Distributed Training Across the World

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

Neurips 19 MLSys

Why Distributed Training?

- Model sizes
  - AlexNet (7 layers) -> VGG (16 layers) -> ResNet (152 layers)
- Dataset sizes
  - CIFAR (50k) -> ImageNet (1.2M) -> Google JFG (300M)

Even with modern GPU, says eight-V100 server,
it still takes ***days*** and even ***weeks*** to train a model.

What is Distributed Training?

Conventional SGD

1. Sample (X, y) from dataset
2. Forward to compute loss.
3. Backward to compute gradients.
4. Apply gradients to update model.

Distributed SGD

1. Sample (X, y) from dataset
2. Forward to compute loss.
3. Backward to compute gradients.
4. **Synchronize gradients**
5. Apply gradients to update model.

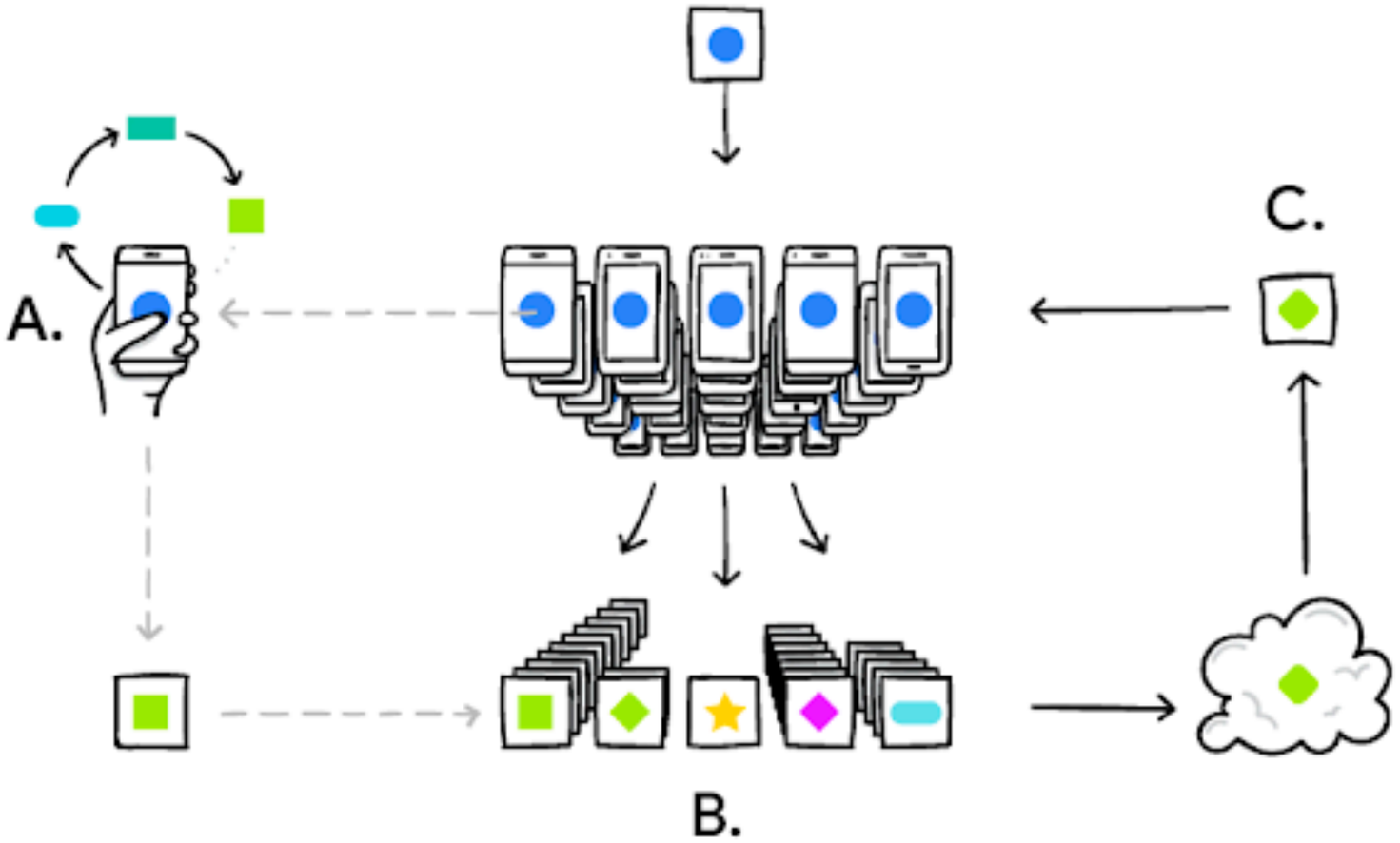
Why Learning across Geographical Locations?

It's not who has the best algorithm that wins.
It’s who has the most data.

—Andrew Ng

However, it is always difficult to collect data (even illegal sometimes).

Why Learning across Geographical Locations?



Collaborative / Federated Learning
Data never leaves local device.

Communication Limits Scalability

Latency

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

Shanghai - Boston:

- **10 Mb/s** with a high variance.
- 78ms (ideal) / >**700ms** (real world)

$$11{,}725\text{km} \times 2/(3 \times 10^8\text{m/s}) = 78.16\text{ms}$$

Bandwidth

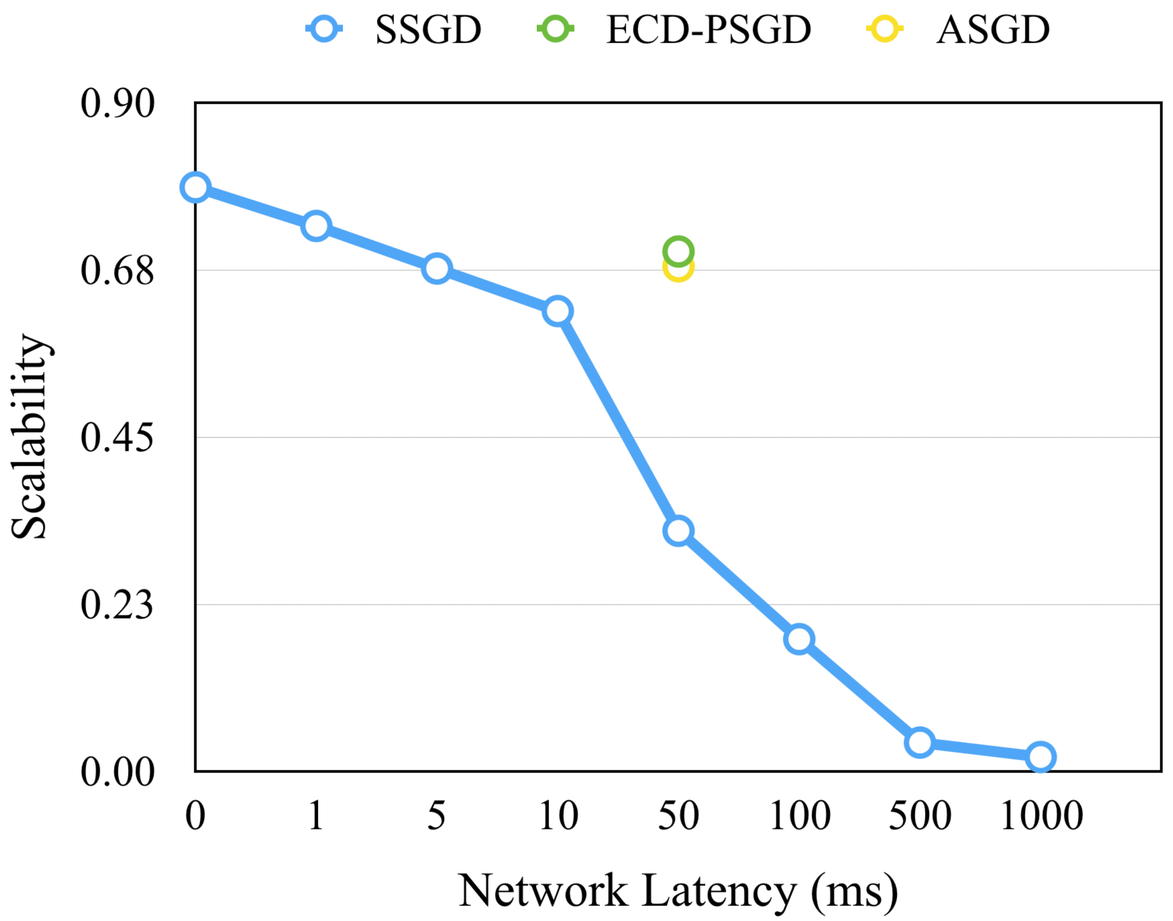
- Infinity band: up to 100 Gb/s
- Mobile network: 100 Mb/s (4G), 1Gb/s (5G)

What we need

- Bandwidth as high **900 Mb/s.**
- Latency as low as **1ms**.

Bandwidth is easy to increase.
Latency is hard to improve : (

Latency is critical



Distributed Training Across the World

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

Neurips 19 MLSys

Delayed Update: sync stale gradients

Conventional Distributed SGD at *step i*

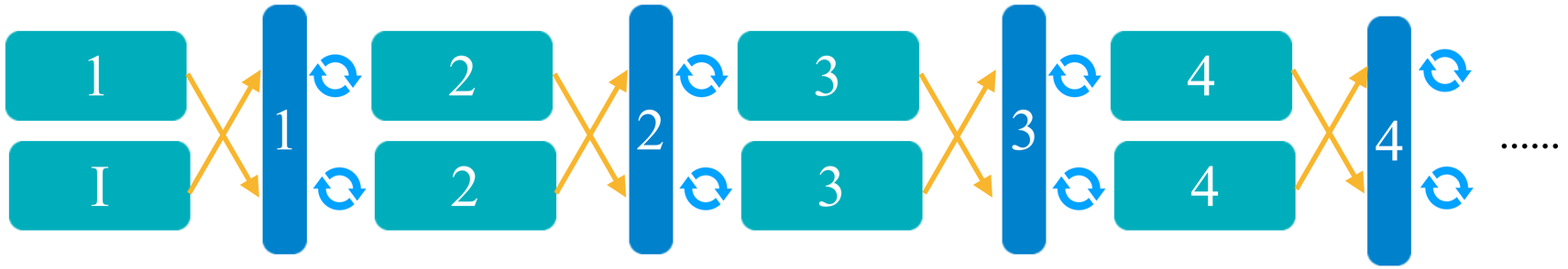
1. Sample (X, y) from dataset
2. Forward to compute loss.
3. Backward to compute gradients.
4. Synchronize *step i*'s gradients
5. Apply gradients to update model.

Delayed Update at *step i*

1. Sample (X, y) from dataset
2. Forward to compute loss.
3. Backward to compute gradients.
4. Synchronize *step (i - t)*'s gradients
5. Apply gradients to update model.

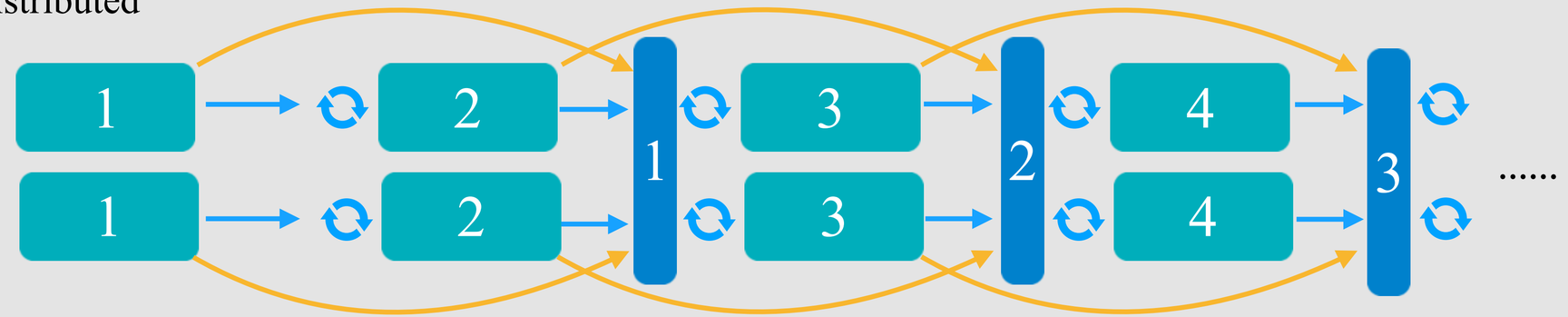
Delayed Update: put off the sync barrier

Normal Distributed



Gradients from **time stamp i** are synced before (i+1)th update.

Delayed Distributed



Gradients from **time stamp (i-t)** are synced before (i+1)^th update.

Preserve Accuracy by Compensation

Sync ***gradients (i-t)*** are synced at *step i* update.

For example, if t = 2

Vanilla SGD

$$w_n = w_0 - \gamma \sum_{i=0}^{n-1} \overline{v_i}$$

$$\overline{w_3} = w_0 - \gamma(\overline{v_0} + \overline{v_1} + \overline{v_2})$$

Delayed Update

$$w_3 = w_0 - \gamma(v_0 + v_1 + v_2) - \gamma(\overline{v_0} - v_0) = w_0 - \gamma(\overline{v_0} + v_1 + v_2)$$

Preserve Accuracy by Compensation

$$w_{n,j} = w_0 + \underbrace{\sum_{i=0}^{n-1-t} \overline{\Delta w_i}}_{\text{Same as normal distributed training}} + \underbrace{\sum_{i=n-t}^{n-1} \Delta w_{i,j}}_{\text{Difference caused by local update}}$$

Global information

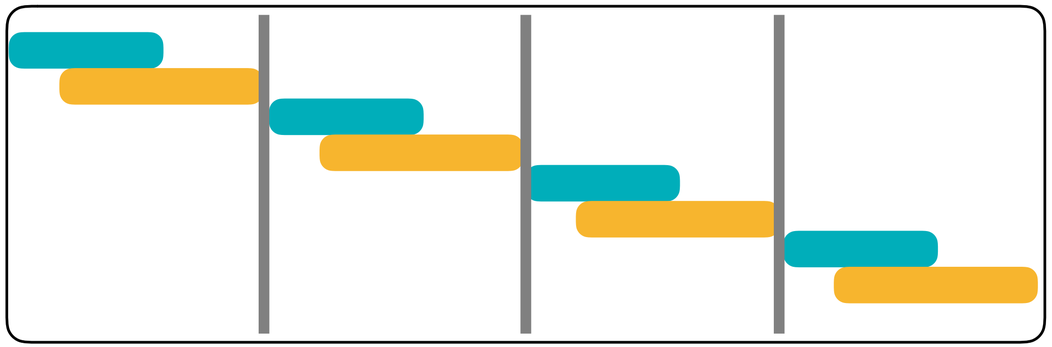
Local gradients

Theoretical Convergence: $O(\frac{1}{\sqrt{NJ}}) + O\frac{t^2 J}{N}$

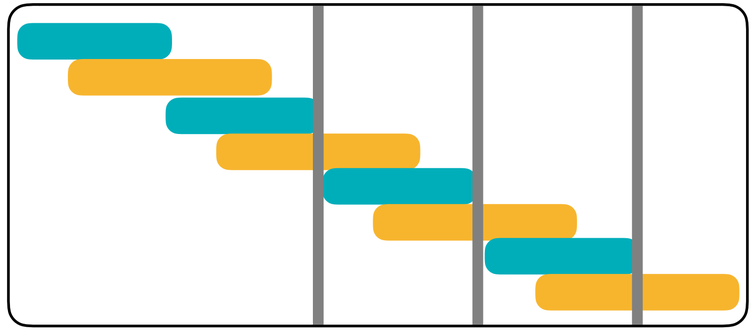
Convergence of SGD: $O(\frac{1}{\sqrt{NJ}})$

Delayed Update: put off the sync barrier

Naive Distributed SGD



Delayed Update

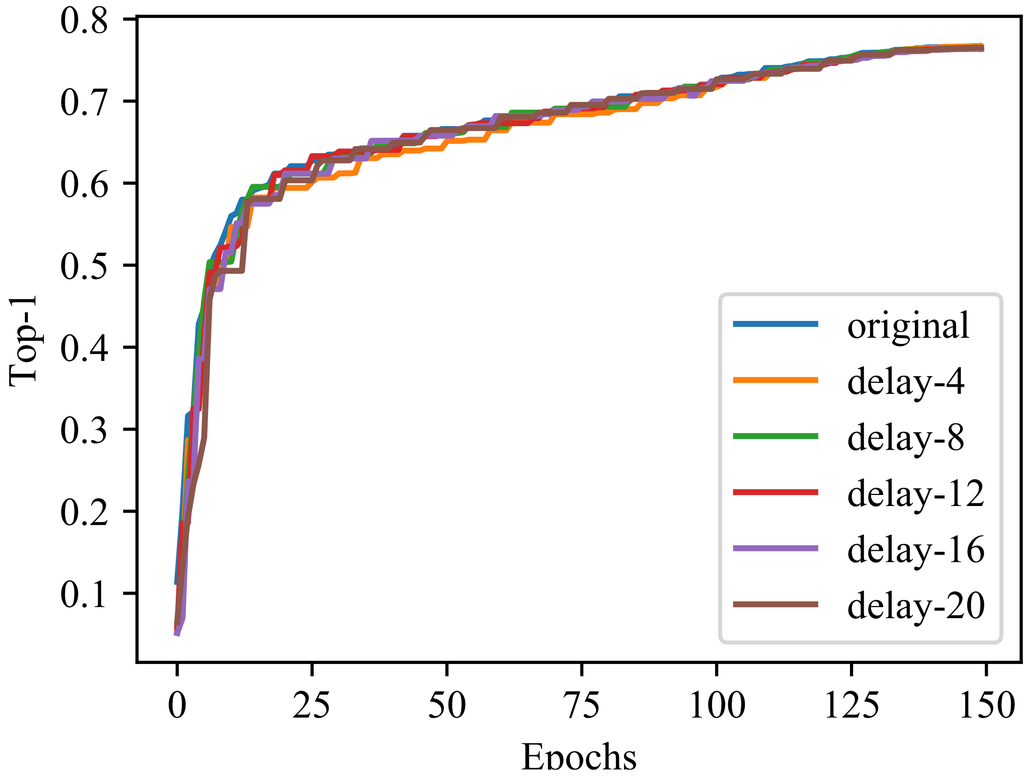


Wait time: $T_{\text{communicate}} - T_{\text{overlap}}$ $\max(0, T_{\text{communicate}} - T_{\text{overlap}} - t \times T_{\text{compute}})$

Delayed update speeds up training

and also tolerate **high latency**!

Delayed Update: put off the sync barrier



Accuracy: promised

Latency issue: solved.

20 delay -> tolerate 6s latency.

Remaining issues:

Bandwidth / Congestions

Temporally Sparse Update: periodically sync

Conventional Distributed SGD at *step i*

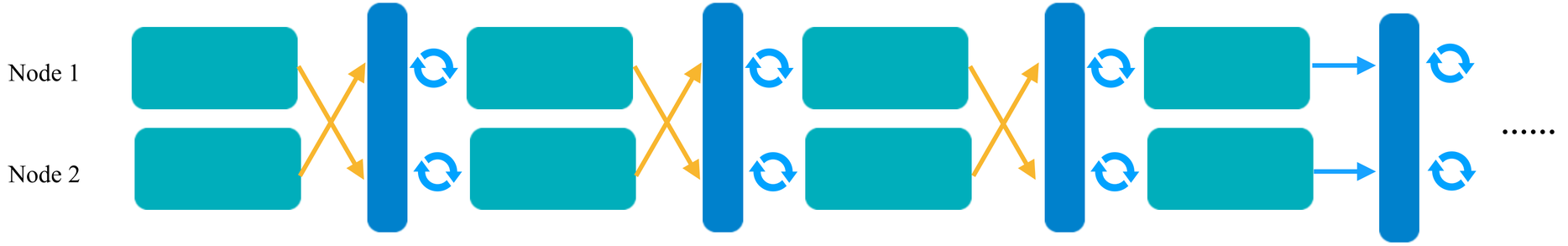
1. Sample (X, y) from dataset
2. Forward to compute loss.
3. Backward to compute gradients.
4. Synchronize *step i*'s gradients
5. Apply gradients to update model.

Temporally Update at *step i*

1. Sample (X, y) from dataset
2. Forward to compute loss.
3. Backward to compute gradients.
4. Synchronize *step [i - d, i)*'s gradients if *i mod d == 0*.
5. Apply gradients to update model.

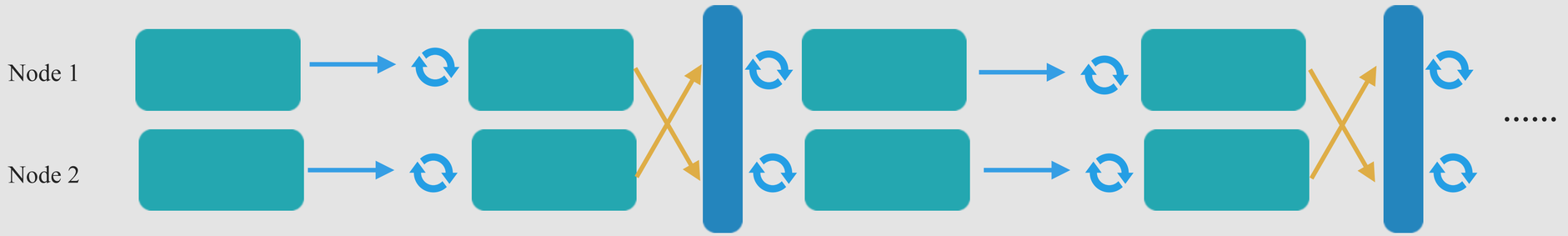
Temporally Sparse Update: reduce sync frequency

Normal Distributed



Gradients from **time stamp i** are synced before **(i+1)^th update.**

Temporal Sparse



Gradients from **time stamp (i-p, i]** are synced before **(i+1)^th update.**

Temporally Sparse Update: reduce sync frequency

Gradients from **time stamp (i-p, i]** are synced before **(i+1)^th update.**
Similarly, we compute the compensation.

$$(v_{n-p+1,j}, v_{n-p+2,j}, ..., v_{n,j}) \leftarrow (\overline{v_{n-p+1}}, \overline{v_{n-p+2}}, ..., \overline{v_n})$$

Vanilla SGD

$$w'_n = w_n + (\sum_{i=n-p+1}^{n-1} \overline{v_i} - \sum_{i=n-p}^{n-1} v_i)$$

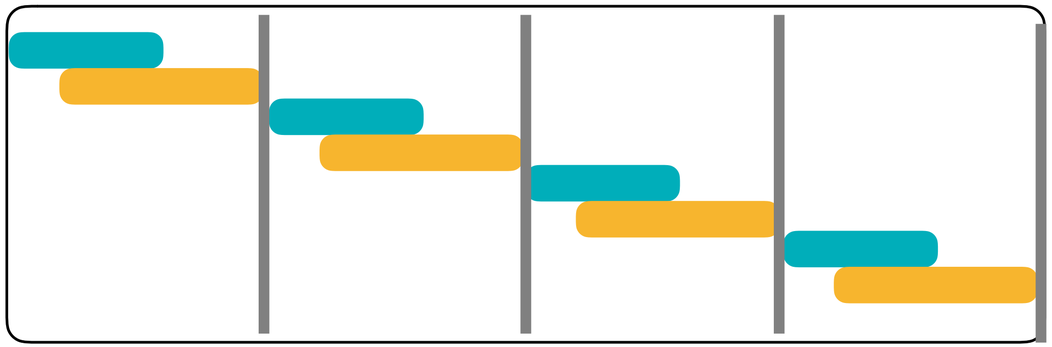
Momentum SGD

$$u'_n = u_n + (\sum_{i=n-p+1}^{n} m^{n-i}\overline{v_i} - \sum_{i=n-p+1}^{n} m^{n-i} v_i)$$

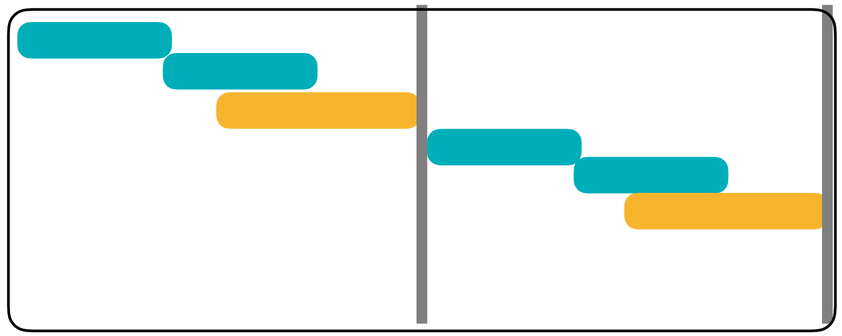
$$w'_n = w_n + (\sum_{i=n-p}^{n} \sum_{j=n-p}^{i} m^{i-j}\overline{v_j} - \sum_{i=n-p}^{n} \sum_{j=n-p}^{i} m^{i-j} v_j)$$

Temporally Sparse Update: reduce sync frequency

Naive Distributed SGD



Temporally Sparse Update



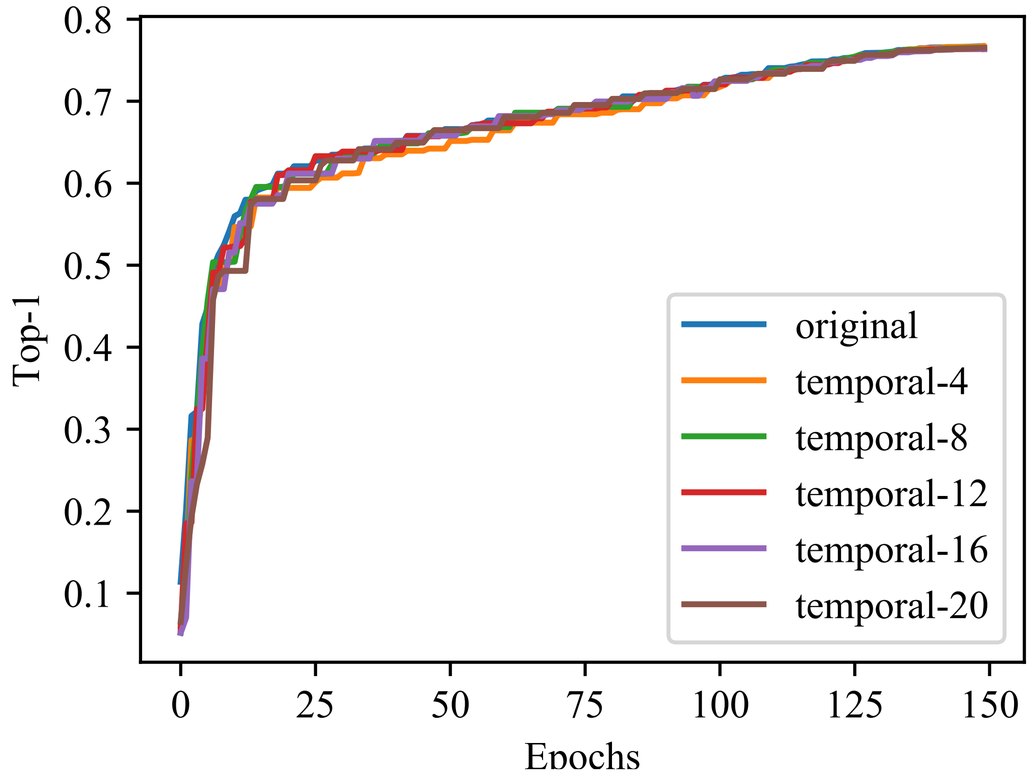
Wait time: $T_{\text{communicate}} - T_{\text{overlap}}$ $\quad \frac{T_{\text{communicate}} - T_{\text{overlap}}}{P}$

Bandwidth: $\frac{||W||}{T_{\text{training}}}$ $\quad \frac{||W||}{P \times T_{\text{training}}}$

Temporal Sparsity alleviates **congestion**

and improves **amortized latency / bandwidth**.

Temporally Sparse Update: reduce sync frequency

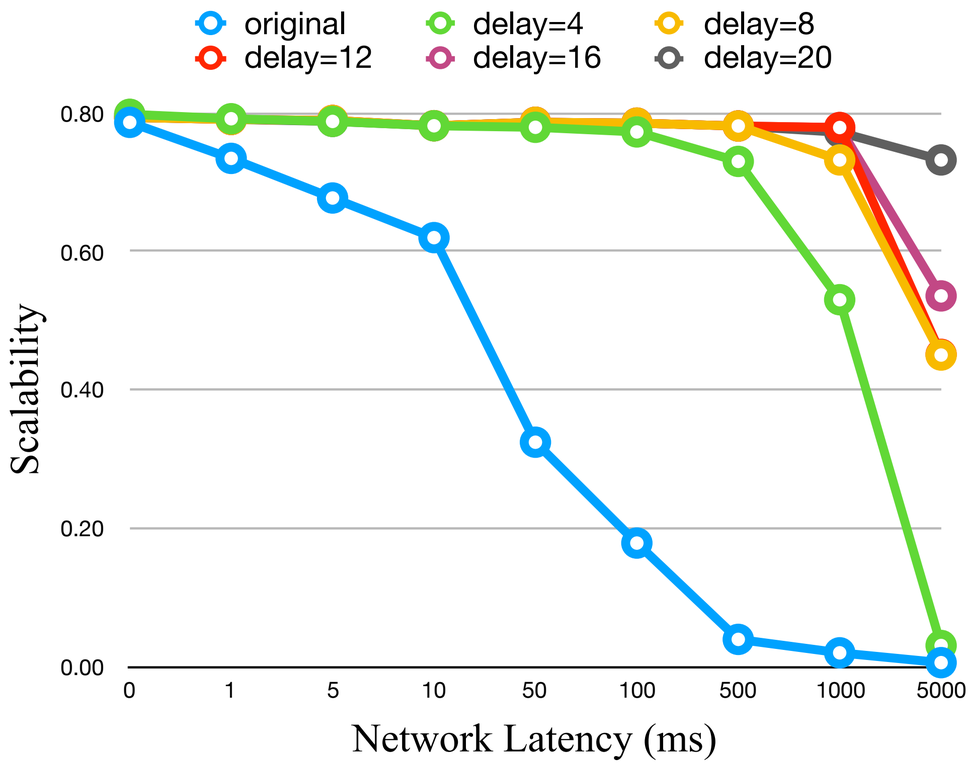


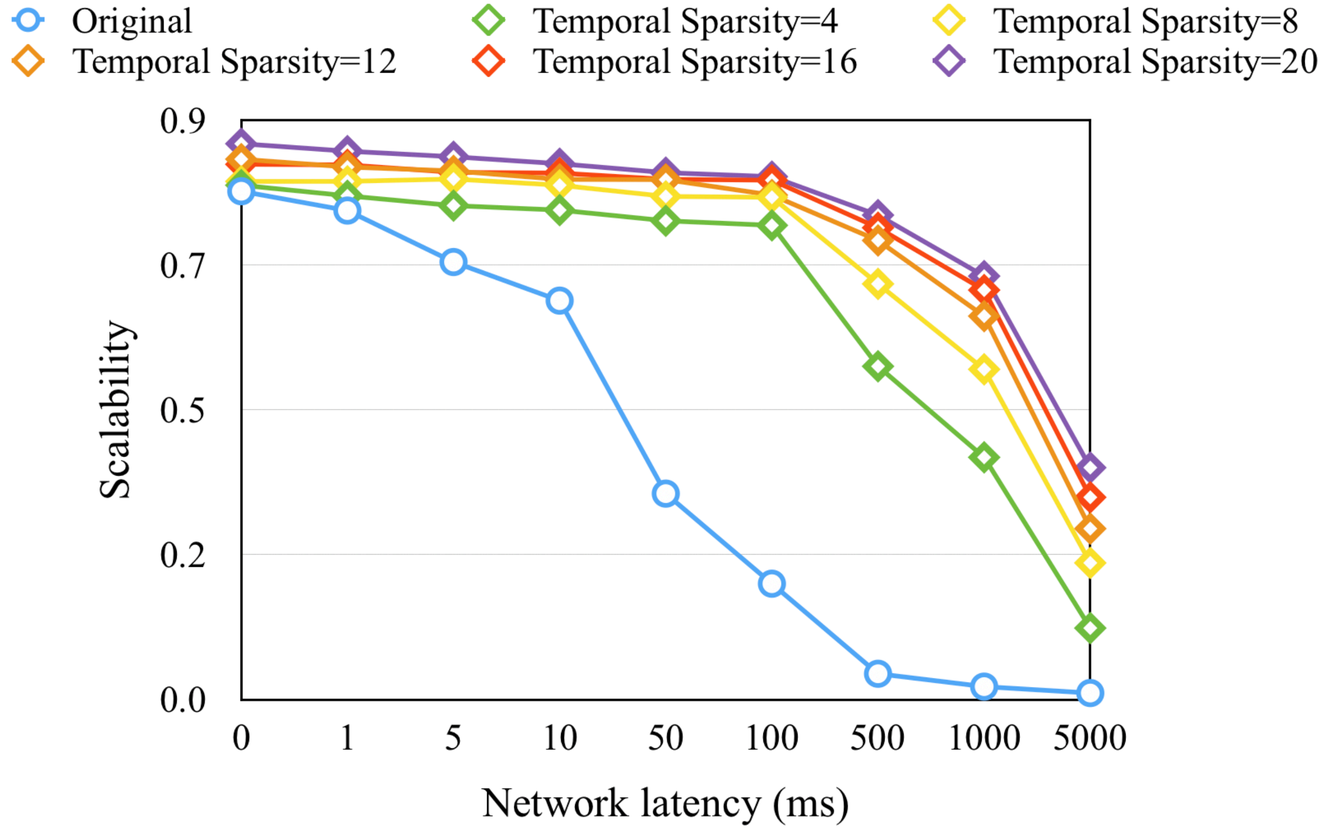
Congestion: solved

Bandwidth: (900 / P)Mb/s

However, even for temporal_sparse=20, 45Mb/s is not always achievable, e.g., cross continent connection.

Results


original
delay=4
delay=8
delay=12
delay=16
delay=20
0.80
0.60
0.40
0.20
0.00
Scalability
0
1
5
10
50
100
500
1000
5000
Network Latency (ms)


Original
Temporal Sparsity=4
Temporal Sparsity=8
Temporal Sparsity=12
Temporal Sparsity=16
Temporal Sparsity=20
0.9
0.7
0.5
0.2
0.0
Scalability
0
1
5
10
50
100
500
1000
5000
Network latency (ms)

Results

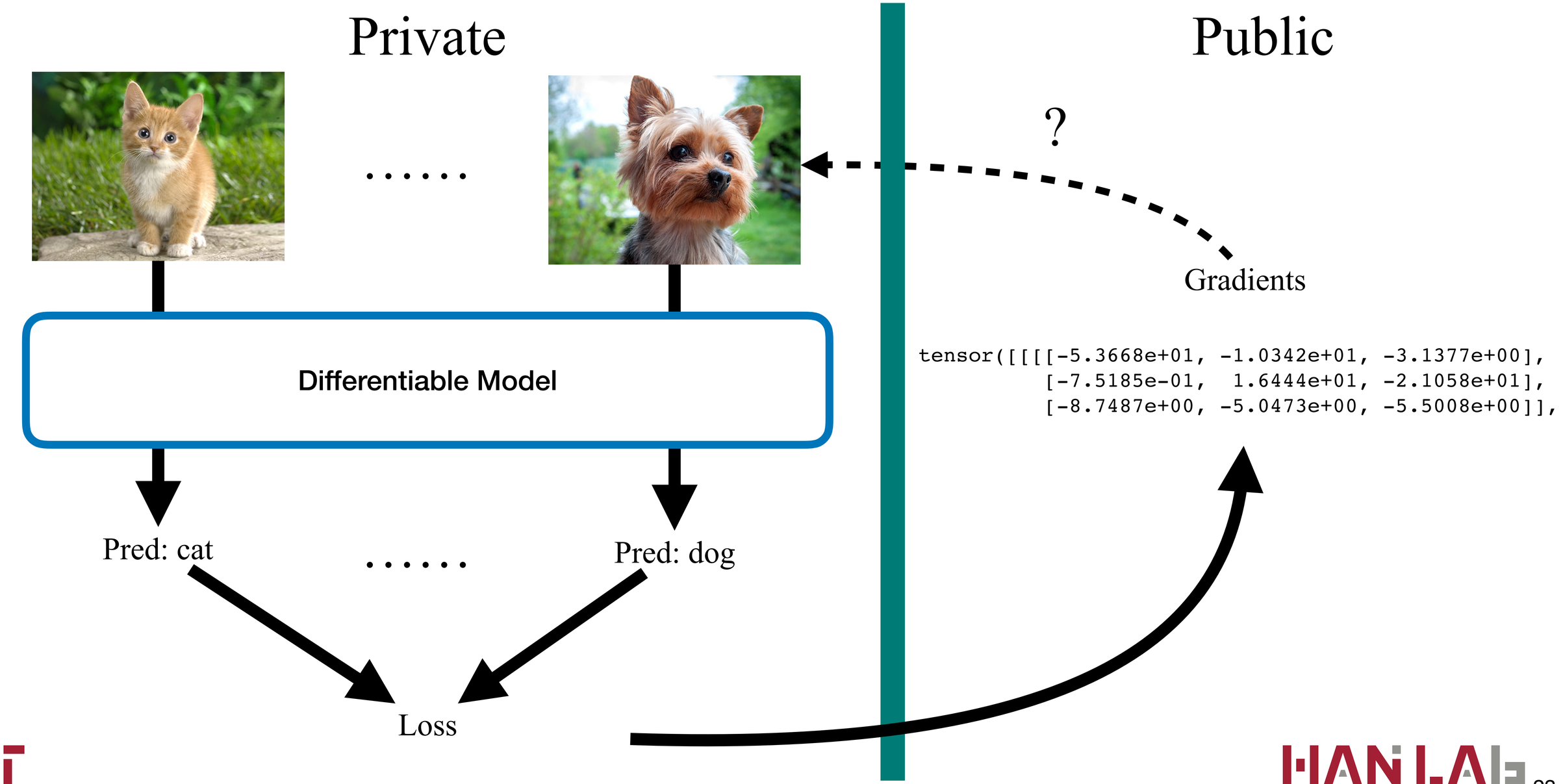
- p3.8x Instances on AWS (4 x V100)
- 4 instances at 4 different places
  - Ohio, California, Tokyo, London
  - Bandwidth: ~15Mb/s Latency: ~300ms
- The scalability of naive training: 0.02
- The scalability of DTC training: **0.72 (36x better!)**

Deep Leakage from Gradients

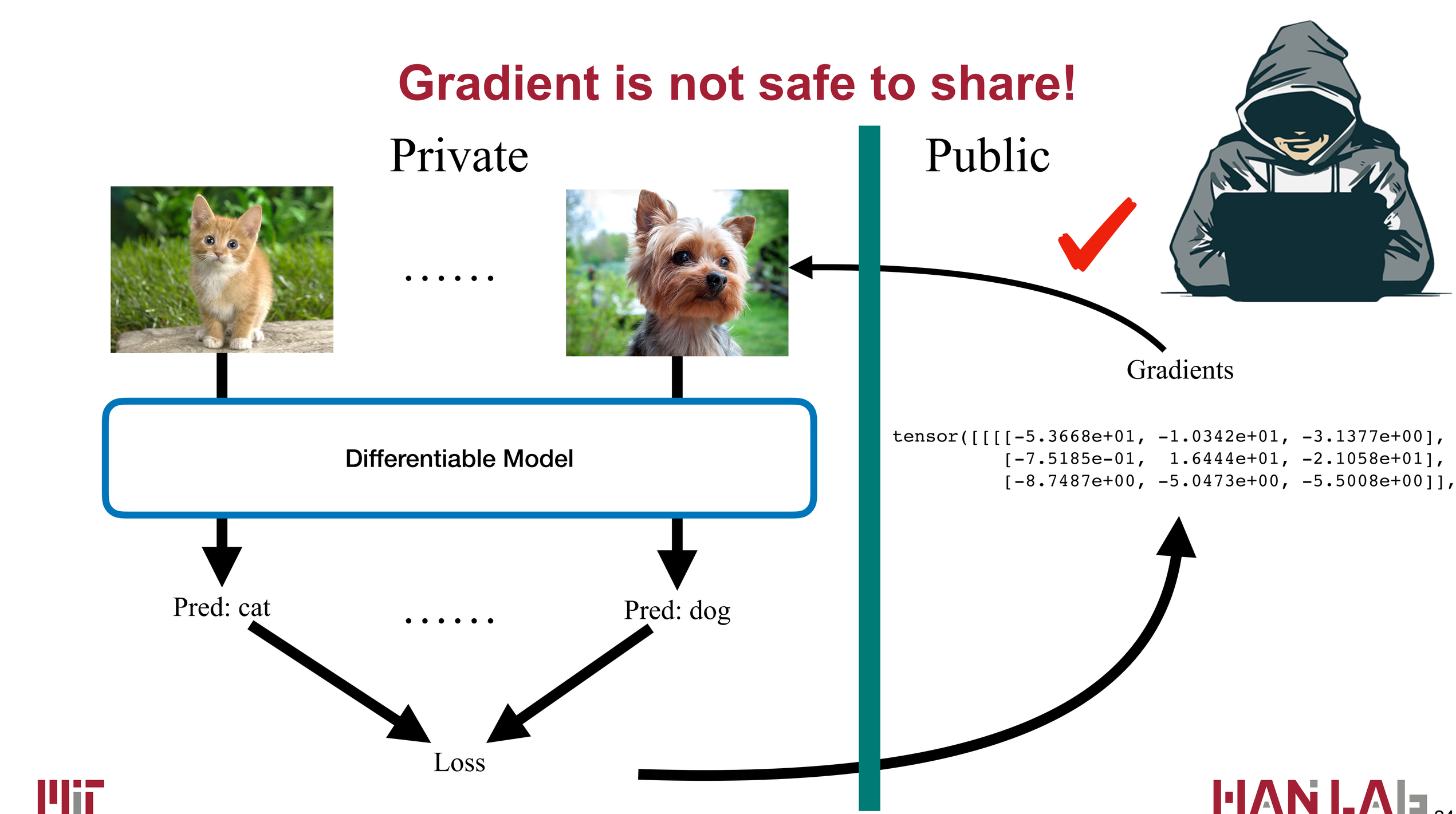
Ligeng Zhu, Zhijian Liu, Song Han

Neurips 19

Is gradient safe to share?



Gradient is not safe to share!



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.

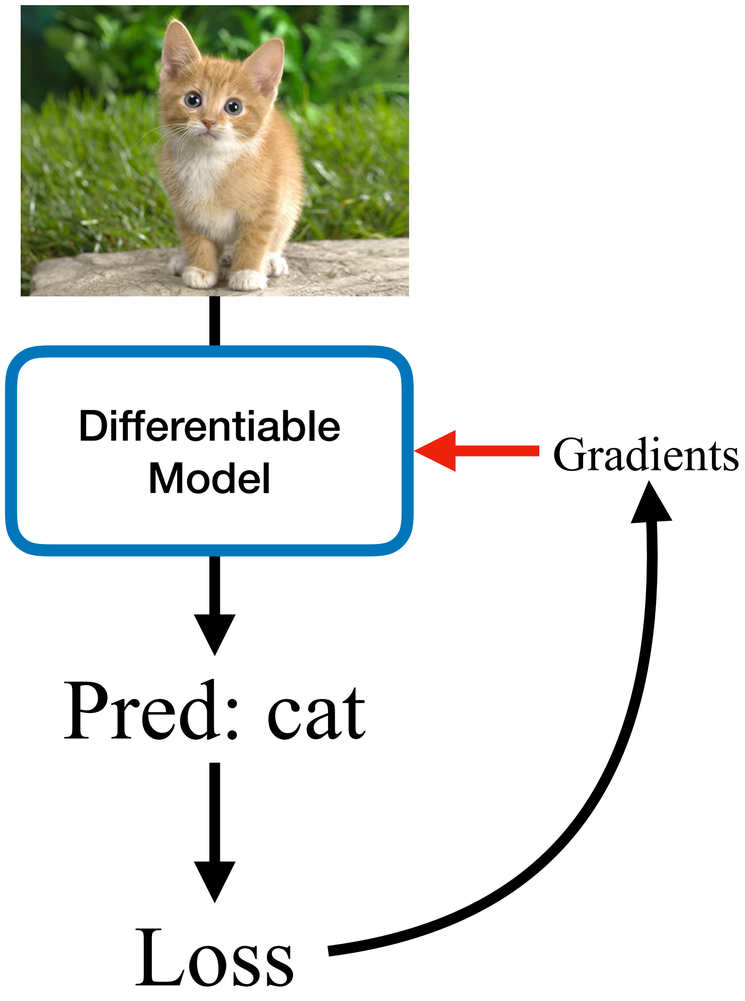
Property Inference
Whether a sample with certain property is in the batch.

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

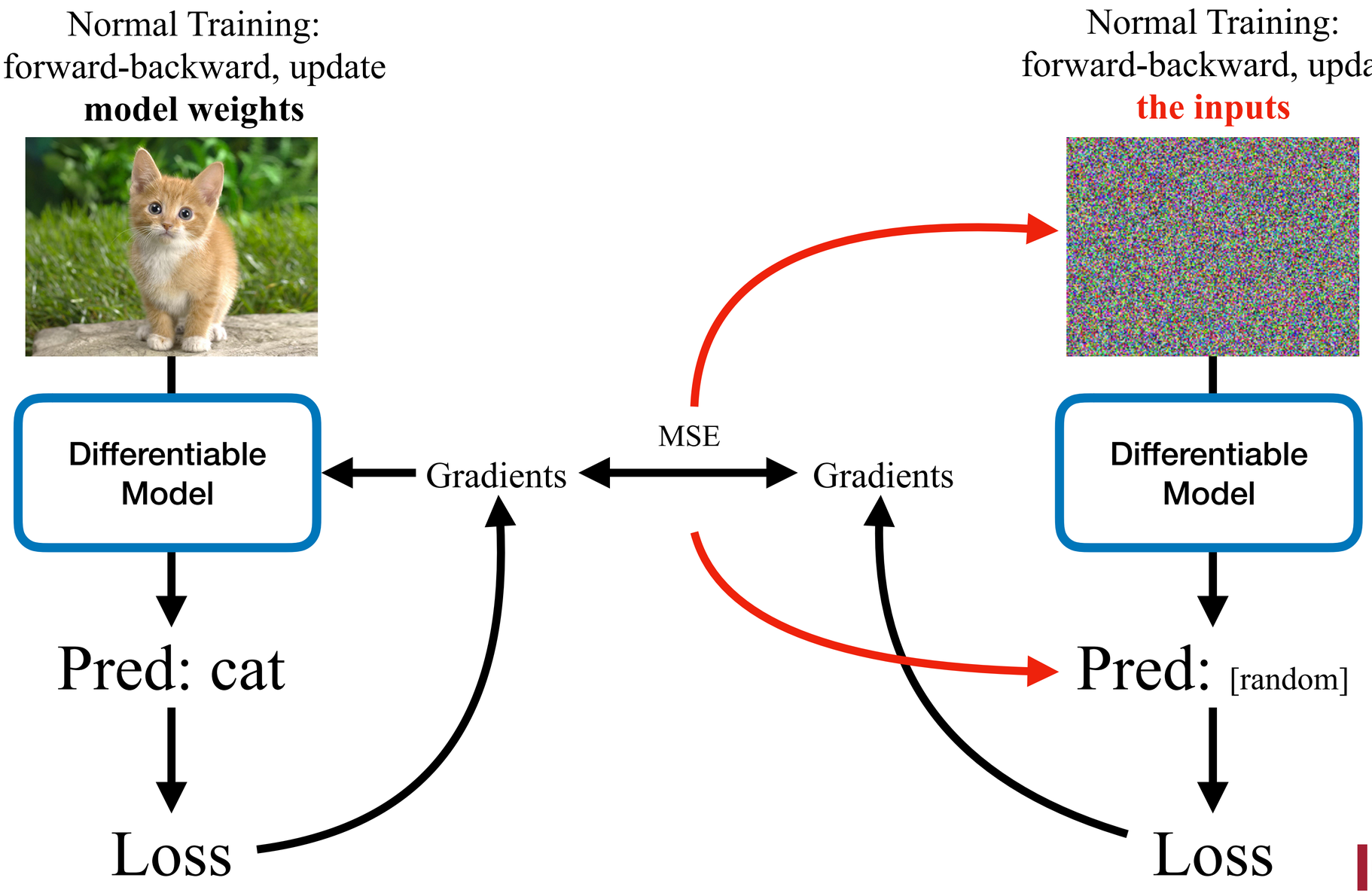
[1] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov. *Exploiting unintended feature leakage in collaborative learning.*
[2] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. *Membership inference attacks against machine learning models.*

Deep Leakage from Gradients

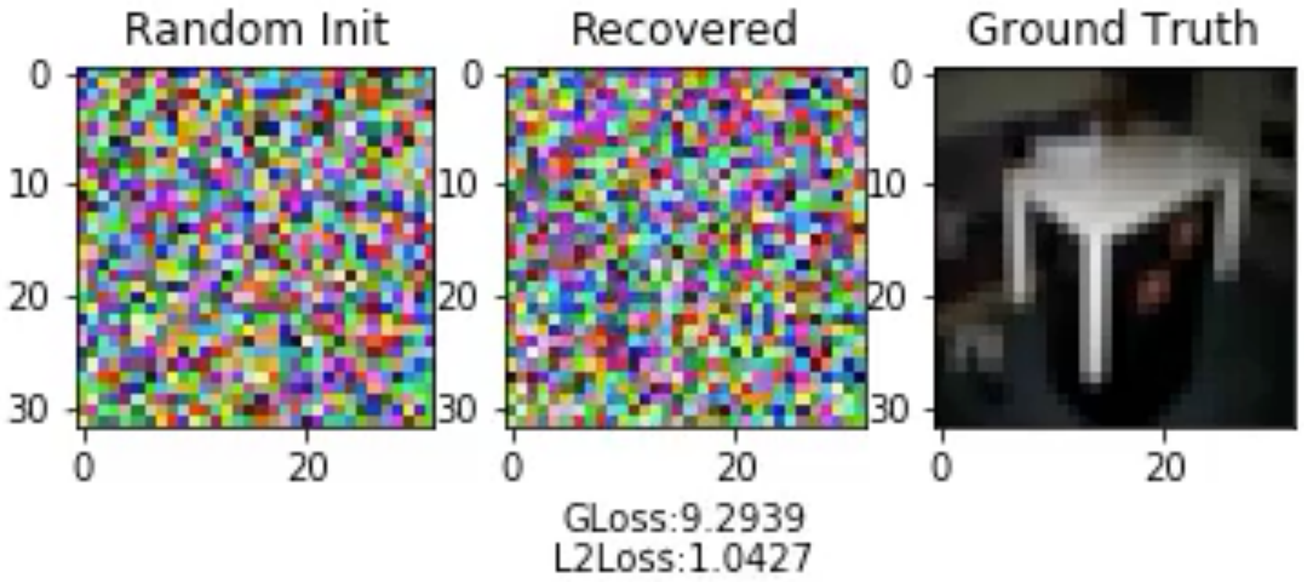
Normal Training:
forward-backward, update **model weights**



Deep Leakage from Gradients

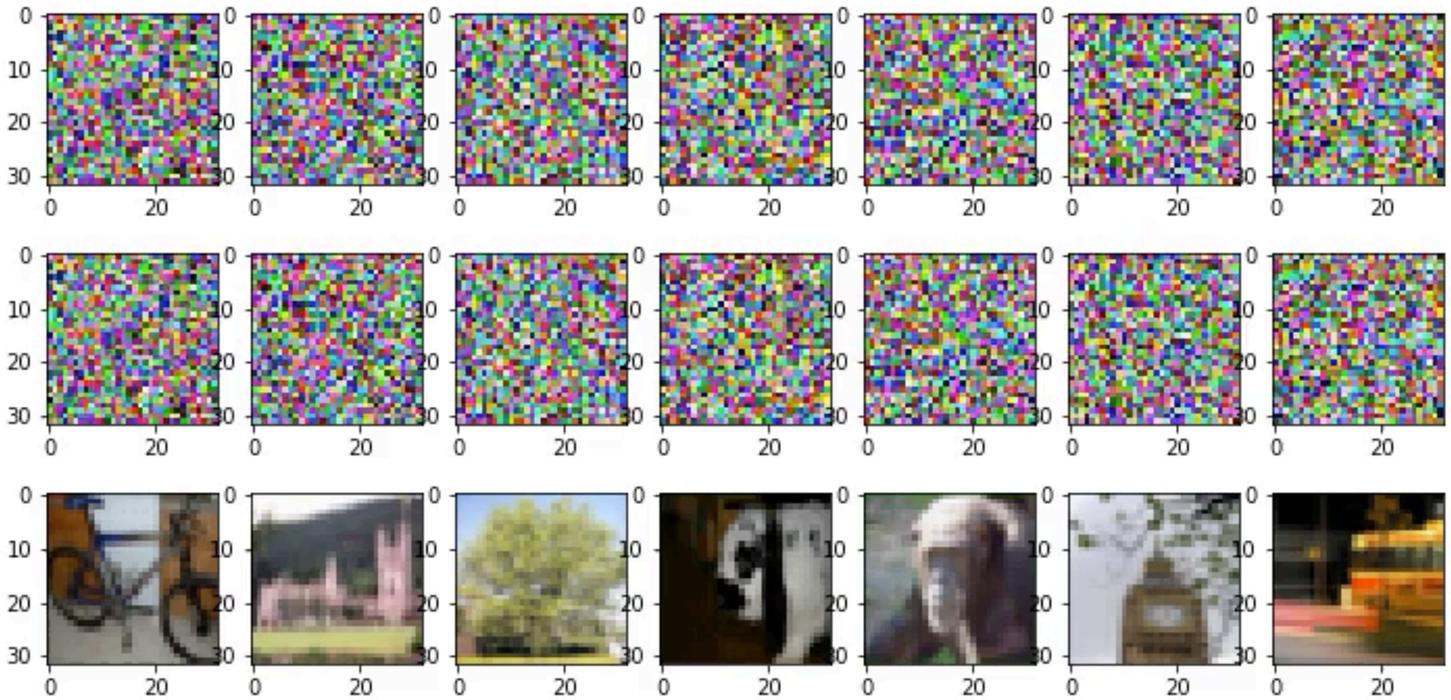


Recovering Visualization (bs=1)



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

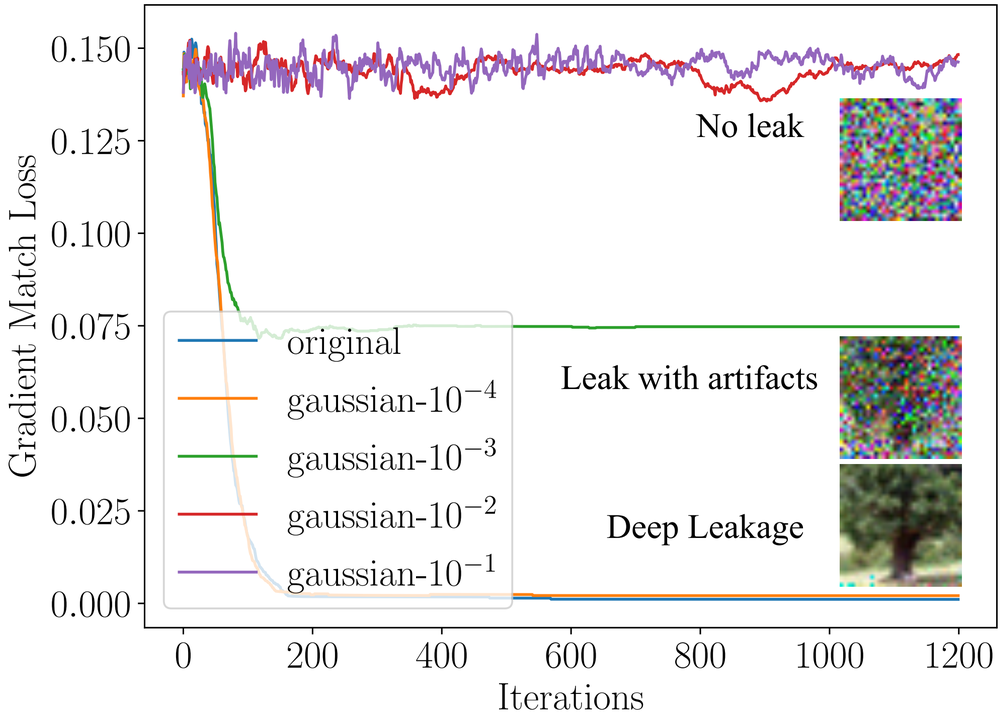
Iters=10: tilting fill given **less full solicitor other ligue shrill living vegas rider treatment carry played sculptures lifelong ellison net yards marne **kali

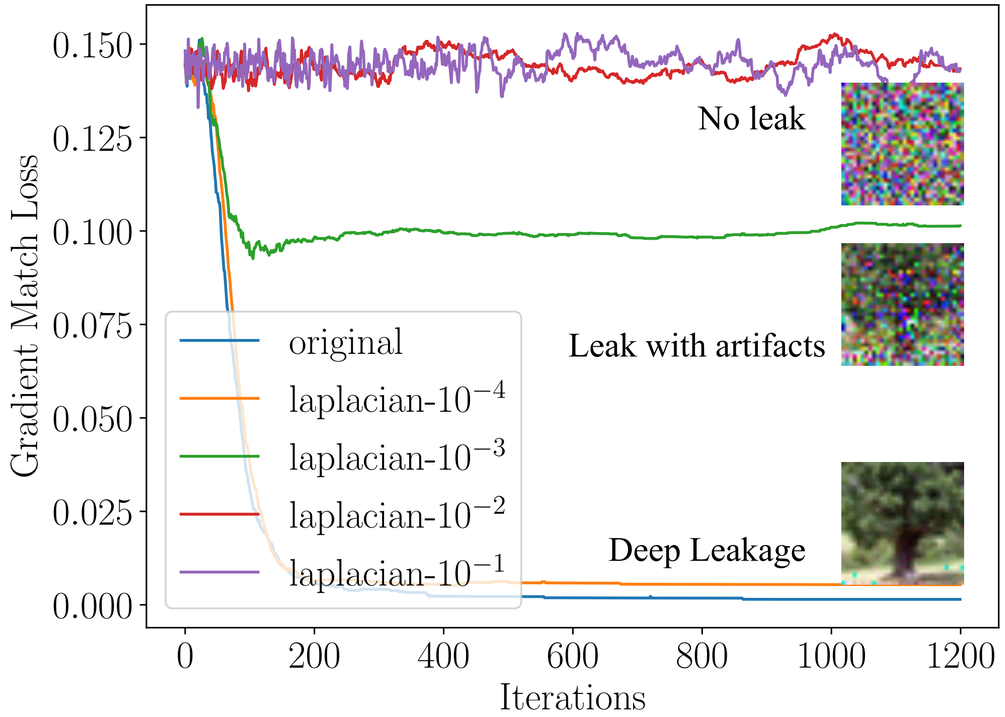
Iters=20: registration , volunteer applications , at student travel application open the ; week of played ; child care will be glare .

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

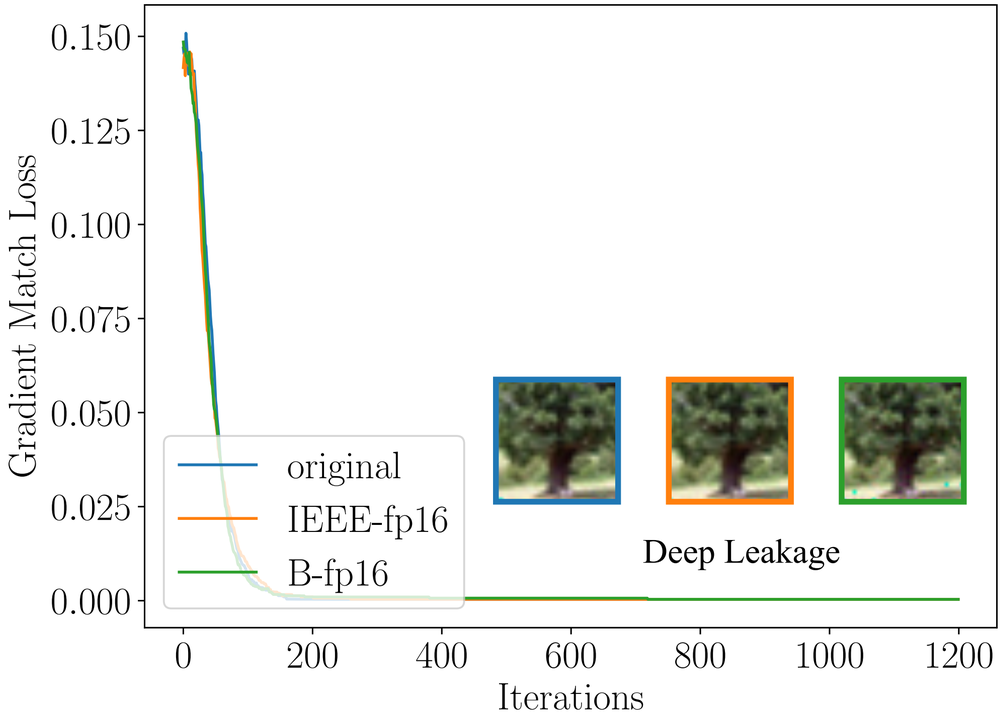
Original text: Registration, volunteer applications, and student travel application open the first week of September. Child care will be available.

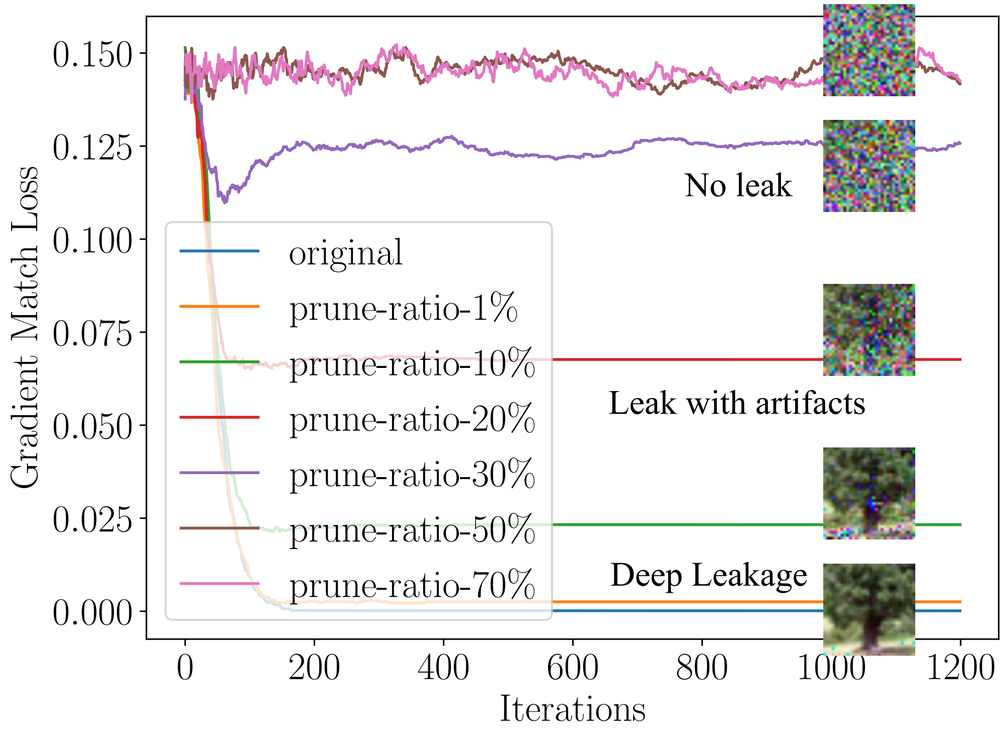
Defense Strategy





Defense Strategy





Thank you!

Advertisement: I am applying for Ph.D.