Transfer Deep Learning for TinyML

Evangelia Fragkou *and* Dimitrios Katsaros

Department of Electrical and Computer Engineering
University of Thessaly, Volos

Abstract

Tiny Machine Learning (TinyML) and Transfer Learning (TL) are two widespread methods of successfully deploying ML models to resource-starving devices. In this work we propose a simple but efficient TL method, applied to three types of Convolutional Neural Networks (CNN), by retraining more than the last fully connected layer of a CNN in the target device, and specifically one or more of the last convolutional layers. Our results shown that our proposed method (*FxC*1) achieves about 19% increase in accuracy and 61% increase in convergence speed, while it incurs a bit larger energy consumption overhead, compared to two baseline techniques.

Introduction & Motivation

Tiny Machine/Deep Learning *(Tiny ML/DL)* is a fast-growing field, which boosts machine learning tasks deployment in devices that lack of computational resources, like sensors, IoT devices or networks [5] etc. Moreover, TinyML lessens the energy cost of the possible and now not necessary communication between the edge device and the server. Furthermore, Transfer Learning *(TL)*, is a method in which pre-knowledge of a source model is used, in order to enhance the learning ability of the device in which this model is further applied. TL means either retraining the whole model or the last fully connected layer to the resource-constrained (target) evice and hence boosting its performance with both reducing the training time [2],[1], needed for the model to converge, as it inherits the already learned features and by giving the opportunity for the model to be trained with sufficient already-existed data.

In this work, we combine the two aforementioned techniques, by proposing algorithms that go the common TL and the already known TinyML optimization techniques a step further. Specifically, our endeavor is inspired by the work [4], in which authors propose the retraining of the fully connected layers to low-powered device. We present methods of retraining, which include also one or more convolutional layers, with the one that uses only one convolutional layer to be our best method, due to the fact that we achieve increase in accuracy levels, with the less energy consumption [3], regarding to the other two methods, proposed in this work.

The basic motivation is extracted by the way a CNN network performs its tasks. Specifically, in an image classification task, the first layers of a CNN network learn the basic concepts of an image, then this knowledge is transferred to the next layers in order for the final ones to be able to learn the high level concepts of the input-image. This means that the final convolutional layer/layers is/are responsible for the classification of the data, being processed by the network. So, when the network is exposed to new data, it is of high importance to update not only the categorization of data, but also the categories themselves, so as to achieve high Accuracy levels.

By this way, we start to enter the core of the network, without needing the resources, that the whole network requires, in order to operate. Overall, our work contributes in the following:

- We develop a technique where not only the final fully connected layers are retrained, but also some of the immediately previous convolutional layer are retrained.
- We investigate the tradeoff between training more layers versus accuracy versus energy consumption which has not been explored so far. Evidently, retraining more of the last layers (both convolutional and fully-connected) will result in improved accuracy, but it will cost in energy consumption. Is there any optimal point with respect to how many retrained layers are enough before consuming too much energy without gaining in accuracy?
- We use homogeneous data to experiment with, but by affecting a convolution layer through retraining, we can more easily deal with heterogeneous data in the future, since the convolutional layers are the ones responsible for the better understanding and categorization of the hierarchical features of image data; we compare the examined methods against two baseline methods: a) the methods reported in [4] (*Baseline-1*) and b) when the whole CNN is retrained (*Baseline-2*)

The Family of *FxCy* Techniques

The most widespread technique in Transfer Learning is to train the model to a dataset, then 'freeze' the weights and after that either retrain the last fully connected layer, or retrain the whole model to the new data. In our research, we conducted experiments in order to observe how different the model will perform when we retrain more layers than just the fully connected ones and compare our results with the already existed TL methods. So, we created three different experimental cases:

- **F*x* C1**. In this case, we train not only the last fully connected layers, but also the last convolution layer, while the remaining model is frozen.
- **F*x* C2**. In this case, we train the fully connected layers and the last two convolution layers, while the remaining model is frozen.
- **F*x* C3+**. In the last case, we train the fully connected layers and up to three convolution layers, while the remaining model is frozen.

In our experiments, the fully- connected layers of the networks, used, are more than one, so in order to elaborately present our Figures (Figure 1, Figure 2), we display exactly the number of both fully-connected and convolutional layers, taking place in training, by using the abbreviation *FxCy*, while *x* is the number of final fully- connected layers and *y* the number of Convolution layers, re-trained in every experiment.
In Table 1, information regarding the datasets, used, is shown.

Acknowledgements

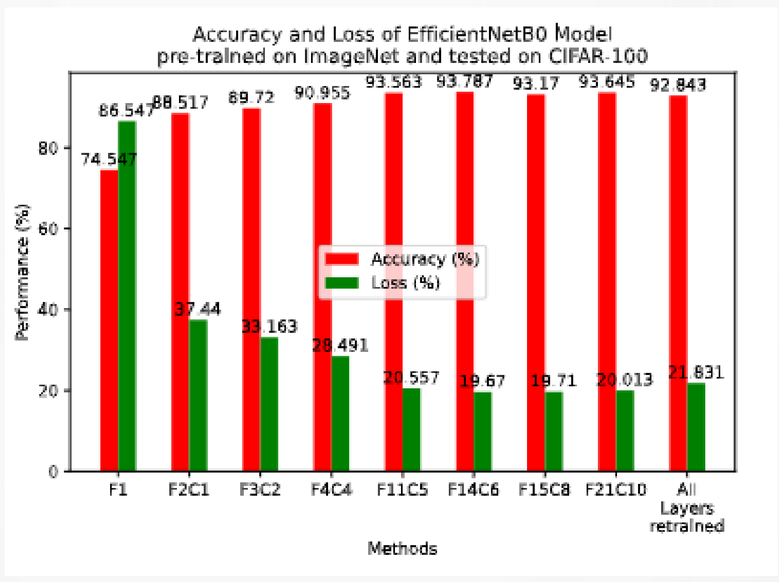

Figure 1: Experimental evaluation (Accuracy – Loss) of EfficientNetB0 model, pre-trained on ImageNet and tested on CIFAR-100.

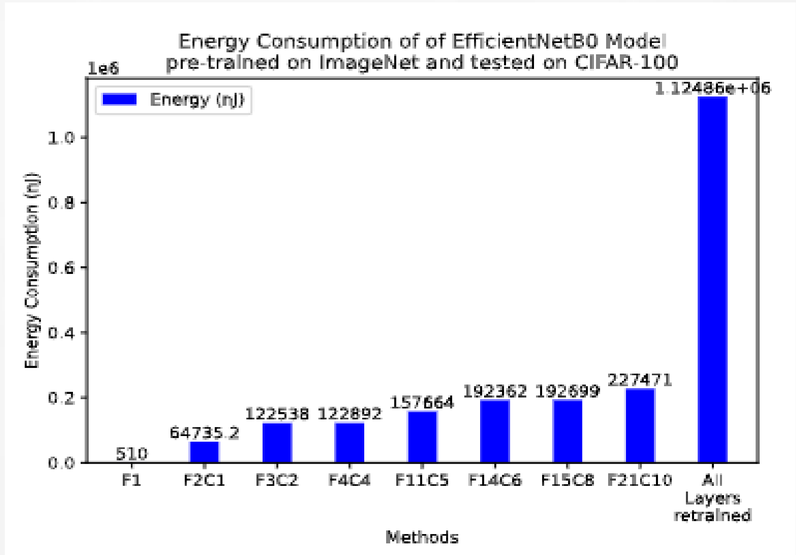

Figure 2: Experimental evaluation (Energy Consumption) of EfficientNetB0 model, pre-trained on ImageNet and tested on CIFAR-100.

Conclusions

Overall, we presented new Transfer Learning methods, deployed to resource-constrained environments. We present a simple but efficient method, in which we don't fine-tune only the last fully connected layer of a pre-trained network, but we also retrain one or more of the convolutional layers of three kinds of a CNN network (a Small CNN, EficientNetB0, EficientNetB2). The results shown that *FxC*1 proposed method achieves approximately 19% increase in network accuracy and about 61% faster convergence.

References

[1] A. Chouliaras, E. Fragkou, and D. Katsaros. 2021. ***Feed Forward Neural Network Sparsification with Dynamic Pruning***. In *25th Pan-Hellenic Conference on Informatics* (Volos, Greece) *(PCI 2021)*. Association for Computing Machinery, New York, NY, USA, 12–17.

[2] E.Fragkou, M. Koultouki, and D. Katsaros. 2022**.** ***Model reduction of feed forward neural networks for resource-constrained devices***. *Applied Intelligence* (2022).

[3] E. Fragkou, V. Lygnos, and D. Katsaros. 2023. ***Transfer Learning for Convolutional Neural Networks in Tiny Deep Learning Environments***. In *Proceedings of the 26th Pan-Hellenic Conference on Informatics* (Athens, Greece) *(PCI '22)*. Association for Computing Machinery, New York, NY, USA, 145–150.

[4] K. Kopparapu and E. Lin. 2021. TinyFedTL: ***Federated Transfer Learning on Tiny Devices***. *ArXiv* abs/2110.01107 (2021).

[5] D. Papakostas, T. Kasidakis, E. Fragkou, and D. Katsaros. 2021. ***Backbones for Internet of Battlefield Things.*** In *2021 16th Annual Conference on Wireless On-demand Network Systems and Services Conference (WONS)*. 1–8.

[6] J. Chen and X. Ran. 2019***. Deep Learning With Edge Computing: A Review***. Proc. IEEE 107, 8 (2019), 1655–1674..

Acknowledgements

The research work is supported by the Hellenic Foundation for Research and Innovation (HFRI) under the 3rd Call for HFRI PhD Fellowships (Fellowship Number: 5631).

Table 1: Datasets Information.

Dataset	Size of Images	Train Samples	Test Samples	Classes
ImageNet	224x224	1, 281, 167	100, 000	1, 000
CIFAR-100	32x32	50, 000	10, 000	100
CIFAR-10	32x32	50, 000	10, 000	10
Intel Image Classification	150x150	14, 000	3, 000	6

Table 2: Summary of experiments.

Model	Accuracy (%)		Energy (nJ)		Percentage Increase in Accuracy, compared	Percentage of faster convergence, compared
Datasets Sets (pre-trained/tested)	FC (*Baseline-1*)	**F$_x$C1**	FC (*Baseline-1*)	**F$_x$C1**	to *Baseline-1*(%)	to *Baseline-1*(%)
Small CNN (550,000 #of params.) (CIFAR-100/CIFAR-10)	69, 5	76, 5	844, 9	31097, 5	10, 01	19, 98
EfficientNetB0 (4,049,571 #of params.) (ImageNet/CIFAR-100)	74, 5	88, 5	510, 0	64735, 2	18, 79	56, 74
EfficientNetB0 (ImageNet/Intel Classification)	86, 9	92, 2	137, 4	64350, 2	6, 09	39, 38
EfficientNetB2 (7,768,569 #of params.) (ImageNet/CIFAR-100)	78, 0	91, 1	560, 9	78273, 5	16, 79	60, 81