strongly stable offloading task queue, i.e. find the optimal offloading policy that solves*

$$\theta_0 \triangleq \arg\min_{\theta\in\Upsilon} \lim_{T\to\infty} \bar{E}_T. \tag{5}$$

Definition 9. Consider the stationary deterministic threshold θ_0 that solves Problem 1. Under this policy, the offloading indicator for any task n with confidence Z_n, is $I_n^{\theta_0} \triangleq \mathbb{1}_{\{Z_n \leqslant \theta_0\}}$. The minimum mismatch error and average offload rate under this optimal stationary deterministic threshold θ_0 are

$$\bar{E}_0 \triangleq \lim_{T\to\infty} \bar{E}_T^{\theta_0}, \qquad \lambda_0 \triangleq \lim_{T\to\infty} \bar{O}_T^{\theta_0}.$$

III. Lyapunov analysis and problem reformulation

We observe that Problem 1 can be formulated as a Markov Decision Process (MDP), where the state is queue length and confidence value for each task arrival, and task offload is the state dependent decision. However, the reward of task offloading is unknown, and so is the reward distribution which is the complementary of mismatch error probability P_Φ. Consequently, we avoid MDP formulation due to (a) the unknown reward distribution, (b) computing the optimal MDP policy is computationally expensive, and (c) the optimal offloading decision obtained by MDP offers no insight in general. Instead, we adopt the Lyapunov optimization framework [26], which enables efficient policy design via drift-plus-penalty analysis.

A. Drift-plus-penalty Analysis

An approach to solve Problem 1 approximately is via the drift-plus-penalty method [26]. To this end, we first define a Lyapunov function and its drift.

Definition 10 (Drift plus penalty [26])**.** We define a Lyapunov function $\mathcal{L} : \mathbb{R} \to \mathbb{R}_+$ for each $x \in \mathbb{R}$, as $\mathcal{L}(x) \triangleq \frac{1}{2}x^2$. For this Lyapunov function, the conditional Lyapunov drift for the offloading queue at each time $t \in \mathbb{N}$ is defined as

$$\Delta_t \triangleq \mathbb{E}[\mathcal{L}(Q_t) - \mathcal{L}(Q_{t-1}) \mid \mathcal{F}_{t-1}]. \tag{6}$$

For the offload queue in (3) and some constant $V \geq 1$, we define the drift plus penalty at each time $t \in \mathbb{N}$ as

$$\Delta_t + V\mathbb{E}[E_t \mid \mathcal{F}_{t-1}]. \tag{7}$$

Lemma 1. *Let $B \triangleq \frac{1}{2}(\lambda_m^2 + \mu_m^2)$, then the drift plus penalty defined in* (7) *is bounded above for each time $t \in \mathbb{N}$, by*

$$B - Q_{t-1}\mathbb{E}[C_t \mid \mathcal{F}_{t-1}] + V\mathbb{E}\Big[\frac{Q_{t-1}}{V}O_t + E_t \mid \mathcal{F}_{t-1}\Big]. \tag{8}$$

Proof: We fix time $t \in \mathbb{N}$. From the definition of Lyapunov function in Definition 10, queue evolution in (3), and the fact that $(\max\{a, 0\})^2 \leqslant a^2$, we observe that

$$\mathcal{L}(Q_t) - \mathcal{L}(Q_{t-1}) \leqslant \frac{1}{2}(Q_{t-1} + O_t - C_t)^2 - \frac{1}{2}Q_{t-1}^2.$$

From the non-negativity of number of offloaded tasks O_t and number of tasks C_t that can be serviced, we observe that $(O_t - C_t)^2 \leqslant O_t^2 + C_t^2$. Since $O_t \leqslant \lambda_m$ and $C_t \leqslant \mu_m$, it follows that $\frac{1}{2}(O_t - C_t)^2 \leqslant B$. Substituting this result in the definition of drift plus penalty (7), we obtain the result. ∎

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.

Definition 11. We define a loss function for task $n \in \mathcal{A}_t$ as

$$\ell_n(I_n) \triangleq I_n \frac{Q_{t-1}}{V} + \bar{I}_n \Phi_n, \tag{9}$$

at each time $t \in \mathbb{N}$. We define the loss aggregated over all arriving classification tasks in a time slot $t \in \mathbb{N}$ as $\ell_t \triangleq \sum_{n \in \mathcal{A}_t} \ell_n(I_n)$ and aggregate loss over a time horizon $[T]$ as $L_T \triangleq \sum_{t=1}^{T} \mathbb{E}\ell_t$. For a stochastic threshold offload policy π, we denote the aggregate loss in a time slot t as ℓ_t^π and the aggregate loss over a horizon $[T]$ as L_T^π. For stationary deterministic threshold policy with unit mass at θ, the aggregate loss in a time slot t is denoted as ℓ_t^θ and the aggregate loss over a horizon $[T]$ as L_T^θ.

Remark 5. For each time slot t, we can write the aggregate loss defined in (9) in terms of the number of offloaded tasks defined in (1) and the aggregate mismatch error defined in (2), as $\ell_t = \frac{Q_{t-1}}{V} O_t + E_t$. From the positivity of service in any time slot t, and the definition of aggregate loss ℓ_t,

$$\Delta_t + V\mathbb{E}[\ell_t \mid \mathcal{F}_{t-1}] \leqslant B + V\mathbb{E}[\ell_t \mid \mathcal{F}_{t-1}].$$

In the above equation, B is a universal upper bound on the arrival and service process. It follows that minimizing an upper bound on drift plus penalty is equivalent to minimizing the conditional mean of aggregate loss in each time slot t.

B. Problem Reformulation

Accordingly, one can reformulate the per-slot optimization problem as an online learning problem for regret minimization. Consider a policy π and the history $\mathcal{F}_{t-1}^\pi$ generated under this policy. For this history, we denote the loss incurred under a stationary deterministic threshold policy θ for each sample $n \in \mathcal{A}_t$ as

$$\ell_n^\pi(I_n^\theta) \triangleq I_n^\theta \frac{Q_{t-1}^\pi}{V} + \bar{I}_n^\theta \Phi_n, \tag{10}$$

and the cumulative losses as

$$\ell_t^\theta(\mathcal{F}_{t-1}^\pi) \triangleq \sum_{n \in \mathcal{A}_t} \ell_n^\pi(I_n^\theta), \quad L_T^\theta(\mathcal{F}_{t-1}^\pi) \triangleq \sum_{t=1}^{T} \ell_t^\theta(\mathcal{F}_{t-1}^\pi).$$

Definition 12. Let θ^* be the threshold that minimizes the loss under a stationary deterministic threshold policy aggregated over a time horizon $[T]$ for a history generated by policy π,

$$\theta^* \triangleq \arg\min_{\theta \in \mathcal{Z}} L_T^\theta(\mathcal{F}_{t-1}^\pi) = \arg\min_{\theta \in \mathcal{Z}} \sum_{t=1}^{T} \ell_t^\theta(\mathcal{F}_{t-1}^\pi). \tag{11}$$

We define the accumulated regret over the time horizon $[T]$ of a policy π that does not choose this optimal stationary deterministic threshold policy θ^*, as

$$R_T^\pi \triangleq \mathbb{E}[L_T^\pi - L_T^{\theta^*}(\mathcal{F}_{T-1}^\pi)]. \tag{12}$$

Remark 6. We note that the regret for any policy is typically defined with respect to a reference stationary deterministic threshold policy. However, in our case, we are defining the regret of any policy with respect to stationary deterministic threshold policy, where the threshold policy is applied in each time slot to a process generated by the original policy under consideration.

Problem 2 (Regret minimization)**.** *Find a causal policy π with sub-linear regret R_T^π defined in* (12)*, i.e.* $\lim_{T\to\infty} \frac{1}{T} R_T^\pi = 0$.

Remark 7. We observe that Problem 2 is a prediction problem with experts' advice, where the decision for each task $n \in \mathcal{A}_t$ is to predict offload given the set of experts being the set of confidence values $\mathcal{Z}$. If the decision is to not offload, then the loss is $E_t^\pi = \sum_{n \in \mathcal{A}_t} \bar{I}_n \Phi_n$. If the decision is to offload, then the loss is $\frac{Q_{t-1}^\pi}{V} \sum_{n \in \mathcal{A}_t} I_n$.

Remark 8. If mismatch error sequence $(\Phi_n, n \in \mathcal{A}_t)$ was known after every decision epoch t, then we would know the offload cost $\ell_t^\theta(\mathcal{F}_{t-1}^\pi)$ for each expert $\theta \in \mathcal{Z}$ after every decision epoch t, i.e. we would have had $(\Phi_n, n \in \mathcal{A}_t) \in \mathcal{F}_t$. In this case, one can find the loss for all experts after the decision has been taken, and one can solve Problem 2 as the prediction with expert's advice. However, this is not the case in our setting, since the mismatched error is unknown for tasks with locally accepted decisions, and the mismatch error is delayed even for the offloaded tasks due to limited service to offload queue.

IV. BOLD HEDGE-Q

Consider the hypothetical *no offload queue* setting corresponding to the mismatch error Φ_n being revealed immediately after offloading any task $n \in \mathcal{A}_t$. Note that, even in this simple no offload queue scenario, mismatch error Φ_n cannot be observed when local inference is accepted (since we don't have remote inference, we can't compute the mismatch error for such tasks). We note that the no offload queue setting is the setting of prediction with experts' advice, where the expert set is $\mathcal{Z}$, but with the additional challenge of estimating the conditional mean of mismatch error $\mathbb{E}[\Phi_n \mid \mathcal{F}_{t-1}]$ for locally accepted tasks $n \in \mathcal{A}_t$. To address this, we offload tasks with high confidence with probability ϵ in an i.i.d fashion, i.e., ϵ fraction of all such tasks is offloaded. This is an importance sampling technique for the exploration of all experts.

A. Hedge-Q Algorithm for No Offload Queue

We first propose an algorithm called *Hedge-Q* in Definition 13 for the hypothetical no offload queue setting.

Definition 13 (Hedge-Q decision)**.** Fix $\epsilon > 0$ and consider an *i.i.d.* random sequence $\xi : \Omega \to \{0,1\}^{\mathbb{N}}$ with common mean $\mathbb{E}\xi_n = \epsilon$ for all tasks $n \in \mathcal{A}_t$ and times $t \in \mathbb{N}$. Consider a piecewise random threshold policy π^H with random independent threshold sequence $\Theta : \Omega \to \mathcal{Z}^{\mathbb{N}}$ with common threshold PMF $\pi_t^H \in \mathcal{M}(\mathcal{Z})$ for all tasks $n \in \mathcal{A}_t$ at each time $t \in \mathbb{N}$. Under Hedge-Q algorithm, the offload indicator for each task $n \in \mathcal{A}_t$, is defined as

$$I_n^H \triangleq \mathbb{1}_{\{Q_{t-1} \leqslant V\}} (I_n^{\Theta_n} + \bar{I}_n^{\Theta_n} \xi_n). \tag{13}$$

In a time slot t under Hedge-Q, the number of offloaded tasks, the aggregate mismatch error, and loss are respectively denoted

$$O_t^H \triangleq \sum_{n \in \mathcal{A}_t} I_n^H, \quad E_t^H = \sum_{n \in \mathcal{A}_t} \Phi_n \bar{I}_n^H, \quad \ell_t^H \triangleq \sum_{n \in \mathcal{A}_t} \ell_n(I_n^H).$$

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.

We note that Q_t corresponds to the queue length at time t under the Hedge-Q policy. Since it is clear from the context, we do not put superscript H on Q_t.

Remark 9. Hedge-Q decision algorithm defined in (13) is designed to minimize loss while still exploring all possible experts. For each task $n \in \mathcal{A}_t$ and threshold $\theta \in \mathcal{Z}$, the minimum loss for offload indicator I_n^θ is $\ell_n(I_n^\theta) = \frac{Q_{t-1}}{V} \wedge \Phi_n$, where the mismatch error $\Phi_n \leqslant 1$. To minimize loss for each task, we should never offload a task if $Q_{t-1} > V$, since the loss due to accepting a local inference will always be smaller. When $Q_{t-1} \leqslant V$, there are two cases: (a) confidence $Z_n \leqslant \theta$, and we offload the task, and (b) confidence $Z_n > \theta$, then the task is offloaded with a random indicator ξ_n.

Remark 10. We observe that the loss $\ell_n(I_n^H)$ incurred under Hedge-Q offload indicator I_n^H is given by

$$\begin{aligned}\ell_n(I_n^H) =& \frac{Q_{t-1}}{V} \mathbb{1}_{\{Q_{t-1} \leqslant V\}} (I_n^{\Theta_n} + \bar{I}_n^{\Theta_n} \xi_n) \\ &+ \Phi_n (\mathbb{1}_{\{Q_{t-1} > V\}} + \mathbb{1}_{\{Q_{t-1} \leqslant V\}} \bar{I}_n^{\Theta_n} \bar{\xi}_n).\end{aligned} \tag{14}$$

Remark 11. The history of the system under Hedge-Q with no offload queue until the end of time slot $t \in \mathbb{N}$ is denoted by $\mathcal{F}_t^H \triangleq \sigma(X_s^H, s \leqslant t)$, where the state X_s^H at time s is defined

$$(C_s, (\Theta_n, \xi_n; n \in \mathcal{A}_s), (\Phi_n I_n^H; n \in \mathcal{A}_s), \mathcal{A}_{s+1}, (Z_n; n \in \mathcal{A}_{s+1})).$$

It follows that $\Phi_n \bar{I}_n^H$ and hence the loss $\ell_n(I_n^H)$ can't be observed when the local inference is accepted and hence we define a pseudo loss function using importance sampling in Definition 14 to update the weights for experts in Hedge-Q.

Definition 14 (Pseudo loss)**.** At any time $t \in \mathbb{N}$, we define *pseudo loss* for each task $n \in \mathcal{A}_t$ and deterministic offload threshold $\theta \in \mathcal{Z}$, as

$$\hat{\ell}_{n,\theta} \triangleq \mathbb{1}_{\{Q_{t-1} > V\}} + \mathbb{1}_{\{Q_{t-1} \leqslant V\}} \Big(\frac{Q_{t-1}}{V} I_n^\theta + \frac{\Phi_n \xi_n}{\epsilon} \bar{I}_n^\theta \Big). \tag{15}$$

For a fixed threshold $\theta \in \mathcal{Z}$, the aggregate pseudo loss at time t and until time t are respectively defined as $\hat{\ell}_t^\theta \triangleq \sum_{n \in \mathcal{A}_t} \hat{\ell}_{n,\theta}$ and $\hat{L}_t^\theta \triangleq \sum_{s=1}^t \hat{\ell}_s^\theta$.

Remark 12. Since $\Phi_n, \xi_n \in \{0, 1\}$, it follows that $\hat{\ell}_{n,\theta} \leqslant \frac{1}{\epsilon}$.

Lemma 2. *For any time t, task $n \in \mathcal{A}_t$, and threshold $\theta \in \mathcal{Z}$, the expectation of pseudo loss with respect to ξ is*

$$\mathbb{E}_\xi \hat{\ell}_{n,\theta} = \mathbb{1}_{\{Q_{t-1} > V\}} + \mathbb{1}_{\{Q_{t-1} \leqslant V\}} \ell_n^H(I_n^\theta). \tag{16}$$

It follows that $\mathbb{E}_\xi \hat{\ell}_{n,\theta} \leqslant 1$.

Proof: Taking expectation on both sides of (15) with respect to *i.i.d.* Bernoulli sequence ξ and using (10), we get (16). Since $\ell_n^H(I_n^\theta) \leqslant 1$, it follows that $\mathbb{E}_\xi \hat{\ell}_{n,\theta} \leqslant 1$. ∎

Remark 13. For each time t, we can compare true loss $\ell_n(I_n^H)$ and pseudo loss $\hat{\ell}_{n,\Theta_n}$ for randomly sampled threshold Θ_n for each task $n \in \mathcal{A}_t$.

(a) Since Hedge-Q algorithm doesn't offload a task when $Q_{t-1} > V$, it accepts local decision and the loss in this case is the mismatch error $\Phi_n \leqslant 1$. Pseudo loss is an upper bound on true loss in this case.

(b) Consider the case when $Q_{t-1} \leqslant V$. If $I_n^{\Theta_n} = 1$ then the true loss and pseudo loss match. If $I_n^{\Theta_n} = 0$ then the true loss is $\Phi_n \bar{\xi}_n + \frac{Q_{t-1}}{V} \xi_n$ but the pseudo loss is $\frac{1}{\epsilon} \Phi_n \xi_n \leqslant \frac{1}{\epsilon}$.

Definition 15 (Hedge-Q threshold PMF)**.** Fix $\eta > 0$, an expert $\theta \in \mathcal{Z}$. At time $t = 0$, we initialize $w_{0,\theta} = 1$. For any time $t \in \mathbb{N}$, we update the weights as

$$w_{t,\theta} \triangleq w_{t-1,\theta} e^{-\eta \hat{\ell}_t^\theta}, \qquad W_t \triangleq \sum_{\theta \in \mathcal{Z}} w_{t,\theta}. \tag{17}$$

From the definition of weights $w_{t,\theta} = e^{-\eta \hat{L}_t^\theta}$ for each $\theta \in \mathcal{Z}$ and time $t \in \mathbb{Z}_+$. We define Hedge-Q threshold PMF $\pi_t^H \in \mathcal{M}(\mathcal{Z})$ at time $t \in \mathbb{Z}_+$ for each threshold $\theta \in \mathcal{Z}$

$$\pi_t^H(\theta) \triangleq w_{t,\theta} / W_t. \tag{18}$$

In the Hedge-Q algorithm, the random thresholds $(\Theta_n; n \in \mathcal{A}_t)$ at time $t \in \mathbb{N}$ are sampled with PMF π_{t-1}^H.

Remark 14. We note that in the no offload queue case, pseudo loss can be observed immediately after the decision is made. Hence, it follows that π_t^H is $\mathcal{F}_t^H$ measurable, and hence Hedge-Q is a causal policy.

B. BOLD Hedge-Q Algorithm for Offload Queue

Recall that the mismatch error of offloaded tasks is delayed due to presence of an offload queue. To overcome this challenge, we adopt the reduction method Black-box Online Learning under Delayed feedback algorithm (BOLD) proposed in [27]. BOLD reduces online learning with feedback delays to solving multiple instances of the problem with no delays. Specifically for our case, each subproblem instance with no offload queue utilizes a Hedge-Q decision.

The main idea we use is to partition the stream of task arrivals into sub-streams, where each sub-stream is obtained by sampling the arrivals such that each offloaded task in a sub-stream must have received the mismatch error feedback before the arrival of the subsequent task of that sub-stream. Each sub-stream can be considered as a system where the mismatch error is available before the arrival of the next task, and hence, in effect, there is no offload queue. Therefore, for each sub-stream, we can use a separate Hedge-Q instance to make offload decisions.

Each sub-stream or problem instance is denoted by an index $i \in \mathbb{N}$. Let $\zeta_{n,i}$ be the indicator that task n is partitioned to sub-stream i, then $\mathcal{A}_t^i \triangleq \{n \in \mathcal{A}_t : \zeta_{n,i} = 1\}$ is the set of tasks that arrived in time slot $t-1$, and were designated to be in sub-stream i. We observe that $\mathcal{A}_t = \cup_{i \in \mathbb{N}} \mathcal{A}_t^i$. We define the aggregate pseudo loss until time t for sub-stream i as $\hat{L}_t^{i,\theta} \triangleq \sum_{s=1}^t \sum_{n \in \mathcal{A}_s^i} \hat{\ell}_{n,\theta}$. For each threshold $\theta \in \Theta$, we have

$$w_{t,\theta}^i \triangleq e^{-\eta \hat{L}_t^{i,\theta}}, \quad W_t^i \triangleq \sum_{\theta \in \mathcal{Z}} w_{t,\theta}^i, \quad \pi_t^{i,H}(\theta) \triangleq \frac{w_{t,\theta}^i}{W_t^i}.$$

For each sub-stream i, the independent random threshold sequence Θ_n^i has PMF $\pi_t^{i,H} \in \mathcal{M}(\mathcal{Z})$ for all $n \in \mathcal{A}_t^i$, and the offload indicator is

$$I_n^{i,H} \triangleq \mathbb{1}_{\{Q_{t-1} \leqslant V\}} (I_n^{\Theta_n^i} + \bar{I}_n^{\Theta_n^i} \xi_n).$$

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.

The set of offloaded tasks for sub-stream i in time slot t is $\mathcal{O}_t^i \triangleq \left\{n \in \mathcal{A}_t^i : I_n^{i,H} = 1\right\}$. From this discussion, it follows that it suffices to describe the partition $(\mathcal{A}_t^i; i \in \mathbb{N})$ of arrivals $\mathcal{A}_t$ for each time slot $t-1 \in \mathbb{Z}_+$. Equivalently, we need to describe $(\zeta_{n,i}; n, i \in \mathbb{N})$ for each task n and sub-stream i.

The main task of BOLD Hedge-Q, presented in Algorithm 1, is to dynamically create the sub-streams $\mathcal{A}_t^i$ as described above and use separate instances of Hedge-Q on each of those sub-streams. To this end, in any round t, for each task $n \in \mathcal{A}_t$, BOLD Hedge-Q invokes a `free` Hedge-Q instance i (or creates a new instance if there is no `free` instance), to make the offloading decision. This task n is added to the set $\mathcal{A}_t^i$. If the decision is to offload the task, the instance enters `wait` state until it receives the remote DL inference. In a future round, when it receives the remote DL inference, then it updates its weights, and the instance enters `free` state. Note that this Hedge-Q instance i will not be invoked until it enters `free` state. By design, the tasks getting added to the set $\mathcal{A}_t^i$ will have the property of the sub-streams described above, and all the instances are thus operating in no offload queue setting.

Algorithm 1 BOLD Hedge-Q policy

1: Initialize: Create Hedge-Q instance $i = 1$. $\mathcal{B} = \{1\}$.
2: **for** each round $t \in [T]$ **do**
3: Select a `free` instance $i \in \mathcal{B}$; if there is no `free` instance, create a new instance $i = |\mathcal{B}| + 1$ and update $\mathcal{B} = \mathcal{B} \cup \{i\}$.
4: **for** each task $n \in \mathcal{A}_t$ **do**
5: Obtain Z_n and $h_\ell(x_n)$ using the on-device model
6: Invoke the selected Hedge-Q instance i with the data t, n, Q_{t-1}, Z_n and $h_\ell(x_n)$
7: If instance i enters `wait` state, then go to line 3.
8: **end for**
9: **end for**

Algorithm 2 Hedge-Q instance i

1: **Input:** t, n, x_n, Q_{t-1}, Z_n, $h_\ell(x_n)$
2: **if** $\mathcal{A}_t^i = \emptyset$ (instance is just created) **then**
3: **Initialize:** $k = 1$, $\tau = t$, $\forall \theta \in \mathcal{Z}$, $w_{t,\theta}^i = 1$, $W_t^i = |\mathcal{Z}|$, and $\epsilon \in (0, 1)$.
4: **else**
5: Assign: Weight $w_{t,\theta}^i = w_{\tau,\theta}^i \forall \theta \in \mathcal{Z}$, cumulative weight $W_t^i \triangleq \sum_{\theta \in \mathcal{Z}} w_{t,\theta}^i$, and PMF $\pi_t^{i,H}(\theta) \triangleq \frac{w_{t,\theta}^i}{W_t^i} \forall \theta \in \mathcal{Z}$.
6: Generate independent $\Theta_n \in \mathcal{Z}$ with PMF $\pi_{t-1}^{i,H}$.
7: Generate independent $\xi_n \in \{0, 1\}$ with $\mathbb{E}\xi_n = \epsilon$.
8: **end if**
9: $\mathcal{A}_t^i = \mathcal{A}_t^i \cup \{n\}$
10: **if** $Q_{t-1} \leqslant V$ **then**
11: **if** $Z_n \leqslant \Theta_n$ **then**
12: Offload x_n
13: **else**
14: **if** $\xi_n = 1$ **then**
15: Offload x_n
16: Enter `wait` state.
17: **if** remote inference $h_r(x_n)$ is received in round $t + d_t$, for some $d_t \geqslant 0$ **then**
18: Enter `free` state.
19: **end if**
20: **else**
21: Accept the local inference $h_\ell(x_n)$
22: **end if**
23: **end if**
24: **else**
25: Accept the local inference $h_\ell(x_n)$.
26: **end if**
27: For all $\theta \in \mathcal{Z}$, compute $\hat{\ell}_{n,\theta}$ using Z_n in (15).
28: Update the weights $w_{t,\theta}^i \triangleq w_{t,\theta}^i e^{-\eta \hat{\ell}_{n,\theta}}$, for all $\theta \in \mathcal{Z}$.
29: $\tau = t$
30: Enter `free` state.

V. Performance Analysis

In our regret analysis, we first prove the bounds with respect to a modified loss function, defined below. Recall that regret for Hedge-Q policy with respect to stationary deterministic policy θ is computed when the observables are generated under Hedge-Q policy for each task.

Definition 16 (Modified loss function)**.** Consider Hedge-Q algorithm with *i.i.d.* random sequence ξ and independent random threshold sequence Θ. For task $n \in \mathcal{A}_t$ in a round t, we define the *modified loss* as $\check{\ell}_{n,\Theta_n}$ where the *modified loss function* for each threshold $\theta \in \mathcal{Z}$ is defined as

$$\check{\ell}_{n,\theta} \triangleq \mathbb{1}_{\{Q_{t-1} > V\}} \Phi_n + \mathbb{1}_{\{Q_{t-1} \leqslant V\}} \ell_n^H(I_n^\theta). \tag{19}$$

We define aggregate modified loss for threshold θ until time t as $\check{L}_t^\theta \triangleq \sum_{s=1}^t \sum_{n \in \mathcal{A}_s} \check{\ell}_{n,\theta}$. For each sub-stream i and threshold $\theta \in \mathcal{Z}$, we define aggregate losses until time t as

$$\check{L}_t^{i,\theta} = \sum_{s=1}^t \sum_{n \in \mathcal{A}_s^i} \check{\ell}_{n,\theta}, \qquad L_t^{i,H} = \sum_{s=1}^t \sum_{n \in \mathcal{A}_s^i} \ell_n(I_n^{i,H}).$$

Remark 15. The motivation for defining the modified loss function is that, when $\frac{Q_{t-1}}{V} > 1$, it is always beneficial to accept the local inference since mismatch error $\Phi_n \leqslant 1$.

Theorem 1. *For a system under Hedge-Q algorithm where there is no offload queue, we have*

$$\mathbb{E}[L_T^H - L_T^{\theta^*}(\mathcal{F}_{T-1}^H)] \leqslant \left(\epsilon + \frac{\eta \lambda_m}{2\epsilon}\right) \sum_{t=1}^T |\mathcal{A}_t| + \frac{1}{\eta} \log |\mathcal{Z}|.$$

The proof of Theorem 1 is omitted due to space constraints.

A. Regret for BOLD Hedge-Q

Recall that $(\mathcal{A}_t^i : i \in \mathbb{N})$ is a partition of task arrivals $\mathcal{A}_t$ in each time slot t. Since the number of task arrivals in each time slot is bounded by λ_m, we have at most $\lambda_m T$ task arrivals until time T.

Corollary 1. *Let* $L_T^{i,\theta}(\mathcal{F}_{T-1}^H) \triangleq \sum_{t=1}^T \sum_{n \in \mathcal{A}_t^i} \ell_n^H(I_n^\theta)$ *and* $\theta_i \triangleq \arg\min_{\theta \in \mathcal{Z}} L_T^{i,\theta}(\mathcal{F}_{T-1}^H)$ *be the loss minimizing threshold*

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.

for each sub-stream i. Then, for each sub-stream i under BOLD Hedge-Q algorithm, we have

$$\mathbb{E}[L_T^{i,H} - L_T^{i,\theta_i}(\mathcal{F}_{T-1}^H) \mid M_T] \leqslant \Big(\epsilon + \frac{\eta\lambda_m}{2\epsilon}\Big) \sum_{t=1}^{T} \big|\mathcal{A}_t^i\big| + \frac{1}{\eta}\log|\mathcal{Z}|\,.$$

Proof: By construction of BOLD Hedge-Q, each task in each sub-stream i gets immediate loss feedback after the decision, and hence each sub-stream is an instance of Hedge-Q algorithm with no offload queue, and we can apply Theorem 1 with sequence of arrivals $(\mathcal{A}_t^i; t \in [T])$. ∎

Definition 17. Let M_t denote the random number of Hedge-Q instances created in t rounds. The cumulative loss of BOLD Hedge-Q in T rounds is given by

$$L_T^H \triangleq \sum_{t=1}^{T} \sum_{n \in \mathcal{A}_t} \sum_{i=1}^{M_T} \ell_n(I_n^{i,H}) = \sum_{i=1}^{M_T} L_T^{i,H}. \tag{20}$$

The regret of BOLD Hedge-Q is defined as $R_T^H \triangleq \mathbb{E}[L_T^H] - L_T^{\theta^*}$, where θ^* is the loss minimizing threshold defined in (11).

Lemma 3. *Under BOLD Hedge-Q, in any round t, (i) $Q_t \leqslant V + \lambda_m$, and (ii) $M_t \leqslant Q_t$.*

Proof: We define the first hitting time for the offload queue to exceed the threshold V as $\tau_0 \triangleq \inf\{t \in \mathbb{N} : Q_t > V\}$. It follows that $Q_t \leqslant V$ for all times $t < \tau_0$. In particular, we have $Q_{\tau_0 - 1} \leqslant V$. Since the maximum number of arrivals is λ_m, we have $Q_{\tau_0} \leqslant V + \lambda_m$. We define $\tau_1 \triangleq \inf\{t > \tau_0 : Q_t \leqslant V\}$. Since Hedge-Q instance never offloads a task $n \in \mathcal{A}_t$ if $Q_{t-1} > V$, it follows that for all times $t \in \{\tau_0 + 1, \ldots, \tau_1\}$, no new task is offloaded to the queue, and hence the offload queue can not increase n this time interval, i.e. $Q_t \leqslant Q_{\tau_0} \leqslant V + \lambda_m$ for all $t \in \{\tau_0 + 1, \ldots, \tau_1\}$. We have now shown that $Q_t \leqslant V + \lambda_m$ for all times $t \leqslant \tau_1$. We can now inductively repeat this argument to conclude that $Q_t \leqslant V + \lambda_m$ for all times $t \in \mathbb{N}$.

Recall that a Hedge-Q instance never offloads a task $n \in \mathcal{A}_t$ to the remote server if $Q_{t-1} > V$. In this case, the instance accepts the local inference, receives a pseudo loss of unity, and enters the `free` state. Also, when a task in the offload queue is served, an instance waiting on that task enters `free` state. Further, when $Q_{t-1} \leqslant V$, task $n \in \mathcal{A}_t$ is offloaded if $Z_n \leqslant \Theta_n$. In this case, the pseudo loss is $\frac{Q_{t-1}}{V}$ which is immediately known to the corresponding instance, and the instance enters a `free` state. That is, some of the tasks in the offload queue may belong to the same Hedge-Q instance. From the above arguments, we conclude that, in round t, the number of instances is upper bounded by the number of tasks in the offload queue, i.e., $M_t \leqslant Q_t$. ∎

Theorem 2. *For a fixed horizon T, the regret of BOLD Hedge-Q is upper bounded as*

$$R_T^H \leqslant \Big(\epsilon + \frac{\eta}{2\epsilon}\Big)\lambda_m T + \frac{1}{\eta}(V + \lambda_m)\log|\mathcal{Z}|\,.$$

Proof: Recall that M_t is the number of Hedge-Q instances created under BOLD Hedge-Q algorithm in t rounds. We note that each task $n \in \cup_{t=1}^{T} \mathcal{A}_t$ is processed by exactly one of the instances $i \in [M_T]$. Note that the set of task arrivals in round t is $\mathcal{A}_t = \cup_{i=1}^{M_T} \mathcal{A}_t^i$, where $\mathcal{A}_t^i = \emptyset$ if instance i is not invoked in round t. By the definition of θ_i, we have

$$L_T^{\theta^*}(\mathcal{F}_{T-1}^H) \geqslant \sum_{i=1}^{M_T} L_T^{i,\theta_i}(\mathcal{F}_{T-1}^H). \tag{21}$$

Summing the regret over time horizon $[T]$ for all M_T instances, applying Corollary 1, exchanging summations, and substituting (21), we obtain

$$\sum_{i=1}^{M_T} \mathbb{E}[L_T^{i,H} - L_T^{\theta^*}(\mathcal{F}_{T-1}^H) \,|\, M_T] \leqslant \Big(\epsilon + \frac{\eta\lambda_m}{2\epsilon}\Big) \sum_{t=1}^{T} |\mathcal{A}_t| + \frac{M_T}{\eta}\ln|\mathcal{Z}|\,.$$

Upper bounding M_T from Lemma 3, taking expectation on both sides of the above equation, using the definition of aggregate loss in (20), and $\sum_{t=1}^{T} |\mathcal{A}_t| \leqslant \lambda_m T$, we obtain the result. ∎

Corollary 2. *Choosing $\eta = (\sqrt{2}(V/\lambda_m + 1)\log|\mathcal{Z}|/T)^{\frac{2}{3}}$ and $\epsilon = \sqrt{\eta/2}$, the regret of Hedge-Q is*

$$R_T^H = O(\lambda_m((V/\lambda_m + 1)\log|\mathcal{Z}|)^{\frac{1}{3}} T^{\frac{2}{3}}).$$

B. Performance Bounds of BOLD Hedge-Q

Corollary 3. *Under any causal policy, we have*

$$\frac{1}{VT}\mathbb{E}Q_T^2 + \mathbb{E}\bar{E}_T \leqslant \frac{B}{V} + \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\ell_t - \frac{Q_{t-1}}{V} C_t].$$

Proof: Taking the expectation on both sides in Lemma 1, summing over all $t \in [T]$, applying tower property of conditional expectation, using the definition of drift Δ_t in (6), the fact that $Q_0 = 0$, definition of loss ℓ_t, and dividing both sides by VT, we obtain the result. ∎

Let $\bar{Q}_T^{\theta_0}$ and $\bar{Q}_T^H$ denote the time average expected queue length until round T under the optimal stationary deterministic threshold policy θ_0 and BOLD Hedge-Q, respectively.

Theorem 3. *Under BOLD Hedge-Q policy, the time-average expected mismatch error is upper bounded as*

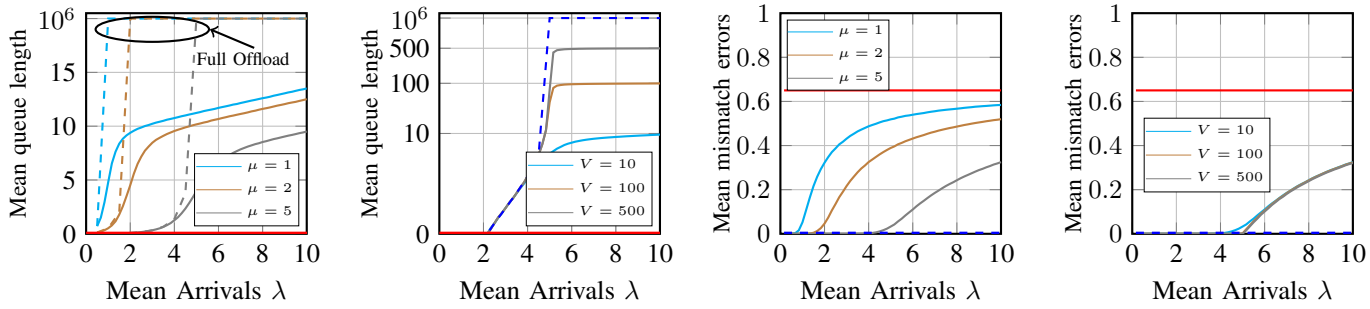
$$\lim_{T\to\infty} \mathbb{E}\bar{E}_T^H \leqslant \Big(\bar{E}_0 + \frac{B}{V}\Big) \wedge 1.$$

Proof: Since BOLD Hedge-Q is a causal policy, Corollary 3 applies for this policy. Recall that regret $R_T^H = \mathbb{E}[L_T^H - L_T^{\theta^*}(\mathcal{F}_{T-1}^H)]$. By definition, we have $L_T^{\theta^*}(\mathcal{F}_{T-1}^H) \leqslant L_T^{\theta_0}(\mathcal{F}_{T-1}^H)$. Since $\mathbb{E}Q_T^2 \geqslant 0$, loss $L_T^{\theta}(\mathcal{F}_{T-1}^H) = \sum_{t=1}^{T} \frac{Q_{t-1}^H}{V} O_t^{\theta} + E_t^{\theta}$, and random sequence C is *i.i.d.* with mean μ we conclude

$$\mathbb{E}\bar{E}_T^H \leqslant \frac{B}{V} + \frac{1}{T}\Big(R_T^H + \mathbb{E}E_T^{\theta_0} + \sum_{t=1}^{T} \mathbb{E}\Big[\frac{Q_{t-1}^H}{V}(O_t^{\theta_0} - C_t)\Big]\Big).$$

Since deterministic threshold $\theta^0 \in \Upsilon$ makes the system strongly stable, $\mathbb{E}[O_t^{\theta_0} - C_t] \leqslant 0$. Further, $O_t^{\theta_0}$ and C_t are independent of Q_{t-1}, and hence $\mathbb{E}[\frac{Q_{t-1}}{V}(O_t^{\theta_0} - C_t)] \leqslant 0$. The result follows by taking limit $T \to \infty$ on both sides, and from the sublinearity of the regret R_T^H shown in Corollary 2. ∎

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.



(a) Queue length vs λ; $V = 10$ (b) Queue length vs λ; $\mu = 5$ (c) Mismatch err. vs λ; $V = 10$ (d) Mismatch err. vs λ; $\mu = 5$

Fig. 2: ImageNet: Average queue length and mismatch errors of BOLD Hedge-Q for varying arrival rate λ with $\epsilon = 0.4$ and $\eta = 0.1$. Dashed lines represent Full-offload; solid red lines represent On-Device.

VI. Simulation Results

In this section, we evaluate the performance of BOLD Hedge-Q under different parameter settings. Existing HI algorithms HIL [5] and Hedge-HI [7] do not consider offload queue and assume remote DL inference is observed immediately after offloading, making them inapplicable for our problem and for comparison. Hence, we consider BOLD Hedge-Q with the following two baselines: (a) *On-Device*: which accepts all local inferences; and (b) *Full Offload*: which offloads all tasks for remote inference. We conduct simulations over $T = 10^6$ rounds, with default parameters $\epsilon = 0.4$, $V = 10$, $\mu = 5$, and learning rate $\eta = 0.1$. Each simulation result is averaged over 100 independent runs.

Dataset, local DL, arrival, and service: We studied two widely used image classification datasets ImageNet [28] and CIFAR-100 [29]. However, we omit the results for CIFAR-100 due to space constraints. In ImageNet, there are 1000 classes, and we use 50000 images in its validation set. For the local DL, we employed an 8-bit quantized *MobileNet* TensorFlow Lite (TFLite) model with size $\approx$ 500 KB and 35% accuracy [30]. For the end device, we used Raspberry Pi equipped with 4 CPU cores, a 1.5 GHz processor, and 8 GB RAM. In our simulation, we assume that the remote DL has 100% accuracy.

We studied arrivals and service generated from uniform and Binomial distributions, and found the results to be of similar nature. Hence, we present results only for the Binomial case for brevity. As such, the numbers of task arrival and tasks serviced in each time slot were assumed to be Binomially distributed with parameters (k, p_a) and (k, p_s) where $k = 10$ for both arrivals and service processes, and $p_a \in \{0, .01, \ldots, 1\}$ and $p_s \in \{0.1, 0.2, 0.5\}$. For this choice of distribution and parameters, we have $\lambda_m = k = 10$.

Performance evaluation: We evaluate performance using average queue length over the simulation horizon and average mismatch error over all tasks. Figures 2a–2b plot average queue length versus mean arrival rate λ, while Figures 2c–2d plot average mismatch error versus λ. In Figures 2a and 2c, V is fixed and the mean service rate is varied; in Figures 2b and 2d, the mean service rate is fixed at $\mu = 5$ and $V \in \{10, 100, 500\}$. As expected, average queue length increases with λ, since the algorithm offloads more tasks to reduce mismatch error, and is higher under lower service rates. In all cases, the maximum queue length remains below the analytical upper bound $V + \lambda_m$.

We observe that the mismatch error of BOLD Hedge-Q is zero when the mean number of task arrivals is smaller than the mean service per time slot. This is due to the fact that all tasks can be offloaded for low mean arrivals, and the number of offloaded tasks decreases as the mean arrival increases. More interestingly, we observe in Fig. 2d, that the mismatch error reduction is almost negligible for larger V values. We can pick an appropriate V depending on the mean number of arrivals and service per time slot to strike a good balance between the average queue length and mismatch error, e.g. $V = 10$ appears to be a reasonable choice under the settings we studied. A smaller V leads to smaller number of Hedge-Q instances. We observed at most 8 Hedge-Q instances are created for $V = 10, \epsilon = 0.9, \lambda = 10, \mu = 5$.

We observe that On-Device has the largest mismatch error equal to 0.65, which is the average error of the local inference for zero offload queue length. Full Offload has zero mismatch error but its queue length is unbounded when $\lambda \geqslant \mu$, as shown in Fig. 2a and Fig. 2b.

VII. Conclusion

In this paper, we study the HI system with an offload queue and formulate the problem of finding an optimal fixed-threshold policy θ_0 that minimizes mismatch error subject to queue stability. We use the drift-plus-penalty method and reformulate the problem into a per-slot optimization problem. Two key challenges to solving the per-slot optimization problem are (i) the mismatch error is not known when the local DL inference is accepted, and (ii) the remote DL inference is not obtained until the task is served from the offload queue. We address these challenges by solving a delayed online learning subproblem. The proposed BOLD Hedge-Q algorithm uses importance sampling and a black-box reduction technique and achieves sub-linear regret for the delayed online learning problem. Our analysis uniquely combines regret analysis with the drift-plus-penalty method to show that, under BOLD Hedge-Q, the mismatch error is within an additive factor of $O(1/V)$ from that of θ_0, and the queue is strictly bounded.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] A. Elhanashi, P. Dini, S. Saponara, and Q. Zheng, "Advancements in TinyML: Applications, limitations, and impact on IoT devices," *Electronics*, vol. 13, no. 17, p. 3562, Sep. 2024.

[2] I. Nikoloska and N. Zlatanov, "Data selection scheme for energy efficient supervised learning at IoT nodes," *IEEE Commun. Lett.*, vol. 25, no. 3, pp. 859–863, Mar. 2021.

[3] G. Al-Atat, A. Fresa, A. P. Behera, V. N. Moothedath, J. Gross, and J. P. Champati, "The case for hierarchical deep learning inference at the network edge," in *Inter. Workshop on Netw. AI Sys. (NetAISys)*, Jun. 2023, pp. 1–6.

[4] A. P. Behera, R. Morabito, J. Widmer, and J. P. Champati, "Improved decision module selection for hierarchical inference in resource-constrained edge devices," in *Ann. Inter. Conf. Mobile Comput. Netw. (ACM MobiCom)*, Oct. 2023, pp. 1–3.

[5] V. N. Moothedath, J. P. Champati, and J. Gross, "Getting the best out of both worlds: Algorithms for hierarchical inference at the edge," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 280–297, Feb. 2024.

[6] H. B. Beytur, A. G. Aydin, G. de Veciana, and H. Vikalo, "Optimization of offloading policies for accuracy-delay tradeoffs in hierarchical inference," in *IEEE Conf. Comp. Commun. (INFOCOM)*, May 2024, pp. 1989–1998.

[7] G. Al-Atat, P. Datta, S. Moharir, and J. P. Champati, "Regret bounds for online learning for hierarchical inference," in *Inter. Symp. Theory, Algo. Found., Proto. Des. Mob. Netw. Mob. Comput. (ACM MobiHoc)*, Oct. 2024, pp. 281–290.

[8] P. Datta, S. Moharir, and J. P. Champati, "Online learning with stochastically partitioning experts," in *Uncert. Artif. Intell. (UAI)*, vol. 286, 2025, pp. 882–896.

[9] A. P. Behera, P. Daubaris, I. naki Bravo, J. Gallego, R. Morabito, J. Widmer, and J. P. Champati, "Exploring the boundaries of on-device inference: When tiny falls short, go hierarchical," *IEEE Internet Things J.*, vol. 12, no. 18, pp. 37 456–37 470, Jul. 2025.

[10] R. Sanchez-Iborra and A. F. Skarmeta, "TinyML-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits Syst. Mag.*, vol. 20, no. 3, pp. 4–18, Aug. 2020.

[11] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Arch. News*, vol. 45, no. 1, pp. 615–629, Apr. 2017.

[12] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.

[13] C. Hu and B. Li, "Distributed inference with deep learning models across heterogeneous edge devices," in *IEEE Conf. Comp. Commun. (INFOCOM)*, May 2022, pp. 330–339.

[14] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan, "Bandwidth-efficient live video analytics for drones via edge computing," in *IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 159–173.

[15] S. S. Ogden and T. Guo, "MDInference: Balancing inference accuracy and latency for mobile applications," in *IEEE Inter. Conf. Cloud Engg. (IC2E)*, Apr. 2020, pp. 28–39.

[16] A. Fresa and J. P. Champati, "Offloading algorithms for maximizing inference accuracy on edge device in an edge intelligence system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 7, pp. 2025–2039, Jul. 2023.

[17] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[18] P. Naor, "The regulation of queue size by levying tolls," *Econometrica: journal of the Econometric Society*, pp. 15–24, 1969.

[19] S. G. Johansen and S. Stidham Jr, "Control of arrivals to a stochastic input–output system," *Advances in Applied Probability*, vol. 12, no. 4, pp. 972–999, 1980.

[20] A. Hordijk and F. Spieksma, "Constrained admission control to a queueing system," *Advances in Applied Probability*, vol. 21, no. 2, pp. 409–431, 1989.

[21] Y. Chen and J. J. Hasenbein, "Knowledge, congestion, and economics: Parameter uncertainty in Naor's model," *Queue. Sys. Theory Appl.*, vol. 96, no. 1–2, pp. 83–99, Oct. 2020.

[22] A. Cohen, V. Subramanian, and Y. Zhang, "Learning-based optimal admission control in a single-server queuing system," *Stochastic systems*, vol. 14, no. 1, pp. 69–107, Jan. 2024.

[23] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Inter. Conf. Learn. Rep. (ICLR)*, Feb. 2017.

[24] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res. (JMLR)*, vol. 18, no. 1, pp. 6869–6898, Jan. 2018.

[25] C. Banbury, V. J. Reddi, P. Torelli, J. Holleman, N. Jeffries, C. Kiraly, P. Montino, D. Kanter, S. Ahmed, D. Pau, U. Thakker, A. Torrini, P. Warden, J. Cordaro, G. D. Guglielmo, J. Duarte, S. Gibellini, V. Parekh, H. Tran, N. Tran, N. Wenxu, and X. Xuesong, "MLPerf Tiny benchmark," *Conf. Neural Info. Process. Sys. (NeurIPS)*, 2021.

[26] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Springer Nature, 2022.

[27] P. Joulani, A. György, and C. Szepesvári, "Online learning under delayed feedback," in *Inter. Conf. Mach. Learn. (ICML)*, vol. 28, no. 3, 2013, pp. 1453–1461.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conf. Comput. Vision Pattern Recog. (CVPR)*, Jun. 2009, pp. 248–255.

[29] A. Krizhevsky and G. Hinton. (2009) Learning multiple layers of features from tiny images. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[30] TensorFlow, "Tflite models," https://www.tensorflow.org/lite/models, 2024.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on August 08,2026 at 08:18:19 UTC from IEEE Xplore. Restrictions apply.