Training Up to 50 Class ML Models on 3 $ IoT Hardware via Optimizing One-vs-One Algorithm (Student Abstract)

Bharath Sudharsan[1 2]

[1]ARM Machine Learning Infrastructure, Galway, Ireland
[2]Confirm SFI Research Centre for Smart Manufacturing, Data Science Institute, NUI Galway, Ireland
bharath.sudharsan@insight-centre.org

Abstract

Multi-class classifier training using traditional meta-algorithms such as the popular One-vs-One (OvO) method may not always work well under cost-sensitive setups. Also, during inference, OvO becomes computationally challenging for higher class counts K as $O(K^2)$ is its time complexity. In this paper, we present *Opt-OvO*, an optimized (resource-friendly) version of the One-vs-One algorithm to enable high-performance multi-class ML classifier training and inference directly on microcontroller units (MCUs). *Opt-OvO* enables billions of tiny IoT devices to self learn/train (offline) after their deployment, using live data from a wide range of IoT use-cases. We demonstrate *Opt-OvO* by performing live ML model training on 4 popular MCU boards using datasets of varying class counts, sizes, and feature dimensions. The most exciting finding was, on the 3 $ ESP32 chip, *Opt-OvO* trained a multi-class ML classifier using a dataset of class count 50 and performed unit inference in super real-time of 6.2 ms.

Introduction

The majority of IoT devices such as smart plugs, thermostats, fitness bands, etc. have MCUs as their brain. Ultra-low-power machine learning (TinyML) is a fast-growing research area committed to democratizing ML for commodity MCU-based devices. Currently, MCUs are not capable to train full ML models due to their resource constraints such as limited memory, low operations per second, parallel processing inability, etc. The top TinyML studies are rapidly advancing only towards the efficient ML inference on MCUs, where the model is first trained on a data center GPU using a historic dataset, then C-code for the trained model is generated and flashed on MCUs. This process impedes the flexibility of billions of deployed ML-powered IoT devices as they cannot learn unseen data patterns (static intelligence) and are impossible to adapt to dynamic scenarios. In this paper, we present *Opt-OvO*, a contribution to the TinyML domain by enabling high-performance training and inference on MCUs.

Optimized One-vs-One Algorithm (Opt-OvO)

Currently, trainable algorithms are attached to an existing model deployed on MCUs (Cai et al. 2020) to perform online/continuous learning. The training of a full multi-class ML classifier on commodity MCUs, using any existing algorithms is currently not feasible. When analyzing the OvO method, we discovered that the OvO's $k(k-1)/2$ base learners/classifiers, for a few datasets, contain classifiers that lack significant contributions to the overall multi-class classification result - this occurs when a classifier is already within a big interdependent group. Hence in *Opt-OvO*, we propose to identify and remove the less important base classifiers to improve the resource-friendliness of OvO.

***Opt-OvO* Algorithm Summary.** In **Step1**, the $k(k-1)/2$ base classifiers b_i belonging to B are trained with the unseen/fresh local data stream using base learner of choice like SVM, LDA, followed by evaluation of all thus trained base classifiers. Here, each base classifiers b_i produces a binary output $\in \{-1, +1\}$ for each input vector $x^{(n)}$. In **Step2**, for all test data, we store outcomes of base learners R_i in R_B. Then, we create a correlation matrix C_m using the output of base classifiers stored in R_B. From C_m, we find $Corr_{class}$, which is the group of highly correlated base classifiers. In **Step3**, from the groups of this found correlated base learners, we create a Probability Table (PT) of each group to know the joint probability of the outcome R_B. These PTs provide the joint probabilities of the outcomes R_B and the groups of correlated classifiers $b_{corr} \subset Corr_{class}$ when evaluating using new/unseen data. In **Step4**, finally, we classify for any new multi-class input $x^{(n)}$ by using thus produced $Corr_{class}$ and set of base classifiers B.

Experiments and Results. We show audience the ML model training on MCU in action with high performance and accurate results transparently from the Serial port of MCUs. We select datasets D1, D2 using which *Opt-OvO* [1] trains classifiers on 4 tiny MCU boards B1-B4. The training performance is presented in Figure 1. a-b. Here, even on the slowest B2, *Opt-OvO* trained using 10 classes D2 in 29.6 sec and 7.6 sec using the 15 classes data of D1. The 3$ ESP32 B1 trained in 0.4 sec for D2 and in 4.7 sec using the 50 classes data of D1. We present inference performance in Figure 1. c. Here, even for high 64 dimensional D2, *Opt-OvO* achieves real-time unit inference of 11.8 ms, even on the slowest B2. The cheapest 3 $ B1 was able to infer for a 50 class input in 6.2 ms. Overall, *Opt-OvO* performed unit inference for multi-class data in su-

[1] *Opt-OvO* implementation, performance report, etc: https://github.com/bharathsudharsan/Optimized-One-vs-One-Algorithm

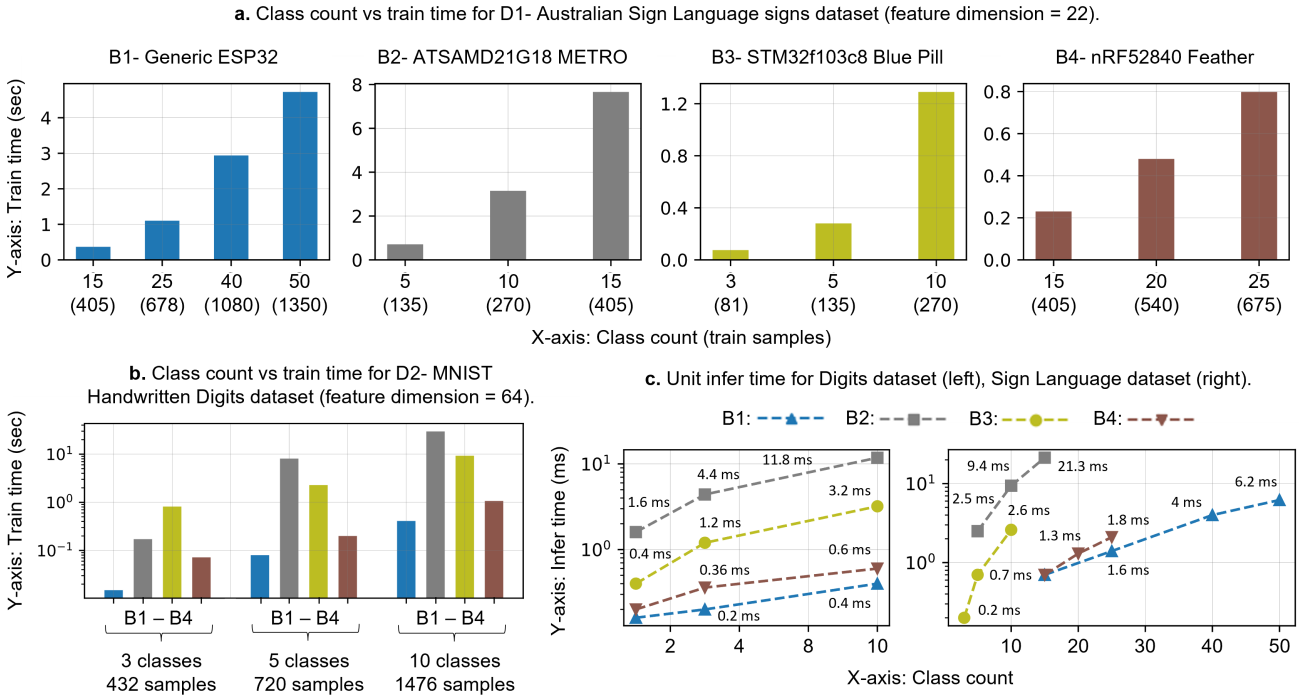

Figure 1: High performance ML model training and inference on MCU boards using *Opt-OvO*.

per real-time, within a second, across B1-B4. We also report that *Opt-OvO* trained models to achieve similar accuracies as the *Python scikit-learn* models.

Opt-OvO Applications

During the IoT device programming phase using Arduino IDE, Atmel Studio, Keil MDK, etc., *Opt-OVO* code needs to be fused with the use-case IoT application. Then, when the labeled data fields that correspond to the low accuracy inference performed are passed to *Opt-OVO*, it can train and update the current classifier version with a superior performance version. *Opt-OVO* is also applicable to other self-learning settings where it can locally train a model from scratch without needing cloud-based ML training services or proprietary datasets. Here, instead of passing only the data that correspond to the low accuracy inference, the complete live data stream should be cleaned, labeled, and passed to *Opt-OVO*. Following are example self-learning use cases.

Self-learning HVACs for Superior Thermal Comfort. Currently, HVACs in smart buildings control internal environment using a standard strategy. Such a one-size-fits-all approach can fail to provide a superior level of thermal comfort for people because every infrastructure has differences (e.g., building size, thermal confinement). In this scenario, if the HVAC control edge devices are equipped with *Opt-OVO*, they can learn the best strategy (offline) to perform tailored control of the HVAC system for any building type, eliminating the need to find and set distinct strategies for each building.

Conclusion and Future Work

We presented *Opt-OvO* algorithm, which achieves reduced computation than OvO by identifying and removing base classifiers that lack significant contributions to the overall multi-class classification result. As demonstrated, *Opt-OvO* enables high-performance ML model training and inference on MCUs. *Opt-OvO* can be used as a key component to enable practicing split-learning, distributed ensemble learning, federated learning, centralized learning by involving even the resource-constrained devices in complex ML-based learning tasks. In future work, similar to the TinyML benchmark (Sudharsan et al. 2021), we plan to use more sophisticated datasets and conduct extensive onboard training plus inference experiments involving the latest pocket-friendly FPGAs, SoCs and AIOT boards.

Acknowledgements

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/16/RC/3918 (Confirm) and also by a research grant from SFI under Grant Number SFI/12/RC/2289_P2 (Insight), with both grants co-funded by the European Regional Development Fund.

References

Cai, H.; Gan, C.; Zhu, L.; and Han, S. 2020. Tinytl: Reduce memory, not parameters for efficient on-device learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Sudharsan, B.; Salerno, S.; Nguyen, D.-D.; Yahya, M.; Wahid, A.; Yadav, P.; Breslin, J. G.; and Ali, M. I. 2021. TinyML benchmark: Executing fully connected neural networks on commodity microcontrollers. In *IEEE 7th World Forum on Internet of Things (WF-IoT)*.