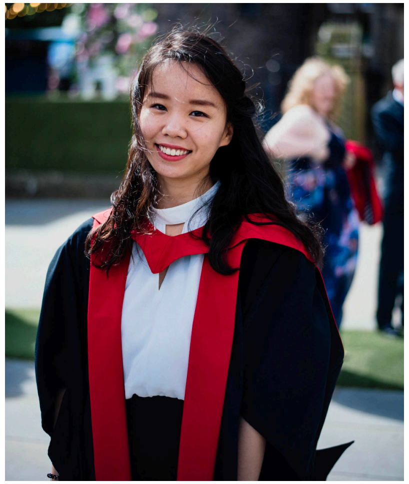
TinyTrain: Resource-Aware Task-Adaptive Sparse Training of DNNs at the Data-Scarce Edge


Young Kwon[1,2] Rui Li[2] Stylianos Venieris[2] Jagmohan Chauhan[3] Nicholas Lane[1] Cecilia Mascolo[1]

[1]*University of Cambridge* [2]*Samsung AI Center - Cambridge* [3]*University of Southampton*

UNIVERSITY OF CAMBRIDGE

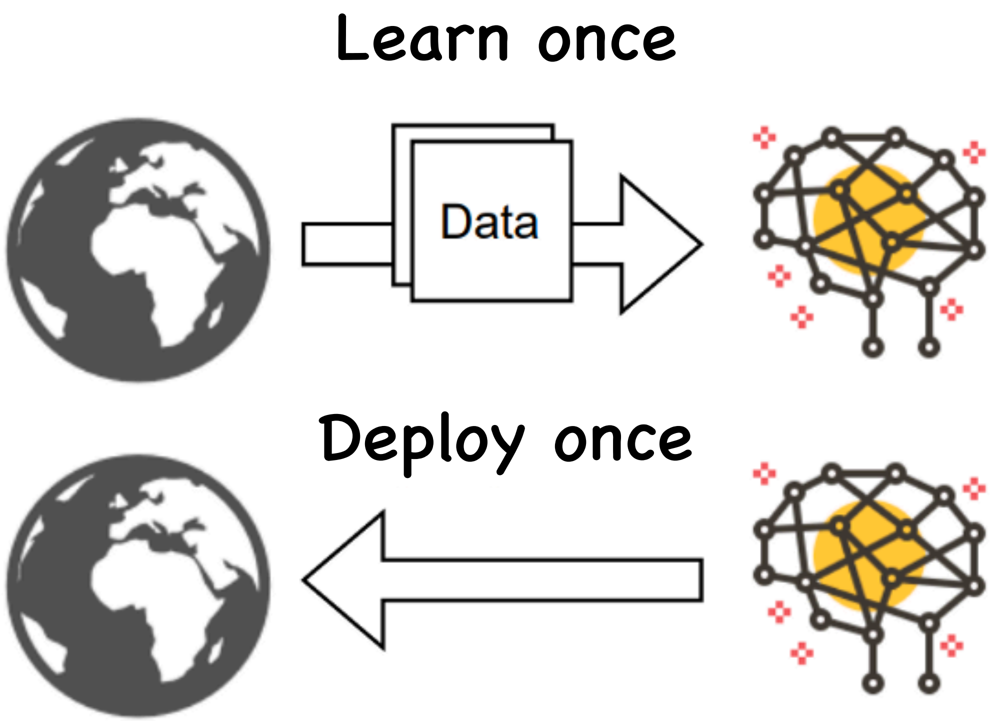
SAMSUNG Research

University of Southampton

AI/Deep Learning on the edge devices

Realistic Scenarios

- On-device training is essential but challenging



Realistic Scenarios

- On-device training is essential but challenging

Unique Challenges

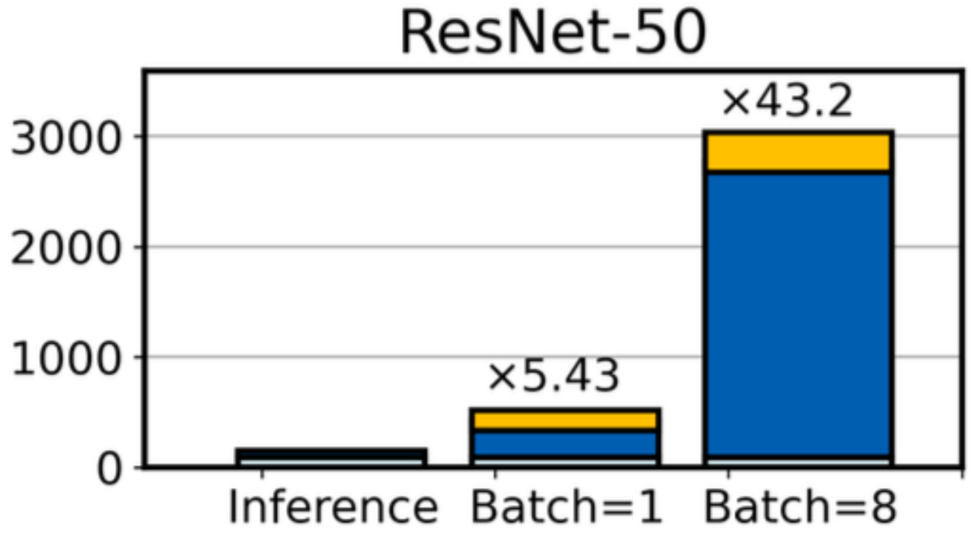
1. **Difficult for Labelling**
   (Lack of labelled data for personalisation)

Unique Challenges

1. Difficult for Labelling
   (Lack of labelled data for personalisation)

2. Training is Expensive in terms of Memory and Computation

   - MCUNet needs almost 1 GB Memory



Peak Memory of Training

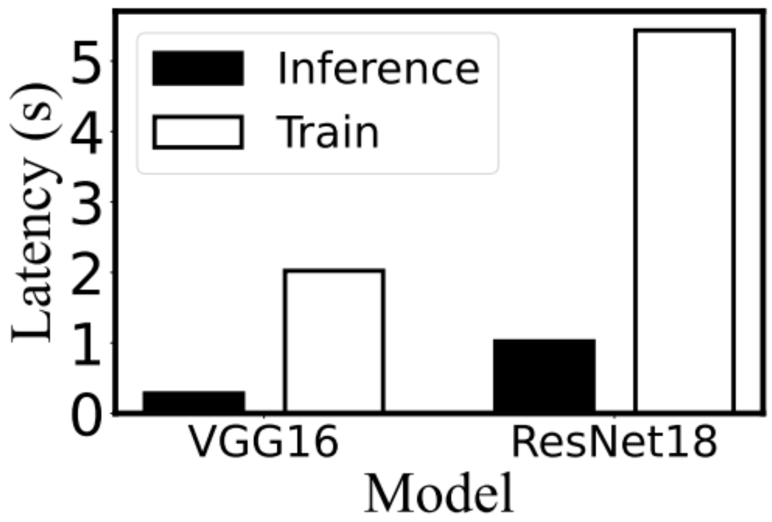
Unique Challenges

1. Difficult for Labelling
   (Lack of labelled data for personalisation)

2. Training is Expensive in terms of Memory and Computation

   - MCUNet needs almost 1 GB Memory

   - Training needs ~3x FLOPs than inference



Latency for Training

Prior Works & Limitations

Fine-tuning Head

- Update enabled with low memory and low compute
- Suffer from drastic accuracy loss

MbV2

Prior Works & Limitations

Fine-tuning Head

- Update enabled with low memory and low compute
- Suffer from drastic accuracy loss

TinyTL

- Update enabled with mid memory and mid compute
- Suffer from moderate accuracy loss

MbV2

Prior Works & Limitations

Fine-tuning Head

- Update enabled with low memory and low compute
- Suffer from drastic accuracy loss

TinyTL

- Update enabled with mid memory and mid compute
- Suffer from moderate accuracy loss

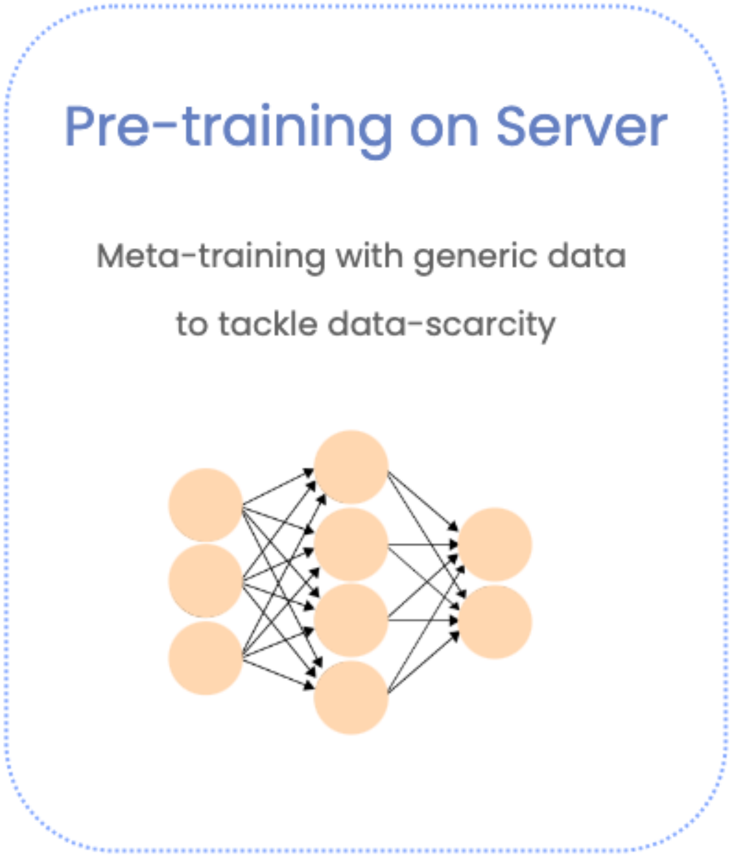
SparseUpdate

- Update enabled with low memory and mid compute
- Burdensome offline search process
- Static channel selection leads to suboptimal results

MbV2

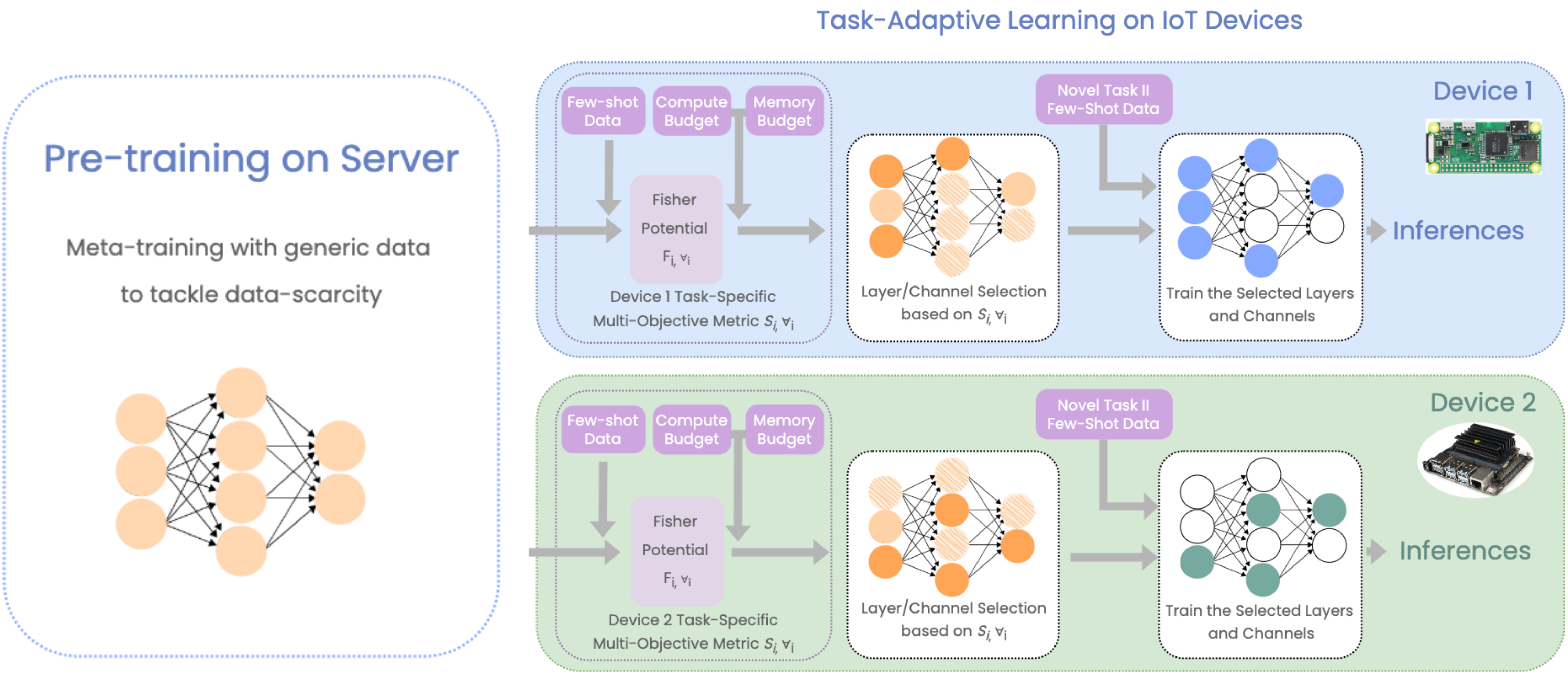
TinyTrain

- Data-, memory-, and compute-efficient adaptive IoT system



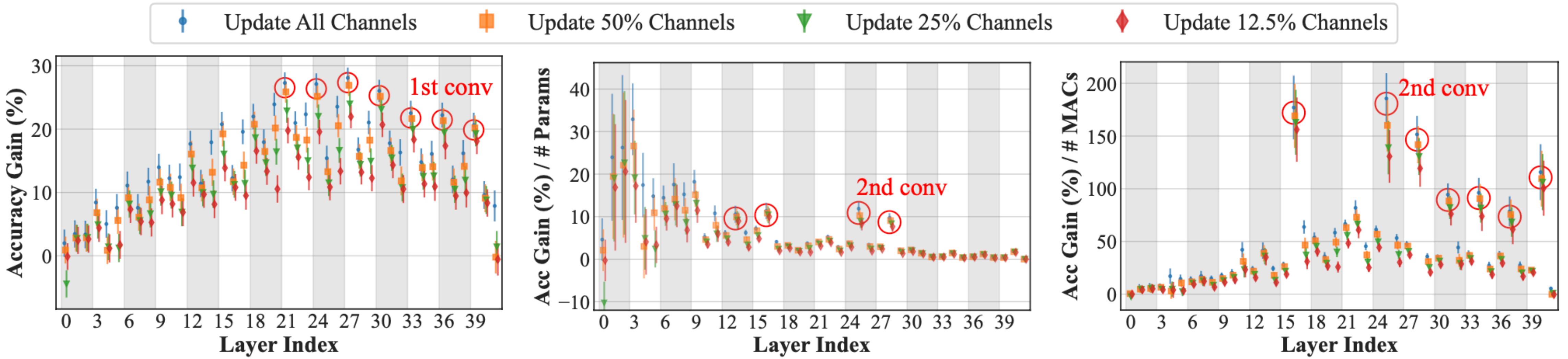
TinyTrain

- **Data-, memory-, and compute-efficient adaptive IoT system**



Task-Adaptive Sparse Update

- **Accuracy, Memory, Computation Trade-off**



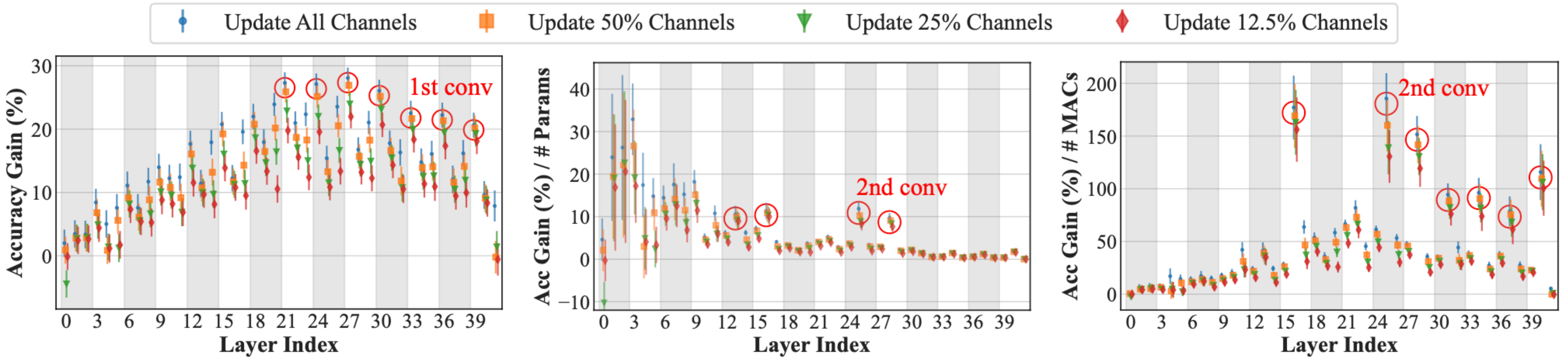
(a) Accuracy Gain (b) Accuracy Gain / Params (c) Accuracy Gain / MACs

Architecture: MCUNet

Dataset: Traffic Sign

Task-Adaptive Sparse Update

- **Accuracy, Memory, Computation Trade-off**



(a) Accuracy Gain (b) Accuracy Gain / Params (c) Accuracy Gain / MACs

Fisher potential of layer i

Multi-objective 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 parameters of layer i

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

Architecture: MCUNet

Dataset: Traffic Sign

Experimental Setup

- **Datasets**

(1) Traffic Sign
(2) Omniglot
(3) Aircraft
(4) Flower
(5) CUB
(6) DTD
(7) Quick Draw
(8) Fungi
(9) MSCOCO

- **Architectures**

(1) MCUNet
(2) MobileNetV2
(3) ProxylessNASNet

- **Baselines**

(1) None
(2) FullTrain
(3) LastLayer
(4) TinyTL
(5) SparseUpdate

Results

- Accuracy

Model	Method	Traffic	Omniglot	Aircraft	Flower	CUB	DTD	QDraw	Fungi	COCO	Avg.
MobileNetV2	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**

TinyTrain achieves 3.6-5.0% higher accuracy compared to FullTrain

Results

- **Accuracy**

Model	Method	Traffic	Omniglot	Aircraft	Flower	CUB	DTD	QDraw	Fungi	COCO	Avg.
MobileNetV2	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**

TinyTrain achieves 3.6-5.0% higher accuracy compared to **FullTrain**

TinyTrain achieves 2.6-7.7% higher accuracy than **SOTA**

Results

- **Memory Footprint & Compute Cost**

Model	Method	Memory	Ratio	Compute	Ratio
MobileNetV2	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×

TinyTrain achieves 987x lower memory & 7.12x lower compute compared to **FullTrain**

Results

- **Memory Footprint & Compute Cost**

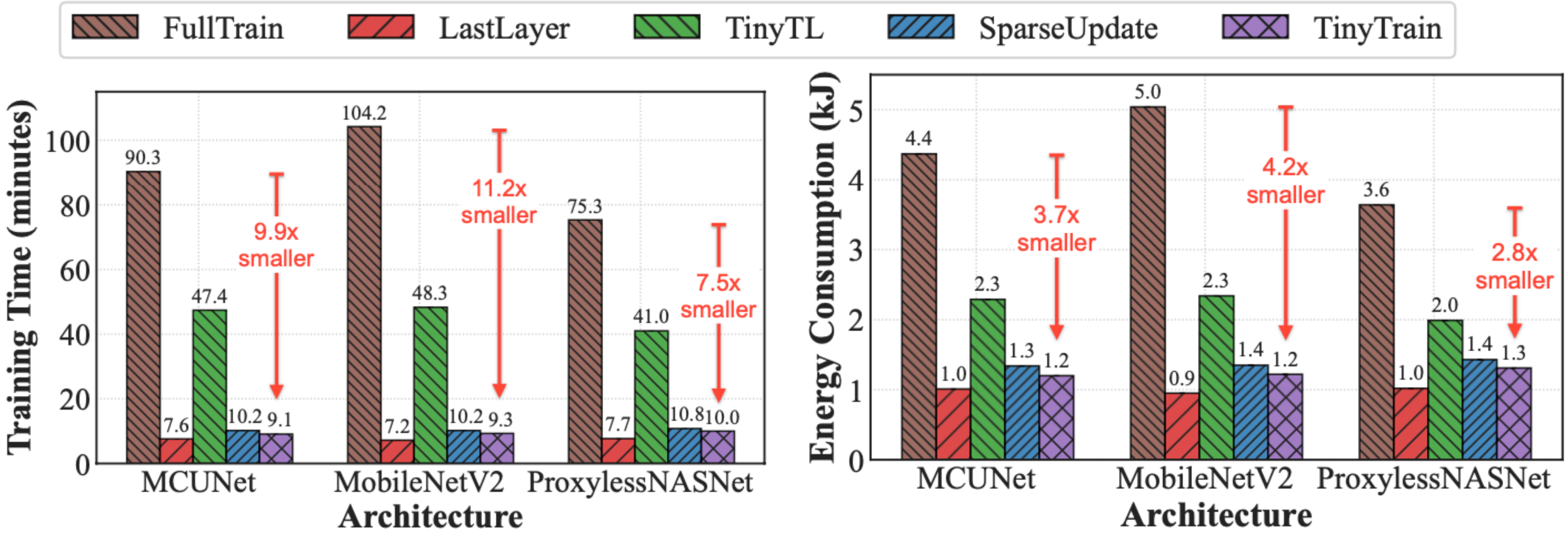
Model	Method	Memory	Ratio	Compute	Ratio
MobileNetV2	FullTrain	1,049 MB	987×	34.9M	7.12×
	LastLayer	1.64 MB	1.54×	0.80M	0.16×
	TinyTL	587 MB	552×	16.4M	3.35×
	SparseUpdate	2.08 MB	1.96×	8.10M	1.65×
	TinyTrain (Ours)	**1.06 MB**	1×	**4.90M**	1×

TinyTrain achieves **987x lower memory & 7.12x lower compute** compared to **FullTrain**

TinyTrain achieves **1.96x lower memory & 1.65x lower compute** compared to **SOTA**

Results

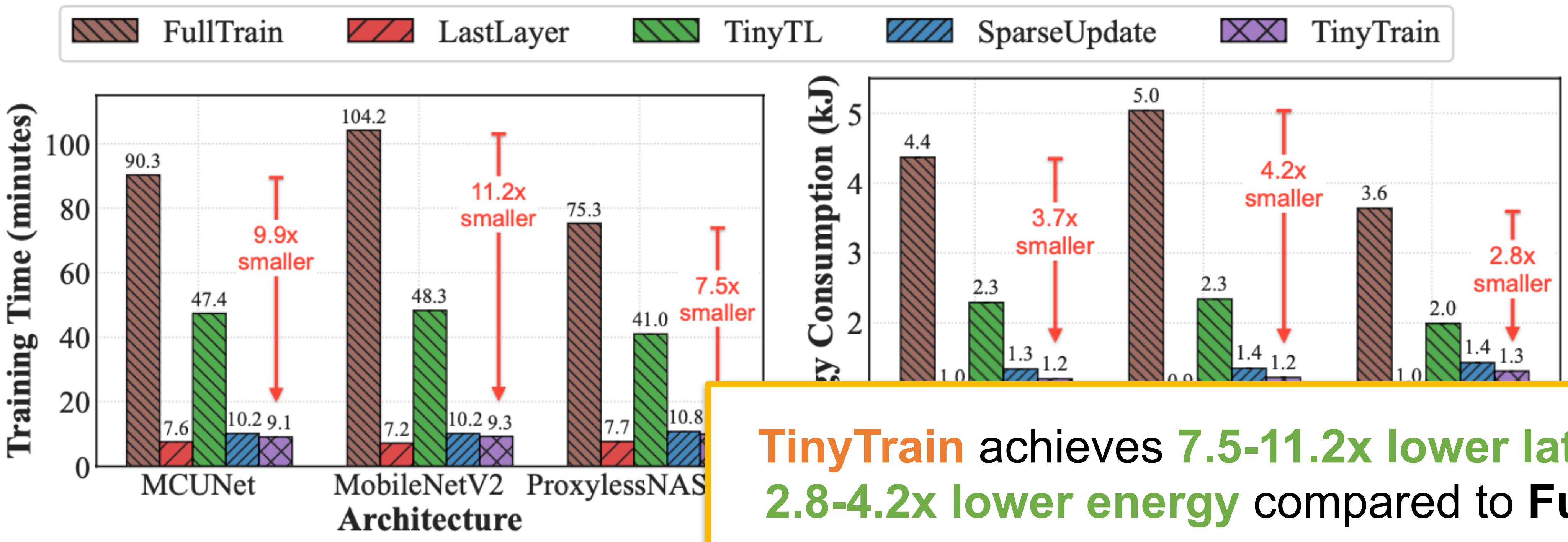
- **End-to-end training time & energy consumption**



(b) Training Time (c) Energy Consumption

Results

- End-to-end training time & energy consumption

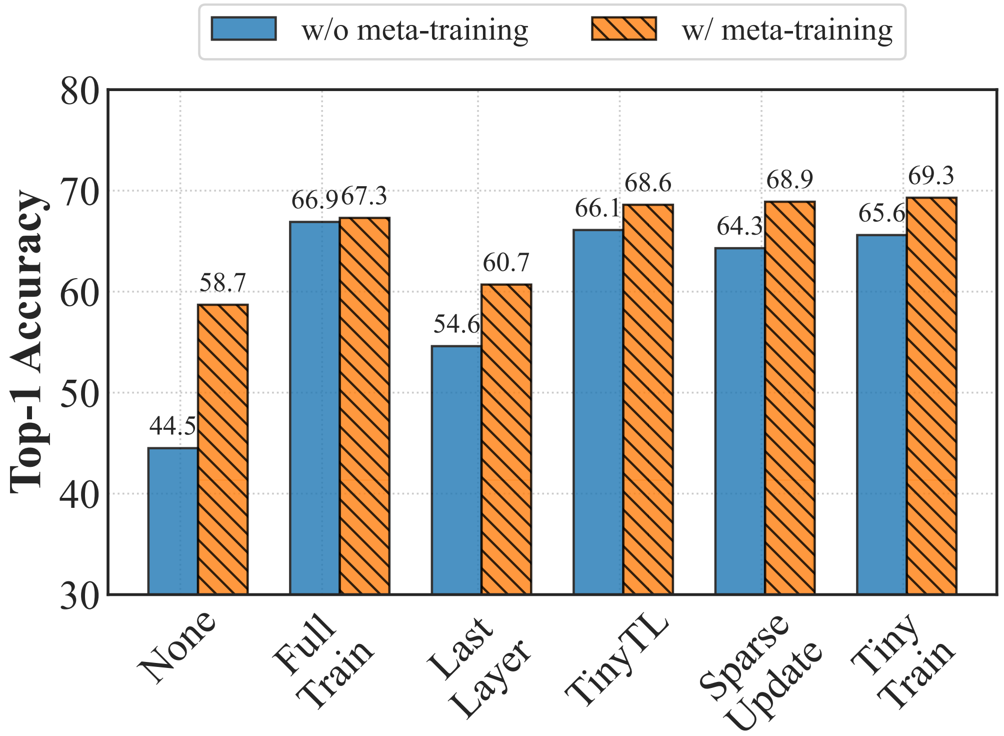


TinyTrain achieves 7.5-11.2x lower latency & 2.8-4.2x lower energy compared to **FullTrain**

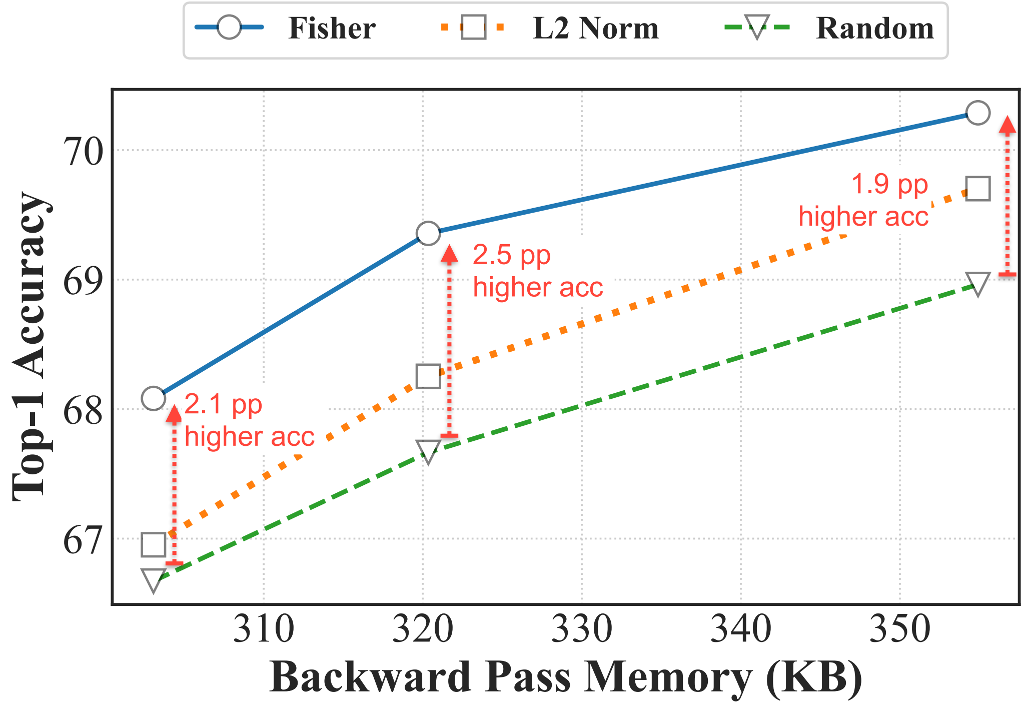
(b) Training Time (c) Energy Consumption

Ablation Study

- **Effect of FSL pre-training and dynamic channel selection**



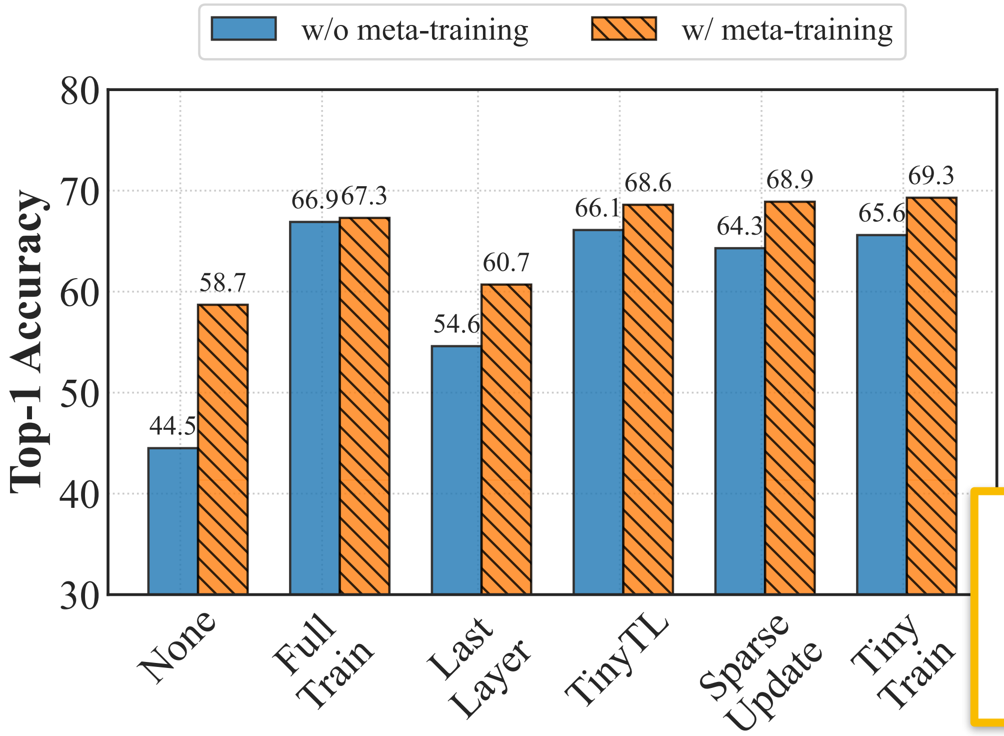
(a) FSL Pre-training



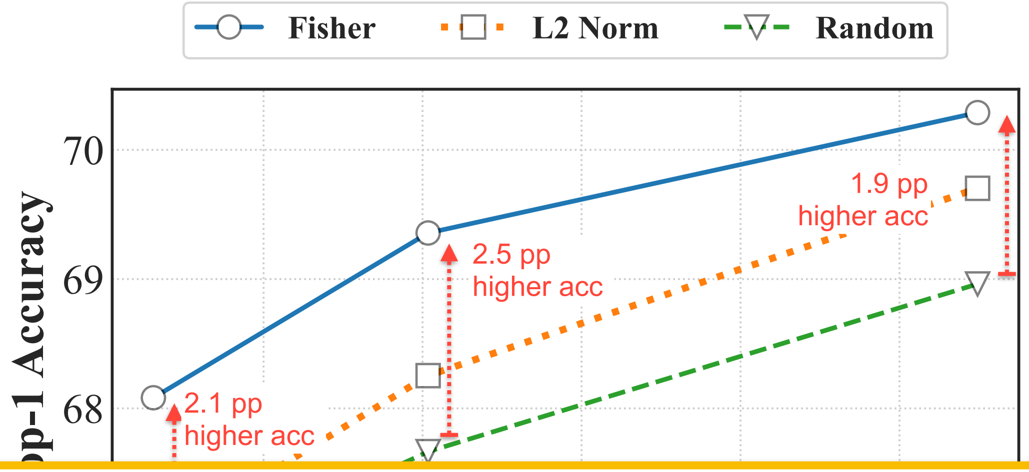
(b) Dynamic Channel Selection

Ablation Study

- **Effect of FSL pre-training and dynamic channel selection**



(a) FSL Pre-training



(b) Dynamic Channel Selection

Our proposed **FSL Pre-training** and **Dynamic Channel Selection** demonstrate **its effectiveness** in **ablation study**

Summary & Take-away Messages

S1. TinyTrain enables Adaptive systems via data-, memory-, and compute-efficient on-device training

Summary & Take-away Messages

S1. TinyTrain enables Adaptive systems via data-, memory-, and compute-efficient on-device training

T1. FSL-pretraining is effective in ensuring high accuracy

T2. Task-adaptive sparse update is effective in ensuring dynamic layer/channel update during deployment

Thank You!

Any questions?

You can find me at:

ydk21@cam.ac.uk

theyoungkwon.github.io/

UNIVERSITY OF CAMBRIDGE

SAMSUNG Research

University of Southampton