Hierarchical Inference at the Edge: A Batch Processing Approach

Afroditi Letsiou
Department of Intelligent Systems
KTH Royal Institute of Technology
Stockholm, Sweden
aletsiou@kth.se

Vishnu Narayanan Moothedath
Department of Intelligent Systems
KTH Royal Institute of Technology
Stockholm, Sweden
vnmo@kth.se

Adarsh Prasad Behera
Department of Intelligent Systems
KTH Royal Institute of Technology
Stockholm, Sweden
adarshbehera.iiita@gmail.com

Jaya Prakash Champati
Computer Science
University of Victoria
British Columbia, Canada
jpchampati@uvic.ca

James Gross
Department of Intelligent Systems
KTH Royal Institute of Technology
Stockholm, Sweden
jamesgr@kth.se

***Abstract*—Deep learning (DL) applications have rapidly evolved to address increasingly complex tasks by leveraging large-scale, resource-intensive models. However, deploying such models on low-power devices is not practical or economically scalable. While cloud-centric solutions satisfy these computational demands, they present challenges in terms of communication costs and latencies for real-time applications when every computation task is offloaded. To mitigate these concerns, hierarchical inference (HI) frameworks have been proposed, enabling edge devices equipped with small ML models to collaborate with edge servers by selectively offloading complex tasks.**

Existing HI approaches depend on immediate offloading of data upon selection, which can lead to inefficiencies due to frequent communication, especially in time-varying wireless environments. In this work, we introduce *Batch* HI, an approach that offloads samples in batches, thereby reducing communication overhead and improving system efficiency while achieving similar performance as existing HI methods. Additionally, we find the optimal batch size that attains a crucial balance between responsiveness and system time, tailored to specific user requirements. Numerical results confirm the effectiveness of our approach, highlighting the scenarios where batching is particularly beneficial.

***Index Terms*—Hierarchical inference, offloading decisions, tiny ML, batching, edge computing, regret bound, responsiveness.**

I. Introduction

In the last decade, deep learning (DL) applications have rapidly evolved to address increasingly complex tasks by leveraging large scale resource-intensive models. This evolution poses challenges for deploying DL solutions on devices with constrained power and processing capabilities, such as micro controller units, IoT sensors, and smartphones. Therefore, most DL applications are cloud centric which involves running the entire model on a remote server with ample computational resources, while the edge devices (EDs) handle only minimal processing tasks. However, this raises concerns about increased communication costs, latency, energy consumption, and potential privacy issues. Therefore, processing and analyzing data locally at the edge device has become essential. To make this feasible, considerable research has been focused into developing compressed DL models using different techniques like pruning, quantization, and knowledge distillation, a field now known as tinyML [1]. This approach enables models to run efficiently on devices with limited memory and processing power, but often at the expense of reduced performance and accuracy compared to the larger, more complex models typically used in cloud or edge servers (ESs).

Somewhere between the two extremes of performing all computations entirely on one location – on the device or on the server, there are collaborative approaches. These strategies include methods such as deep neural network (DNN) partitioning [2], [3], inference load balancing [4], [5], and Hierarchical Inference (HI) [6]. While DNN-partitioning splits a DNN across the two locations allowing different layers to be processed at different locations, inference load balancing involves distributing the computational workload by dynamically allocating inference tasks based on device capabilities, job execution times, and average test accuracy of the DL or

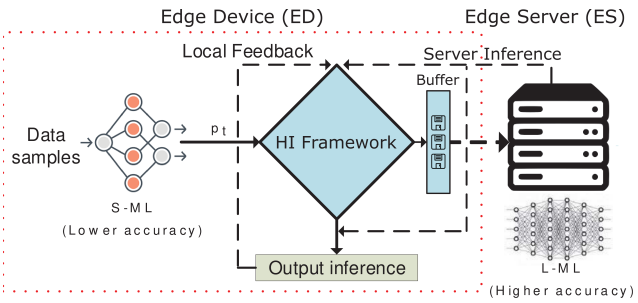


Fig. 1: Illustration of the hierarchical inference framework.

This work has been supported by VR grant 2022-03922 - Optimal Sampling for Interactive Networked Applications.

machine learning (ML) models. In contrast, HI aims to balance accuracy and system and resource costs by allowing tasks to be offloaded only when needed thereby achieving a sweet spot between the high accuracy achieved with offloading and the low system costs of the local processing.

In HI framework, as illustrated in Fig. 1, the edge devices are equipped with a constrained small size ML (S-ML) model that has lower accuracy. In contrast, the server is equipped with a state-of-the-art large ML size model (L-ML) which provides higher accuracy if explicitly requested by the edge device. The HI in the middle makes these explicit decisions, by comparing a metric – the softmax value output by the S-ML – with a threshold. However, with an unknown inference accuracy distribution, finding this threshold is a stochastic optimisation problem and needs to be learned dynamically in an online setting. Achieving this with sublinear regret bounds is the key contribution of our earlier work [6].

In the recent years, HI and closely related works have gained considerable attention [6]–[12]. In [7], a threshold for the maximum softmax value based on the device's transmission energy constraints is determined for offloading decision. In [11] HI framework was expanded to handle multiple devices, focusing on optimizing accuracy while adhering to a constraint on offloading costs. The effectiveness of HI in several use cases is shown in [8] by comparing it with other DL inference techniques. Furthermore, in [12], the authors proposed Hedge-HI and Hedge-HI-Restart algorithms for the HI learning problem, achieving improved regret bounds and lower computational complexity, particularly in resource constrained environments. In [9], it is shown that using a simple linear regression model on the two highest softmax values instead of using a fixed threshold would further enhance the efficiency of offloading decision. [10] presents a measurements based system implementation that systematically compares HI with on-device inference, measuring accuracy, latency, and energy consumption across multiple devices using different image classification datasets.

Existing works on HI learning rely on immediately offloading data when such a decision is made. However, this approach causes the next sample to wait for feedback, where the time depends not only on the known image size and data rate but also on network delay, which can vary significantly in modern wireless systems, especially with a wide range of possible server location. This uncertainty is unfavorable for many applications where an HI system would otherwise be suitable.

The major contribution of this work is the introduction of *Batch HI*, an approach that tackles the above challenge by offloading samples in batches of adjustable size. We show that the proposed algorithm achieves similar regret bounds while introducing a responsiveness trade-off. In addition, we compute the optimal batch size that provides a customized balance tailored to specific user requirements. We summarise these contributions in the following:

1) Introduction of *Batch* HI: We propose a new algorithm that extends the framework of hierarchical inference to process samples in batches of adjustable size according to the requirements of the specific application.
2) Derivation of updated regret bounds and optimal batch size: We perform analysis and derive regret bounds.
3) Numerical evaluation: We perform comprehensive numerical investigations across various parameters, analyzing how batching adapts to different real-time conditions and application requirements.

TABLE I: Table of Abbreviations.

ED	Edge device	ES	Edge server
S-ML	Small ML model	L-ML	Large ML model
HI	Hierarchical inference	HIL	HI Learning

The remainder of this work is organised as follows: In Section II, we briefly discuss the background of HI and the extended system model used in this work. In Section III, we provide a proof expository to the regret bounds and compute the optimum batch size. Finally, we show the performance of batch HI in Section IV and conclude in Section V.

II. Background and System Model

This section is divided into two parts. The first part discusses briefly about the state-of-the-art referred to as *vanilla* HI. In the second part we first extend the system model to include batch processing, followed by the problem statement.

A. Background

Offloading incurs costs related to communication latency, energy consumption, and resource utilization. On the other hand, relying solely on the S-ML may lead to misclassifications, impacting the overall accuracy of the system. HI facilitates balancing the cost of offloading data to the ES against the risk of incorrect local inferences by the S-ML via making strategic decisions to selectively offloading only complex samples.

In round t, HI makes decisions using a threshold on a confidence metric, which in our case is the softmax output p_t. If the confidence metric is below the threshold, HI deems the sample to be *complex* and offloads it for better inference; otherwise, it accepts the S-ML inference, declaring the sample to be *simple*. While an optimal threshold can be obtained offline via brute-force search, finding it in real-time in an online manner is a stochastic optimization problem. In HI, the threshold θ_t at round t is learnt continuously by modeling it as a continuous *expert* in a Prediction with Expert Advice (PEA) setting [13], using the correct sample labels that is assumed to be available later.

The associated costs for the decision are formulated as $0, \beta$ or 1 for a correct inference, offload, and an incorrect inference, respectively. Here β is a fixed value representing the offloading costs such as resource usage, energy consumption, and latency. The cost of round t is denoted by $l(\theta_t, Y_t)$, where Y_t is the binary random variable denoting the correctness of the inference.

The basic steps of the algorithm(s) proposed in [6] are as follows:

1) The S-ML processes the sample t and outputs p_t.
2) If $p_t \geq \theta_t$, the ED accepts the S-ML 's inference; otherwise, it offloads the sample.
3) The ES processes the offloaded samples using the L-ML and feeds the result back.
4) The ED updates its threshold θ_t based on feedback to improve future decisions.

Both algorithms achieve sublinear regret bounds, ensuring that the average regret per sample diminishes as the number of samples n increases. These bounds are extended later section III to build their counterparts for the batch HI. We summarize important abbreviations in TABLE I.

B. System Model Extension and Problem Statement

Consider a system as shown in Fig. 1 deploying HI for image (equivalently referred to as sample) classification applications. The samples arrive at the request of ED, (or ED takes the samples), whenever a S-ML processing opportunity is available. However, instead of offloading *complex* samples immediately, the S-ML processing and decision-making is performed successively on M samples using the same threshold, and a subset m of them are offloaded together at the end. Once they are processed by the ES, feedback is sent to the ED to update the weights, that would be then used to learn the new threshold. The process is then repeated for the next batch. Let τ_{ed} and τ_{es} denote the processing times of the S-ML and L-ML models at the ED and ES, respectively.

This system differs from vanilla HI in a few key performance aspects. First, as the feedback is generated at the end of the batch, samples that require additional server processing must wait until the end of the batch for getting the classification result, thus reducing the system responsiveness for these samples. In contrast, batching improves the total classification time for M samples due to offloading in a single batch, thus providing a trade-off that is the key aspect of this work. Finally, while the weights are continuously updated in vanilla HI; updates occur only at the end of the batch in our model, causing a slightly different threshold for all decisions except the first one in each batch. This difference can lead to varied offloading decisions and classification outcomes for the remaining $M-1$ samples.

Define response time (of a sample) as the average time between start of S-ML processing (the arrival) and the moment when the ED can use the classification output for the sample (the departure). For instance, the response time of a locally processed sample is equal to τ_{ed} and in vanilla HI with $M=1$, the response time for an offloaded sample is the sum τ_{ed}, τ_{es} and the total communication delay. Define t_{r} as the average response time, that we henceforth refer to as response time for simplicity. Smaller batch sizes lead to shorter t_{r} and improved system responsiveness. Unlike the vanilla HI, the response time depends on the number of remaining sample in the batch for batch HI.

Next, define average system time t_{s} as the average contribution of one sample to the total system time. This is the ratio of the time to complete one batch to the batch size.

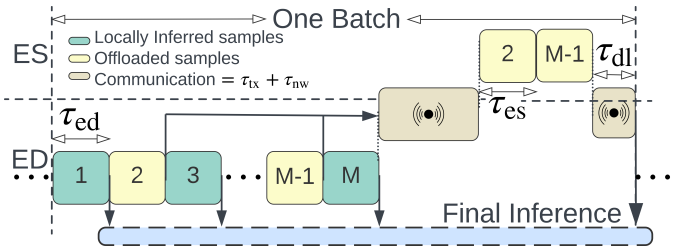


Fig. 2: Timing diagram of the batch HI. Samples 2 and $M-1$ are offloaded. Cumulative communication delay, processing times of samples whose S-ML inference is accepted, and that of samples that are offloaded are shown in grey, green and yellow blocks, respectively, as shown in the legends.

Unlike response time, the average system time decreases with larger batch sizes due to reduced cumulative communication delays. In vanilla HI, the number of transmission attempts per batch depends on m, while in batch HI, there is at most one transmission per batch. This trade-off between responsiveness and system time is the focus of this paper. Note that one could of course consider using the total time for modeling the trade-off but taking this ratio instead ensures that the time is better scaled for optimization.

Now we move on to modeling the communication delay. The total communication delay for an offloaded sample depends on the transmission delay denoted by τ_{tx}, and the cumulative network delay denoted by τ_{nw}. For a fixed sample size B and datarate R, $\tau_{\text{tx}} = {}^{B}/_{R}$. On the other hand, τ_{nw} includes propagation, protocol, retransmissions, and queuing delays in both uplink and downlink combined. While cumulative transmission delay depends on the total payload size, it is unaffected by the number of transmissions. In contrast, the network delay depends on the number of transmissions but not on the payload size. This distinction is a key factor in our optimization problem and hence the reason for this modeling. The timing diagram of the batch HI is illustrated in Fig. 2.

In this work, we introduce a design parameter $\alpha \in (0,1)$ that controls the trade-off between responsiveness and system time. A lower α prioritizes the average responsiveness of the classification task, while a higher α emphasizes minimizing the total task duration. Thus we pose our trade-off as an optimisation problem for a given α:

$$\begin{aligned} M^* &\coloneqq \arg\min_{M\in\mathbb{N}} f(M), \\ \text{where, } f(M) &\coloneqq \alpha\, t_{\text{s}}(M) + (1-\alpha)\, t_{\text{r}}(M). \end{aligned} \tag{1}$$

We summarize important notations in TABLE II and consolidate the problem statement as follows:

1) Establish regret bounds for the batch HI algorithms.
2) Derive the response time and system time as functions of batch size, and establish the trade-off.
3) Obtain optimum batch size by solving (1).

III. Regret Bounds and Batch Size Selection

In this section, we discuss about the batch HI algorithm that offloads the samples in batches. The algorithm outline is

TABLE II: Table of notations.

t	Round (sample) index
p_t	soft-max value output by the S-ML at round t
θ_t	Decision threshold at round t
β	Normalised offload cost
Y_t	Binary random variable denoting the classification correctness
M	Batch size
ρ	Ratio of number of offloaded samples
α	Relative weight given for system responsiveness
t_{s}	Ratio of average system time of a batch to the batch size
t_{r}	Average response time of an sample
τ_{ed}	S-ML processing time at the ED
τ_{es}	L-ML processing time at the ES
τ_{tx}	Transmission time per sample
τ_{nw}	Network time per transmission for offload

Algorithm 1 Batch HI algorithm.

Initialise the weights to 1 and threshold to 0.5.
for $k = 1, 2, 3, \ldots$ **do**
  for $t = (k-1)M + 1, (k-1)M + 2, \ldots kM$ **do**
    S-ML outputs p_t.
    Accept the local inference if $p_t \geq \theta_k$.
    Add the sample to the buffer if $p_t < \theta_k$.
  end for
  Offload and empty the buffer as a batch and obtain the L-ML inferences as feedback.
  Update the weights and the get threshold θ_{k+1}.
end for

given in Algorithm 1, and more details and regret bounds are discussed in the upcoming part of this section.

A. Regret Bounds

Now we look into the regret bounds of the algorithm and the approach to compute it. The regret is defined as the difference between the cumulative loss incurred by the algorithm and the cumulative loss of an hypothetical algorithm that uses the optimum threshold throughout. That is,

$$R_n = \sum_{t=1}^{n} l(\theta_t, Y_t) - \sum_{t=1}^{n} l(\theta^*, Y_t),$$

where θ^* is the unknown optimum θ. This definition remains unchanged between vanilla and batch HIs, while the values of the loss function differs due to the differences in the threshold used at every round as a result of the changes in weight updation frequency.

Due to space constraints and for reducing redundancy, we will not go through the proofs completely. Instead, we provide a brief outline of the proofs and explain where the key differences arise with batching.

1) Proof Expository: We make use of the following: let $\lambda_{\min} = \min_{t_1,t_2}(p_{t_1} - p_{t_2}) : p_{t_1} - p_{t_2} > 0$ denotes the minimum positive difference between any two sof-max values, η denotes the learning rate, $w_t(\theta_t)$ denotes the weight function as a function of the continuous experts θ_t at round t and $W_t \coloneqq \int_0^1 w_t(\theta)\mathrm{d}\theta$ be the normalisation factor.

As the weights are updated only once per batch, they are unchanged within a batch and we have,

$$w_{(k-1)M+i} = w_{kM}, \forall k \geq 1, 1 \leq i \leq M,$$

where k is the batch index. Define $\hat{w}_k$, and $\hat{W}_k$ corresponding to a batch:

$$\hat{w}_k = w_{kM},$$
$$\hat{W}_k = W_{kM}.$$

The weights of each batch are computed as $\hat{w}_k(\theta) \coloneqq e^{-\eta \sum_{\tau=1}^{(k-1)M} l(\theta, Y_\tau)}$ by using an exponentially weighted average forcaster (EWF) [13].

Usage of these weights corresponding to a batch – instead of the weights of each round – is one of the few key difference in the derivation of regrets bounds for the batch HI. The derivation steps are as follows:

1) Write the ratio $\ln(\hat{W}_{n+1}/\hat{W}_1)$ in terms of the EWF and rewrite them in terms of the loss functions:
$$\ln \tfrac{\hat{W}_{n+1}}{\hat{W}_1} \geq -\eta \min_{\theta_t \in [0,1]} \sum_{t=1}^{n} l(\theta, Y_t) - \ln(1/\lambda_{\min}). \quad (2)$$
2) By using Hoeffding's lemma on $\ln(\hat{W}_{k+1}/\hat{W}_k)$ and expanding telescopically, bound the same ratio $\ln(\hat{W}_{n+1}/\hat{W}_1)$ in terms of the expected cost $\overline{l}(Y_t)$, where the expectation is taken over $q_t = (1/\hat{W}_t) \int_0^{p_t} \hat{w}_t(x)\,\mathrm{d}x$:
$$\ln\left(\tfrac{\hat{W}_{n+1}}{\hat{W}_1}\right) \leq -\eta \sum_{t=1}^{n} \overline{l}(Y_t) + \tfrac{nM\eta^2}{8}. \quad (3)$$
3) Combine (2) and (3) to get the regret bound:
$$R_n \leq \tfrac{1}{\eta} \ln(1/\lambda_{\min}) + \tfrac{nM\eta}{8}. \quad (4)$$
4) Find the learning rate that minimizes the regret bound (albeit not necessarily the regret), and use that to get the optimum regret bound:
$$R_n \leq \sqrt{nM \ln(1/\lambda_{\min})/2}. \quad (5)$$

The key difference from vanilla HI arises as a result of the usage of Hoeffdings Lemma in steps (2). As a result of considering the losses of a batch, the upperbound goes from 1 in vanilla HI to M. Further, the number of batches for a fixed number of samples scales inversely with M. In combination, we obtain a looser bound in the order of $\sqrt{M}$.

B. Optimum Batch Size

Now we will look into the batch size in more detail. As already discussed, batching reduces the system time at the cost of responsiveness, a trade-off posed as an optimisation problem given in (1).

Recall that the average system time t_{s} is the ratio of the expected time to complete one batch divided by the batch size. Recall also that the time to complete a batch includes M S-ML processing times, the transmission delay, the network delay and the L-ML processing time for all offloaded samples. We assume the feedback to be of negligible size and that, on average $\rho \cdot M$ samples are offloaded per batch. While the

number of offloaded samples in any given batch is necessarily a whole number, we use the average number characterised by ρ that need not be an whole number. We have

$$t_{\mathrm{s}}(M) = \frac{1}{M}\left(M\tau_{\mathrm{ed}} + \rho M(\tau_{\mathrm{es}} + \tau_{\mathrm{tx}}) + \tau_{\mathrm{nw}}\right) = \tau_{\mathrm{ed}} + \rho(\tau_{\mathrm{es}} + \tau_{\mathrm{tx}}) + \frac{1}{M}\tau_{\mathrm{nw}}. \quad (6)$$

Conversely, the response time t_{r}, that deteriorates with larger batch size, is the average time between the start of S-ML processing and the time the ED can use the classification output for a sample. Here, we assume the scenario where all the offloaded samples are clustered at the start of the batch. As all the initial samples need to wait for the feedback, this scenario presents us with the worst case in terms of responsiveness while not affecting t_{s}. To calculate the (worst case) average responsiveness t_{r}, we consider the time from when the frame is captured on the edge device to when the classification result is available for use at the ED. If the offloading decision is to accept the local inference, this delay corresponds to just τ_{ed}. On the other hand, for the offloaded samples, the time is the time between the start of the S-ML processing and the reception of the feedback from the ES at the end of the batch and thus depends on the number of remaining samples in the batch. Therefore, we have:

$$t_{\mathrm{r}}(M) = \frac{1}{M}\left(\sum_{i=1}^{m}\Big((M-i+1)\tau_{\mathrm{ed}} + m(\tau_{\mathrm{nw}} + \tau_{\mathrm{es}})\Big) + (M-m)\tau_{\mathrm{ed}}\right).$$

Now, we assume that the ratio of offloaded samples to the total number of samples converges to a known value of $\rho \in (0,1)$ as the number of samples increases indefinitely. This implies that the expected number of offloaded samples m, for a batch of size M is known, with $\rho = \frac{m}{M}$. By simplification, we get,

$$t_{\mathrm{r}}(M) = \left(\rho\left(1-\frac{\rho}{2}\right)\tau_{\mathrm{ed}} + \rho^2(\tau_{\mathrm{es}} + \tau_{\mathrm{tx}})\right)M + \rho\tau_{\mathrm{nw}} + (1-0.5\rho)\tau_{\mathrm{ed}}. \quad (7)$$

We obtain the the objective function $f(M)$ from (6) and (7):

$$f(M) = \frac{1}{M}\alpha\tau_{\mathrm{nw}} + M(1-\alpha)\left(\rho\left(1-\frac{\rho}{2}\right)\tau_{\mathrm{ed}} + \rho^2(\tau_{\mathrm{es}} + \tau_{\mathrm{tx}})\right) + (1-\alpha)\big(\rho\tau_{\mathrm{nw}} + (1-0.5\rho)\tau_{\mathrm{ed}}\big) + \alpha\big(\tau_{\mathrm{ed}} + \rho(\tau_{\mathrm{tx}} + \tau_{\mathrm{es}})\big). \quad (8)$$

Now consider a relaxed version of the integer optimisation problem (1), where instead of positive integers M can take any real positive number. We find M^* by first finding the optimum argument to the relaxed problem, denoted by $M^\dagger$, and then looking for M^* at the closest integers, given by $\lfloor M^\dagger \rfloor$ and $\lceil M^\dagger \rceil$. The relaxed optimisation problem is defined as:

$$M^\dagger := \arg\min_{M\in\mathbb{R}^+} f(M),$$
$$M^* = \arg\min_{M\in\{\lfloor M^\dagger\rfloor, \lceil M^\dagger\rceil\}} f(M).$$

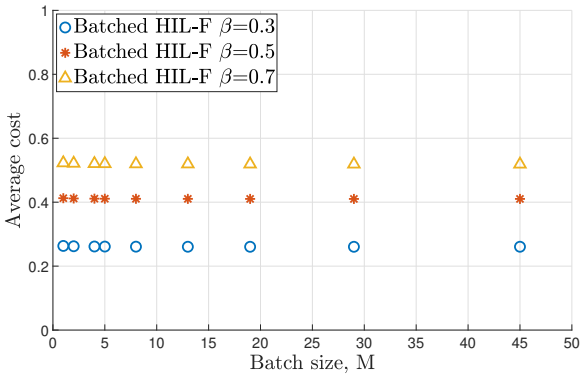


Fig. 3: Average cost vs. the batch size M for different offload cost β, for the Imagenette dataset.

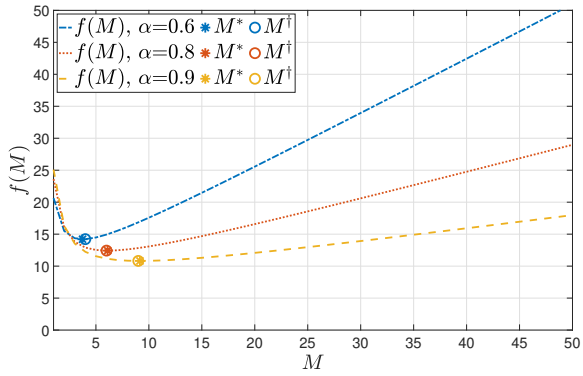
Observe that the objective function (8) is convex in real M. Therefore, we can find $M^\dagger$ by simply using the first derivative:

$$M^\dagger = \sqrt{\frac{\alpha\tau_{\mathrm{nw}}}{(1-\alpha)\left(\rho\left(1-\frac{\rho}{2}\right)\tau_{\mathrm{ed}} + \rho^2(\tau_{\mathrm{es}} + \tau_{\mathrm{tx}})\right)}}. \quad (9)$$

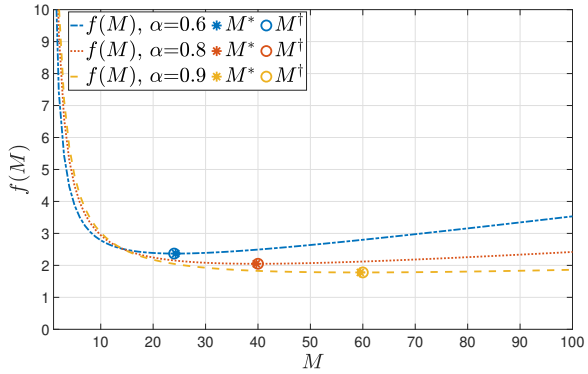
IV. Numerical Evaluation

In this section, we investigate the performance of the proposed batch HI and the effect of batch sizes using numerical simulations, where we use the Imagenette [14] and CIFAR10 [15] datasets. Unless otherwise specified, we use the following parameterizations: The normalised offload cost β for the simulations related to Imagenette is chosen as 0.7 and to CIFAR10 as 0.5, while the batch size M is a variable. While ρ depends on β we have validated that it largely unaffected with the batch size. For processing times, we refer to [10] and choose $\tau_{\mathrm{ed}} = 1.48$ms, $\tau_{\mathrm{es}} = 4.41$ms for Imagenette and $\tau_{\mathrm{ed}} = 0.39$ms, $\tau_{\mathrm{es}} = 4.29$ms for CIFAR10 which corresponding to Jetson 15W edge device with AlexNet ML model for both datasets, and NVIDIA A100 server with ConvText ML model for Imagenette and ViT-H/14 model for CIFAR10. For the nework delay, τ_{nw}, we refer to [16] and use a value of 20ms. We assume that τ_{nw} is equal to the RTT from this work due to the very short packet of 172 Bytes used. This value varies significantly depending on the network condition, and for this reason we do the same in one of our simulations. The average image size is observed to be 30.08 KB for the 3925 Imagenette samples and 3 KB for the 10000 CIFAR10 samples. We used an upink data rate of 50 Mbps, resulting in $\tau_{\mathrm{tx}} = 4.93$ ms and $\tau_{\mathrm{tx}} = 0.49$ ms accordingly.

In Fig. 3, we plot the average cost as a function of the batch size M for the Imagenette dataset. We choose three values of 0.794, 0.671 and 0.421 for the offload ratio ρ, that we obtained from the vanilla HI with $\beta = 0.3, 0.5$ and 0.7, respectively. We observe across different parameterisations that the average cost per round remains largely unaffected by an increase in M in the relatively small sizes of batch size considered. Similar results were observed also for the CIFAR10 dataset, where the average cost for the different parametrizations remains almost constant, simillar to vanillaHI, for relatively small values of



(a) $f(M)$ vs. M for Imagenette dataset with different α.



(b) $f(M)$ vs. M for CIFAR10 dataset with different α.

Fig. 4: The relaxed version of the objective function $f(M)$ as a function of the batch size $M \in \mathbb{R}^+$, for different α, for different datasets. The optimums $M^\dagger$ and M^* are shown.

M. This indicates that while batching improves overal system time, it does not significantly affect the incurred average cost.

In Fig. 4, we examine the objective functions against the batch size for both datasets, for different values of α. First we observe the existence of the optimums M^* and $M^\dagger$, as shown in the figures. It is interesting to note the behaviour of the objective, before and after the optimum. As a result, it is imperative that if we end up choosing an incorrect batch size, for instance due to estimation errors of the parameters in (9), choosing a smaller batch size is costlier that choosing a larger batch size for both cases. In other words, if an error is unavoidable, it is always preferrable to move towards a larger batch size. This further shows the benefits of batching in comparison with the vanilla HI with $M = 1$. Secondly, we also observe from Fig. 4 that as the need for shorter system times increases, the need for larger batch sizes also tends to increase, which is the primary motivation of the batch HI. Thirdly, we observe that the CIFAR10 dataset requires a larger batch size under similar parameter settings. This is due to the significantly lower resolution of its images compared to Imagenette, leading to a smaller τ_{tx} and highlights that batching becomes even more critical in such cases, with much larger batch sizes needed to offset network overhead.

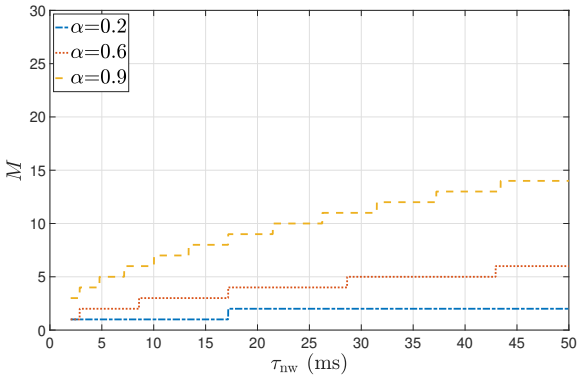


Fig. 5: Optimal batch Size M vs. the network delay τ_{nw}, for different α.

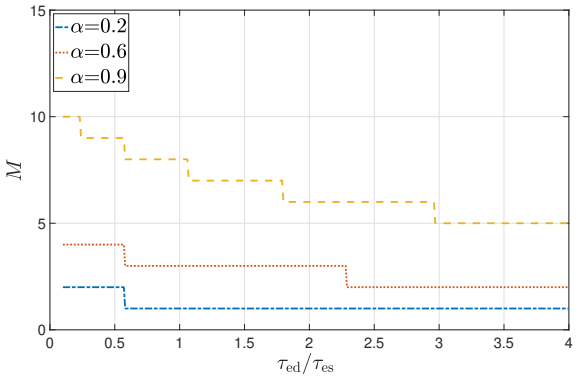


Fig. 6: Optimal batch Size M vs. the ratio of S-ML and L-ML processing times for varying $\tau_{\text{ed}}/\tau_{\text{es}}$, for different α.

Finally, in Fig. 5 and Fig. 6, we investigate how the optimal batch size varies with the system parameters, particularly focusing on how it changes with the network time, τ_{nw}, and the ratio of inference durations at the ED and at the ES, $\tau_{\text{ed}}/\tau_{\text{es}}$. This analysis aims to understand the effect of batching in different real-world setups, taking into account different edge devices, ML models and wireless scenarios. Here, we use Imagenette dataset and an offloading ratio $\rho = 0.421$ corresponding to the offload cost $\beta = 0.7$. In Fig. 5 we vary τ_{nw} between 2 and 50 ms. We observe that as network latencies increase, the optimal batch size also increases. This is because, with higher latencies, offloading in bigger batches reduces the frequency of encountering these delays, thereby decreasing the overall system time. In Fig. 6, we use the default τ_{es} of 4.29 ms and a varying τ_{ed} in the range of 0.429 to 21.45 ms, thereby forcing the ratio $\tau_{\text{ed}}/\tau_{\text{es}}$ to span between 0.1 and 5. We observe that larger the ratio is, the smaller the optimum batch size becomes. As before, this is because, for a relatively larger L-ML processing time – that reduces the ratio – we would want to encounter these delays for a fewer number of times, thus favouring bigger batch sizes.

V. Conclusion

In this paper, we introduced the *Batch* HI framework to address some challenges associated with immediate offloading in traditional HI systems. By offloading samples in adjustable-sized batches of size M, our approach reduces communication overhead and system time while maintaining similar regret bounds to existing HI methods. We established updated regret bounds for the *Batch* HI that scales with $\sqrt{M}$.

Furthermore, we derive analytical expressions for the average system time and the average response time as functions of M, and showed an underlying trade-off that is crucial for systems where these times must be balanced. We formulated the trade-off as an optimization problem to determine the optimal batch size M^* that achieves this balance for a user-defined parameter α. Our solution indicates that the optimal batch size increases with a higher priority on system efficiency and decreases when responsiveness is prioritised.

Numerical evaluations using the Imagenette and CIFAR10 datasets confirmed the effectiveness of the proposed batch HI framework. Unlike the regret bound, the average cost remained largely unaffected by the batch size, indicating that batching does not compromise the cost with smaller ranges of batch sizes. Additionally, we observed that the optimal batch size that minimizes the objective function falls in this range.

In future work, we plan to explore directions such as adaptive batch sizing, where the system adjusts the batch size based on real-time conditions, allowing it to meet varying demands dynamically while maintaining service-level objectives. We also plan to extend this work to deal with the average responsiveness in a probabilistic manner rather than considering the worst case for simplicity.

References

[1] C. Banbury, V. J. Reddi, P. Torelli, J. Holleman, N. Jeffries, C. Kiraly, P. Montino, D. Kanter, S. Ahmed, D. Pau *et al.*, "Mlperf tiny benchmark," *arXiv preprint arXiv:2106.07597*, 2021.

[2] 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 Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[3] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "Spinn: synergistic progressive inference of neural networks over device and cloud," in *Proceedings of the 26th annual international conference on mobile computing and networking*, 2020, pp. 1–15.

[4] Y. Xu, T. Mohammed, M. Di Francesco, and C. Fischione, "Distributed assignment with load balancing for dnn inference at the edge," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 1053–1065, 2022.

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

[6] V. N. Moothedath, J. P. Champati, and J. Gross, "Getting the best out of both worlds: Algorithms for hierarchical inference at the edge," *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[7] I. Nikoloska and N. Zlatanov, "Data selection scheme for energy efficient supervised learning at iot nodes," *IEEE Communications Letters*, vol. 25, no. 3, pp. 859–863, 2020.

[8] 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 *Proc. Workshop on Networked AI Systems, ACM MobiSys*, 2023, pp. 1–6.

[9] A. P. Behera, R. Morabito, J. Widmer, and J. P. Champati, "Improved decision module selection for hierarchical inference in resource-constrained edge devices," in *Proc. ACM MobiCom*, 2023, pp. 1–3.

[10] A. P. Behera, P. Daubaris, I. Bravo, J. Gallego, R. Morabito, J. Widmer, and J. P. Champati, "Exploring the boundaries of on-device inference: When tiny falls short, go hierarchical," *arXiv preprint arXiv:2407.11061*, 2024.

[11] H. B. Beytur, A. G. Aydin, G. de Veciana, and H. Vikalo, "Optimization of offloading policies for accuracy-delay tradeoffs in hierarchical inference," in *Proc. IEEE INFOCOM*. IEEE, 2024, pp. 1989–1998.

[12] G. Al-Atat, P. Datta, S. Moharir, and J. P. Champati, "Regret bounds for online learning for hierarchical inference," in *Proc. ACM MobiHoc (to appear)*, 2024.

[13] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge University Press, 2006.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE CVPR*, 2009, pp. 248–255.

[15] A. Krizhevsky, "The CIFAR-10 dataset," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[16] S. Mostafavi, G. P. Sharma, and J. Gross, "Data-driven latency probability prediction for wireless networks: Focusing on tail probabilities," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 4338–4344.