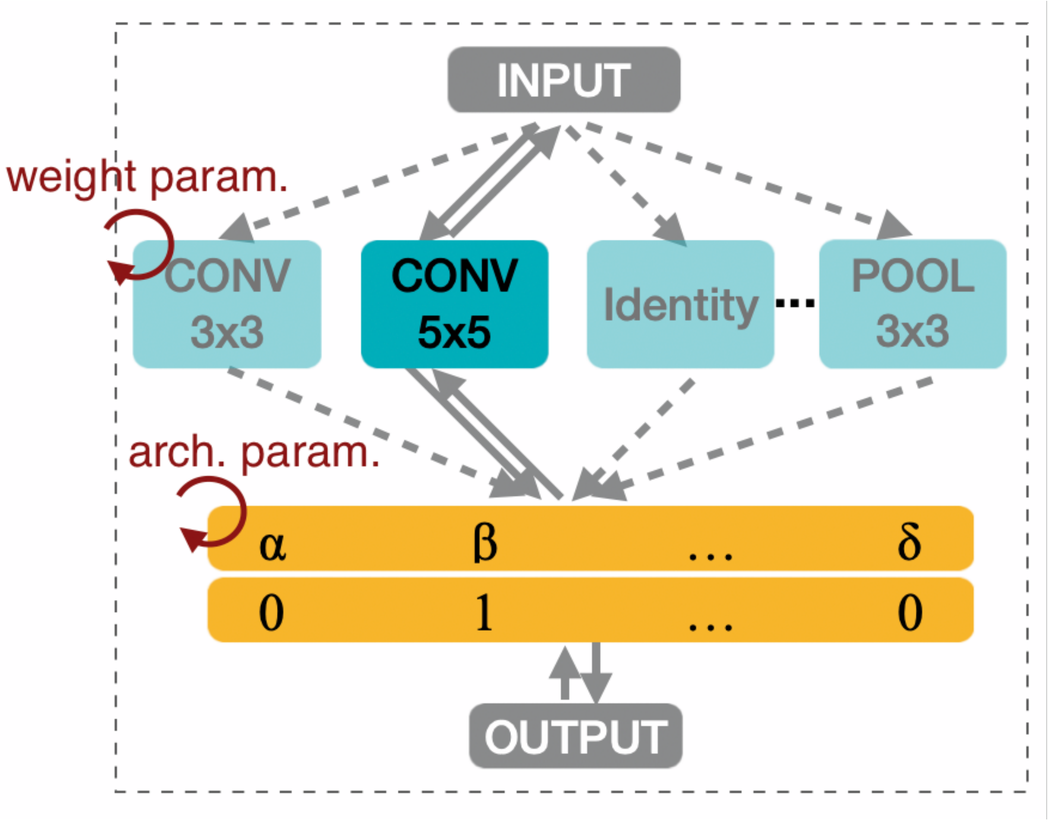
Efficient and Scalable Deep Leaning: Automated and Federated

Ligeng Zhu

Brief Bio

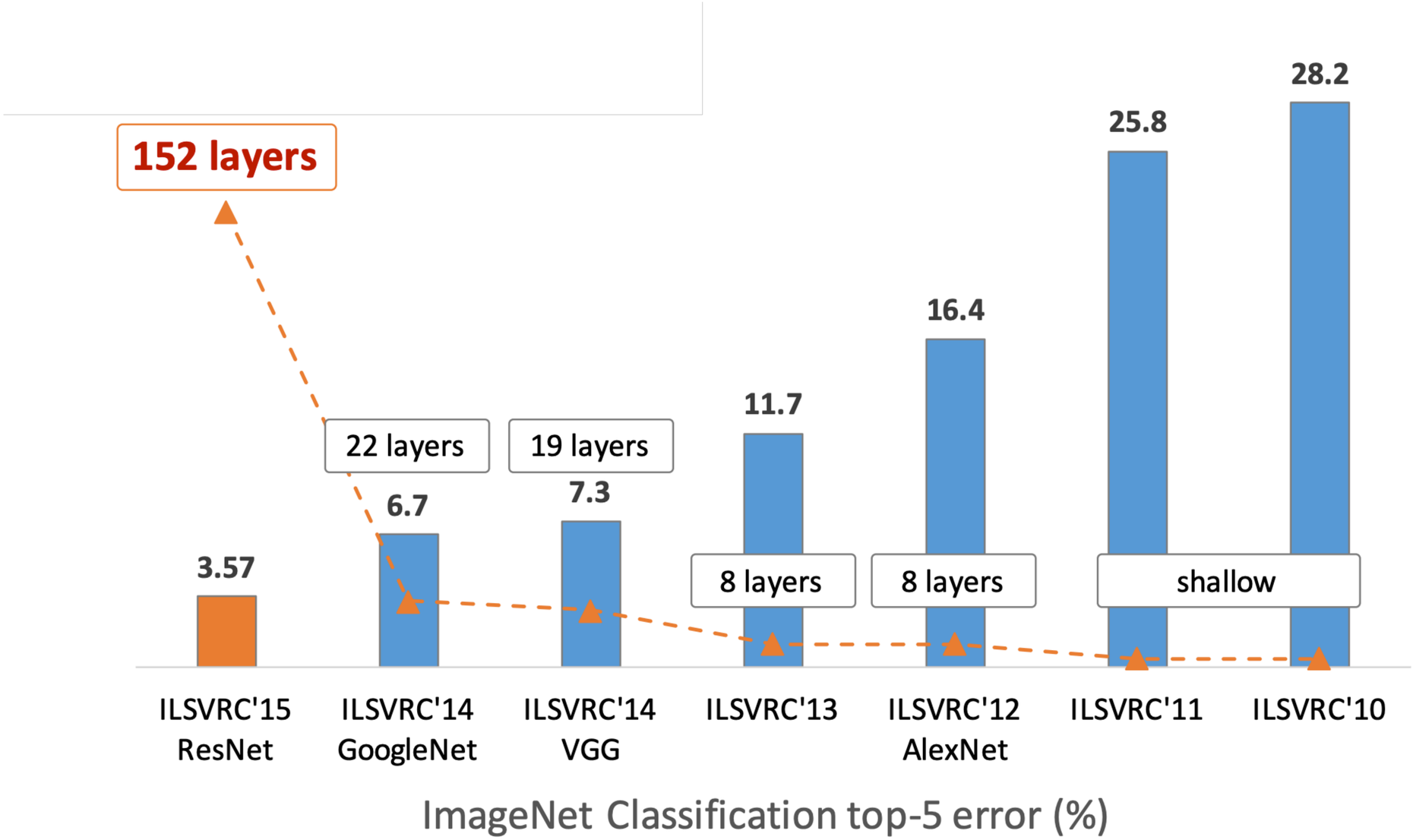
- Was born in Taizhou, Zhejiang Province, China.
- Entered Zhejiang University to pursue study in CS.
- Dual degree program at Simon Fraser University, also major in CS.
- Intern at TuSimple in 17’s summer, love the weather in SD.
- Visit MIT (host: Song Han) in 18-19.
- Work as a Data Scientist at Intel AI Labs.

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware



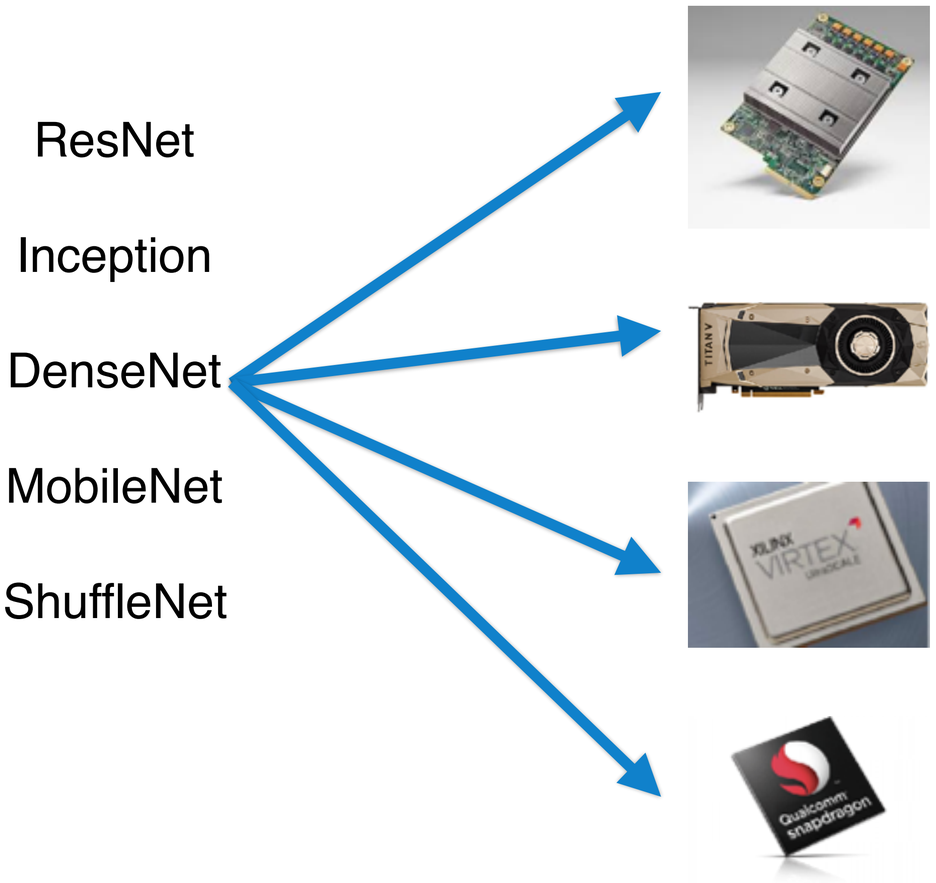
Han Cai, Ligeng Zhu,, Song Han

History of CNN Architectures



Generalization v.s. Specialization

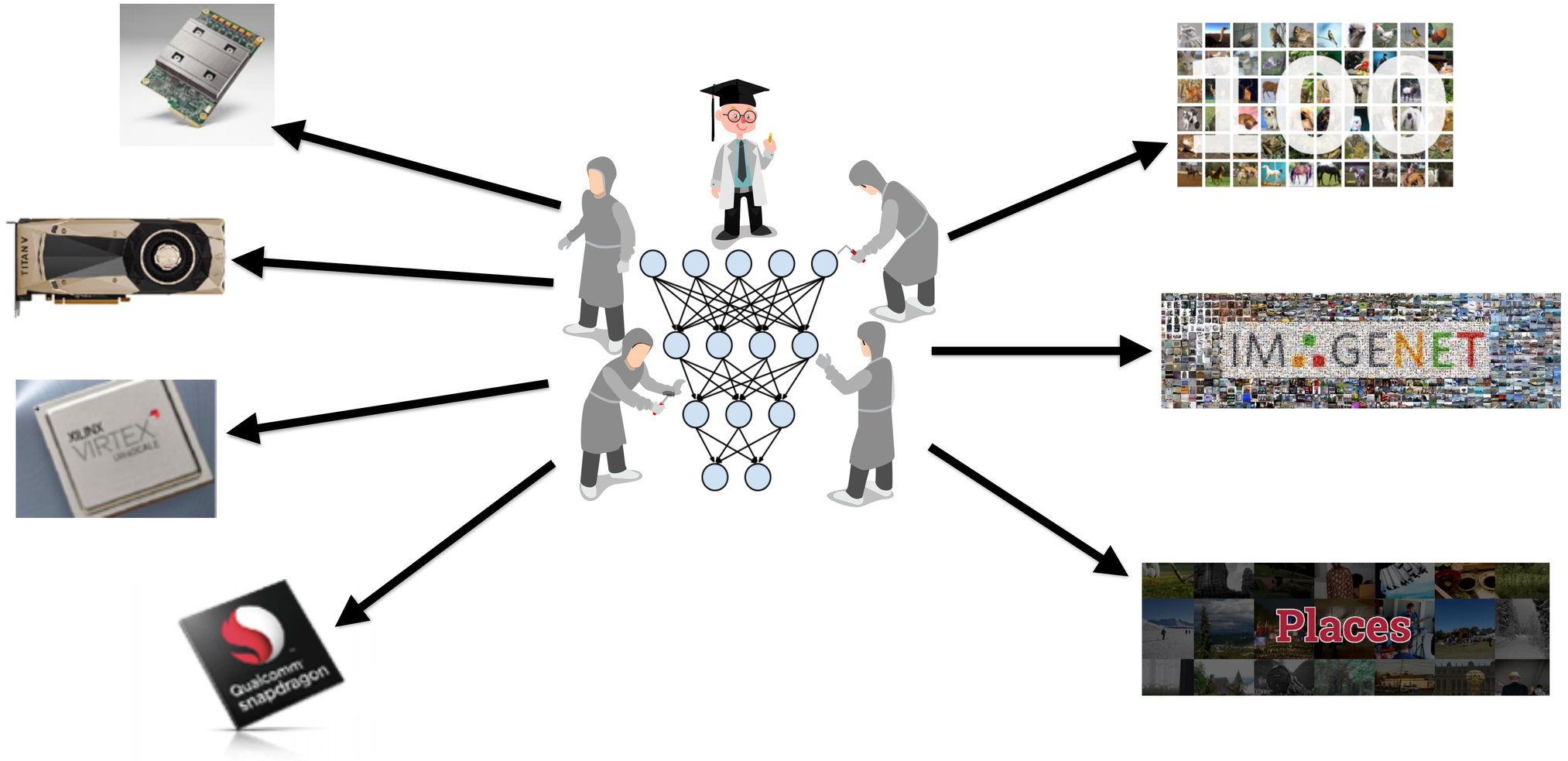
- Previously, people tend to design a single efficient CNN for all platforms and all datasets.
- But, different dataset in fact has different features, e.g., size of object, scale, rotation.
- But, different platform in fact has different properties, e.g. degree of parallelism, cache size, #PE, memory BW.
- Machine learning wants **generalization**
  Hardware efficiency needs **specialization**
  A **generalized** model to handle **specialized** is not ideal!



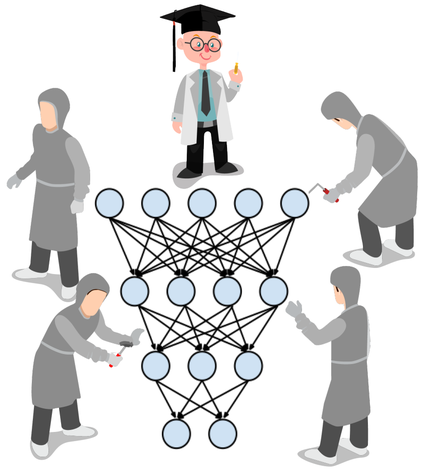
Case by case Design — Expensive!

Different Platforms

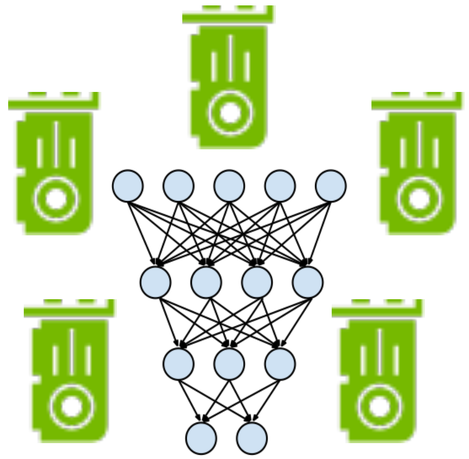
Different datasets



From Manual Design to Automatic Design



Use Human Expertise



Use Machine Learning

Manual Architecture Design

ResNet / DenseNet / Inception / …

Automatic Architecture Search

Reinforcement Learning / Mento Carlo / …

From General Design to Specialized CNN

Previous Paradigm:
One CNN for all platforms.

ResNet

Inception

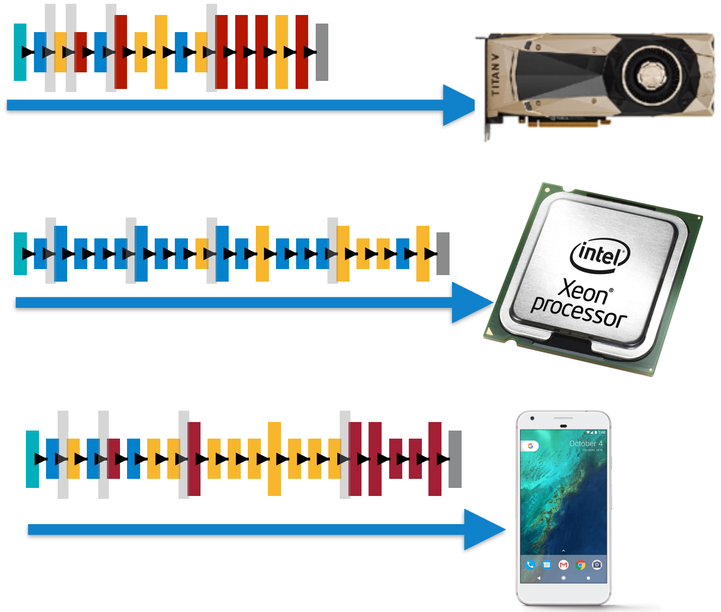
DenseNet

MobileNet

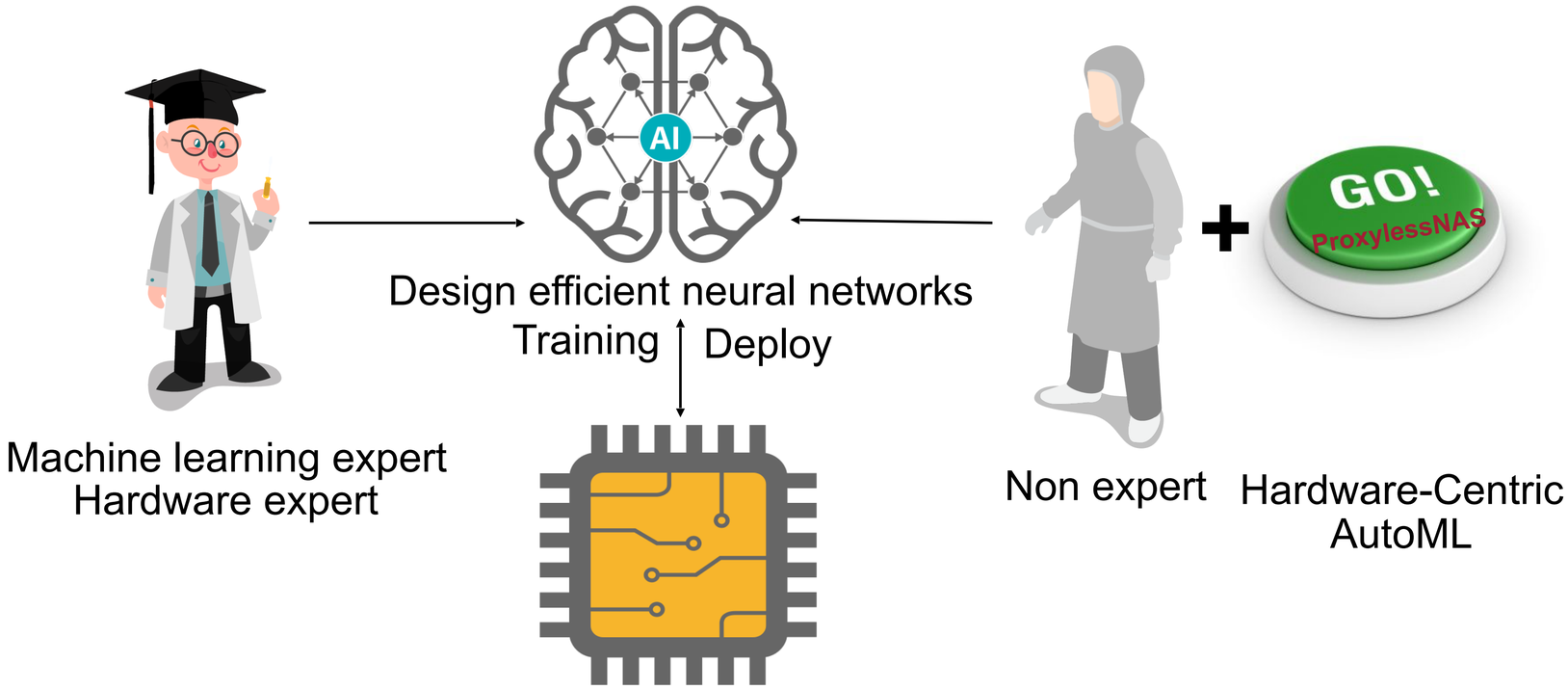
ShuffleNet

Our Work:
Customize CNN for each platform.

Proxyless
NAS

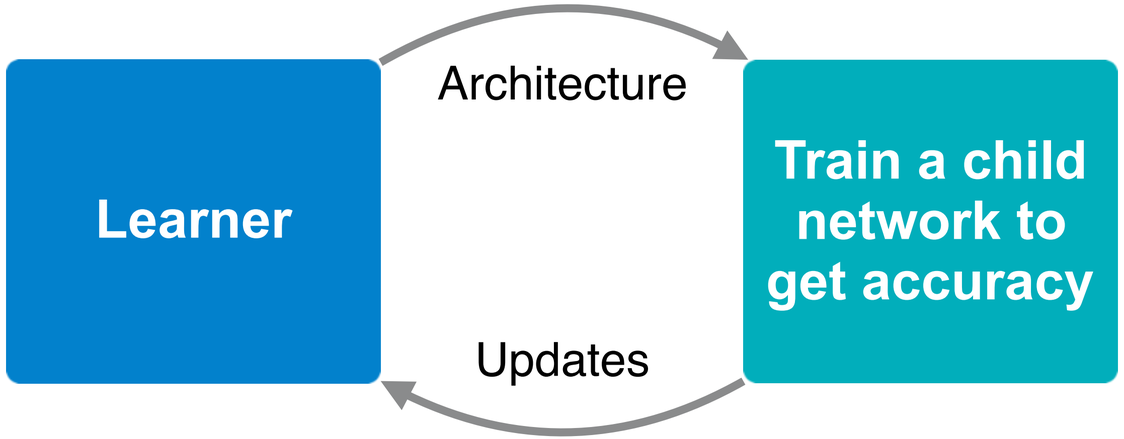


Design Automation for Hardware Efficient Nets



Hardware-Centric AutoML allows non-experts to efficiently design neural network architectures with a push-button solution that runs fast on a specific hardware.

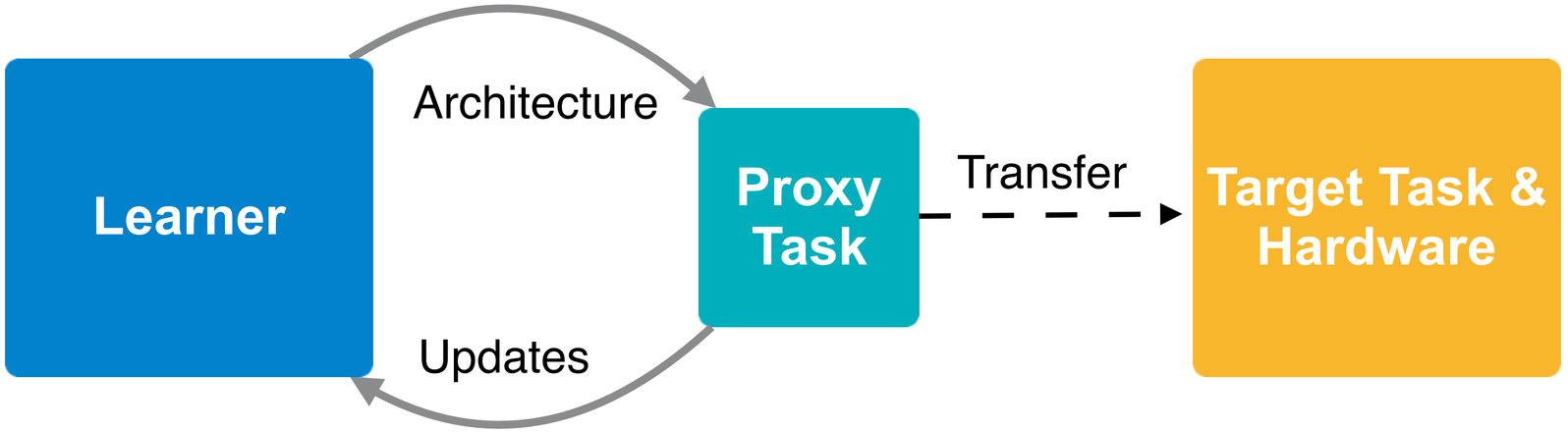
Conventional NAS: Computationally Expensive



VERY EXPENSIVE.

- NASNet: 48,000 GPU hours ≈ 5 years on single GPU
- DARTS: 100Gb GPU memory* ≈ 9 times of modern GPU

Conventional NAS: Proxy-Based



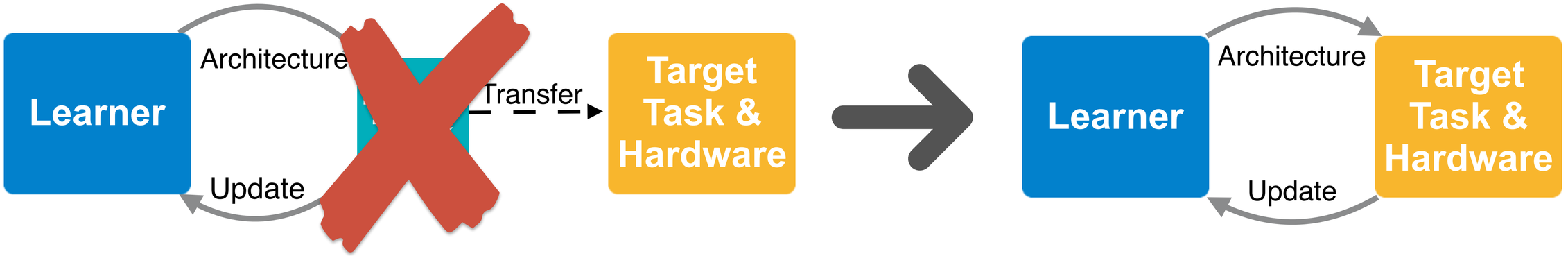
Therefore, previous work have to utilize **proxy tasks:**

- CIFAR-10 -> ImageNet
- Small architecture space (e.g. low depth) -> large architecture space
- Fewer epochs training -> full training

Limitations of Proxy

- **Suboptimal** for the target task
- **Blocks are forced to share the same structure.**
- Cannot optimize for **specific hardware.**

Our Work: Proxyless, Save GPU Hours by 200x



Goal: Directly learn architectures on the **target task** and **hardware**, while allowing all blocks to have different structures. We achieved by

1. Reducing the cost of NAS (GPU hours and memory) to the **same level of regular training**.
2. Cooperating **hardware feedback** (e.g. latency) into the search process.

Model Compression

Neural Architecture Search

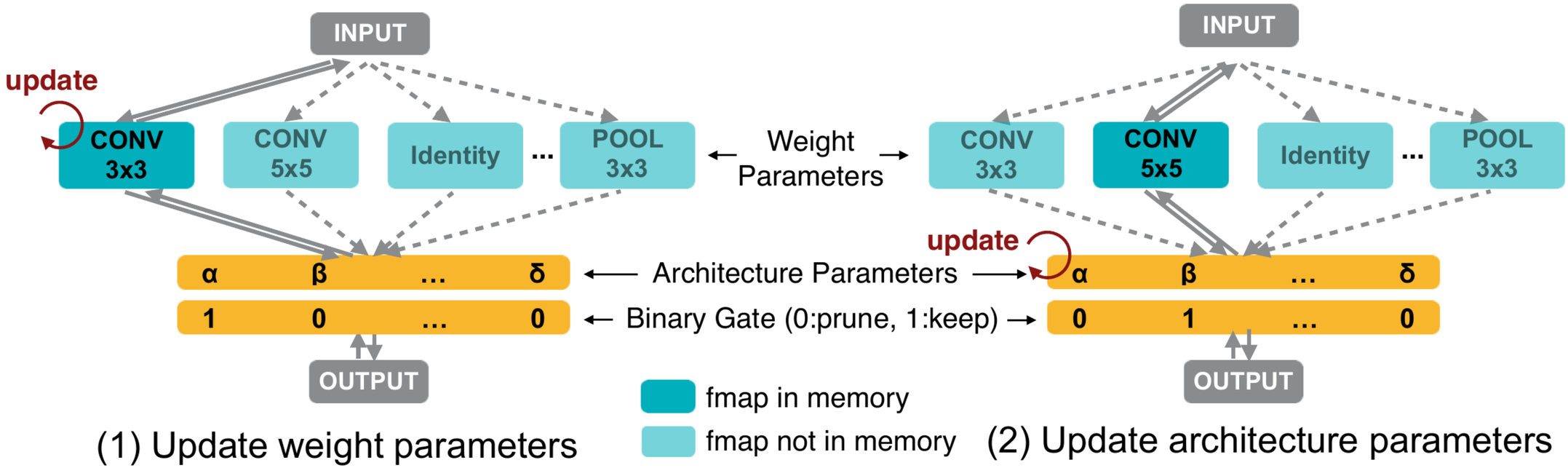
Pruning

Binarization

Save GPU hours

Save GPU Memory

Save GPU Hours



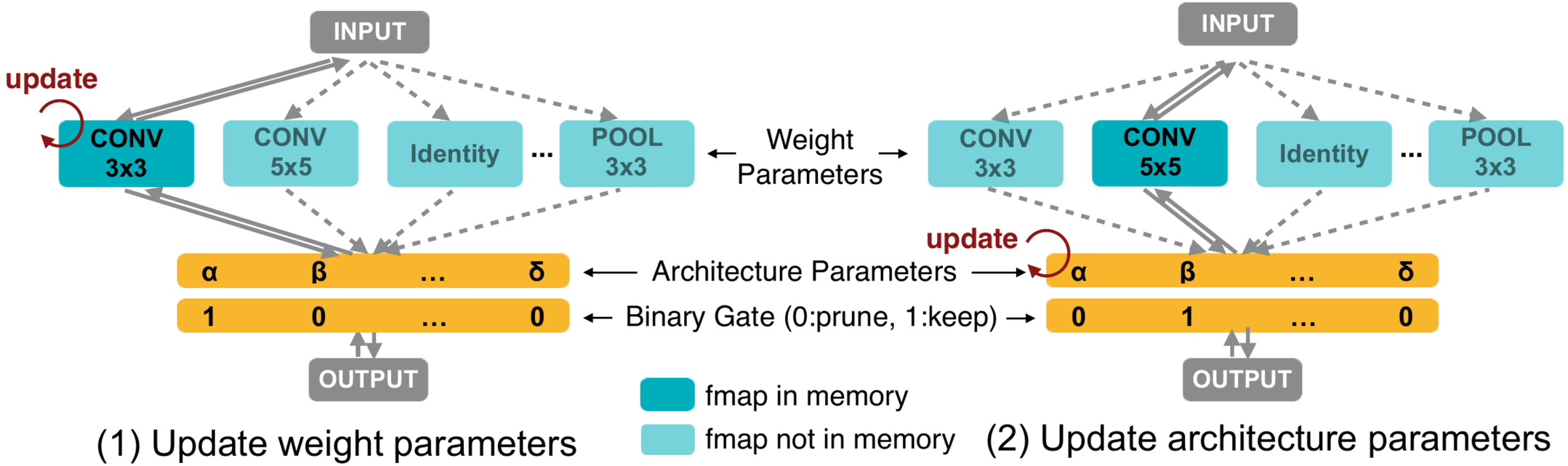
Pruning redundant paths based on architecture parameters

Simplify NAS to be a **single training process** of a over-parameterized network.

No meta controller. Stand on the shoulder of giants.

Build the cumbersome network **with all candidate paths**

Save GPU Memory



Binarize the architecture parameters and allow only **one path of activation to be active** in memory at run-time.

We propose **gradient-based** and **RL** methods to update the **binarized parameters**.

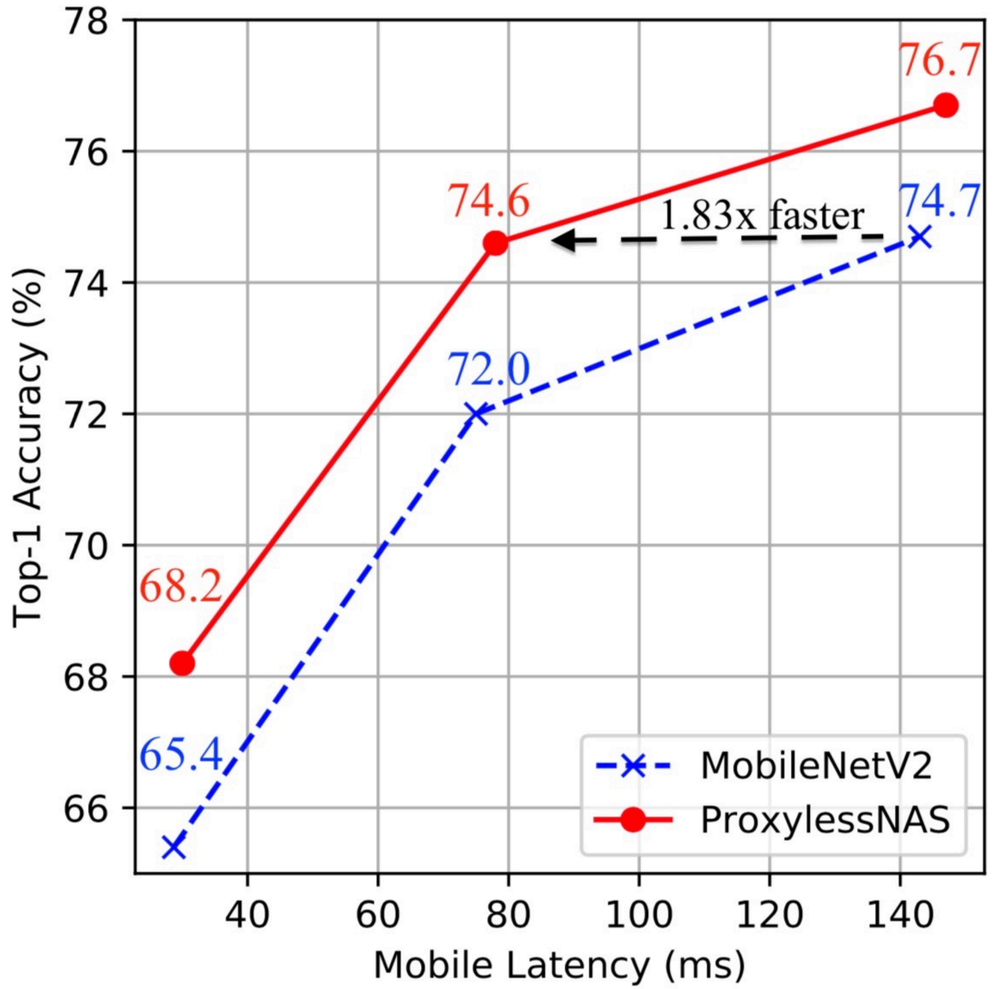
Thereby, the memory footprint reduces from **O(N) to O(1)**.

Results: ProxylessNAS on CIFAR-10

Model	Params	Test error
DenseNet-BC (Huang et al., 2017)	25.6M	3.46
PyramidNet (Han et al., 2017)	26.0M	3.31
Shake-Shake + c/o (DeVries & Taylor, 2017)	26.2M	2.56
PyramidNet + SD (Yamada et al., 2018)	26.0M	2.31
ENAS + c/o (Pham et al., 2018)	4.6M	2.89
DARTS + c/o (Liu et al., 2018c)	3.4M	2.83
NASNet-A + c/o (Zoph et al., 2018)	27.6M	2.40
PathLevel EAS + c/o (Cai et al., 2018b)	14.3M	2.30
AmoebaNet-B + c/o (Real et al., 2018)	34.9M	2.13
Proxyless-R + c/o (ours)	5.8M	2.30
Proxyless-G + c/o (ours)	5.7M	**2.08**

- Directly explore a huge space: 54 distinct blocks and possible architectures
- State-of-the-art test error with 6X fewer params (Compared to AmeobaNet-B)

Results: ProxylessNAS on ImageNet, Mobile Platform



- With >74.5% top-1 accuracy, ProxylessNAS is **1.8x faster** than MobileNet-v2, the current industry standard.

Results: ProxylessNAS on ImageNet, Mobile Platform

	Model	Top-1	Latency	Hardware Aware	No Proxy	No Repeat	Search Cost
Manually Designed	MobilenetV1	70.6	113ms	-	-	x	-
	MobilenetV2	72.0	75ms	-	-	x	-
NAS	NASNet-A	74.0	183ms	x	x	x	48000
	AmoebaNet-A	74.4	190ms	x	x	x	75600
	MNasNet	74.0	76ms	yes	x	x	40000
ProxylessNAS	ProxylessNAS-G	71.8	83ms	yes	yes	yes	200
	ProxylessNAS-G + LL	74.2	79ms	yes	Yes	yes	200
	ProxylessNAS-R	74.6	78ms	yes	Yes	yes	200
	ProxylessNAS-R + MIXUP	75.1	78ms	yes	yes	yes	200

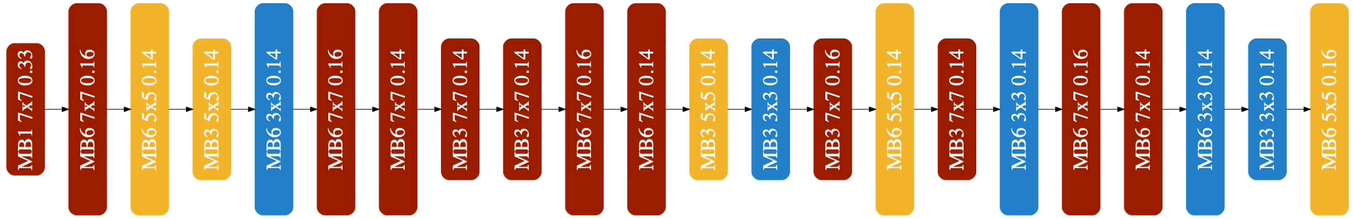
ProxylessNAS achieves state-of-the art accuracy (%) on ImageNet (under mobile latency constraint ≤ 80ms) with 200× less search cost in GPU hours. “LL” indicates latency regularization loss.

Results: Proxyless-NAS on ImageNet, GPU Platform

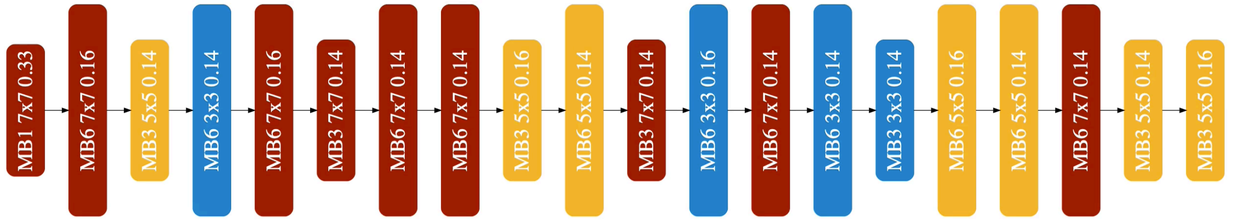
Model	Top-1	Top-5	GPU latency
MobileNetV2 (Sandler et al., 2018)	72.0	91.0	6.1ms
ShuffleNetV2 (1.5) (Ma et al., 2018)	72.6	-	7.3ms
ResNet-34 (He et al., 2016)	73.3	91.4	8.0ms
NASNet-A (Zoph et al., 2018)	74.0	91.3	38.3ms
DARTS (Liu et al., 2018c)	73.1	91.0	-
MnasNet (Tan et al., 2018)	74.0	91.8	6.1ms
Proxyless (GPU)	**75.1**	**92.5**	**5.1ms**

When targeting GPU platform, the accuracy is further improved to 75.1%. 3.1% higher than MobilenetV2.

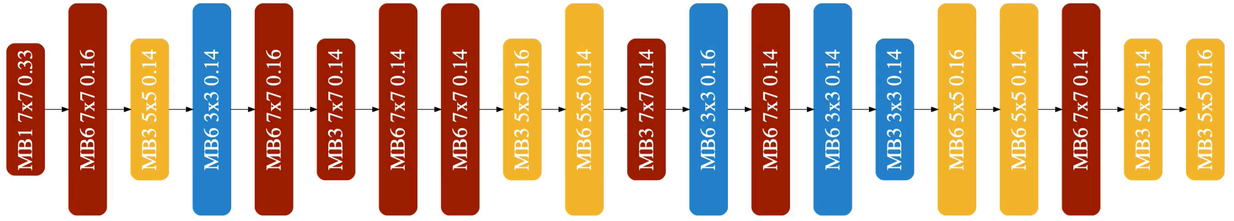
The History of Architectures



(1) The history of finding efficient Mobile model



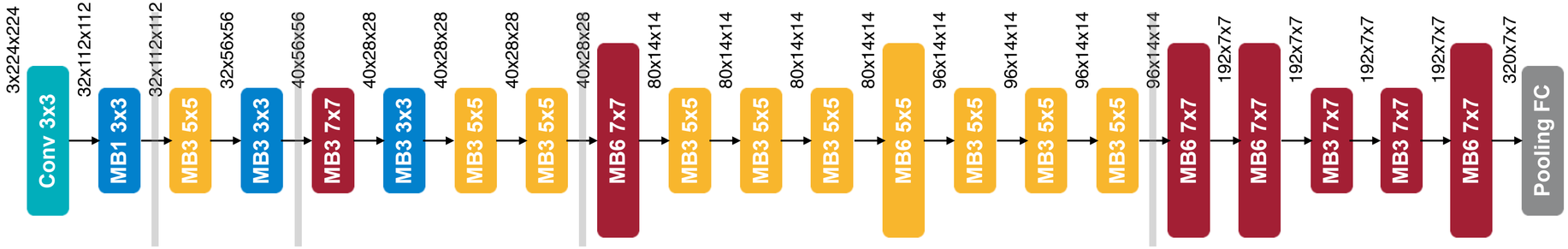
(2) The history of finding efficient CPU model



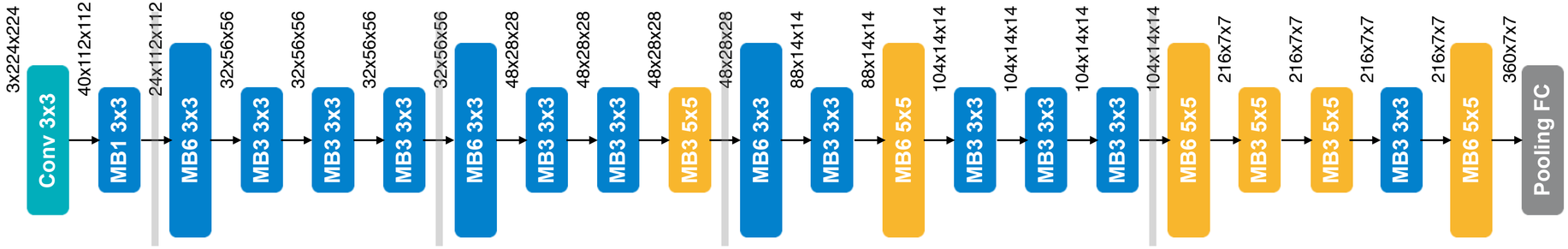
(3) The history of finding efficient GPU model

Epoch-00

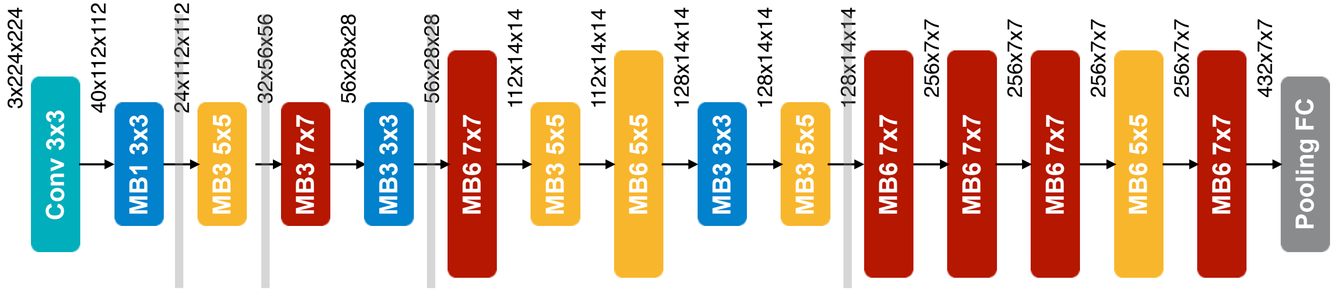
Detailed Architectures



(1) Efficient mobile architecture found by Proxy-less NAS.



(2) Efficient CPU architecture found by Proxy-less NAS.



(3) Efficient GPU architecture found by Proxy-less NAS.

ProxylessNAS for Hardware Specialization

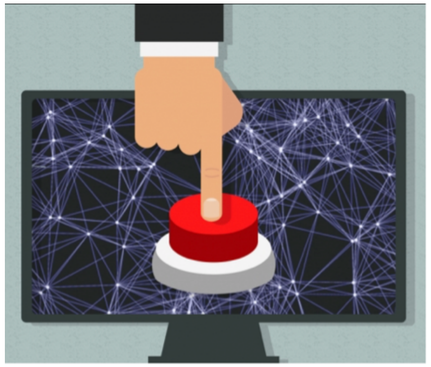

Model	Top-1	GPU	CPU	Mobile
Specialized for GPU	75.1	**5.1ms**	204.9ms	124ms
Specialized for CPU	75.3	7.4ms	**138.7ms**	116ms
Specialized for Mobile	74.6	7.2ms	164.1ms	**78ms**

Achievements of Design Automation

- **The first place** in the Visual Wake-up Word Challenge@CVPR'19
  - with <250KB model size, <250KB peak memory usage, <60M MAC
- **The third place** in the classification track of the LPIRC@CVPR.
  - image classification within 30ms latency on a Pixel-2 phone.

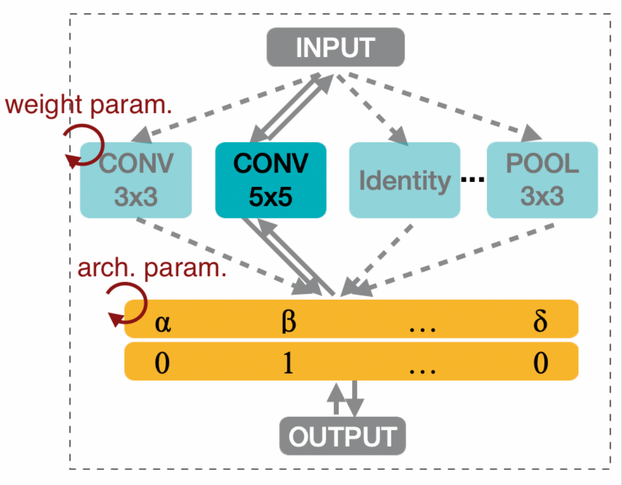
Both powered by Design Automation!

Embrace Open-source

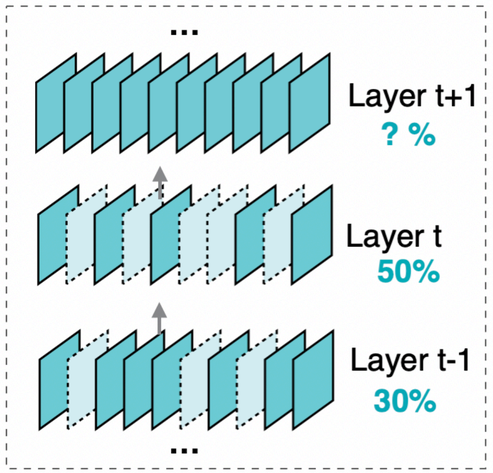


All codes are now public at

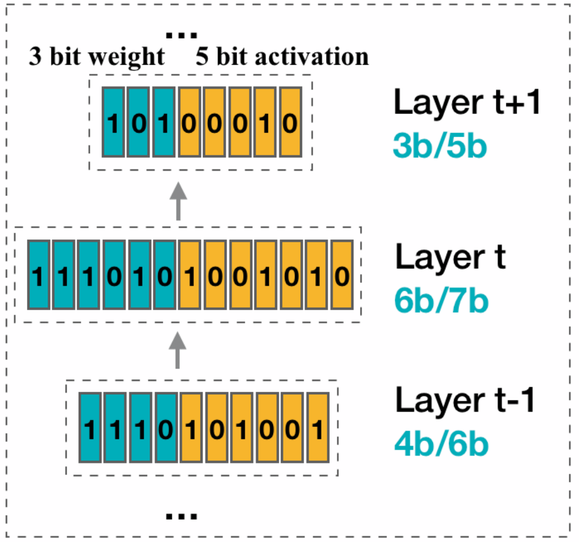
https://github.com/MIT-HAN-LAB



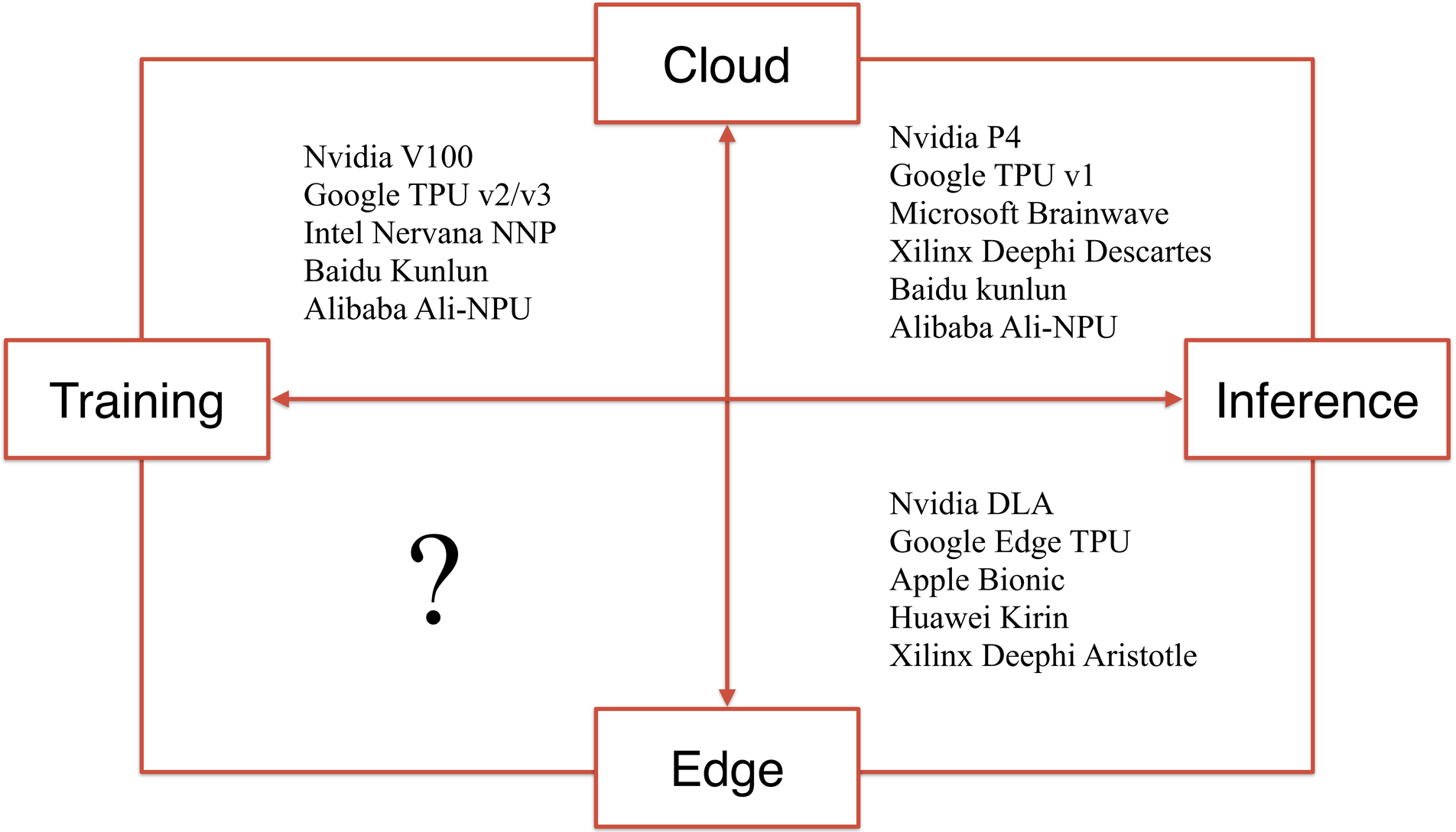
Proxyless Neural Architecture Search
[ICLR 2019]



AMC: AutoML for Model Compression
[ECCV 2018]



HAQ: Hardware-aware Automated Quantization
[CVPR 2019] Oral


Cloud
Nvidia V100
Google TPU v2/v3
Intel Nervana NNP
Baidu Kunlun
Alibaba Ali-NPU
Nvidia P4
Google TPU v1
Microsoft Brainwave
Xilinx Deephi Descartes
Baidu kunlun
Alibaba Ali-NPU
Training
Inference
?
Nvidia DLA
Google Edge TPU
Apple Bionic
Huawei Kirin
Xilinx Deephi Aristotle
Edge

Distributed Training Across the World

Ligeng Zhu, Yao Lu, Hangzhou Lin, Yujun Lin, Song Han

Conventional Distributed Training

…… ……

[2011] Niu et al. *Hogwild!: A Lock-Free Approach to Parallelizing Stochastic*
[2012] Google. *Large Scale Distributed Deep Networks*
[2012] Ahmed et al. *Scalable inference in latent variable models*.
[2014] Li et al. *Scaling Distributed Machine Learning with the Parameter Server.*
[2017] Facebook. *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour.*

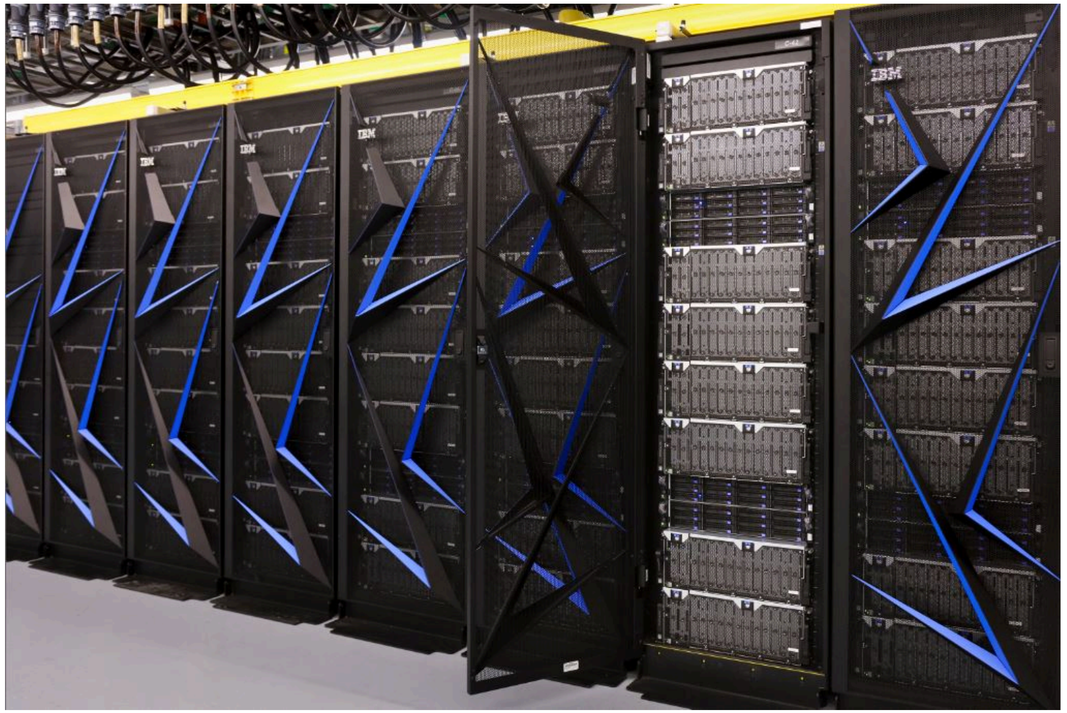
…… ……

Almost all of them are performed **within a cluster.**

Why distributed within a cluster?

- Scalability
  - Network bandwidth > 10Gbps
  - Network latency < 1ms

- Easy to manage
  - Hardware failure
  - System upgrade



Why distributed between clusters?

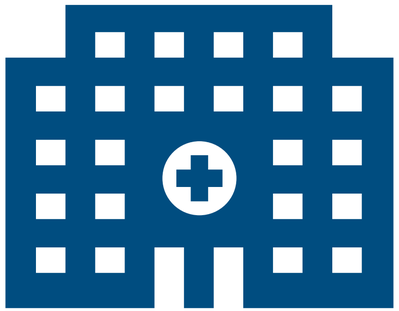
- Customization
  - I.e., Different users will have a different tone for speech recognition

Amazon
Alexa

Apple
Home Pod

Google
Home

- Security
  - Data cannot leave device because of security and regularization.



Limitation on Scalability (across clusters)

Bandwidth

- Infinity band: up to 100 Gb/s
- Normal ethernet: up to 10 Gb/s
- Mobile network: 100 Mb/s (4G), 1Gb/s (5G)

Latency

- Infinity band: < 0.002 ms
- Normal ethernet: ~0.200 ms
- Mobile network: ~50ms (4G) / ~10ms (5G)

What we need

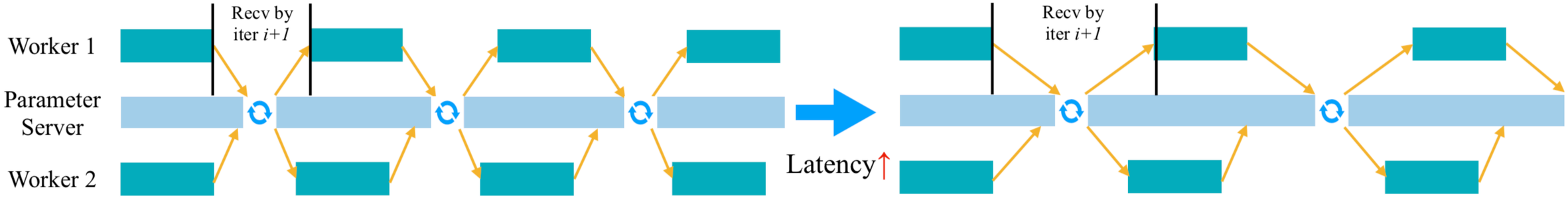
- ResNet 50: 24.37MB, 0.3s / iter (v100)
- At least 600 Mb/s bandwidth and 1ms latency.

Limitation on Scalability (across clusters)

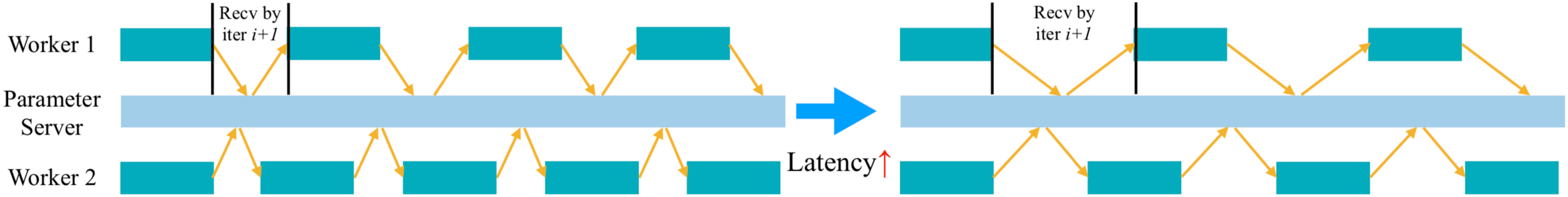
- Bandwidth can be always improved by
  - Hardware upgrade (Wired: fiber, Wireless: 5G)
  - Gradient sparsification (e.g., DGC, one-bit)

- Latency is hard to reduce because **physical laws**.
  - I.e. Shanghai to Boston, 11,725km, even considering the speed of light, still takes 78ms.

Conventional Algos suffer from high latency

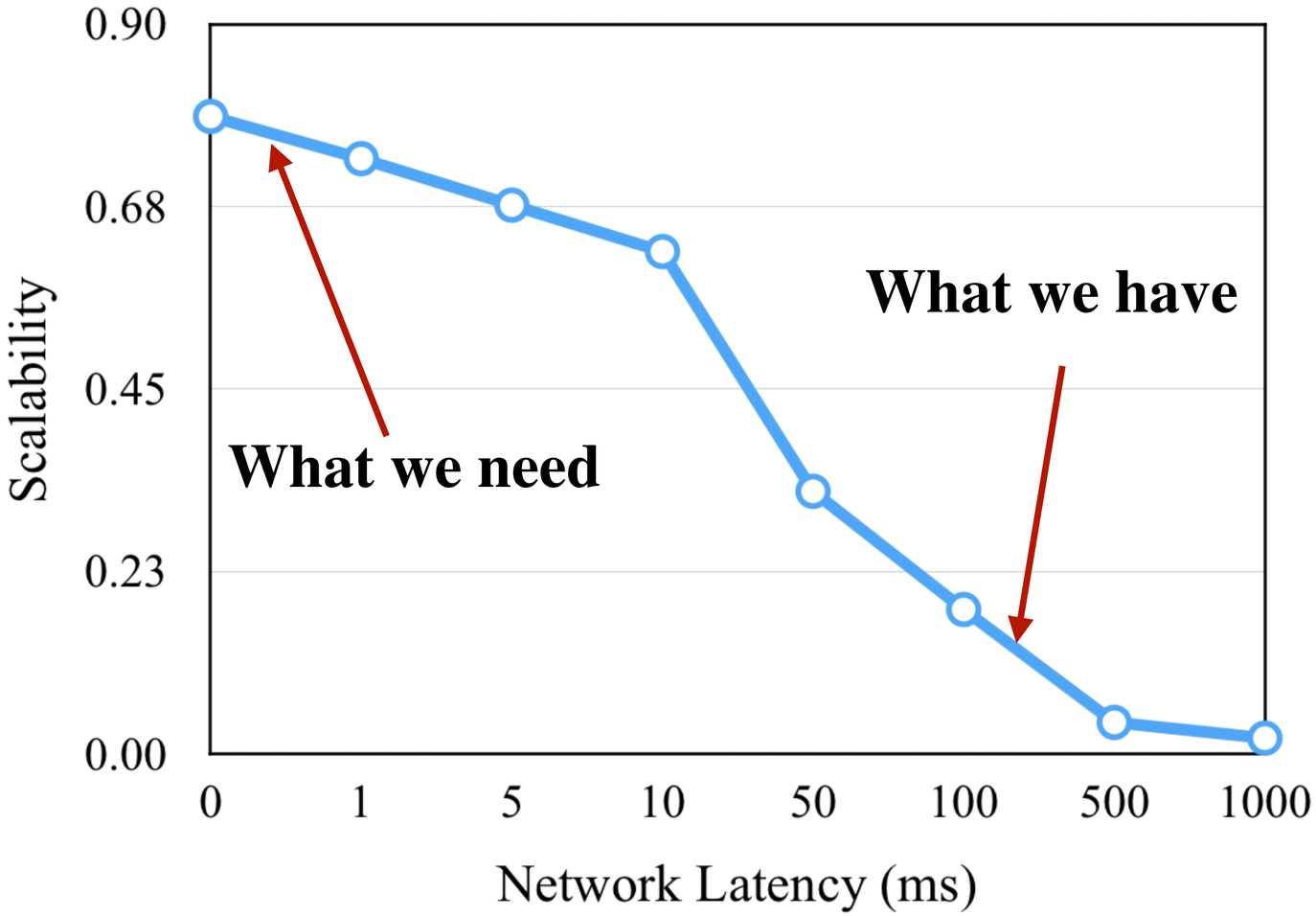


(a) Synchronous SGD synchronizes current gradients before next round of computation, thus not scalable when the latency increases.



(b) Asynchronous SGD has no synchronization barrier but each worker still needs to wait a long time when pulling latest weights.

Scalability degrades quickly with latency



Delayed Synchronous SGD

Algorithm 1 Delayed Synchronous Stochastic gradients Descent (DS-SGD)

Initialize each worker with $w_{0,j} = w_0$ for $j \in [1, J]$.
for $n = 0, 1, \cdots$ **do**
  On local worker j**:**
  Sample stochastic gradients $g_{(n,j)}$ at $w_{(n,j)}$
  Send $g_{(n,j)}$ to all other workers.
  if $n < d$ **then**
    Update with standard SGD rule locally:

$$w_{(n+1,j)} = w_{(n,j)} - \eta g_{(n,j)}.$$

  else
    Update with variance reduction rule:

$$w_{(n+1,j)} = w_{(n,j)} - \eta(g_{(n,j)} - g_{(n-d,j)} + \overline{g_{(n-d)}})$$

    where

$$\overline{g_{(n-d)}} = \frac{1}{J}\sum_{j=1}^{J} g_{(n-d,j)}$$

  end if
end for
Return $\overline{w}_n = \frac{1}{J}\sum_{j=1}^{J} w_{(n,j)}$

Keypoint: synchronize **stale** gradients.

Synchronous SGD

$$w_{(i+1)} = w_{(i)} - \overline{\nabla w_{(i)}}$$

Delayed Synchronous SGD (d=4)

$$w_{(i+1)} = w_{(i)} - \eta(\nabla w_{(i,j)} - \nabla w_{(i-d,j)} + \overline{\nabla w_{(i-d)}})$$

Synchronous SGD

$$\overline{w_{(i+1)}} = w_{(0)} - \eta \sum_{i=0}^{n} \overline{\nabla w_{(i)}}$$

$$\overline{w_{(i+1)}} = w_{(0)} - \eta \sum_{i=0}^{n-d} \overline{\nabla w_{(i)}} - \eta \sum_{i=n-d+1}^{n} \nabla w_{(i,j)}$$

Only differs in a small range

Delayed Synchronous SGD

DSSGD guarantees the convergence

- Assumption 1: the loss function $F(w; x, y)$ is L-Lipchitz smooth

$$\nabla f_j(x) - \nabla f_j(y)\,|| \leq L\,||x - y\,||\,. \quad \forall x, y \in \mathbb{R}^d$$

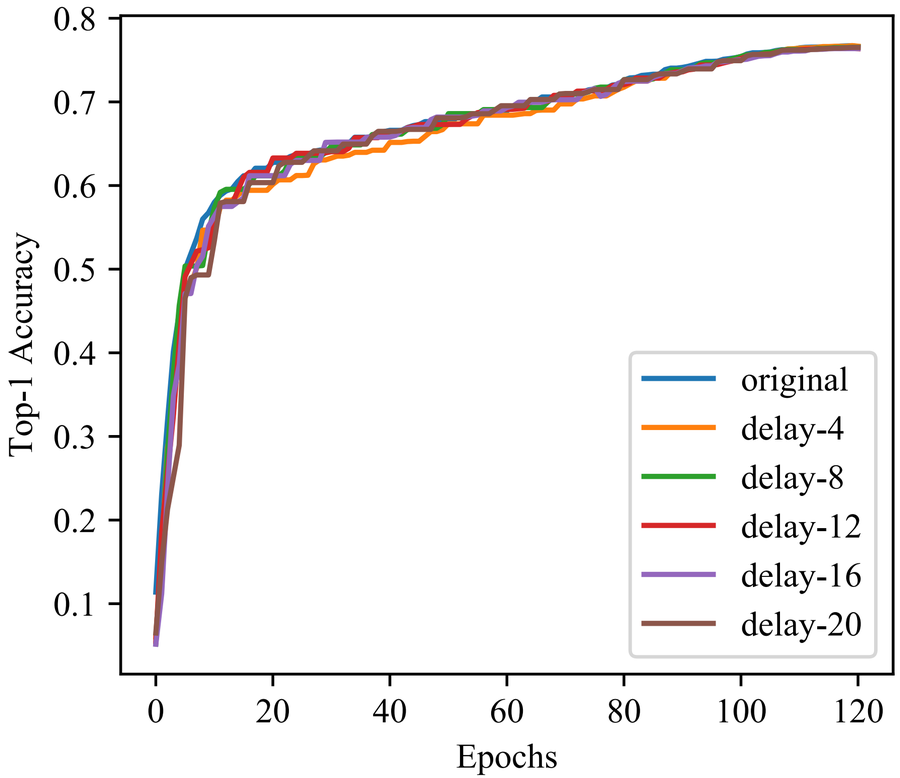
- Assumption 2: Bounded gradients and variances

$$\mathbb{E}_{\zeta_j}\,||\,\nabla F_j(w; \zeta_i)\,||^2 \leq G^2, \forall w, \forall j,\ \mathbb{E}_{\zeta_j}\,||\,\nabla F_j(w; \zeta_j) - \nabla f_j(w)\,||^2 \leq \sigma^2, \forall w, \forall j\,.$$

The convergence rate of DSSGD is $O(\frac{\Delta + \sigma^2}{\sqrt{JN}} + \frac{Jd^2}{N})$ is no slower than SGD $O(\frac{\Delta + \sigma^2}{\sqrt{JN}})$

When $d < O(N^{\frac{1}{4}} J^{-\frac{3}{4}})$ [the first term dominates.

DSSGD yields the same accuracy



Latency issue:

Forward / backward -> 300ms

20 delay -> tolerate 6s latency.

Remaining issues:

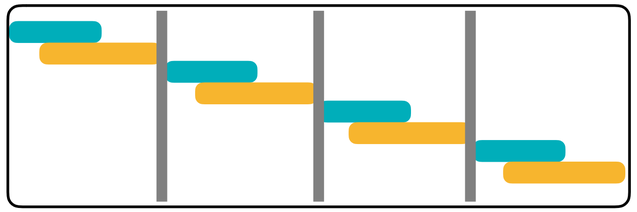
Bandwidth / Congestions

Up to *t* gradients are transmitting simultaneously

DSSGD tolerates high latency

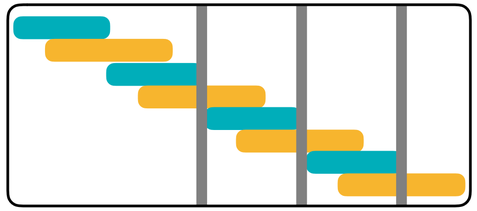
Naive Distributed SGD

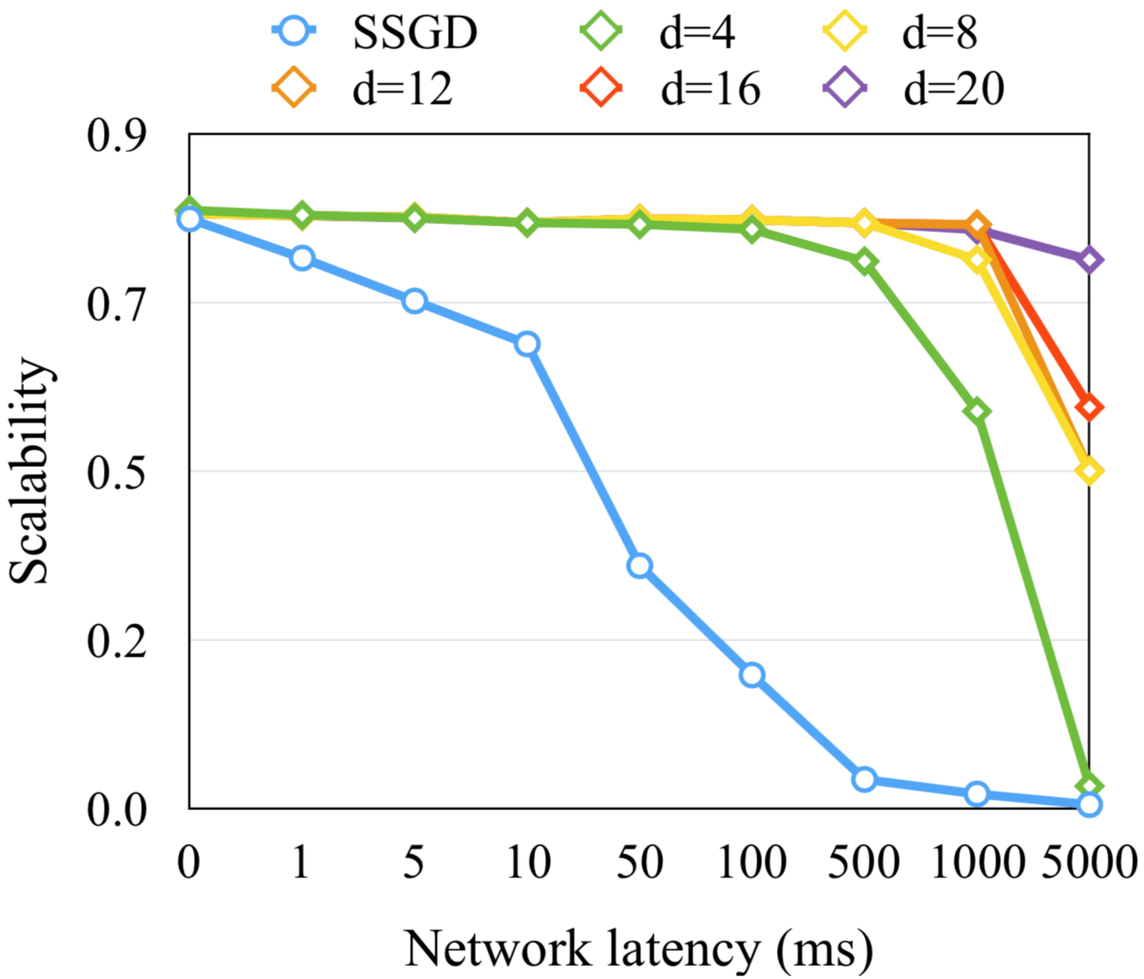
Wait time: $T_{\text{communicate}} - T_{\text{overlap}}$



Delayed Update

$$\max(0, T_{\text{communicate}} - T_{\text{overlap}} - t \times T_{\text{compute}})$$



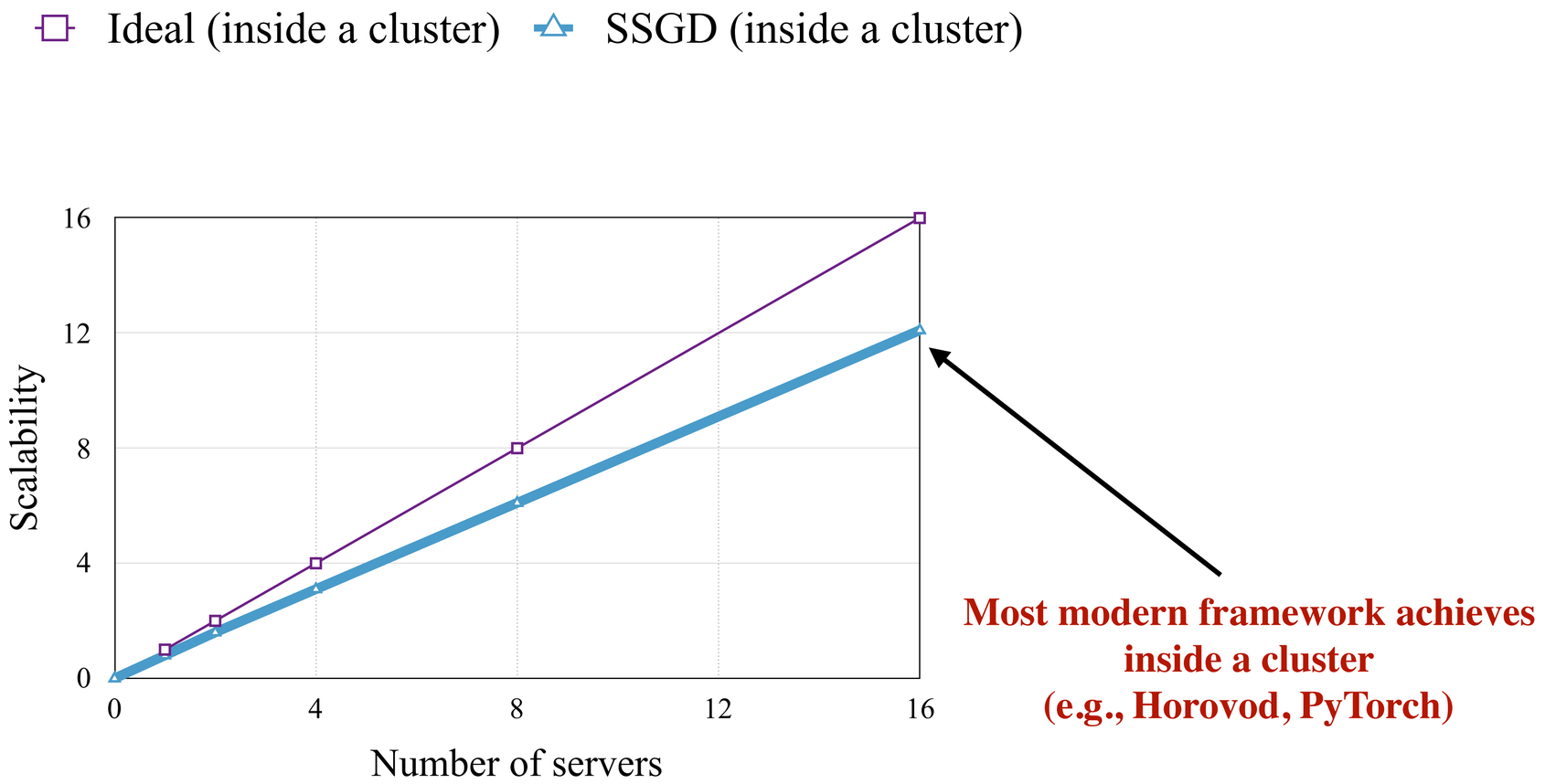


Distributed Training Across the World

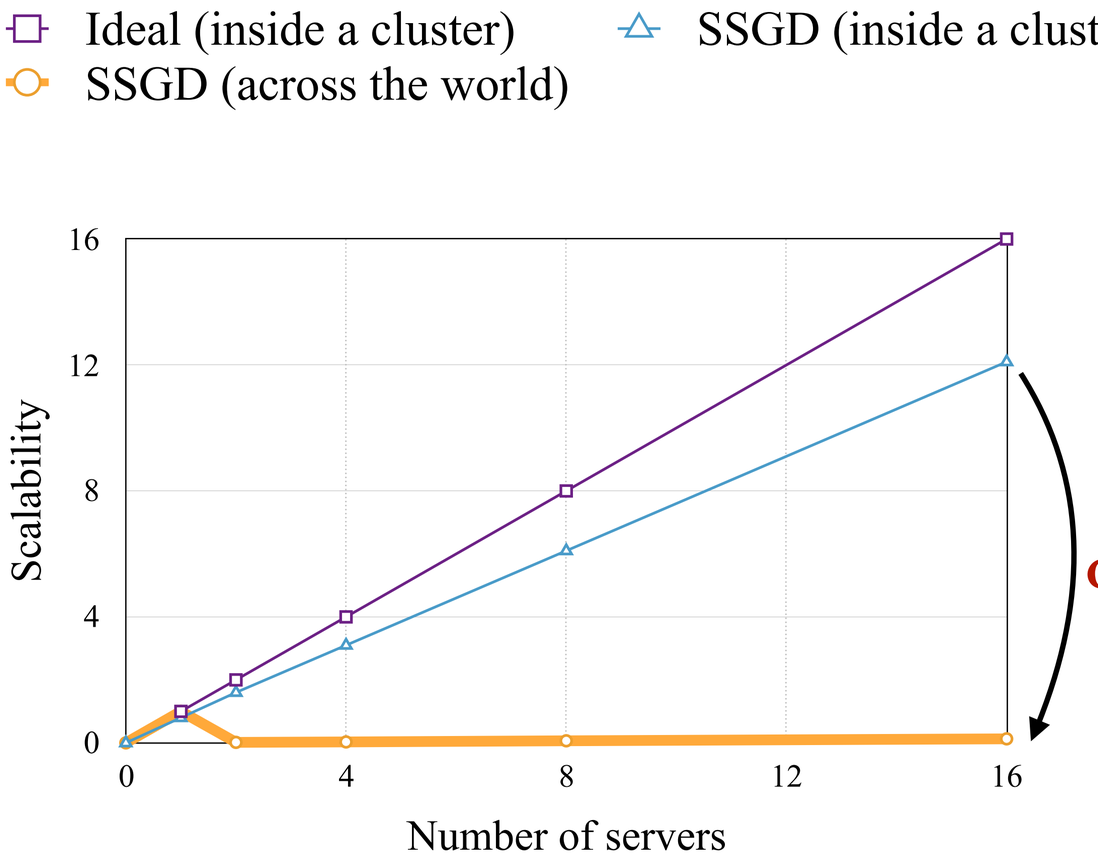
Experiment environments

- p3.16x Instances on AWS (8 x V100)
- 8 instances at 4 different geographical locations
  - Ohio, Oregon, London, Tokyo
  - Latency: ~480ms (based on ring all reduce)
- The scalability of naive training: 0.008
  - Training on 100 machines is slower than single one.

Scalability of SSGD when inside a cluster

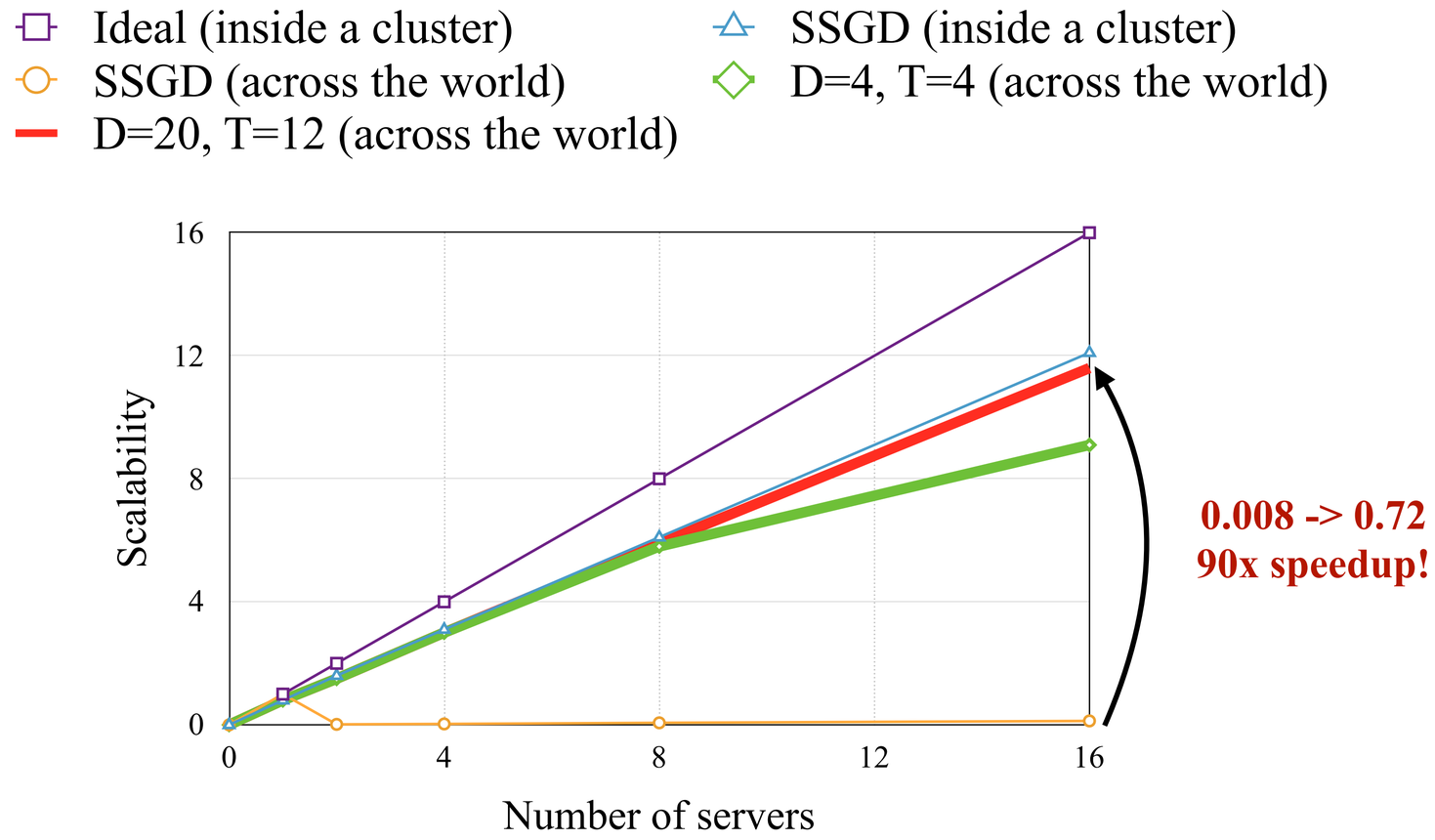


Scalability of SSGD when across the world



0.8 -> 0.008
Conventional algorithms fails to scale under high latency

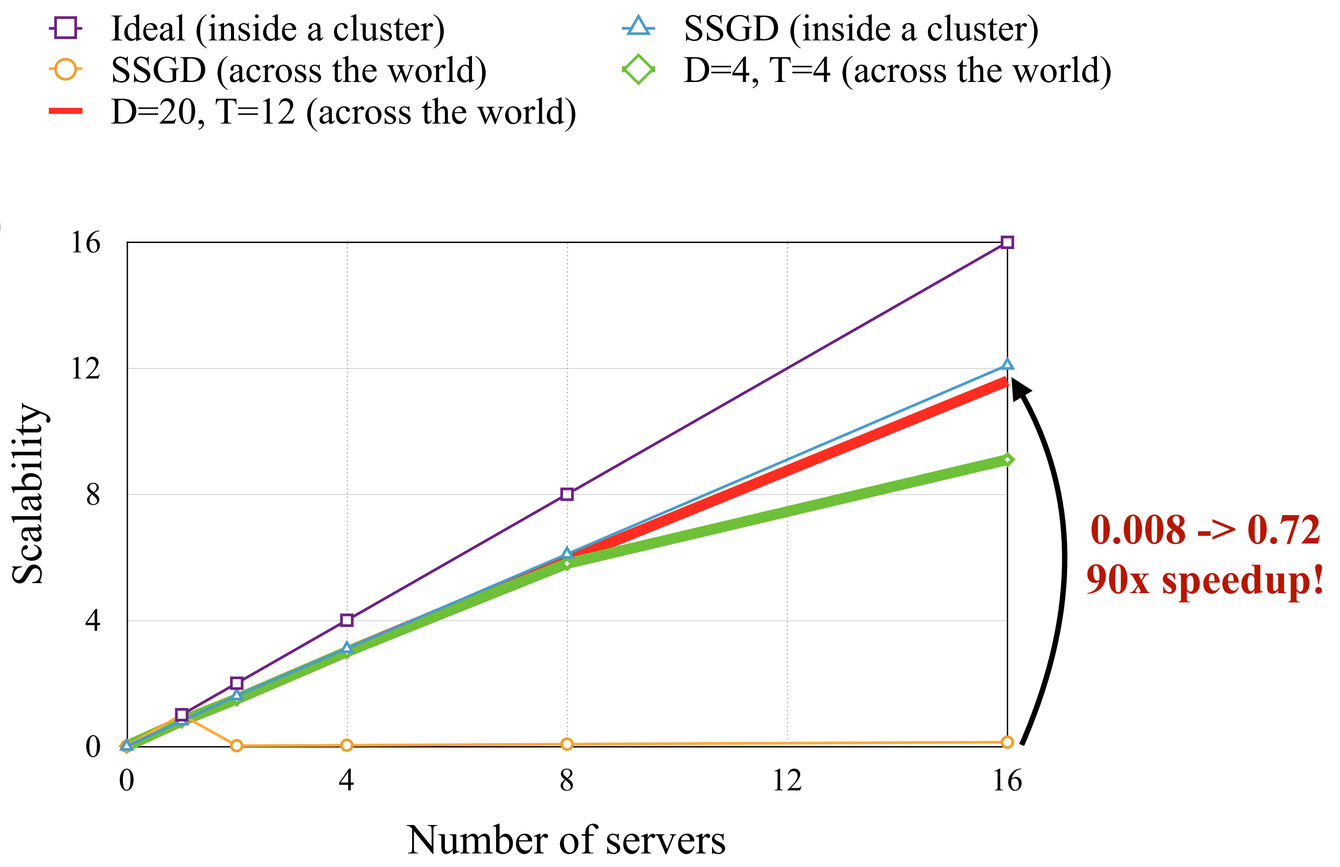
Scalability of SSGD when across the world



Scalability of DTS when across the world

- Delayed update (tolerate latency)
- Temporally sparse update (amortize latency)
- Gradient compression[1] (reduce transferred data)

	Top-1%
Original SGD	76.63
D=4, T=4, C=1%	76.15
D=8, T=8, C=1%	76.32
D=12, T=8, C=1%	76.18
D=20, T=12, C=1%	75.81



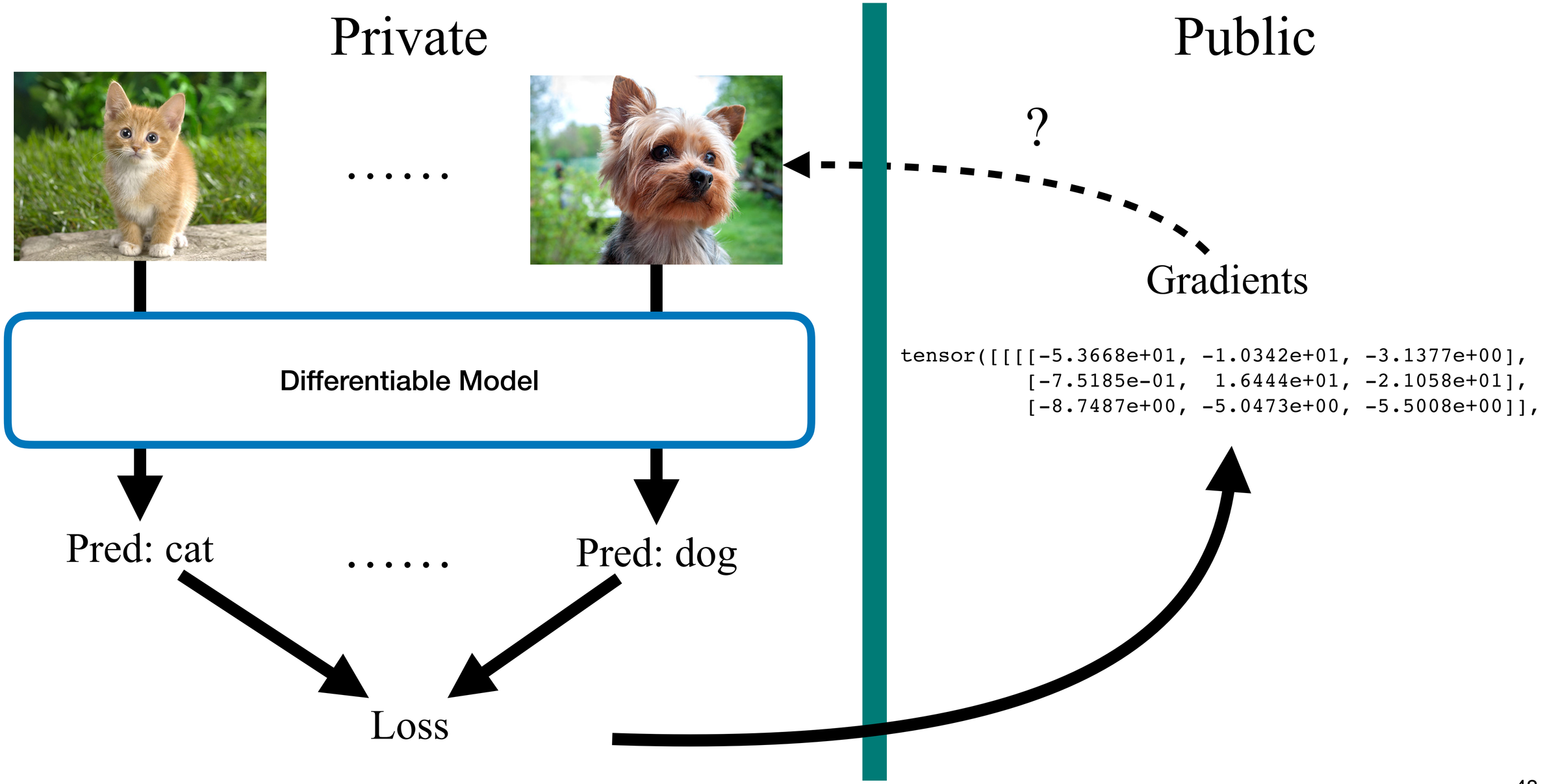
[1] *Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training.* Yujun Lin, Song Han, Yu Wang, Bill Dally. ICLR 18

Deep Leakage from Gradients

Ligeng Zhu, Zhijian Liu, Song Han

NeurIPS’19

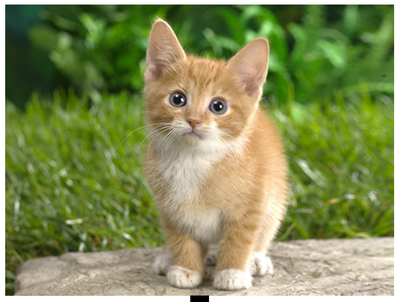
Is gradient safe to share?



Gradient is not safe to share!

Private

Public



.....

Differentiable Model

Pred: cat

.....

Pred: dog

Loss

Gradients

```
tensor([[[[-5.3668e+01, -1.0342e+01, -3.1377e+00],
          [-7.5185e-01,  1.6444e+01, -2.1058e+01],
          [-8.7487e+00, -5.0473e+00, -5.5008e+00]],
```

Conventional Shallow Leakage

Gradients

```
tensor([[[[-5.3668e+01, -1.0342e+01, -3.1377e+00],
          [-7.5185e-01,  1.6444e+01, -2.1058e+01],
          [-8.7487e+00, -5.0473e+00, -5.5008e+00]],
```

Membership Inference

Whether a record is used in the batch.

Property Inference

Whether a sample with certain property is in the batch.

But, can we obtain **the original training data**?

[1] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov. *Exploiting unintended feature leakage in collaborative learning*.

[2] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. *Membership inference attacks against machine learning models*.

Deep Leakage from Gradients

Normal Training:
forward-backward, update **model weights**

Differentiable Model

Gradients

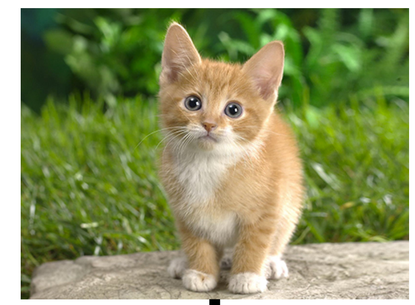
Pred: cat

Loss

Deep Leakage from Gradients

Normal Training:
forward-backward, update **model weights**

Normal Training:
forward-backward, update **the inputs**



Differentiable Model

Differentiable Model

MSE

Gradients

Gradients

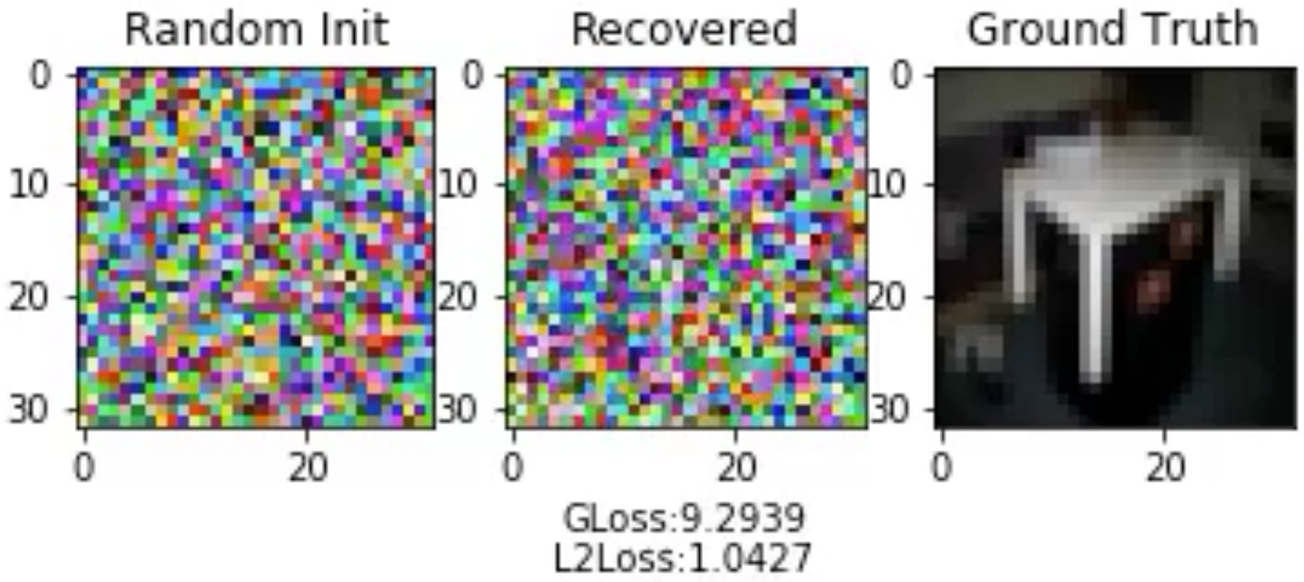
Pred: cat

Pred: [random]

Loss

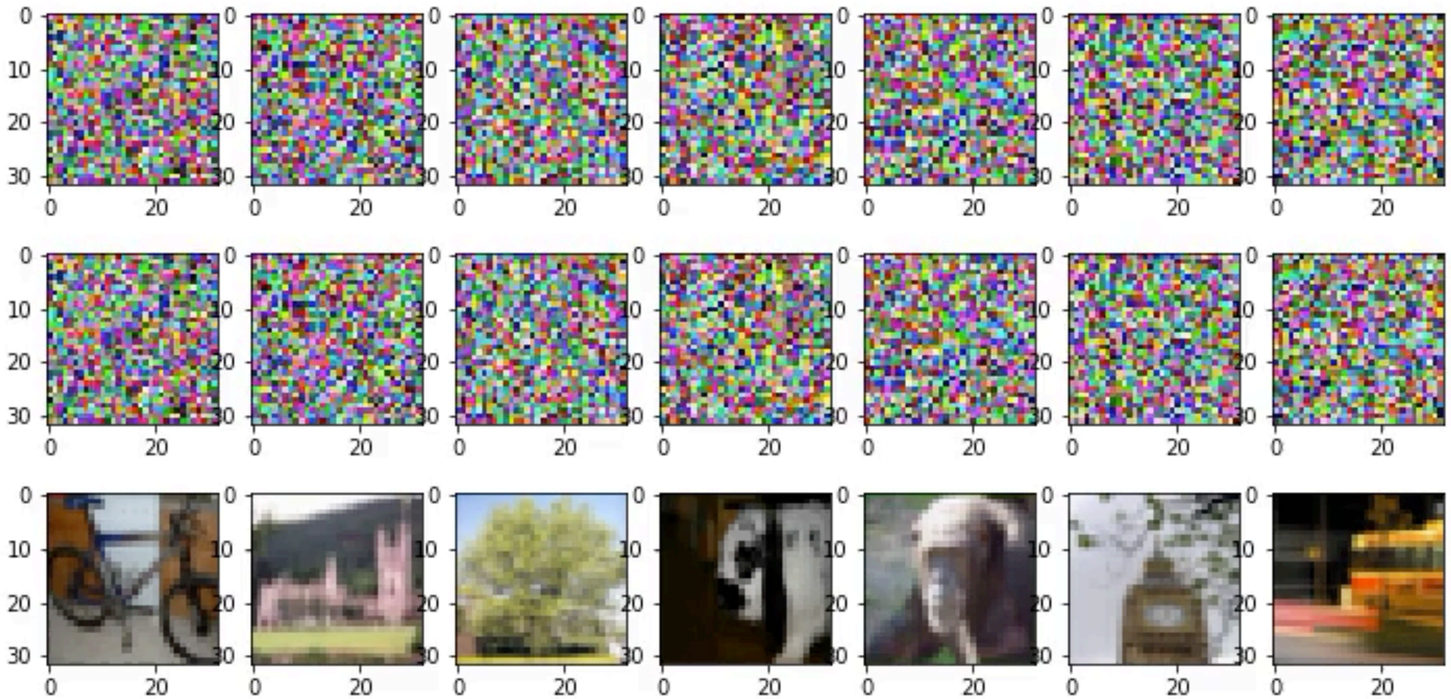
Loss

Recovering Visualization (bs=1)



Model: ResNet18 Dataset: CIFAR100 Optimizer: LBFGS 300 iters

Recovering Visualization (bs=8)



Model: ResNet18 Dataset: CIFAR100 Optimizer: LBFGS 300 iters

Experiment on Bert

- For discrete word, the embeddings are taken as input.

Random Init: 【a2 furnished angel compromise springsteen ##lice ##ulated sal ##n ##ory moshe unitary ##tori commercial】

DLG: 【. who is jim henson ? . jim henson was a puppet ##eer .】

GT: 【[CLS] Who was Jim Henson ? [SEP] Jim Henson was a puppeteer [SEP]】

Experiment on Bert

Iters=0: tilting fill given **less word **itude fine **nton overheard living vegas **vac **vation *f forte **dis cerambycidae ellison **don yards marne **kali

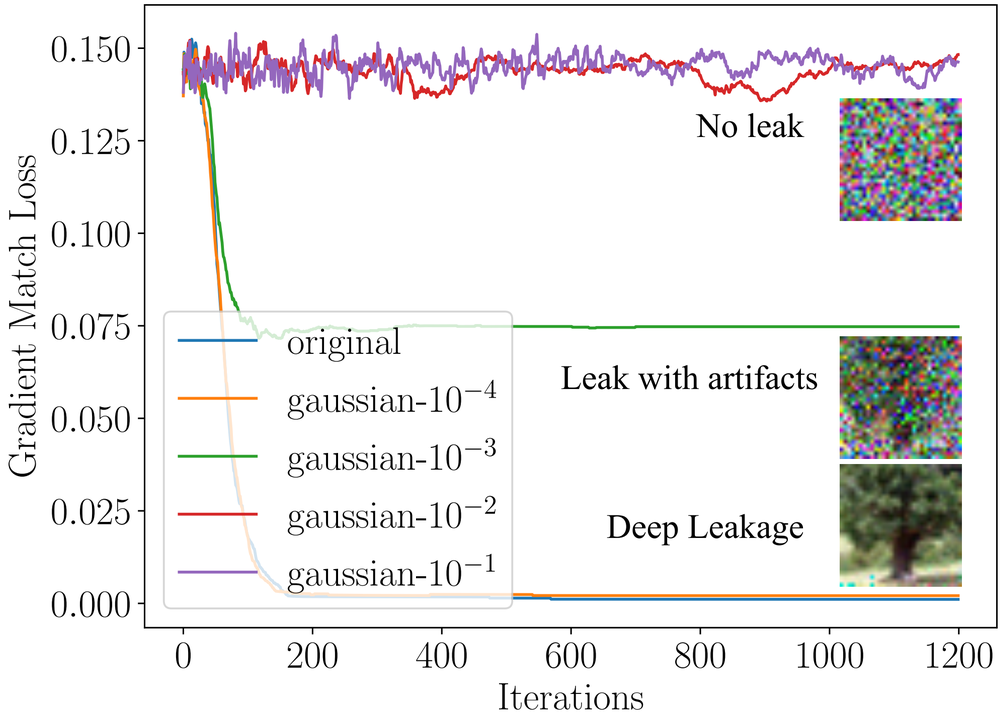
Iters=10: tilting fill given **less full solicitor other ligue shrill living vegas rider treatment carry played sculptures lifelong ellison net yards marne **kali

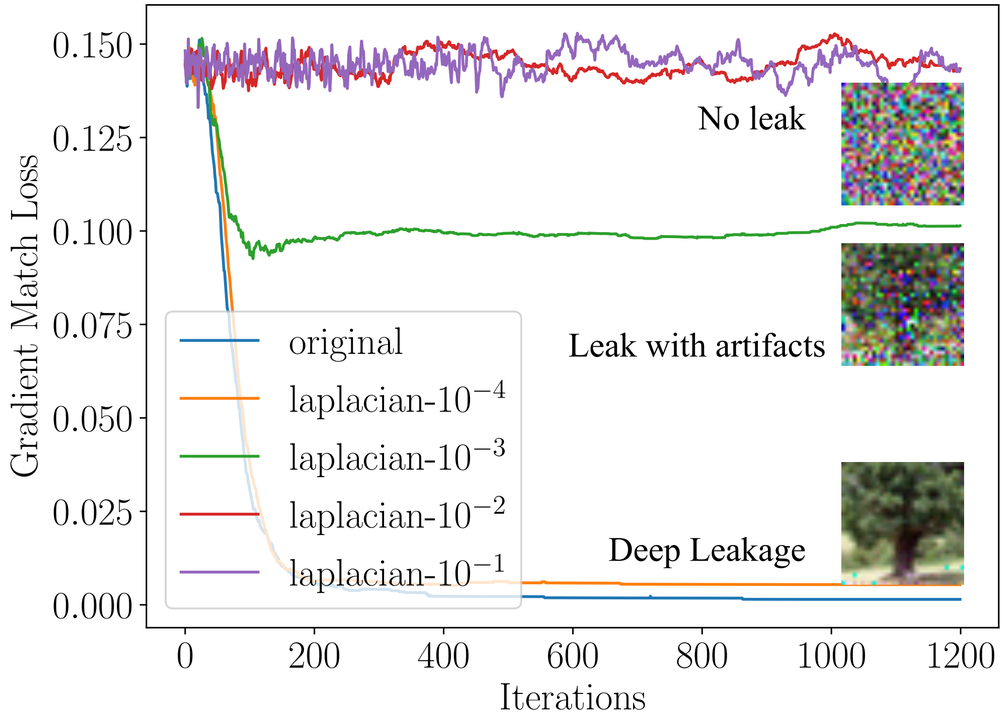
Iters=20: registration , volunteer applications , at student travel application open the ; week of played ; child care will be glare .

Iters=30: registration, volunteer applications, and student travel application open the first week of september . child care will be available

Original text: Registration, volunteer applications, and student travel application open the first week of September. Child care will be available.

Defense Strategy





Defense Strategy