TinyTrain: Resource-Aware Task-Adaptive Sparse Training of DNNs at the Data-Scarce Edge

Young D. Kwon, Rui Li, Stylianos I. Venieris, Jagmohan Chauhan, Nicholas D. Lane, Cecilia Mascolo

Contact: {yd.kwon, rui.li}@samsung.com

UNIVERSITY OF CAMBRIDGE · University of Southampton · SAMSUNG Research

ICML International Conference On Machine Learning

TL;DR

A data-, memory-, and compute-efficient on-device training approach at the edge that dynamically adapts to target tasks on the fly.

Intro

📱 On-device training is essential for user personalisation and privacy.

🏠 Extremely resource-constrained consumer platforms are ubiquitous, but training DNNs on these platforms is so far impossible or takes impractically long or with substantial accuracy loss.

🎯 Existing efforts focus on addressing the first two challenges (compute & memory) while assuming abundant labelled data are available.

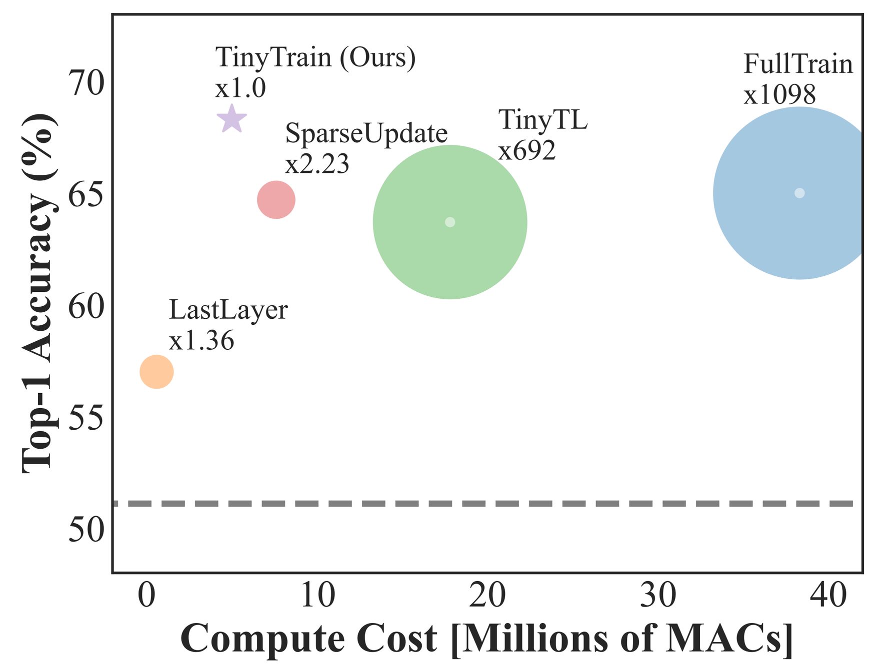

Figure 1: Cross-domain accuracy (y-axis) and compute cost in MAC count (x-axis) of *TinyTrain* and existing methods, targeting ProxylessNASNet on Meta-Dataset. The radius of the circles and the corresponding text denote the increase in the memory footprint of each baseline over *TinyTrain*. The dotted line represents the accuracy without on-device training.

Challenges in the targeted extreme-Edge AI training:

Compute-, **Memory-**, and **Data-Scarcity**

TinyTrain

We present TinyTrain, a novel framework that enables efficient training of DNNs on data-scarce, memory-severely-limited, compute-constrained edge platforms. This is enabled by:

I. A dynamic and **task-adaptive sparse-update** approach that fine-tunes only part of the model's parameters.

II. A **multi-objective parameter selection criterion** for layer/channel selection* that co-optimises accuracy, compute and memory footprint, specially designed for resource-constrained platforms.

Feature dim of each channel

N: #samples

$$P_i = \sum_o \Delta_o \text{ where } \Delta_o = \frac{1}{2N}\sum_n^N \Big(\sum_d^D a_{nd} g_{nd}\Big)^2$$

activations

gradient

Fisher potential of layer i [4]

Multi-objective Parameter Selection criterion

$$s_i = \frac{P_i}{\frac{\|W_i\|}{\max_{l \in \mathcal{L}}(\|W_l\|)} \times \frac{M_i}{\max_{l \in \mathcal{L}}(M_l)}}$$

number of multiply-accumulate (MAC) operations in layer i

number of parameters of layer i

normalised by max value across all layers L of the model

*This is carried out efficiently on-device with a single back-propagation per task, avoiding the burdensome search process through a few thousand tests of different configurations [1].

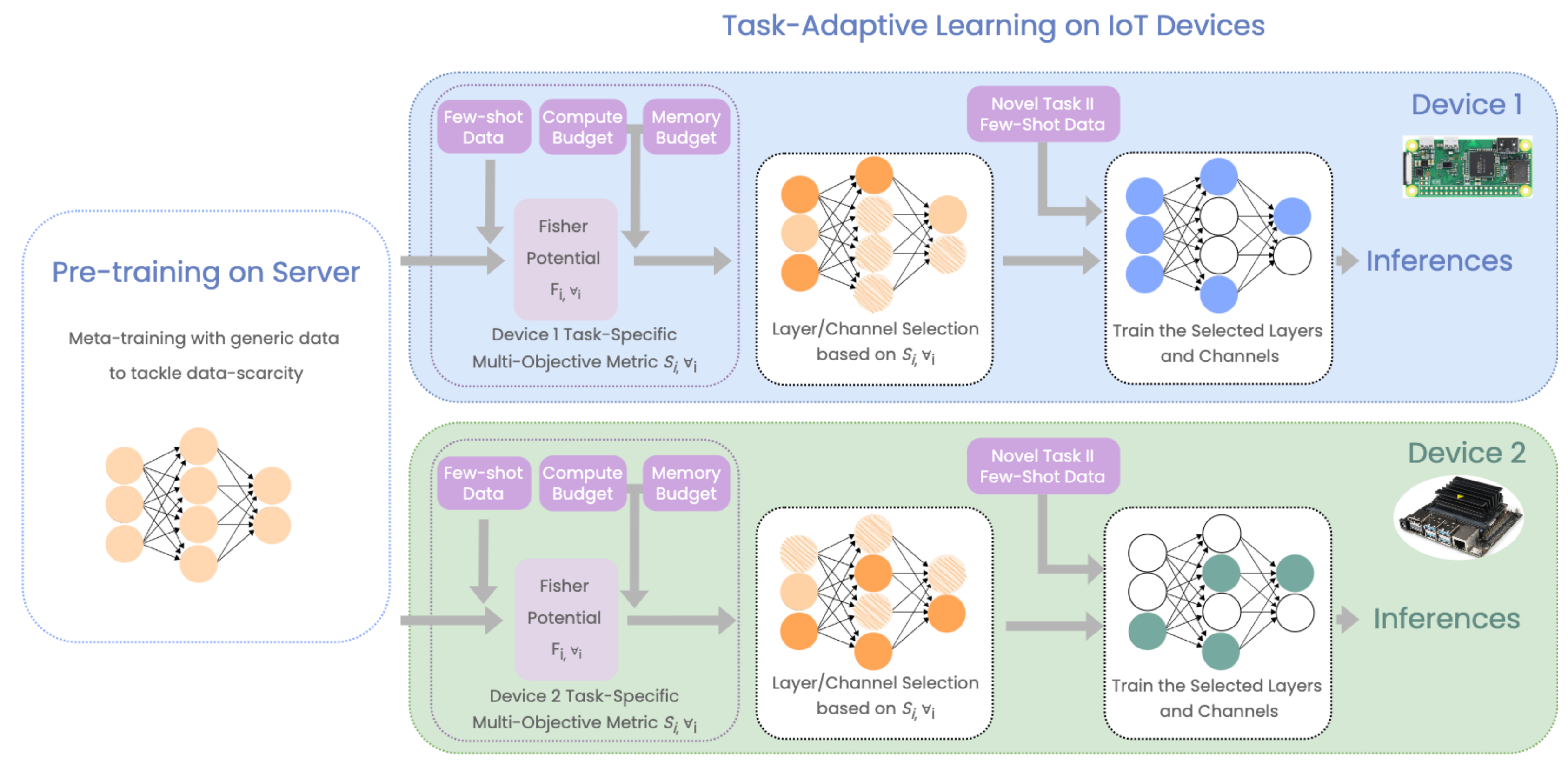

Figure 2: Overview of TinyTrain.

Evaluation Settings

- Three NN architectures: MCUNet, MobileNet, and ProxylessNASNet.
- Baselines: None, FullTrain, LastLayer (Training the last layer only), TinyTL [2], and SparseUpdate [1]
- Meta-datasets [3]: 9 cross-domain datasets e.g. Traffic Signs, Flowers, Aircrafts.
- Target platforms: Raspberry Pi Zero 2 and Jetson Nano

Evaluation Results

Table I. TinyTrain outperforms all the baselines w.r.t. top-1 accuracy with three architectures on nine cross-domain datasets

Model	Method	Traffic	Omniglot	Aircraft	Flower	CUB	DTD	QDraw	Fungi	COCO	Avg.
MCUNet	None	35.5	42.3	42.1	73.8	48.4	60.1	40.9	30.9	26.8	44.5
	FullTrain	**82.0**	72.7	75.3	90.7	66.4	74.6	64.0	40.4	36.0	66.9
	LastLayer	55.3	47.5	56.7	83.9	54.0	72.0	50.3	36.4	35.2	54.6
	TinyTL	78.9	73.6	74.4	88.6	60.9	73.3	67.2	41.1	36.9	66.1
	SparseUpdate	72.8	67.4	69.0	88.3	67.1	73.2	61.9	41.5	37.5	64.3
	TinyTrain (Ours)	79.3	**73.8**	**78.8**	**93.3**	**69.9**	**76.0**	**67.3**	**45.5**	**39.4**	**69.3**
Mobile NetV2	None	39.9	44.4	48.4	81.5	61.1	70.3	45.5	38.6	35.8	51.7
	FullTrain	75.5	69.1	68.9	84.4	61.8	71.3	60.6	37.7	35.1	62.7
	LastLayer	58.2	55.1	59.6	86.3	61.8	72.2	53.3	39.8	36.7	58.1
	TinyTL	71.3	69.0	68.1	85.9	57.2	70.9	**62.5**	38.2	36.3	62.1
	SparseUpdate	77.3	**69.1**	72.4	87.3	62.5	71.1	61.8	38.8	35.8	64.0
	TinyTrain (Ours)	**77.4**	68.1	**74.1**	**91.6**	**64.3**	**74.9**	60.6	**40.8**	**39.1**	**65.6**
Proxyless NASNet	None	42.6	50.5	41.4	80.5	53.2	69.1	47.3	36.4	38.6	51.1
	FullTrain	78.4	73.3	71.4	86.3	64.5	71.7	63.8	38.9	37.2	65.0
	LastLayer	57.1	58.8	52.7	85.5	56.1	72.9	53.0	38.6	38.7	57.0
	TinyTL	72.5	**73.6**	70.3	86.2	57.4	71.0	65.8	38.6	37.6	63.7
	SparseUpdate	76.0	72.4	71.2	87.8	62.1	71.7	64.1	39.6	37.1	64.7
	TinyTrain (Ours)	**79.0**	71.9	**76.7**	**92.7**	**67.4**	**76.0**	**65.9**	**43.4**	**41.6**	**68.3**

Table II. Comparison of the memory footprint and computation cost for a backward pass.

Model	Method	Memory	Ratio	Compute	Ratio
MCUNet	FullTrain	906 MB	1,013×	44.9M	6.89×
	LastLayer	2.03 MB	2.27×	1.57M	0.23×
	TinyTL	542 MB	606×	26.4M	4.05×
	SparseUpdate	1.43 MB	**1.59**×	11.9M	**1.82**×
	TinyTrain (Ours)	**0.89 MB**	1×	**6.51M**	1×
Mobile NetV2	FullTrain	1,049 MB	987×	34.9M	7.12×
	LastLayer	1.64 MB	1.54×	0.80M	0.16×
	TinyTL	587 MB	552×	16.4M	3.35×
	SparseUpdate	2.08 MB	**1.96**×	8.10M	**1.65**×
	TinyTrain (Ours)	**1.06 MB**	1×	**4.90M**	1×
Proxyless NASNet	FullTrain	857 MB	**1,098**×	38.4M	7.68×
	LastLayer	1.06 MB	1.36×	0.59M	0.12×
	TinyTL	541 MB	**692**×	17.8M	3.57×
	SparseUpdate	1.74 MB	**2.23**×	7.60M	**1.52**×
	TinyTrain (Ours)	**0.78 MB**	1×	**5.00M**	1×

TinyTrain achieves:

- ☑ 2.6-7.7% higher accuracy than **SOTA**
- ☑ 3.6-5.0% higher accuracy compared to **FullTrain**

while requiring:

- ☑ 987x smaller memory & 7.12x smaller compute compared to **FullTrain**
- ☑ 1.96x smaller memory & 1.65x smaller compute compared to **SOTA**

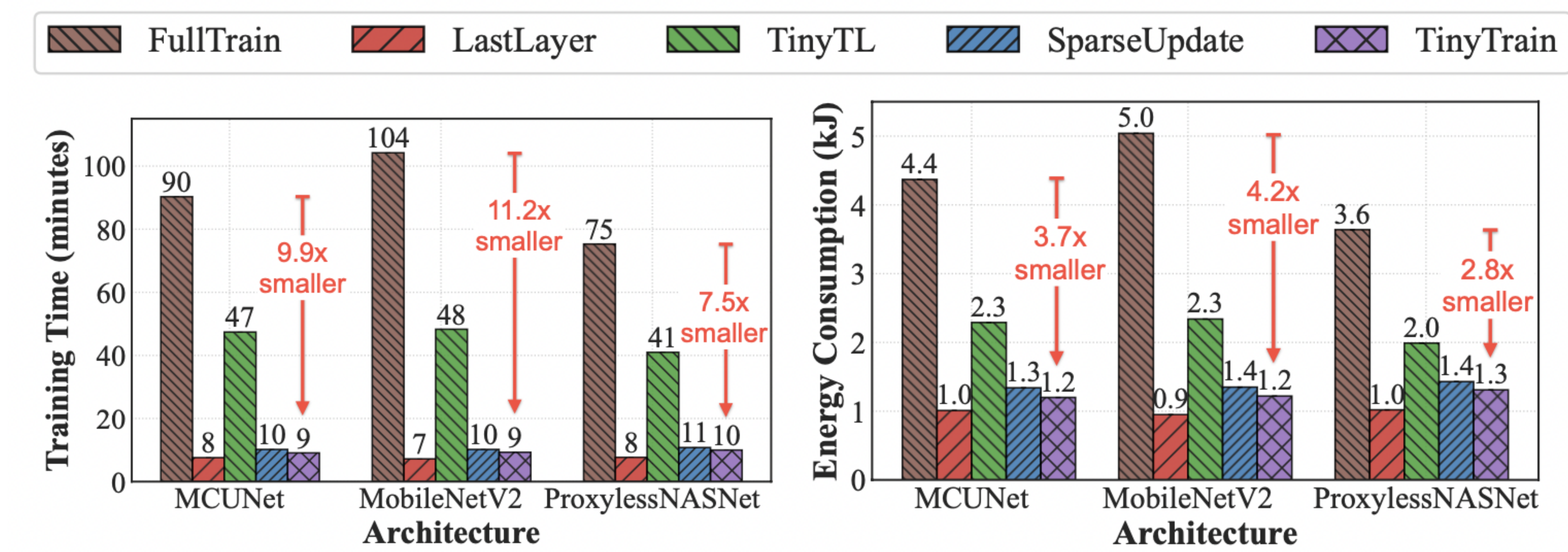

Fig 3. End-to-End Latency (left) and Energy Consumption (right) of the on-device training methods on three architectures.

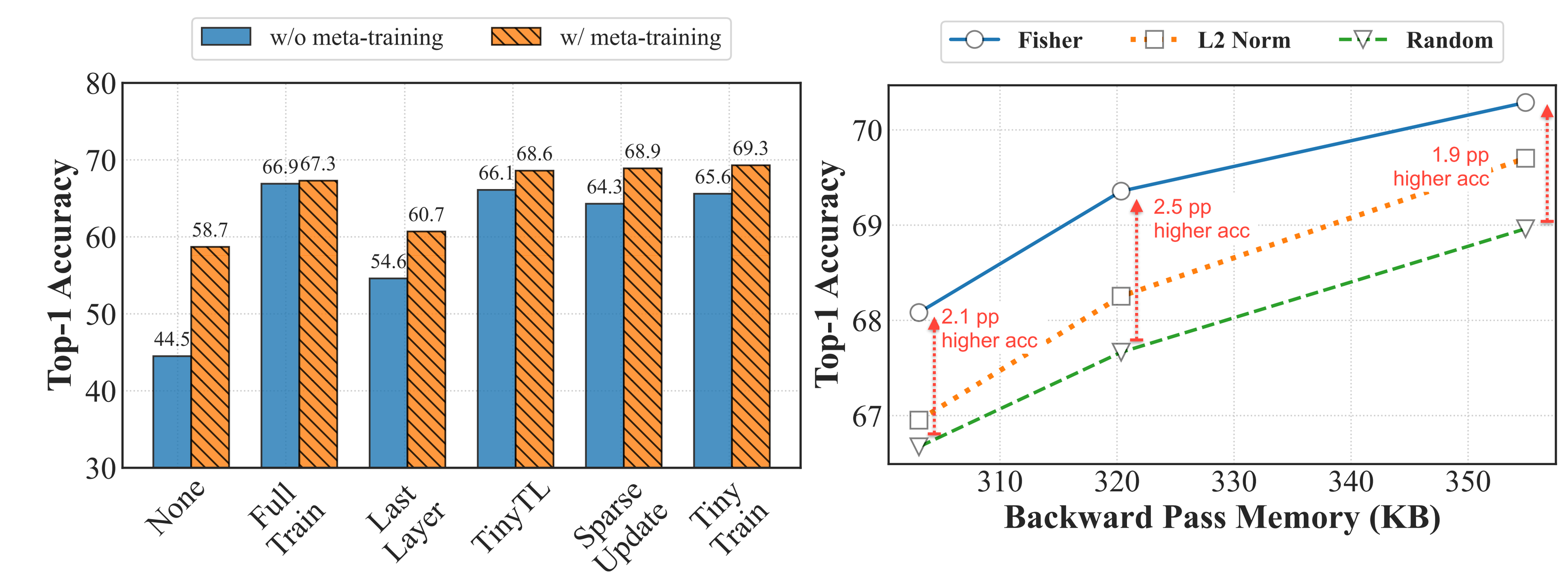

Fig 4. Ablation Study: Effect of Meta-training (left) and Dynamic Channel Selection (right).

- ☑ **TinyTrain** achieves **7.5-11.2x lower latency & 2.8-4.2x lower energy consumption** compared to **FullTrain.**
- ☑ Our Ablation study suggests i) **Offline meta-training** increases TinyTrain's **accuracy by 5.6 pp** on average; ii) **Dynamic channel selection** increases **accuracy by 0.8-1.7 pp and 1.9-2.5 pp** on average compared to static channel selection based on L2-Norm and Random, respectively.

Conclusions

💻 We have developed the first realistic on-device training framework, TinyTrain, solving practical challenges in terms of data, memory, and compute constraints for edge devices.

🏌 TinyTrain meta-learns in a few-shot fashion during the offline learning stage and dynamically selects important layers and channels to update during deployment.

🏋 Targeting broadly used real-world edge devices, TinyTrain achieves 9.5× faster and 3.5× more energy-efficient training over status-quo approaches, and 2.23× smaller memory footprint than SOTA methods, while remaining within the 1 MB memory envelope of MCU-grade platforms.

References

[1] 'SparseUpdate' Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., Gan, C., and Han, S. On-Device Training Under 256KB Memory. In *NeurIPS* 2022.
[2] 'TinyTL' Cai, H., Gan, C., Zhu, L., and Han, S. TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning. In *NeurIPS* 2020.
[3] 'MetaDataset' Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.-A., and Larochelle, H. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. In *ICLR*, 2020.
[4] Turner, J., Crowley, E. J., O'Boyle, M., Storkey, A., and Gray, G. BlockSwap: Fisher-guided Block Substitution for Network Compression on a Budget. In International Conference on Learning Representations (ICLR), 2020.