Towards Ubiquitous Learning: A First Measurement of On-Device Training Performance

Dongqi Cai[1], Qipeng Wang[2], Yuanqiang Liu[2], Yunxin Liu[3, 4], Shangguang Wang[1, 5] and Mengwei Xu[1]

[1] Beijing University of Posts and Telecommunications

[2] Peking University

[3] Institute for AI Industry Research (AIR)

[4] Tsinghua University

[5] Shenzhen Research Institute

北京郵電大學

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

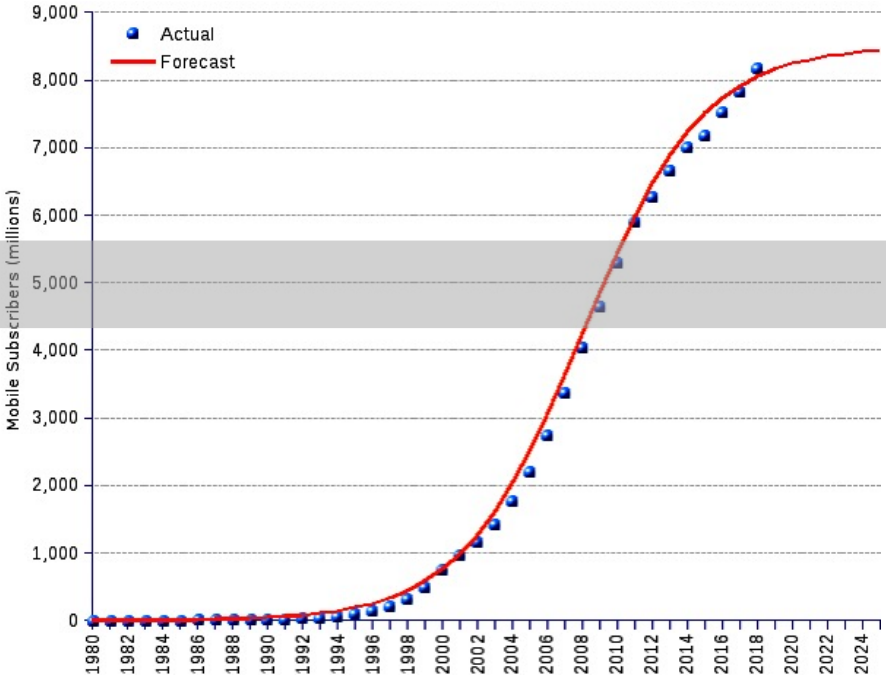
北京大學

PEKING UNIVERSITY

清華大學

Tsinghua University

Background



Galaxy S (Released in 2010)
RAM: 512MB
Storage: 16GB

Galaxy Note20 (Released in 2020)
RAM: 12GB
Storage: 512GB

(Source:https://stats.areppim.com/stats/stats_mobilex2019.htm)

Deep Learning on edge devices

usage	detailed usage	as core feature
image: 149	photo beauty: 97	94 (96.9%)
	face detection: 52	44 (84.6%)
	augmented reality: 19	5 (26.3%)
	face identification: 8	7 (87.5%)
	image classification: 11	6 (54.5%)
	object recognition: 10	9 (90%)
	text recognition:11	4 (36.3%)
text:26	word&emoji prediction: 15	15 (100%)
	auto-correct: 10	10 (100%)
	translation: 7	3 (42.8%)
	text classification: 4	2 (50%)
	smart reply: 2	0 (0%)
audio: 24	speech recognition: 18	7 (38.9%)
	sound recognition: 8	8 (100%)
other: 19	recommendation: 11	2 (18.1%)
	movement tracking: 9	4 (44.4%)
	simulation: 4	4 (100%)
	abnormal detection: 4	4 (100%)
	video segment: 2	1 (50%)
	action detection: 2	0 (0%)
total: 211		171 (81.0%)

Table 1: The DL usage in different apps. Note: as one app may have multiple DL uses, the sum of detailed usage (column 2) might exceed the corresponding coarse usage (column 1).

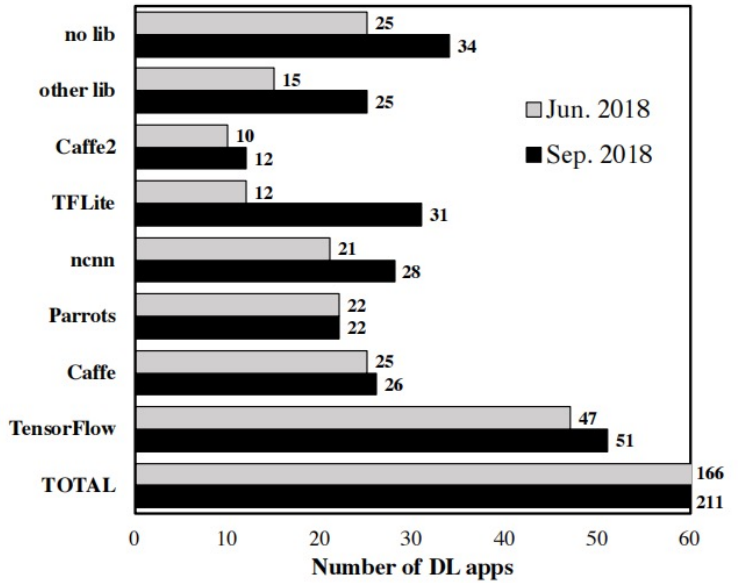
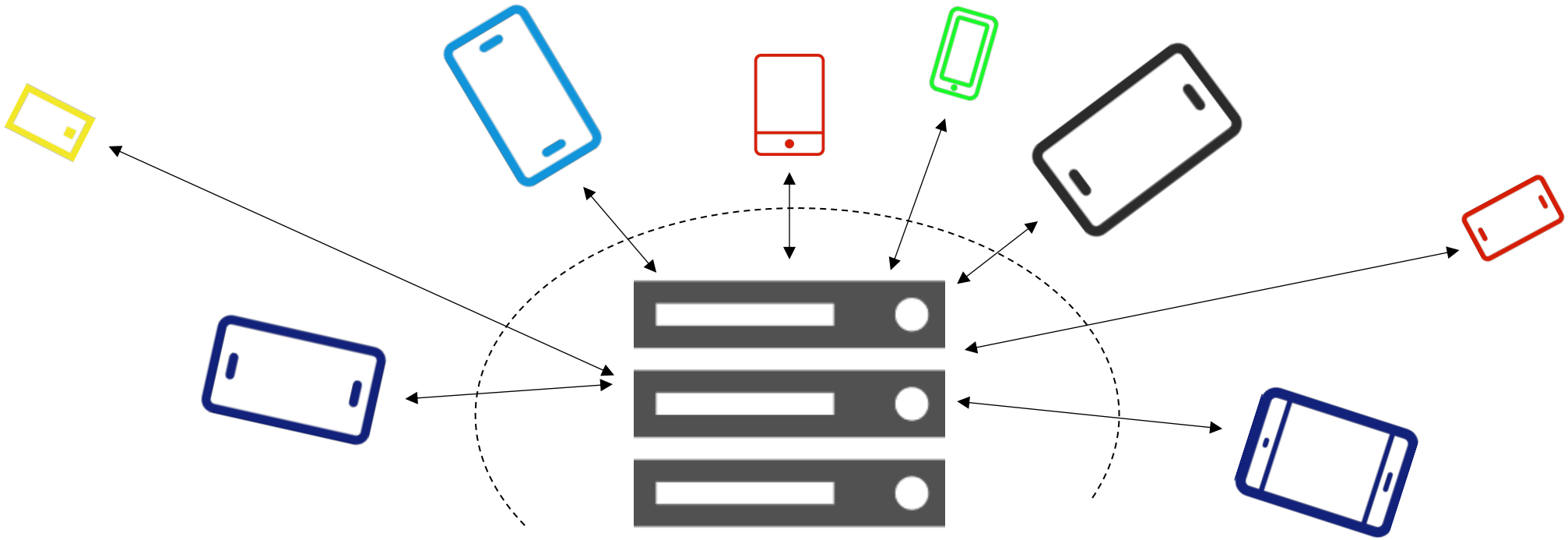


Figure 4: Numbers of DL apps using various mobile DL frameworks. "other lib": the DL apps developed on the frameworks in Table 2 but not itemized here, e.g. *mace, SNPE, and xnn*. "no lib": apps with DL functions but using no DL frameworks from Table 2. Note that the number in "TOTAL" is lower than the sum of others since some DL apps have integrated multiple frameworks.

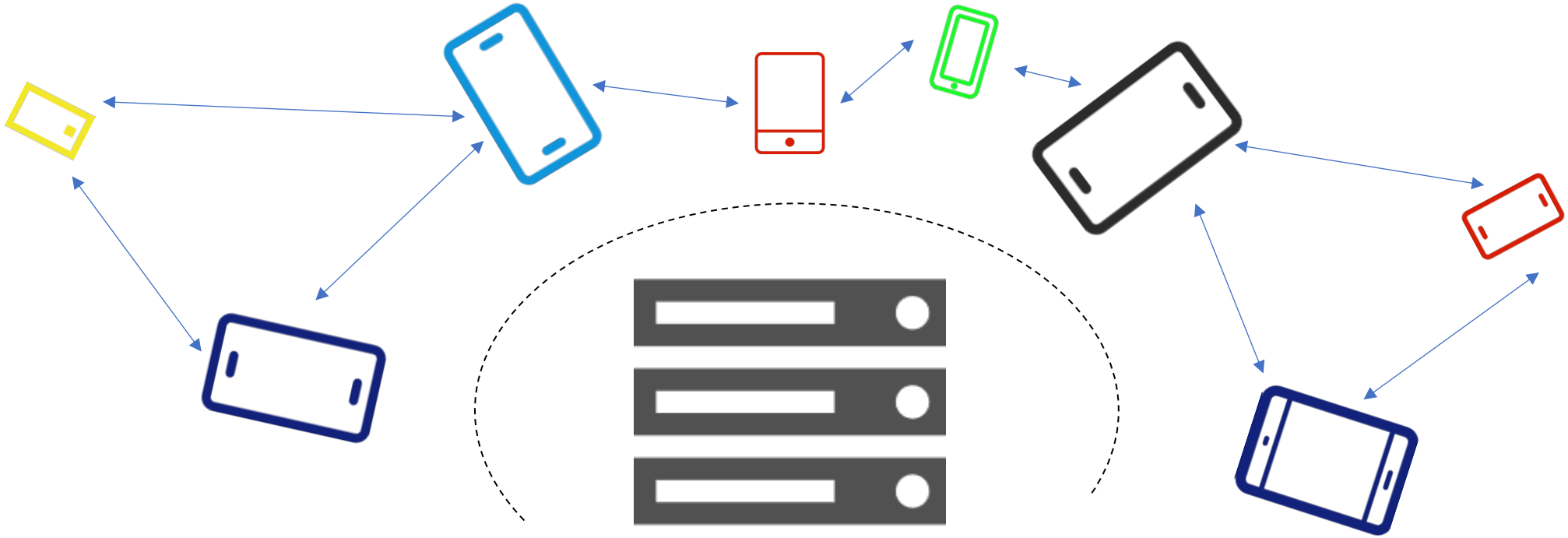
Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, and Xuanzhe Liu. A first look at deep learning apps on smartphones. In *The World Wide Web Conference*, pages 2125–2136, 2019.

Open Issue

However, the training stage of deep learning is still commonly placed on data centers.



Ubiquitous Learning



Methods: Federated Learning, Split Learning, Local Transfer Learning, etc.

- Whether edge devices can really afford training modern NN models?
- And if so, do current libraries efficiently support that?

Mobile Neural Network (MNN)

- Lightweight
- Versatility
- High performance
- Easy to use

Testing platform	Training library	Training time (ms)		
		BS = 1	BS=2	BS=4
Samsung Note 10	MNN	516	812	1365
	DL4J	3,032	6,129	OOM
RPI 3B+	MNN	6698	10,651	OOM
	TensorFlow	10,468	14,157	27,574
	PyTorch	48,274	79,097	OOM

https://github.com/alibaba/MNN

Experiment Setups

- Devices

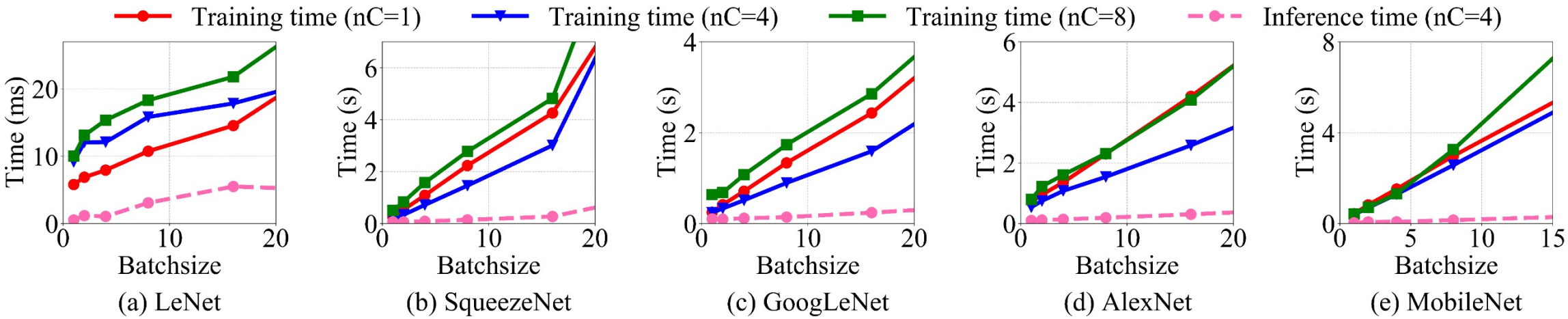
Device	Specifications	Yr.
Redmi Note 9 Pro	Snapdragon 720G, 6GB RAM	2020
Xiaomi MI 9	Snapdragon 855, 6GB RAM	2019
Huawei Mate 30	Kirin 990, 8GB RAM	2019
Meizu 16T	Snapdragon 855, 6GB RAM	2019
Samsung S8 Plus	Snapdragon 835, 6GB RAM	2017
Huawei Honor 8	Kirin 950, 3GB RAM	2016

- Models

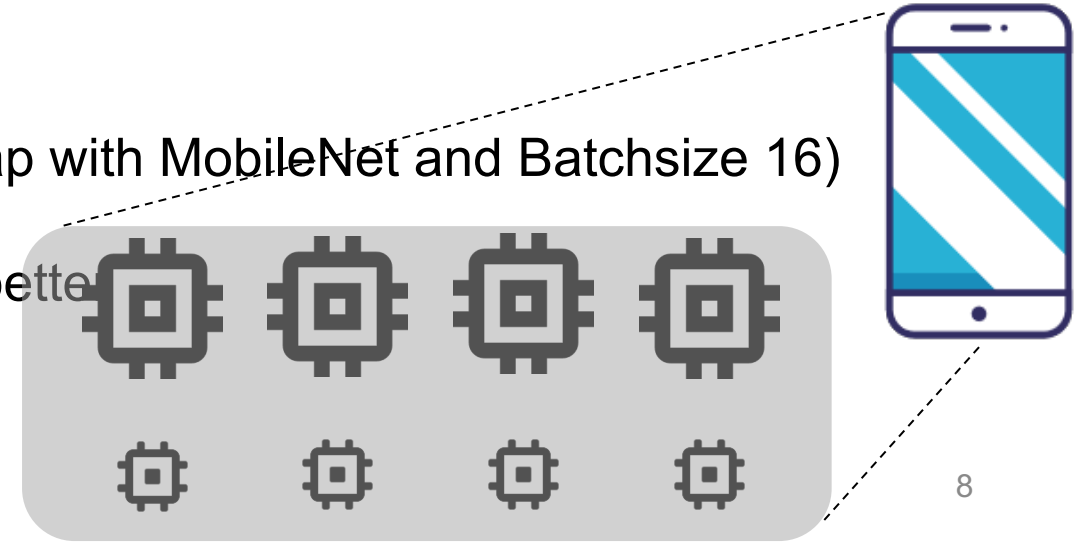
Model	Convs	Parameter
LeNet	2	3.2K
AlexNet	5	61M
MoileNetv2	53	3.4M
SqueezeNet	18	411.2K
GoogLenet	22	6.8M

https://github.com/UbiquitousLearning/Benchmark-On-Device-Training

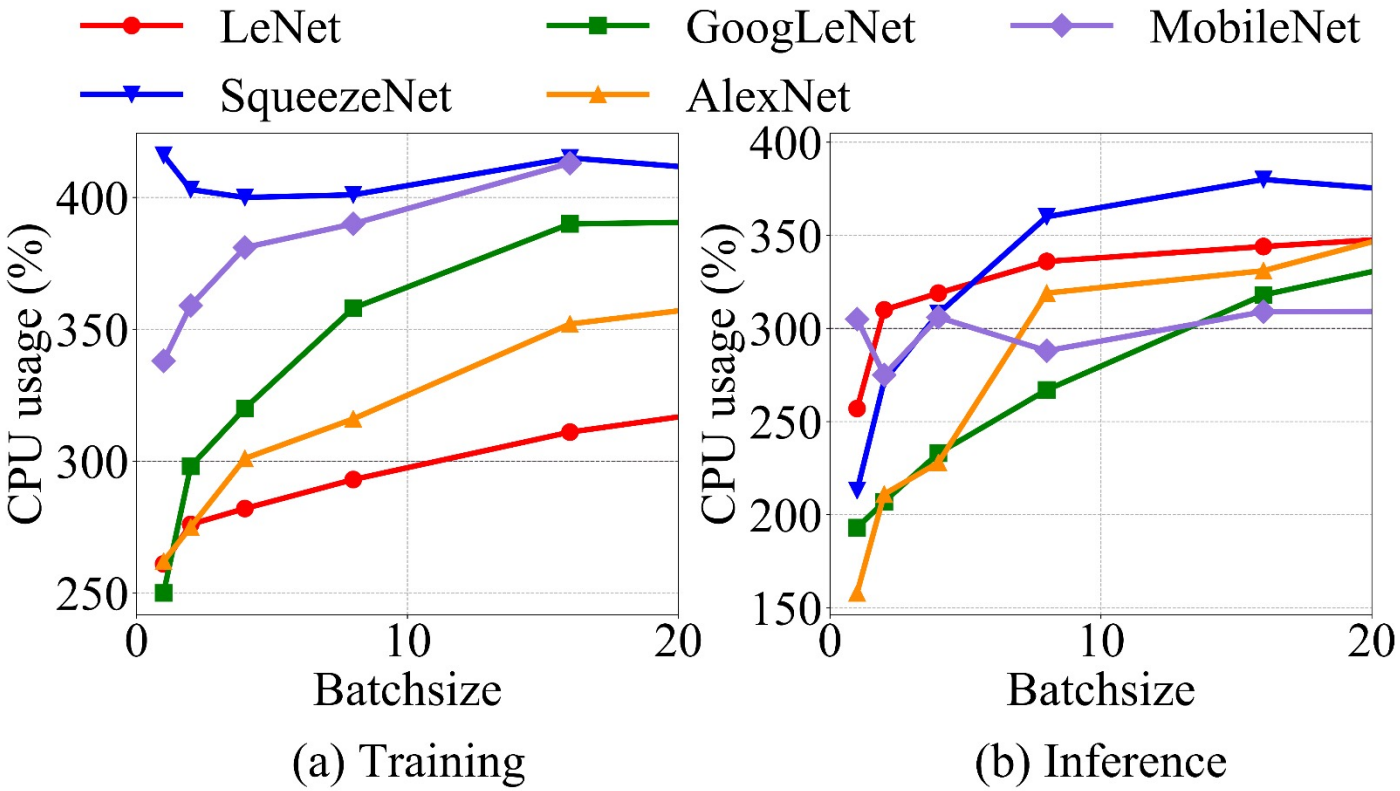
Overall Latency



- Training time **>>** Inference time (Up to **17.8×** gap with MobileNet and Batchsize 16)
- Setting the number of CPU core as **4** performs bette

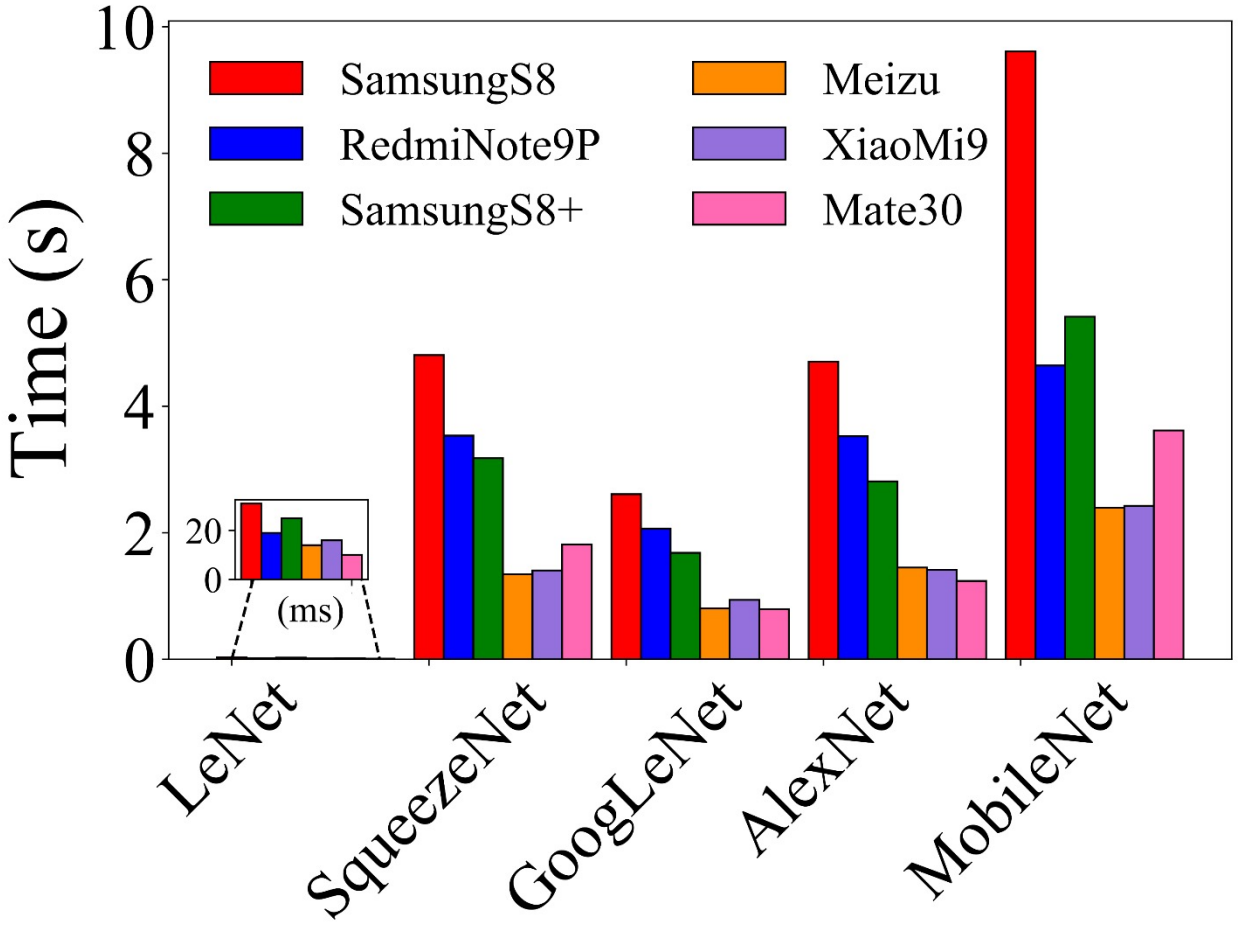


CPU Usage



The CPU usage during training is already close to **maximal** with relatively **small batchsize**

Cross-device Comparison



- **Performance disparity**

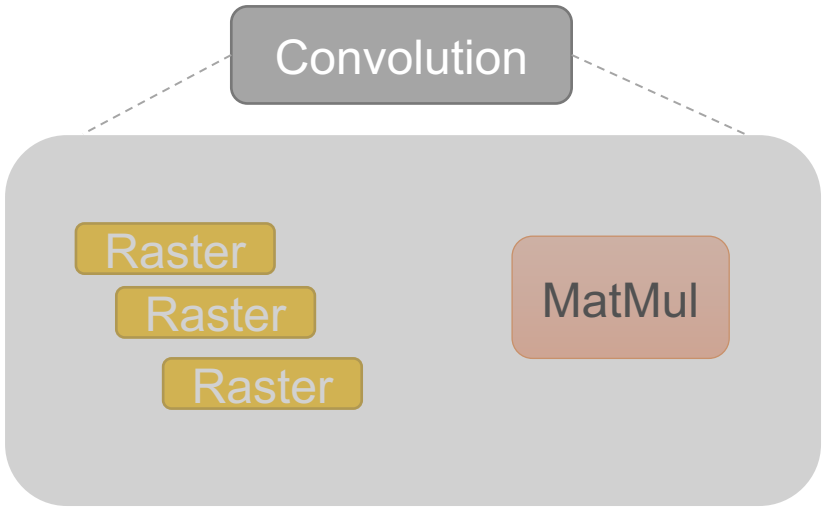
It is pervasive across devices and can be up to **4.0×** (MobilenetV2 model among Meizu 16T and Samsung S8+)

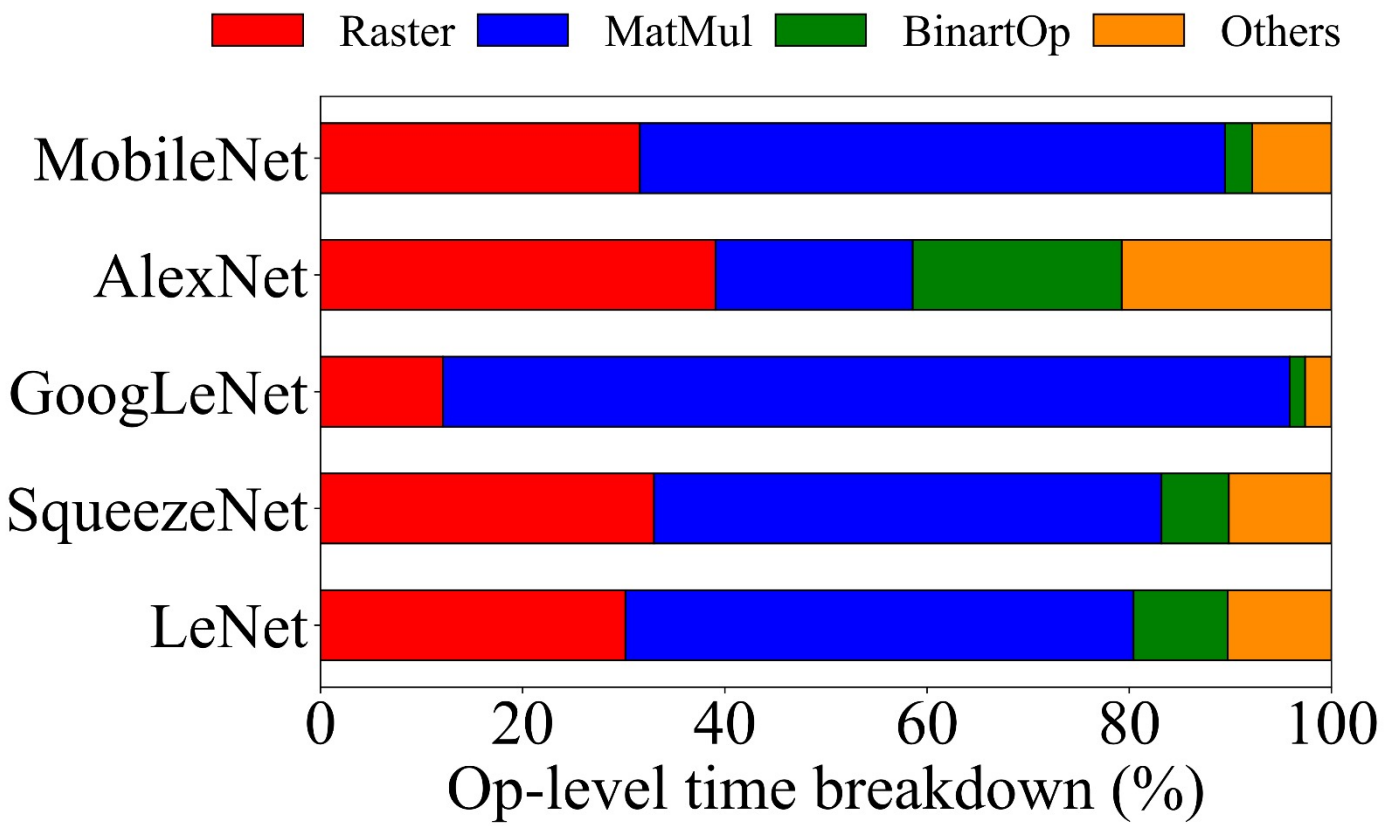
- **Model affinity**

SqueezeNet trains fastest on Meizu 16T while AlexNet trains fastest on Huawei Mate 30

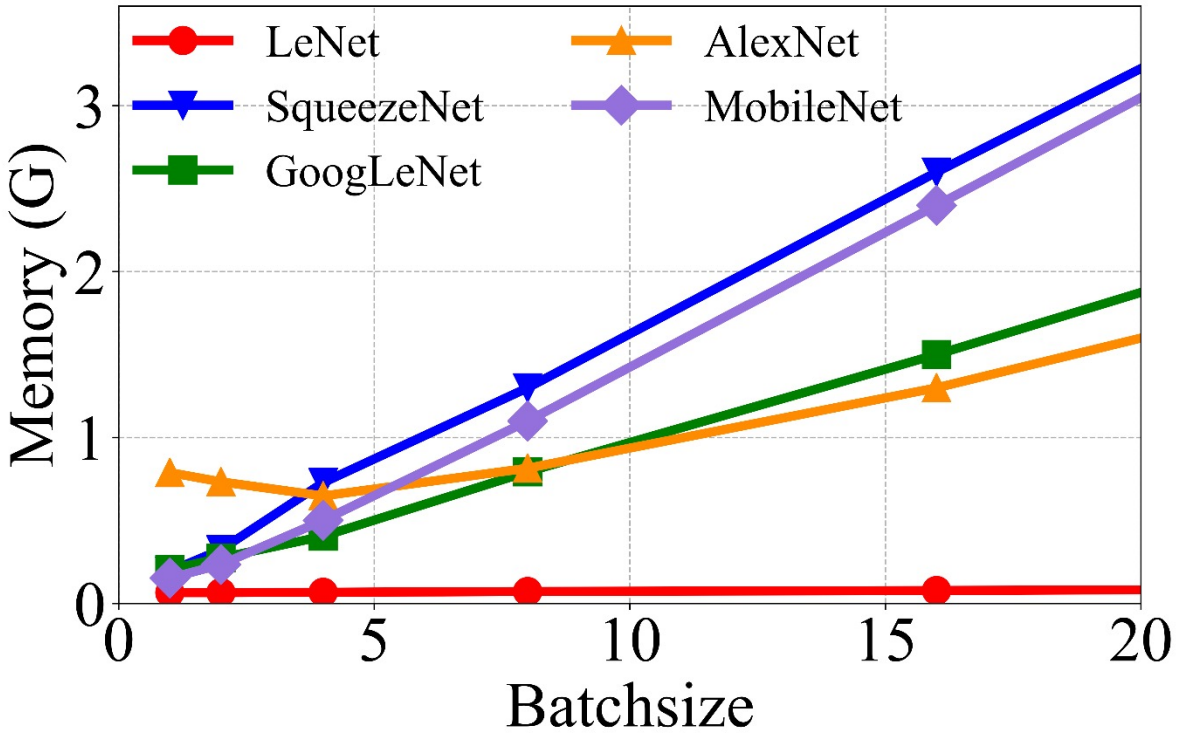
Latency Breakdown

Raster operator is a unified implementation of all traditional operators that are related to tensor shape transformations, including *slice, concat, reshape, broadcast, etc*.





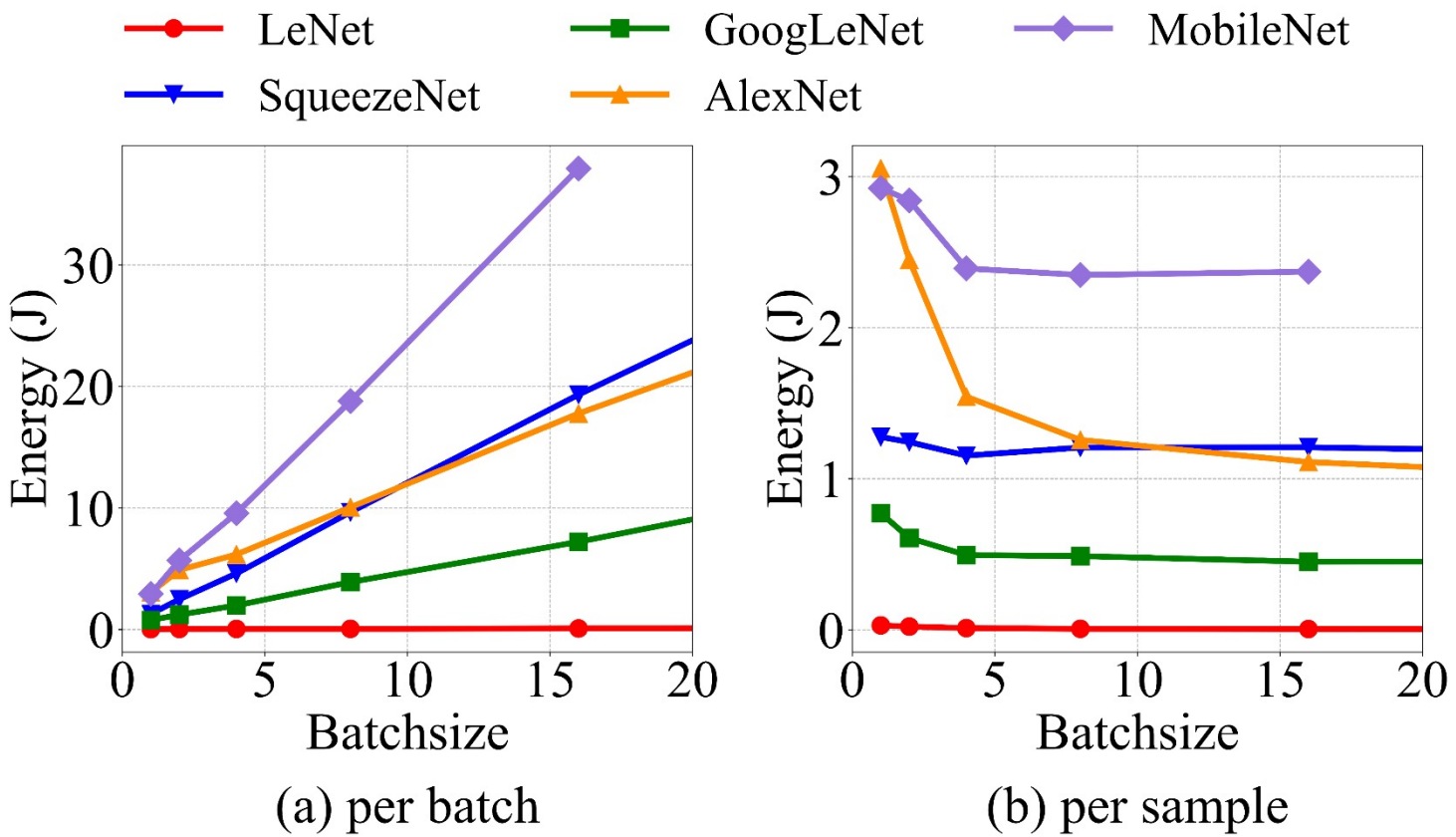
Memory footprint



To keep the memory usage under 1GB, the upper bound of batchsize may be limited to 4!

Energy consumption

The training time doesn't linearly scale with the batchsize since larger batchsize benefits the **intra-operator parallelism.**



Using larger batchsize can be more energy-efficient.

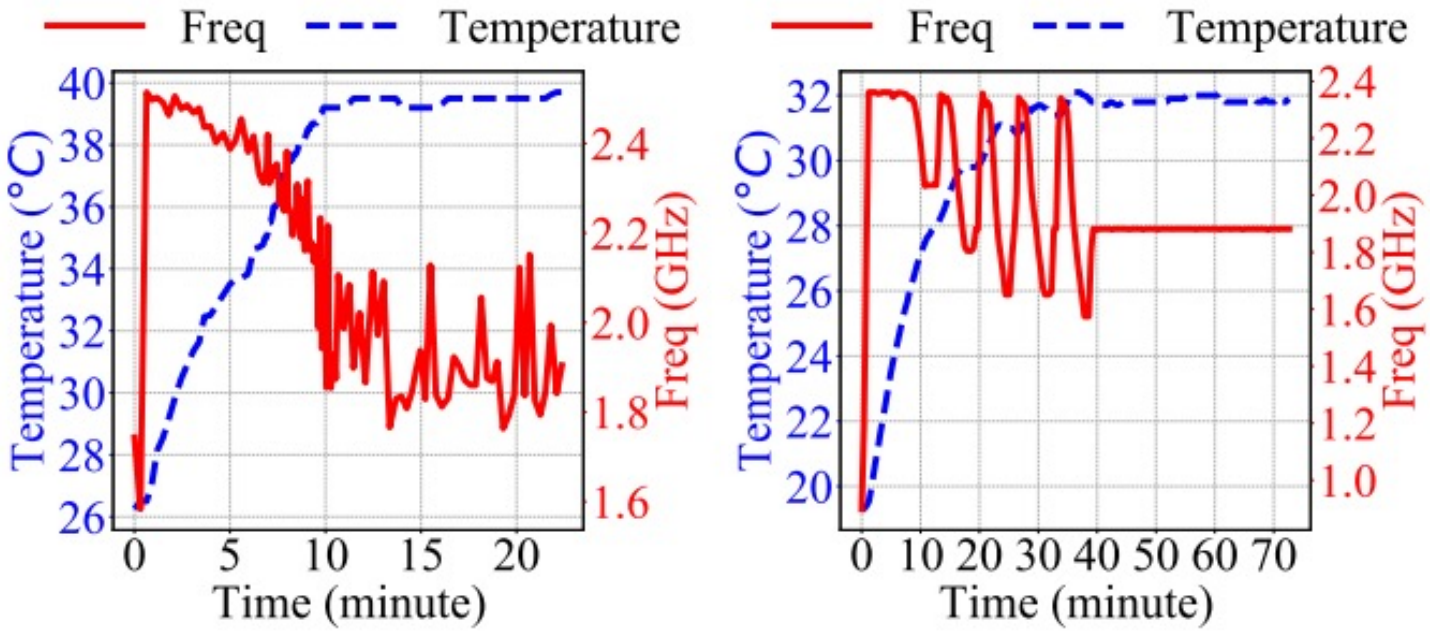
Impacts of CPU parameters

CPU Conf.		Time (s)			Energy (J)		
		H	M	L	H	M	L
Big	1×	4.2	5.4	10.8	10.6	8.0	6.9
	2×	2.6	3.2	6.4	8.9	7.7	7.0
	4×	2.0	3.3	8.4	7.1	8.7	8.2
Small	1×	25.0	33.9	57.8	10.4	7.2	3.1
	2×	13.3	18.0	31.8	10.1	8.4	4.8
	4×	8.0	11.0	52.3	11.4	9.6	8.2
Hybrid	8×	3.8	6.5	50.4	13.4	13.9	14.4

- Using 4 big cores with the highest frequency achieves **the best performance**.
- Using only one small core with the lowest CPU frequency leads to **the lowest usage of energy consumption**. Its energy consumption is only 43.7% of the optimal case of training time, despite it runs 28.9× slower.

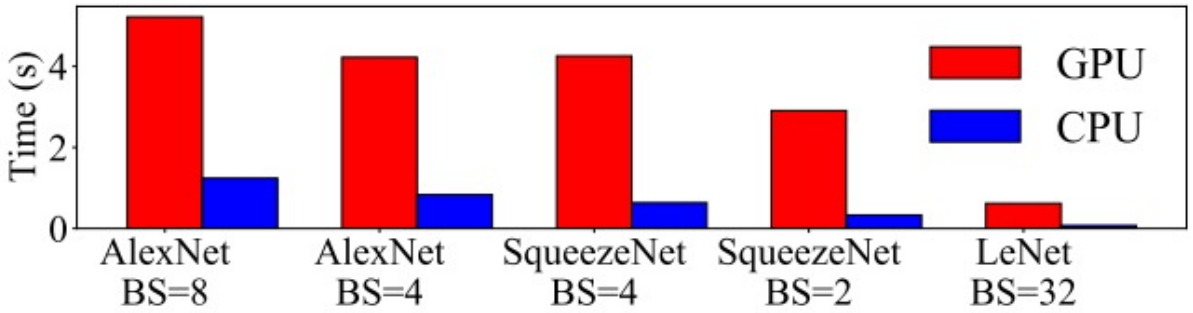
Thermal dynamics

Such **heterogeneous** thermal dynamics may complicate the ubiquitous learning process, e.g., the device selection in federated learning.

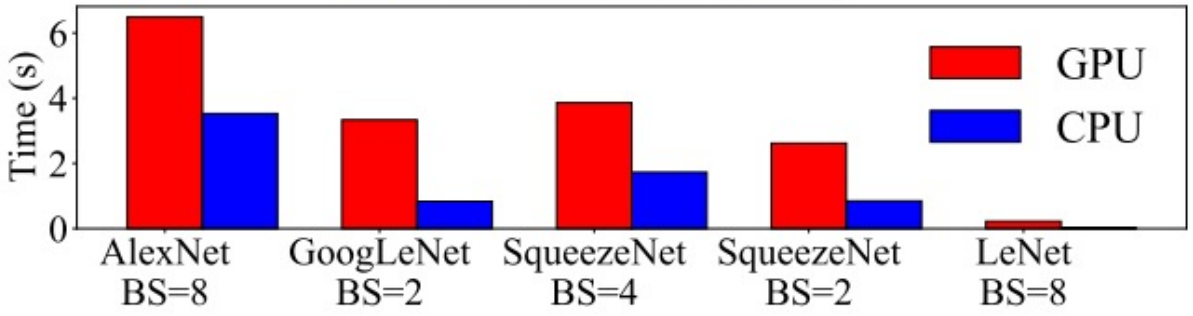


(a) Meizu 16T (b) Samsung S8+

Mobile CPU vs. Mobile GPU



(a) Huawei Mate 30 (Mali)



(b) Redmi Note 9 Pro (Adreno)

Both GPU Mali and Adreno run much slower than CPU (1.8×–12.0×)

Conclusion

- First measurement of on-device-training performance
- At the dawn of "learning everywhere and anything"

Future Work

- Generating efficient operators
- Memory optimizations
- Tuning system parameters

Thank you

https://github.com/UbiquitousLearning/Benchmark-On-Device-Training