北京郵電大學
Beijing University of Posts and Telecommunications

The Way towards Ubiquitous Learning: a System Perspective

Mengwei Xu（徐梦炜）
CS Dept of BUPT

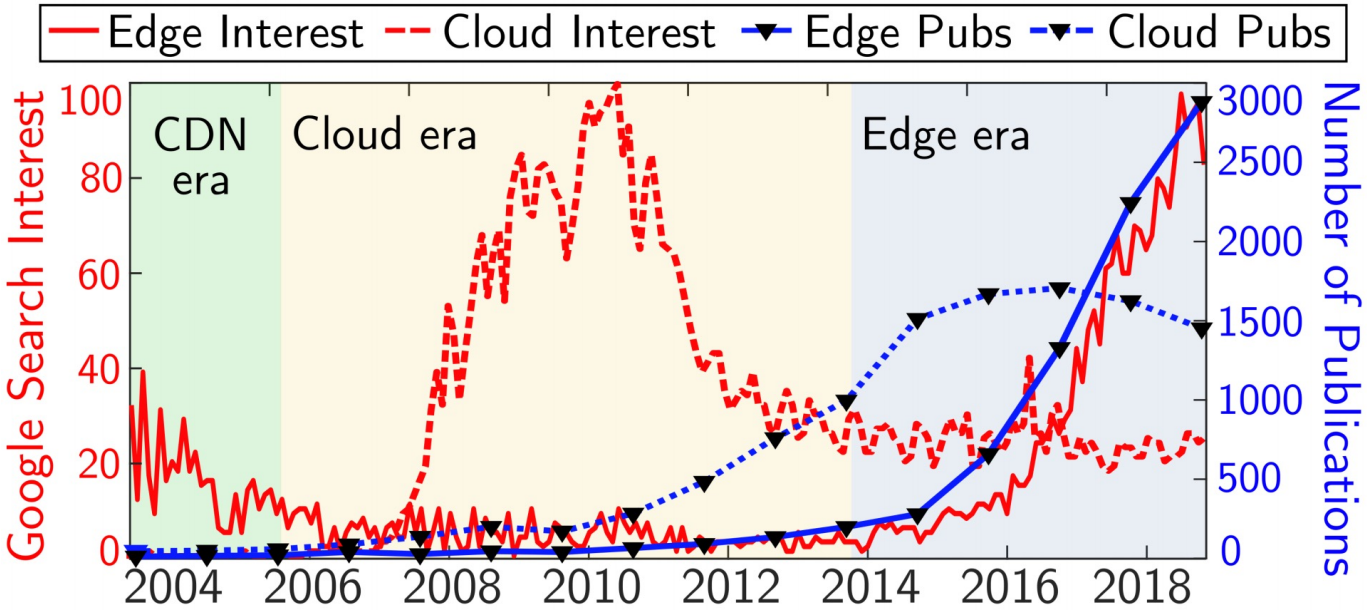
Outline

- Ubiquitous Learning (UL): What and Why?
  - A system software perspective
  - The status quo

- A measurement study of UL
- Memory optimizations of UL
- Other on-going research of UL

Some trends

- Edge devices (smartphones, IoTs, etc) are becoming important computing platforms, not just user equipment [1].



Edge is the new field for emerging techniques like AI[2]

- ❑ Preserving privacy
- ❑ Low delay
- ❑ Personalization
- ❑ etc..

[1] Mengwei Xu, et al. “A case for camera-as-a-service”, IEEE Pervasive Computing, 2021.
[2] N. Mohan, et al. "Pruning Edge Research with Latency Shears." HotNets, 2020.

My definition of Ubiquitous Learning

- Ubiquitous ~~computing~~ learning is the growing trend of embedding ~~computational~~ learning capability (generally in the form of microprocessors) into everyday objects to make them effectively communicate and perform useful tasks in a way that minimizes the end user's need to interact with computers as computers.
- Unlike ~~desktop computing~~ datacenter-based learning, ubiquitous ~~computing~~ learning can occur with any device, at any time, in any place and in any data format across any network.

Why Ubiquitous Learning?

- Rich applications, mostly driven by privacy needs
- Personalization for each device
- Offloading the computation & storage expense to devices
  - Workflow simplified: no need to maintain complex, clumsy datacenter systems (for both training and inference)
  - Risk minimized: benefits of decentralization

Some concrete evidences

- FL is widely adopted on billions of phones already.
  - Google uses it for input method (Gboard)[1]
  - Alibaba uses it for e-commerce recommendations (Taobao)[2]
  - Apple uses it for voice assistant (Siri)[3]
  - …
- IoT companies are seeking to train and adapt ML models on IoT devices (smart home scenario).

[1] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html
[2] Chaoyue Niu, et al. "Billion-scale federated learning on mobile clients: A submodel design with tunable privacy." MobiCom, 2020.
[3] https://www.technologyreview.com/2019/12/11/131629/apple-ai-personalizes-siri-federated-learning/

A review of DL system evolution on cloud/edge

AlexNet, 2012
- With GPU used

Parameter Server, 2014
- Many GPUs

Since then…

TensorFlow (Google), PyTorch (FB), PaddlePaddle (Baidu), Petuum (CMU), MXNet (Apache), DMTK (MS), Horovod, etc.

TF Serving, 2016
- For deployment

TVM, 2018
- Auto Compilation

…

TensorRT (Nvidia), PRETZEL (SNU), Rammer (MSRA), TorchScript (FB), DeepCPU (MS), etc.

A review of DL system evolution on cloud/edge

AlexNet, 2012
- With GPU used

Parameter Server, 2014
- Many GPUs

Since then…

TensorFlow (Google), PyTorch (FB), PaddlePaddle (Baidu), Petuum (CMU), MXNet (Apache), DMTK (MS), Horovod, etc.

TF Serving, 2016
- For deployment

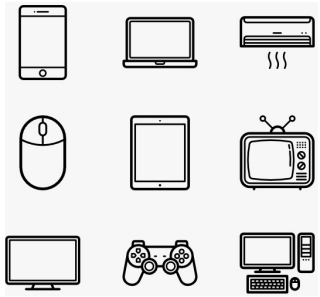
TVM, 2018
- Auto Compilation

… **TensorRT (Nvidia), PRETZEL (SNU), Rammer (MSRA), TorchScript (FB), DeepCPU (MS), etc.**

Federated Learning (Google), 2017
- A killer usage for on-device training
- A special case of decentralized learning

MNN (Alibaba), Core ML (Apple), 2019
- Preliminary support: only CPU and a few ML operators & optimizers

LONG WAY TO GO



DL for mobile sensing, 2015

TensorFlow Lite, 2017
- Rich support for heterogeneous processors and ML operators

Caffe2 (FB), Core ML (Apple), ncnn (Tencent), Paddle Lite (Baidu), MACE (Xiaomi), SNPE (Qualcomm), etc.

A review of DL system evolution on cloud/edge

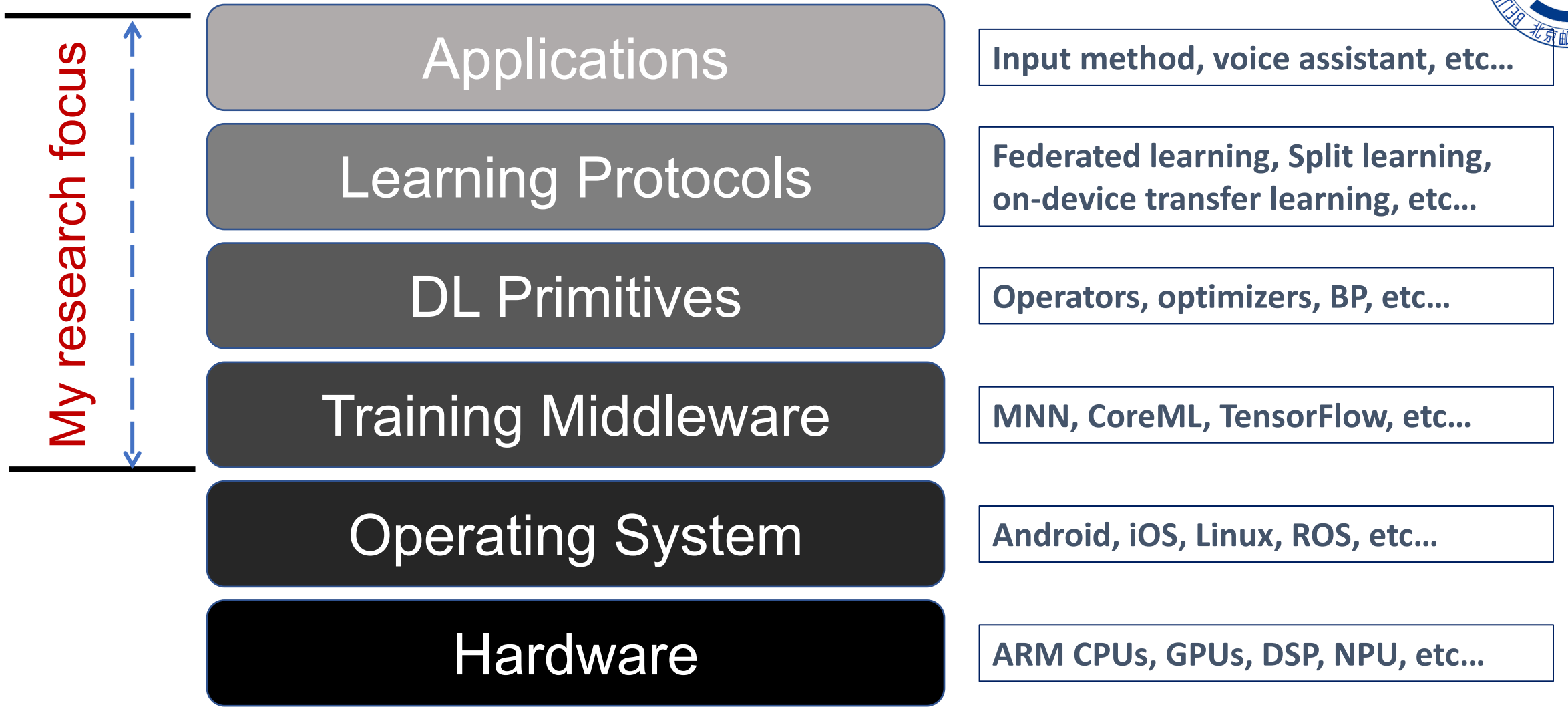
Our first attempt of on-device training
- **Input prediction personalization**
- **Published at UbiComp'18**

Heterogeneous FL platform
- **Published at WWW'21**

More to come…

Many attemps at on-device inference:
[MobiCom'18] DeepCache, [WWW'19] An empirical study, [MobiSys'20] Elf, [TMC'19] DeepWear

HW-SW stack for UL



2 Key research questions in UL

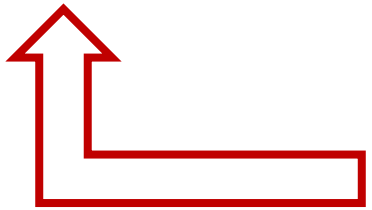
1. Training data is limited, non-IID, or even not labelled
   - Model accuracy heavily relies on data!

2. Devices have constrained hardware resources
   - Training a ML model is notoriously resource-hungry!

2 Key research questions in UL

1. Training data is limited, non-IID, or even not labelled
   - Model accuracy heavily relies on data!

2. Devices have constrained hardware resources
   - Training a ML model is notoriously resource-hungry!



We need a system-algorithm co-design

Outline

- Ubiquitous Learning (UL): What and Why?
  - A system software perspective
  - The status quo

- A measurement study of UL
- Memory optimizations of UL
- Other on-going research of UL

On-device training: a measurement study

- Target library: MNN[1] by Alibaba
- 6 Android devices
- 5 classic CNN models
  - LeNet, AlexNet, MobileNetv2, SqueezeNet, GoogLeNet
- CPU by default

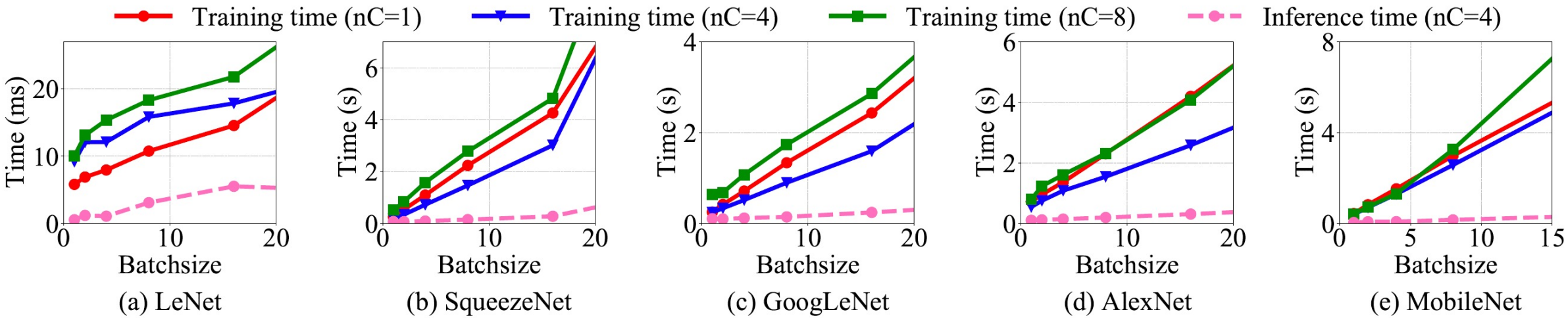
Testing platform	Training library	Training time (ms)		
		BS = 1	BS=2	BS=4
Samsung Note 10	MNN	516	812	1365
	DL4J	3,032	6,129	OOM
RPI 3B+	MNN	6698	10,651	OOM
	TensorFlow	10,468	14,157	27,574
	PyTorch	48,274	79,097	OOM

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+	Snapdragon 835, 6GB RAM	2017
Huawei Honor 8	Kirin 950, 3GB RAM	2016

[1] https://github.com/alibaba/MNN

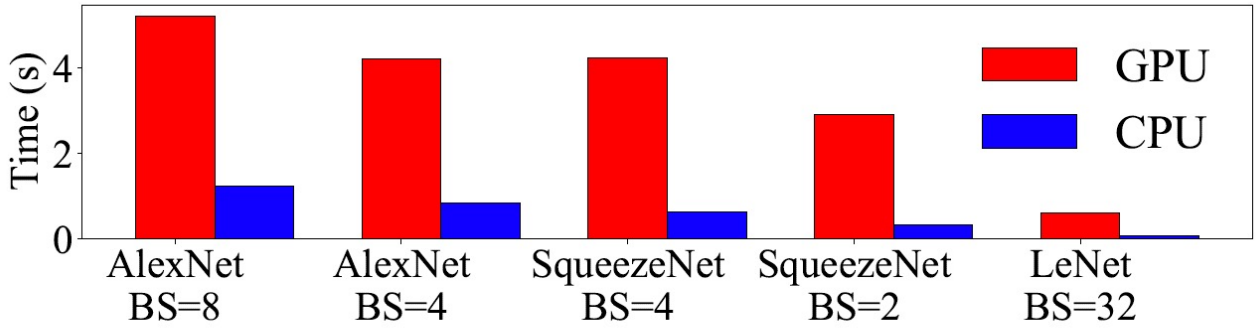
Training time

- Training takes much more time than inference
  - Up to 17.8x gap, much larger than the FLOPs-gap

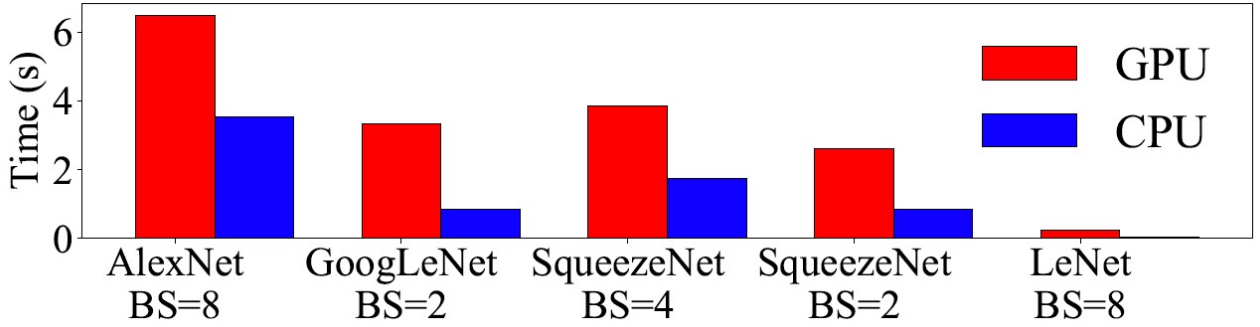


Training time

- Training takes much more time than inference
- GPU cannot speedup

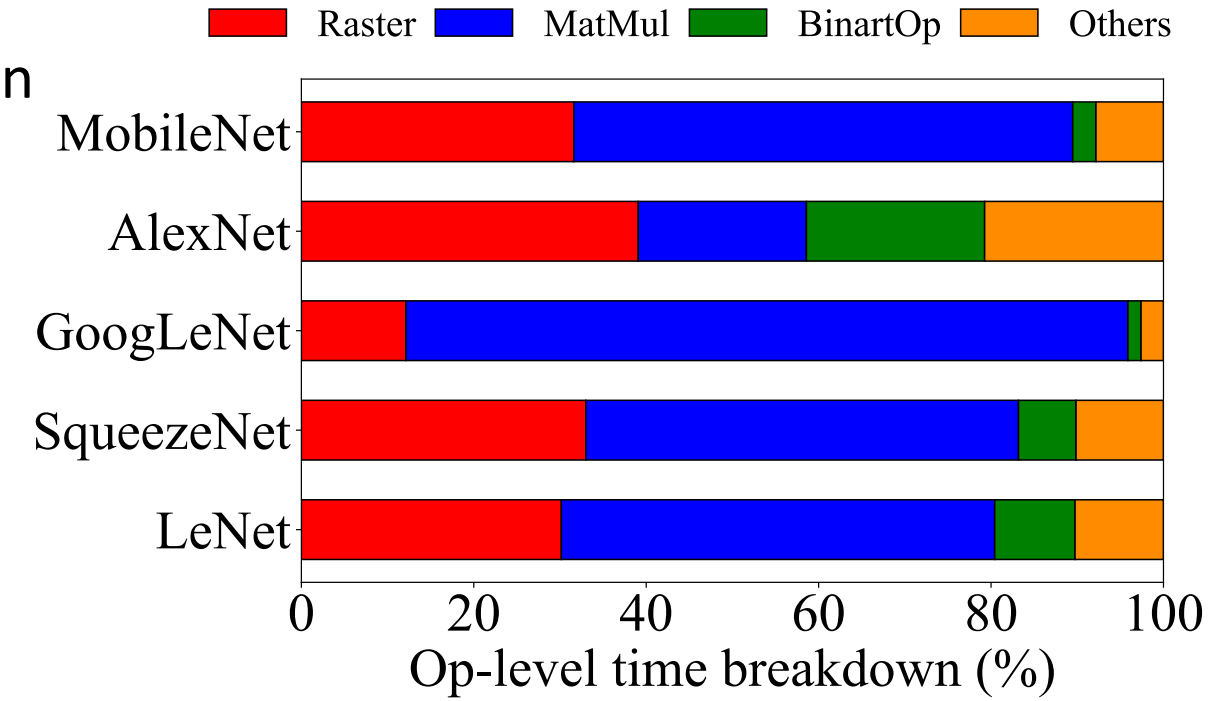


(a) Huawei Mate 30 (Mali)



Training time

- Training takes much more time than inference
- GPU cannot speedup
- Op-level breakdown
  - Raster: ops with shape transformation
  - Conv = 3 x Raster + MatMul



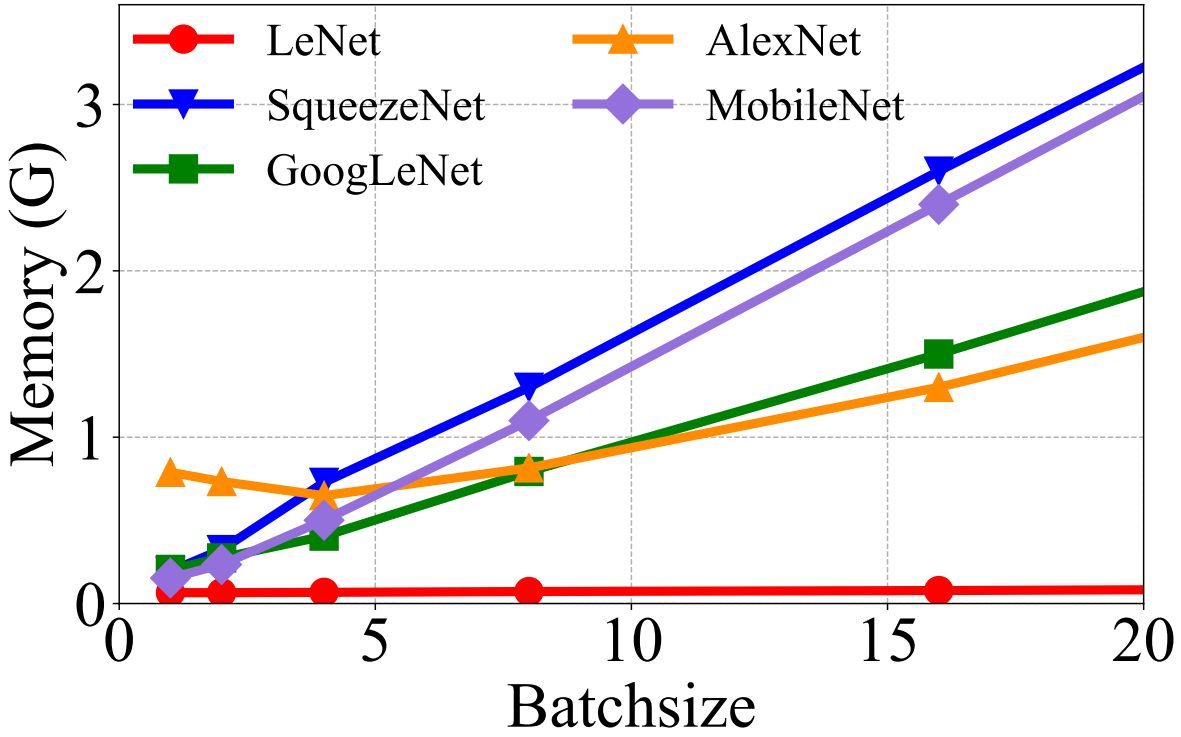
Training time

- Training takes much more time than inference
- GPU cannot speedup
- Op-level breakdown

- Why?
  - The training support of MNN is still at very preliminary stage
  - Training is far more complex than inference: much more operators, dynamic weights update, variable batch size, etc…

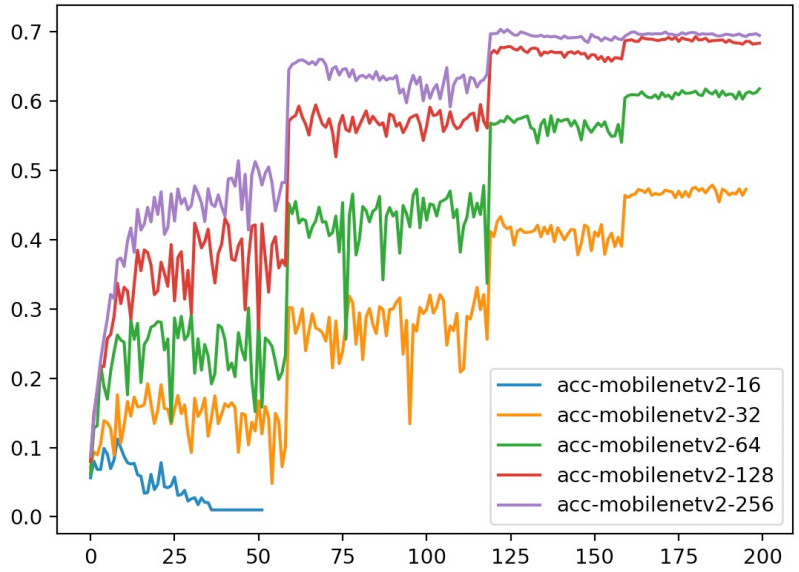
Memory footprint

- Training is very memory-intensive
  - 16-32 is typically the max batch size supported by a high-end device (6~8 GBs)

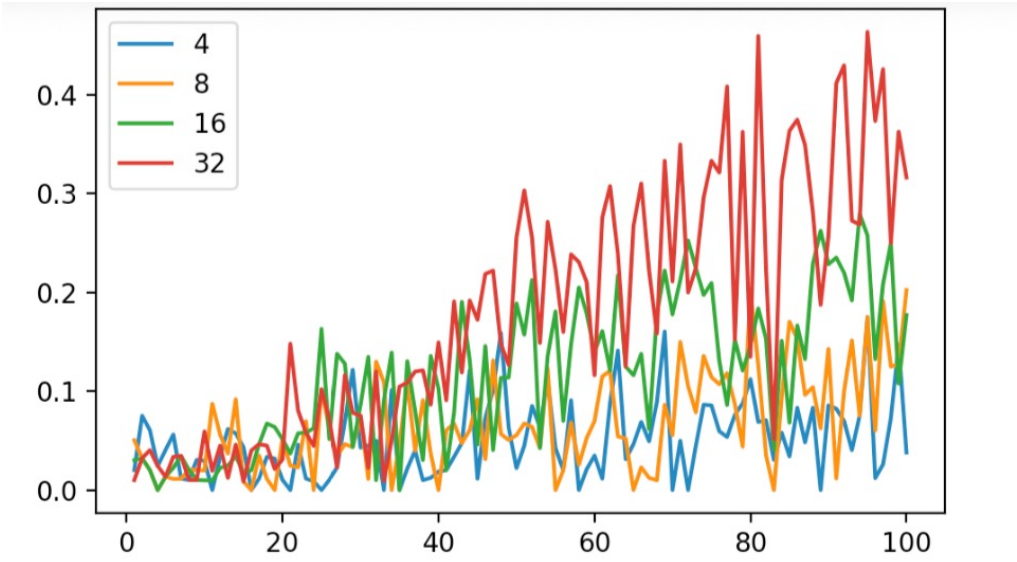


Memory footprint

- Training is very memory-intensive
  - 16-32 is typically the max batch size supported by a high-end device
  - Enough for a good convergence? NO!



Single-machine setting (MobileNet-v2)



Federated setting (MobileNet-v2)

[1] Smith, Samuel L., et al. "Don't decay the learning rate, increase the batch size." *arXiv preprint arXiv:1711.00489* (2017).

Memory footprint

- Training is very memory-intensive
  - 16-32 is typically the max batch size supported by a high-end device
  - Enough for a good convergence? NO!

A ”clean” environment.
In practice? NO!

System parameters

- CPU parameters open rich trade-off among latency and energy

Lowest latency

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

Lowest energy

Implications and future directions

- Generating efficient training operators
  - NN compilers?

- **Memory optimizations**
  - ➢ **The next chapter of this talk**

- Tuning system parameters
  - Model-level or operator-level?

Outline

- Ubiquitous Learning (UL): What and Why?
  - A system software perspective
  - The status quo
- A measurement study of UL
- Memory optimizations of UL
- Other on-going research of UL

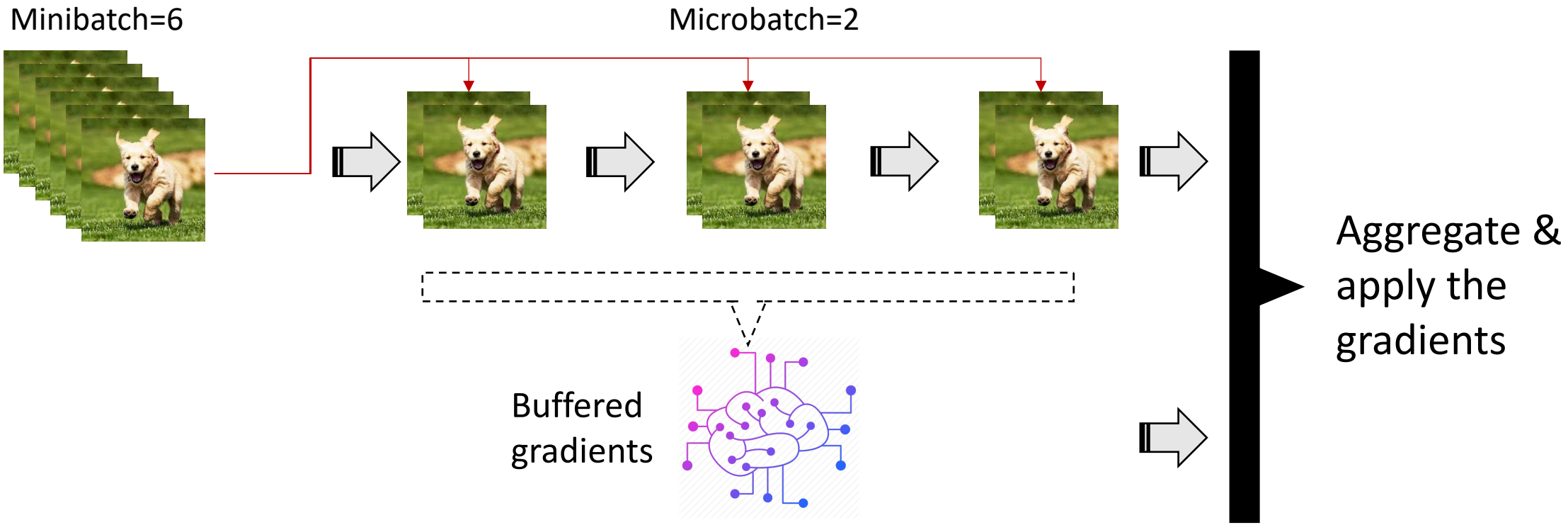
Design goals and principles

- Goal: supporting larger batch size training with given upper bound of peak memory usage

- Principles
  1. Don’t sacrifice the accuracy – No quantization & sparsity
  2. Sacrifice latency (energy) as less as possible – Making tradeoffs

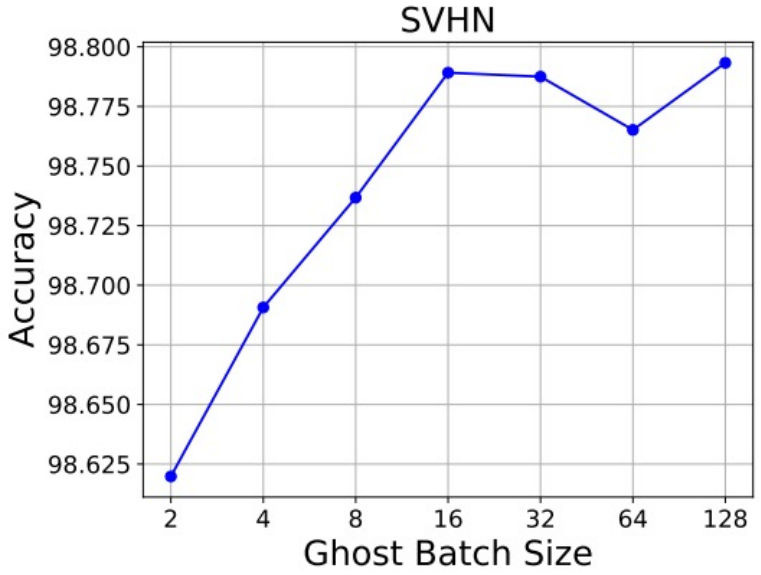
Splitting the minibatch

- A technique called microbatch (or virtual batch, ghost batch)

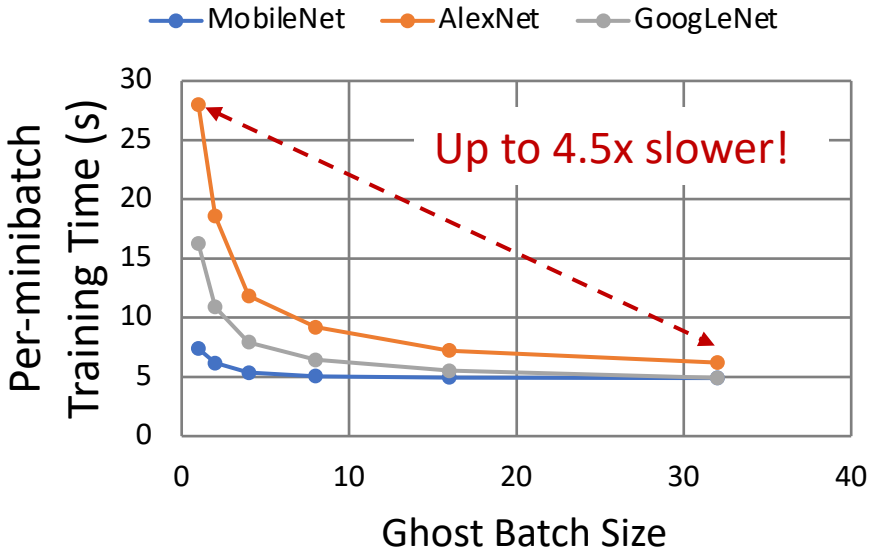


Splitting the minibatch

- A technique called microbatch (or virtual batch, ghost batch)
- Why it's not enough?



Uncertainty from BN layer



Performance sacrificed

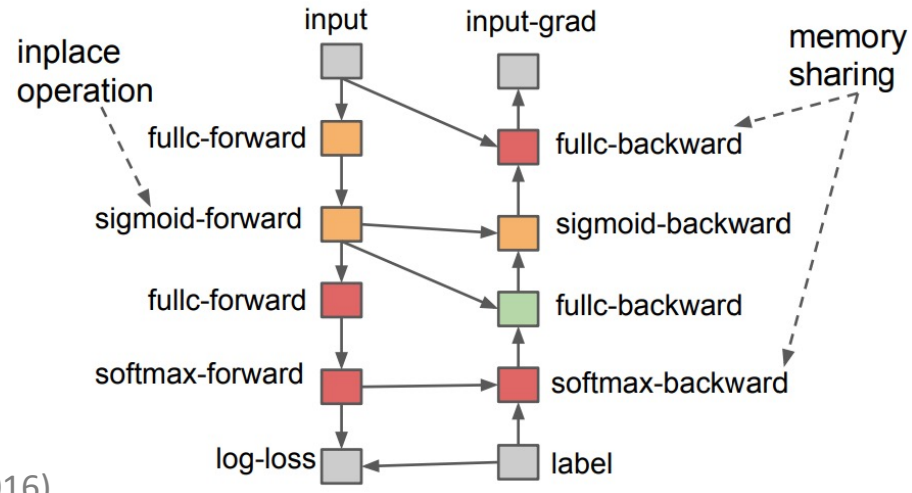
[1] Summers, Cecilia, and Michael J. Dinneen. "Four things everyone should know to improve batch normalization." *ICLR 2020.*

How about another technique

- (✗) Memory swapping?
  - Mobile devices have integrated memory for heterogeneous processors
  - 40x slower according to our preliminary experiments
- (?) Recompute[1]?
  - Under implementation but seems more promising
  - An efficient memory pool is needed

Memory reduced to $O(\log n)$

Forward latency increased to $O(n \log n)$



[1] Tianqi Chen, et al. "Training deep nets with sublinear memory cost." arXiv:1604.06174 (2016).

Combining them

- Determining the appropriate ghost batch size and recomputed nodes to achieve the optimal performance
  - Hardware-specific, large search space

Outline

- Ubiquitous Learning (UL): What and Why?
  - A system software perspective
  - The status quo

- A measurement study of UL
- Memory optimizations of UL
- Other on-going research of UL

Some on-going research to share

1. DSP-based, integer-only on-device training algorithm and library

2. Adaptive network topology and protocol for cross-silo federated learning

Takeaways

- Machine (deep) learning is happening everywhere at anytime
- The system support for such ubiquitous learning is still at very preliminary stage – so many open problems!
- Open to discussion and collaboration on it

Outstanding PhD students from BUPT/PKU working on UL ⇒

Our code ⇒ https://github.com/UbiquitousLearning