On the Reproducibility of Experiments achieved by TinyRCE

On the Reproducibility of Experiments achieved by TinyRCE

Danilo Pau[1]*, Andrea Pisani[1]**, and Alessandro Carra[1]

1 System Research and Applications, STMicroelectronics Italia, Via Paracelso 20, 20864 Agrate Brianza, MB, Italy

** Fellow, IEEE and AAAIA*

*** Member, IEEE*

Abstract—TinyRCE is a hyperspherical classifier aimed at Continual Learning On-Tiny-Devices, a challenging task in which a Machine Learning model is required to learn from continuous streams of data while being directly installed on a (tiny) device with limited computational resources. The classifier has so far been applied to several use cases, including Human Activity Recognition, Ball Bearing Anomaly Classification, Keyword Spotting and Image Classification. The proposed work in this paper focuses on the reproducibility of TinyRCE's experimental results already published on other papers. This to prove that all the published results are quantitatively reproducible. All the experiments have been executed on two independent computing machines to profile the impact on accuracy of the computations. As the outcomes are matching, the experimental reproducibility of TinyRCE's accuracy over all the use cases has been positively verified.

Index Terms—Experimental Reproducibility, Hyperspherical Classifier, Image Classification, Keyword Spotting, Restricted Coulomb Energy Classifier, TinyML, Visual Wake Words.

I. INTRODUCTION

In the context of Machine Learning (ML), the complexity of problems that modern models can solve is steadily growing. Model complexity, in terms of memory footprint and required computational power, is unfortunately steadily growing as well. Thus, deploying ML on the Edge, i.e. on low-power Internet of Things (IoT) devices, is a challenging task [1]. The TinyML Foundation*, a growing community of research groups and tech companies, was established to foster the development of low-complexity ML technologies, including hardware, software, tools and new models. These models are meant to solve their targeted problems accurately, and also to be deployable on Edge devices such as microcontrollers (MCU) and/or sensors.

Currently, the majority of TinyML models are trained on the Cloud [2] or on a computer that is equipped with a powerful graphical processing unit (GPU) in a supervised fashion, and then deployed on the target MCU or sensor for the inference phase only. On-Tiny-Device Learning models are instead capable of being trained directly on the target device, in real time w.r.t the acquisition of data [3]. Thus, their model complexity estimations must include the training phase as well.

Continual or Incremental Learning (CL) [4] is a branch of ML where the source of information is a continuous stream of data. In such context, a learning agent needs to neverendingly adapt to incoming information.

TinyRCE is a Hyperspherical Classifier targeted at the execution of On-Tiny-Device CL. It has been tested on several devices and use cases. In particular, it was tested over CL workflows regarding Human Activity Recognition (HAR) and Ball Bearing Anomaly Classification (BBAC) in [5], while in [6] it was tested against MLCommons'** standard benchmark networks over Keyword Spotting (KWS) and Image Classification (IC).

The proposed work demonstrates that the results claimed in [5] and [6] are fully reproducible: in fact, the tests were executed on two different machines, yielding to the same results.

The problem this work was set to solve is described in Section II; Section III explains the experimental structure, while the results are listed and discussed in Section IV. Finally, Section V draws several conclusions and lists a few potential future developments.

II. PROBLEM STATEMENT

The experiments this work describes targeted the following question:

'***Are the experimental results related to TinyRCE fully reproducible regardless of the hardware on which the programs are executed?***'

Reproducibility is the cornerstone of the scientific method's reliability: when experiments can be repeated with consistent results not only are the initial findings validated, but, most importantly, the experiment's hypotheses are verified. Furthermore, any incongruences found in the result may lead to the identification of potential errors or biases. Thus, testing any model for reproducibility is of paramount importance.

Corresponding author: D. Pau (danilo.pau@st.com).

* https://www.tinyml.org/

** https://mlcommons.org/en/

III. STRUCTURE OF THE EXPERIMENT

TinyRCE's learning algorithm has been coded in a Python class. Each experimental use case (HAR, BBAC, KWS, and IC) had its own separate directory, complete with Python scripts for dataset preprocessing, feature extraction, and training/testing workflows. All the experiments were run using the same Conda environment, which included several open Python libraries, installed using `pip`. Python version 3.9.12 was used, along with version 2.12.0 of TensorFlow. The following is a list of the included libraries:

- Matplotlib;
- NumPy;
- TensorFlow;
- TensorFlow Datasets;
- CuPy;
- Tqdm;
- Scikit Learn;
- Bayesian Optimization (exact name: bayesian-optimization);
- Pandas;
- Seaborn;
- KerasCV.

The installation of the libraries triggered the automatic installation of their dependencies, as well. As the installation of dependencies is an automated process, they are not included in the list.

The experiments were run independently on two different Linux workstations, codenamed W1 and W2. Details on the specifications of these workstations, including processor model, provider, clock frequency, RAM and Flash memories, and GPU model and provider, as well as operative system (OS) version, are provided in Table 1.

Table 1. Property comparison between workstations.

Property	W1	W2
Linux Version	Ubuntu 20.04	Ubuntu 22.04
CPU model	Intel® Core™ i7-6700	Intel® Xeon® Silver 4114
CPU cores	8	20
CPU frequency	3.40 GHz	2.20 GHz
GPU model	NVIDIA Quadro P400	NVIDIA Quadro RTX 4000
RAM capacity	16.0 GiB	32.0 GiB
Flash capacity	1.00 TB	1.00 TB

IV. RESULTS

The experimental results categorized by use case are reported in this section. Please refer to [5], [6] for a detailed description of the performed experiments.

A. HAR

HAR experiments were performed over two different datasets (SHL [7] and PAMAP2 [8]). Both involved simulating a continuous data stream. The Feature Extractor (FE) was a custom-made Extreme Learning Machine (ELM) based on a Convolutional Neural Network (CNN) topology. Classes were presented sequentially with a train-test split of 20%-80%, the opposite of the classical 80%-20% one adopted by supervised learning. The maximum number of performed training epochs was 6. Results are reported in Table 2. As there was no testing apart from the CL one, test set accuracy was not measured.

Table 2. Results for HAR experiments.

Dataset	Avg single class accuracy %	
	W1	W2
SHL	99.62	99.51
PAMAP2	91.51	90.91

B. BBAC

The BBAC experiments involved the CWRU [9] dataset. All other experimental conditions were equal to those reported in section a. Results are reported in Table 3.

Table 3. Results for BBAC experiments.

Dataset	Avg single class accuracy %	
	W1	W2
CWRU	95.14	95.68

C. KWS

The KWS experiments were performed over the Speech Commands v2 [10] dataset. After a preprocessing pipeline that transforms the raw audio data into Mel Frequency Cepstral Coefficients (MFCC), the data is fed into a CNN FE, which is an off-the-shelf Depthwise-Separable CNN (DS-CNN), trained for 36 epochs. In this context, TinyRCE has been tested for both Continual and K-Fold Learning. In CL, a 20%-80% train-test split has been used, just like in the previous experiments. In K-Fold Learning, the dataset was divided into 5 folds. The results achieved by TinyRCE have been compared to those of a SoftMax classifier. All said results are reported in Table 4, where non-matching results are highlighted in bold.

Table 4. Results for KWS experiments.

Classifier	Learning method	Avg single class accuracy %		Test set accuracy %	
		W1	W2	W1	W2
TinyRCE	CL	95.25	95.25	89.49	89.49
TinyRCE	K-Fold	**-**	-	**74.57**	**87.47**
SoftMax	CL	**17.01**	**34.52**	**4.94**	**27.71**
SoftMax	K-Fold	-	-	97.16	97.16

The mismatch between results in the experiments involving TinyRCE and K-Fold learning have several possible causes, involving most likely the installed versions of NumPy, the interaction between the NumPy library and the workstations' OS versions, or some difference in the exploitation of hardware-level algorithm optimizations. Nevertheless, there is evidence backing the claim that the experiment is reproducible: in workstation W1, TinyRCE showed identical results to those obtained in W2 for all folds except the first one, where it scored a lower accuracy, for reasons, as mentioned, yet to be fully understood.

The mismatch between results involving SoftMax and CL were caused by the random initialization of weights in the SoftMax layer, which brought to different results over the singular experiments.

When repeating the experiments multiple times, the averages of the obtained results were the same for both workstations: 6.22% for test set accuracy and 14.22% for average single class accuracy.

D. IC

The IC experiments were performed over the CIFAR-10[11] dataset. Images were encoded as NumPy matrices and fed to an off-the-shelf FE based on ResNet-v1, which had been trained for 500 epochs. For CL, TinyRCE has also undergone a process of Bayesian Hyperparameter Optimization in this case. A detailed description of the hyperparameters that were used can be found in [6]. Said hyperparameters included the train-test split, which was set to 57%-43%. Once again, the results were compared to those obtained by a SoftMax classifier. All results are reported in Table 5.

Table 5. Results for IC experiments.

Classifier	Learning method	Avg single class accuracy %		Test set accuracy %	
		W1	W2	W1	W2
TinyRCE	CL	87.17	87.16	78.50	78.50
SoftMax	CL	100.0	100.0	10.0	10.0

V. CONCLUSIONS AND FUTURE WORKS

In this work, the reproducibility of the experiments regarding TinyRCE has been verified. All use cases have been tested on two separate and independent workstations, and the results that have been achieved highlight that the experiments were fully reproducible, given the same code.

Future works regarding TinyRCE will investigate improvements in the FEs and experimentation over other use cases.

REFERENCES

[1] N. Kukreja *et al.*, “Training on the Edge: The why and the how,” *ArXiv Distrib. Parallel Clust. Comput.*, Feb. 2019.

[2] F. Samie, L. Bauer, and J. Henkel, “From Cloud Down to Things: An Overview of Machine Learning in Internet of Things,” *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4921–4934, Jan. 2019, doi: 10.1109/jiot.2019.2893866.

[3] Danilo Pau and Prem Kumar Ambrose, “Automated Neural and On-Device Learning for Micro Controllers,” *2022 IEEE 21st Mediterr. Electrotech. Conf. MELECON*, Jun. 2022, doi: 10.1109/melecon53508.2022.9843050.

[4] M. Delange *et al.*, “A continual learning survey: Defying forgetting in classification tasks.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 1, pp. 1–1, Sep. 2019, doi: 10.1109/tpami.2021.3057446.

[5] Danilo Pau, Prem Kumar Ambrose, Fabrizio Maria Aymone, and Andrea Pisani, “TinyRCE: Forward Learning under Tiny Constraints,” presented at the IEEE MetroXRAINE 2023, Milan, Oct. 2023.

[6] D. P. Pau, A. Pisani, F. M. Aymone, and G. Ferrari, “TinyRCE: Multi Purpose Forward Learning for Resource Restricted Devices,” *IEEE Sens. Lett.*, pp. 1–4, 2023, doi: 10.1109/LSENS.2023.3307119.

[7] H. Gjoreski *et al.*, “The University of Sussex-Huawei Locomotion and Transportation Dataset for Multimodal Analytics With Mobile Devices,” *IEEE Access*, vol. 6, pp. 42592–42604, Jul. 2018, doi: 10.1109/access.2018.2858933.

[8] “Time Series Models - PAMAP2 DataSet.” https://kaggle.com/code/avrahamcalev/time-series-models-pamap2-dataset (accessed Feb. 10, 2023).

[9] “Bearing Data Center | Case School of Engineering | Case Western Reserve University,” *Case School of Engineering*, Aug. 05, 2021. https://engineering.case.edu/bearingdatacenter (accessed Feb. 10, 2023).

[10] P. Warden, “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition,” 2018, doi: 10.48550/ARXIV.1804.03209.

[11] “CIFAR-10 and CIFAR-100 datasets.” http://www.cs.toronto.edu/~kriz/cifar.html (accessed Mar. 07, 2023).