Improved Decision Module Selection for Hierarchical Inference in Resource-Constrained Edge Devices

Adarsh Prasad Behera
IMDEA Networks Institute
Madrid, Spain
adarsh.behera@imdea.org

Roberto Morabito
University of Helsinki
Helsinki, Finland
roberto.morabito@helsinki.fi

Joerg Widmer
IMDEA Networks Institute
Madrid, Spain
joerg.widmer@imdea.org

Jaya Prakash Champati
IMDEA Networks Institute
Madrid, Spain
jaya.champati@imdea.org

ABSTRACT

The Hierarchical Inference (HI) paradigm has recently emerged as an effective method for balancing inference accuracy, data processing, transmission throughput, and offloading cost. This approach proves particularly efficient in scenarios involving resource-constrained edge devices like micro controller units (MCUs), tasked with executing tinyML inference. Notably, it outperforms strategies such as local inference execution, inference offloading, and split inference (i.e., inference execution distributed between two endpoints). Building upon the HI paradigm, this work explores different techniques aimed at further optimizing inference task execution. We propose three distinct HI approaches and evaluate their utility for image classification.

CCS CONCEPTS

• **Computing methodologies** → **Distributed artificial intelligence**.

KEYWORDS

Edge Computing, Deep Learning, Hierarchical Inference.

[1]Funding Information

[1]This work has been supported by Ministry of Economic Affairs and Digital Transformation, European Union NextGeneration-EU, project TSI-063000-2021-59, and through MSCA-PF project "DIME: Distributed Inference for Energy-efficient Monitoring at the Network Edge" under Grant Agreement No. 101062011

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3615732

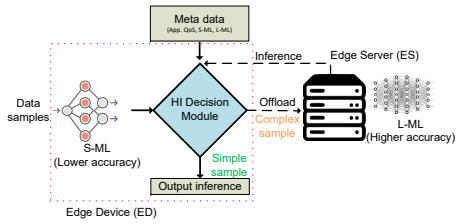


Figure 1: HI framework for DL inference at the edge.

1 INTRODUCTION

Machine Learning (ML) inference has become vital due to its growing application, leading to research expansion, particularly at the network edge. Performing ML inference on resource-constrained Edge Devices (EDs) is challenging because of limited computation and energy resources. Strategies to overcome these include tinyML, edge intelligence offloading to more powerful servers, and Deep Neural Network (DNN) partitioning between an ED and an Edge Server (ES). While these approaches can mitigate the challenges of executing ML inference on EDs, they can introduce other issues such as latency and increased communication costs.

Hierarchical Inference (HI) emerges as a novel framework for distributed Deep Learning (DL) inference at the edge [1]. We consider a system in which an ED is equipped with a tinyML model (referred to here as Small ML or S-ML) and leverages ESc or cloud computing resources, where a state-of-the-art large-size ML model (L-ML) is available. The HI paradigm, shown in Fig. 1, offloads only complex data samples to the ES or cloud, classifying them based on the S-ML's inference requirements. Specifically, a data sample is deemed *simple data sample* if S-ML inference suffices. Differently, a *complex data sample* necessitates L-ML inference. The decision module of HI utilizes the S-ML inference output to determine whether a sample is complex or simple, and subsequently, whether to offload it or not. Unlike traditional tinyML research, HI introduces the flexibility of inference offloading. It scrutinizes the S-ML inference first before deciding whether offloading is necessary, in contrast to existing inference offloading algorithms [5]. With HI, it is possible to take advantage of the resource efficiency of S-MLs on EDs

while maintaining the option to utilize more robust, state-of-the-art L-MLs. However, HI can be implemented effectively if the S-ML fulfills certain conditions: *(i)* the size of S-ML should be small enough for seamless execution on EDs, *(ii)* the energy required for S-ML inference should be lower than that needed for transmitting a data sample, and *(iii)* the accuracy of S-ML should be such that the proportion of *simple data samples* exceeds that of *complex data sample*.

Another critical challenge in the implementation of HI is the identification of complex data samples for offloading. Previous research on HI [1] used a threshold on the highest softmax value from the final layer of the tinyML model embedded in the ED. However, they had to offload more than 35% of total data samples for CIFAR-10 image classification, which increased the cost per image significantly. While HI improves accuracy, its potential can be further tapped through better differentiation of complex and simple data samples. In this work, false negatives (FN) are mistakenly offloaded simple samples, and false positives (FP) are complex ones that are not offloaded. Both add to cost (energy/delay), so minimizing them is crucial for optimal performance. We propose three novel HI approaches to differentiate these samples, evaluating performance with a state-of-the-art tinyML model for CIFAR-10 classification.

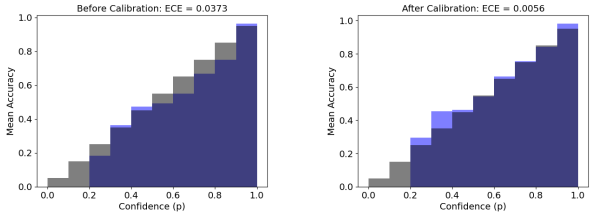
2 PROPOSED METHODOLOGIES

We explore different approaches aimed at overcoming these limitations and increasing the effectiveness and utility of the HI paradigm: *(i)* calibration of tinyML model with fixed threshold, *(ii)* use of classical ML classifiers after tinyML inference, and *(iii)* use of classical ML classifiers before tinyML inference. In this work, we use simple ML classifiers like Logistic Regression (LR), Support Vector Machine (SVM) or Random Forest (RF) due to the limited memory constraints of MCUs used as EDs in our resource constrained set up as DNNs require much more resources. A customized ResNetv1 developed by the MLPerf group [2] is used as our S-ML model on the ED. The model's size is 311 KB, with an impressive benchmark accuracy of 85%, making it the ideal choice for our experiments. Moreover, we have considered a pre-trained ViT-H/14 [3] with a test accuracy of 99.5% as our L-ML model present on the cloud or ES.

2.1 Model Calibration with Fixed Threshold

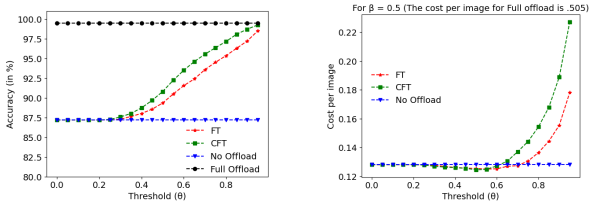
Confidence calibration in probabilistic ML models reflects of the congruence between model predictions and the actual probabilities or levels of confidence [6]. ML models often exhibit overconfidence [4], which results in incorrect predictions. Calibration can be be mathematically defined as:

$$\mathbb{P}\left(\hat{y} = y | \hat{p} = p\right) = p, \quad \forall p \in [0, 1] \tag{1}$$



(a) Before Calibration **(b) After Calibration**

Figure 2: Mean Accuracy vs Confidence in each bin with respective ECE before and after calibration.



(a) Accuracy vs threshold (θ) **(b) Selection of θ^* for $\beta = 0.5$**

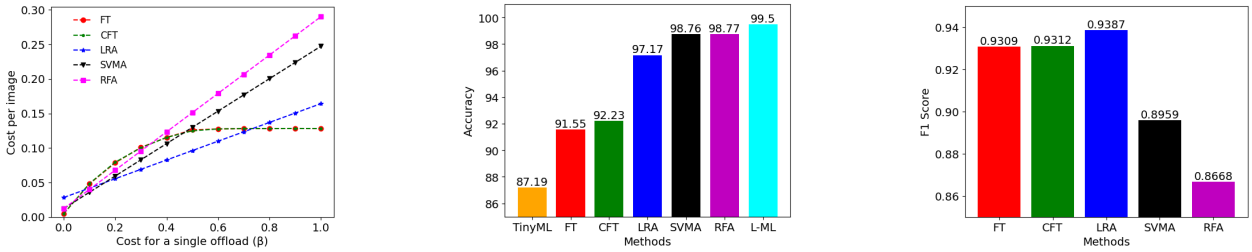
Figure 3: Optimal threshold (θ^*) selection.

where y and $\hat{y}$ represent ground truth and predicted class of any data sample respectively and $\hat{p}$ represents the confidence of the neural network (NN) for that particular prediction. In this approach, we calibrate the tinyML model using "Temperature Scaling" [4] to make the probability estimates more reliable and decide an optimal threshold θ^* for offloading decisions with minimal FPs and FNs.

To estimate the expected accuracy from finite samples, we group predictions into 10 interval bins (each of size 0.1). Fig. 2 shows the tinyML model's mean accuracy relative to each bin's confidence. As discussed earlier, the energy consumption per tinyML inference (denoted as α), is considered negligible compared to the energy consumption associated with offloading each sample to the ES or cloud (represented by β, here $\beta = 0.5$) and the cost of an incorrect inference (denoted as γ, here $\gamma = 1$). Fig. 3 shows the optimal threshold (θ^*) selection, based on the cost per image for a fixed value of $\beta = 0.5$.

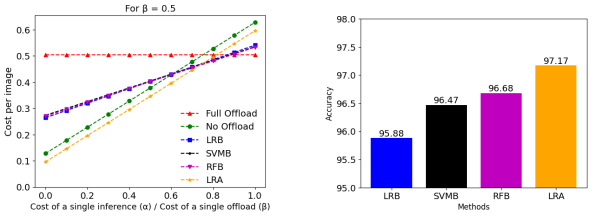
2.2 Use of Classifiers after TinyML

In this approach, all images are processed at the ED. The tinyML's softmax layer provides probability estimates, which are learned by conventional ML classifiers like LR, SVM, and RF on the edge device. The offloading decision is made based on these estimates. At first, the tinyML is trained on 50000 training data and tested on the rest 10000 images. The complex data samples are identified in these 10000 images and correspondingly assigned a class (e.g. class 1) and the rest of the data samples are assigned another class (e.g. class 0). Due to significant class imbalance, down-sampling is performed on simple samples. These images are divided into



(a) CPI for HI classifiers after tinyML **(b) Accuracy comparison** **(c) F1 Score comparison**

Figure 4: Comparative Performance Analysis of various approaches under HI classifiers after tinyML, when $\alpha \longrightarrow 0$.



(a) CPI comparison with LRA ($\beta = 0.5$) **(b) Accuracy comparison with LRA**

Figure 5: Comparative Performance Analysis of ML algorithms before TinyML.

an 80 : 20 train-test ratio, and the classifiers are trained, with the weights saved.

2.3 Use of Classifiers before TinyML

When α is significant compared to β, identifying complex samples in advance for offloading to the ES or cloud may be beneficial. We train classifiers to detect complex samples before tinyML inference, directly offloading them without ED processing. The classifiers are trained on images to execute a binary task, predicting whether samples are complex or simple for the given tinyML.

3 RESULTS

Fig. 4 shows the performance of calibration with fixed threshold (CFT) and ML classifiers after tinyML (LRA, SVMA, RFA) to the fixed threshold method (FT) proposed previously [1]. For this case, we assume α is negligible with respect to β, hence set $\alpha = 0$. It can be observed from Fig. 4(a), as β increases, the CPI increases initially for all the techniques. However, after β reached 0.5, the CPI for FT and CFT stagnated while for the ML classifiers CPI increases linearly with β. It can also be observed that for the majority of the region ($0.2 \leq \beta \leq 0.7$) the CPI for LRA is the lowest. Furthermore, in Fig. 4(c) it can be noticed LRA outperformed all other techniques in terms of $F1$ score. Although the overall accuracies of SVMA and RFA are a little higher than that of LRA as seen in Fig. 4(b), LRA outperforms its counterparts due to a lower number of offloads while also restricting the number of FP and FN. In Fig. 5, we consider α is non-negligible relative to β and present the comparative performance analysis of ML classifiers (LRB, SVMB and RFB) before tinyML. Fig. 5(a) shows CPI for varying α/β for a constant β value of 0.5. It is worth noting that as previously discussed one important assumption for HI is the energy required for tinyML inference should be lower than that needed for transmitting a data sample or $\alpha \leq \beta, \forall \beta$. In Fig. 5(a), it can be observed that the CPI for each technique increases linearly with an increase in α/β value. All three classifiers LRB, SVMB, and RFB perform almost identically. Interestingly, compared to the classifiers before tinyML, CPI for LRA is the lowest till α/β reaches 0.7. There is a small margin $0.7 < \alpha/\beta < 0.9$, where classifiers before tinyML techniques outperform their counterparts. As α approaches β (for $\alpha/\beta \geq 0.9$), CPI for full offload became the lowest, suggesting it is better to offload all the data samples rather than doing any inference at the ED. The overall system accuracy for each classifier before tinyML, compared to logistic regression after tinyML, is shown in Fig. 5(b).

4 CONCLUSION AND FUTURE WORKS

In this work, we proposed three HI selection strategies aimed at optimizing offloading performance. Experiments showed that using logistic regression after tinyML (LRA) often produced the best outcome in HI, considering cost and $F1$ score. In the future, we plan to embed these models into MCUs to assess energy needs and validate accuracy in system implementations.

REFERENCES

[1] Ghina Al-Atat, Andrea Fresa, Adarsh Prasad Behera, Vishnu Narayanan Moothedath, James Gross, and Jaya Prakash Champati. 2023. The Case for Hierarchical Deep Learning Inference at the Network Edge. In *Proceedings of the 1st International Workshop on Networked AI Systems.*

[2] Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Jeremy Holleman, Nat Jeffries, Csaba Kiraly, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, et al. 2021. Mlperf tiny benchmark. *arXiv preprint arXiv:2106.07597* (2021).

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR.*

[4] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *ICML*. PMLR, 1321–1330.

[5] Han Xiao, Changqiao Xu, Yunxiao Ma, Shujie Yang, Lujie Zhong, and Gabriel-Miro Muntean. 2022. Edge intelligence: A computational task offloading scheme for dependent IoT application. *IEEE Transactions on Wireless Communications* 21, 9 (2022), 7222–7237.

[6] Xu-Yao Zhang, Guo-Sen Xie, Xiuli Li, Tao Mei, and Cheng-Lin Liu. 2023. A survey on learning to reject. *Proc. IEEE* 111, 2 (2023), 185–215.