BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning

Chengliang Zhang†, Suyi Li†, Junzhe Xia†, Wei Wang†, Feng Yan‡, Yang Liu*

†Hong Kong University of Science and Technology

‡University of Nevada, Reno

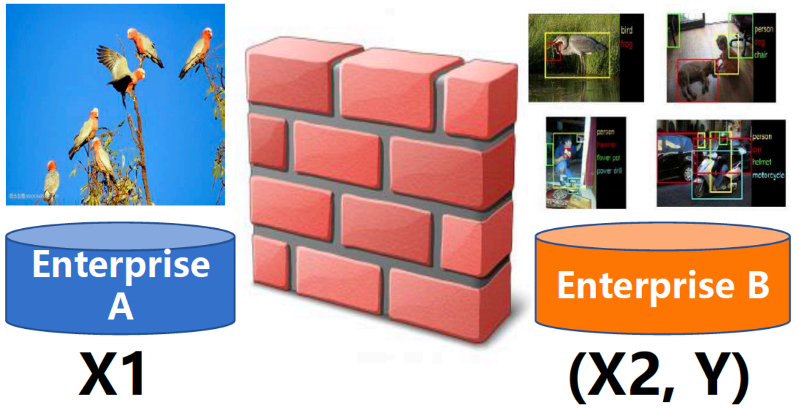
* WeBank

Federated Learning

- Privacy concerns
  - Data breaches
- Government regulations
  - GDPR
  - CCPA

Emerging challenge:
small & fragmented data

Data Silos



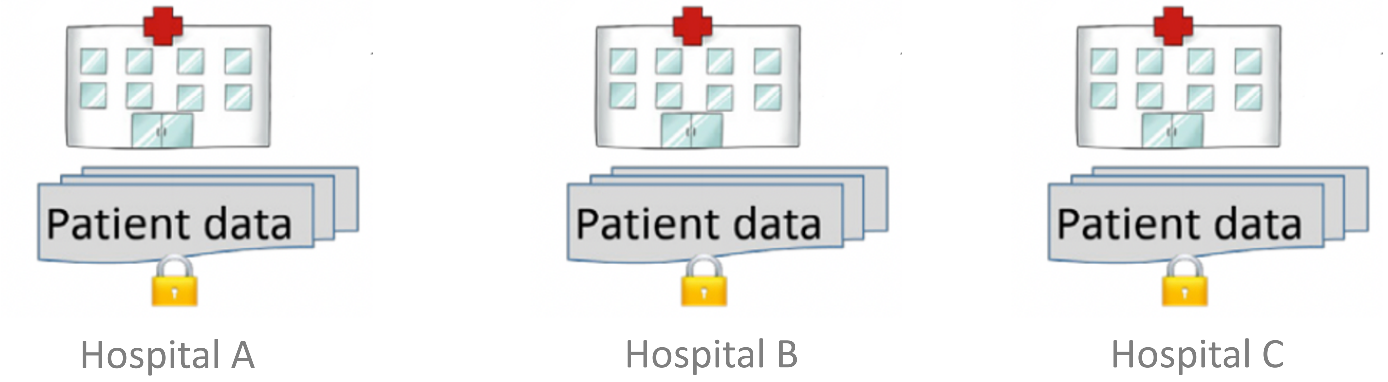
Solution: Federated Learning

Collaborative Machine Learning without Centralized Training Data [1]

[1] Bonawitz, Keith, et al. "Towards federated learning at scale: System design." arXiv preprint arXiv:1902.01046 (2019).

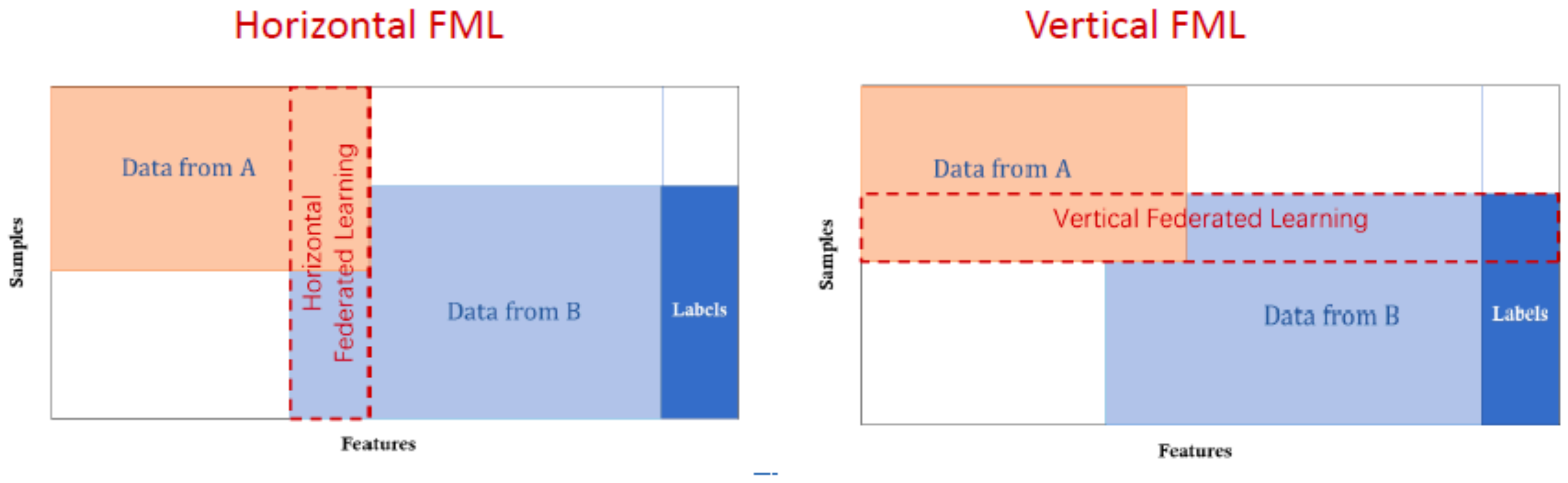
Target Scenario: Cross-Silo Horizontal FL

- Cross-Silo: among organizations / institutions
  - Banks, hospitals…
  - Reliable communication and computation
  - Strong privacy requirements
  - As opposed to cross-device: edge devices



Target Scenario: Cross-Silo Horizontal FL

- Horizontal: datasets share same feature space [2]

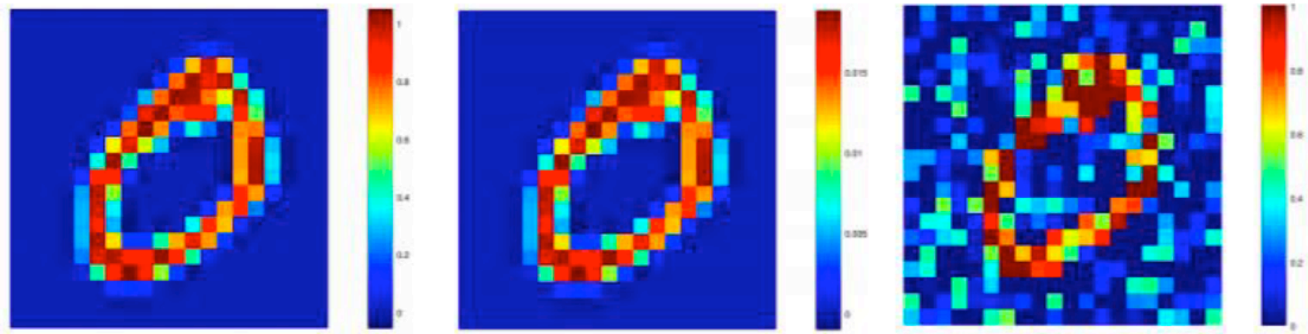


- Large overlap of features of the two data sets
- Large overlap of sample IDs (users) of the two data sets

- Objective: train a model together without revealing private data to third party (aggregator) and each other

[2] Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19.

Repurpose datacenter distributed training?



(**a**) Original 20x20 image of hand-written number 0, seen as a vector over $\mathbb{R}^{400}$ fed to a neural network.

(**b**) Recovered image using 400/10285 (3.89%) gradients (see Sect.3, Example 2). The difference with the original (**a**) is only at the value bar.

(**c**) Recovered image using 400/10285 (3.89%) gradients (see Sect.3, Example 3). There are noises but the truth label 0 can still be seen.

Gradients are not safe to share in plaintext [3]

[3] Aono, Yoshinori, et al. "Privacy-preserving deep learning via additively homomorphic encryption." IEEE Transactions on Information Forensics and Security 13.5 (2017): 1333-1345.

Federated Learning Approaches

Method	Differential Privacy	Secure Multi Party Comput.	Secure Aggregation [7]	Homomorphic Encryption
Efficiency		🚫 [6]	🚫	🚫
Strong Privacy	🚫 [4]		🚫	
No accuracy loss	🚫 [5]			

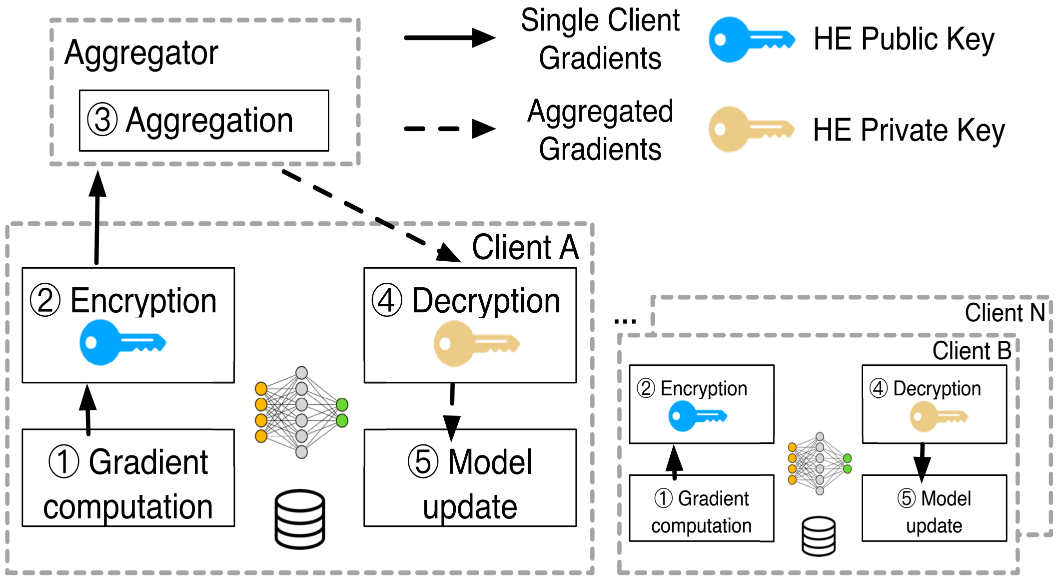
[4] Gehrke, Johannes, Edward Lui, and Rafael Pass. "Towards privacy for social networks: A zero-knowledge based definition of privacy." TCC 2011.
[5] Bagdasaryan, Eugene, Omid Poursaeed, and Vitaly Shmatikov. "Differential privacy has disparate impact on model accuracy." NIPS. 2019.
[6] Du, Wenliang, Yunghsiang S. Han, and Shigang Chen. “Privacy-preserving multivariate statistical analysis: Linear regression and classification.” SDM 2004.
[7] Bonawitz, Keith, et al. “Practical secure aggregation for privacy-preserving machine learning.” CCS 2017.

Additively Homomorphic Encryption for FL

- Allow computation over ciphertexts

  decrypt(encrypt(a) + encrypt(b)) = a + b

- Enables oblivious aggregation

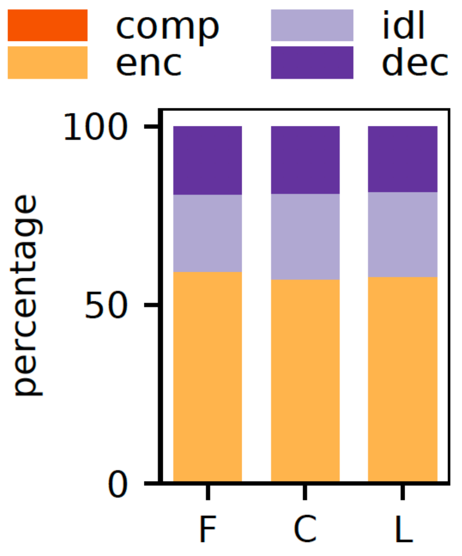


[8]

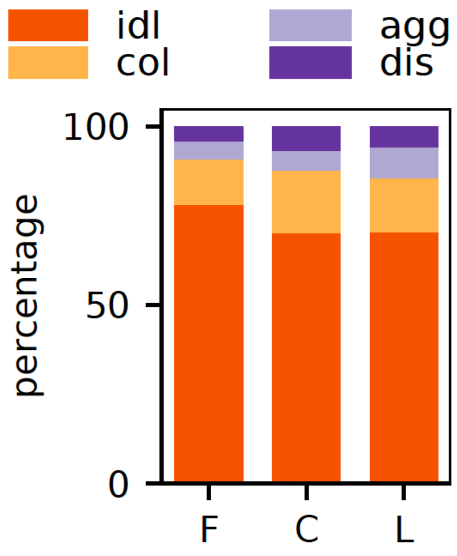
1. Clients produce gradients
2. Encrypt gradients and upload them to Aggregator
3. Aggregator summarizes all gradient ciphertexts
4. Clients receive aggregated gradients
5. Clients decrypt and apply model update

[8] Aono, Yoshinori, et al. "Privacy-preserving deep learning via additively homomorphic encryption." IEEE Transactions on Information Forensics and Security 13.5 (2017): 1333-1345.

Characterization: FL with HE



(a) Client: **comp**ute, **enc**rypt, **idl**e, **dec**rypt



(b) Aggregator: **idl**e, **col**lect, **agg**regate, **dis**patch

Time breakdown of one iteration
Run on FATE, models are **F**MNIST, **C**IFAR10, and **L**STM

Why is HE expensive:

- Computation
- Communication
  - Plaintext: 32bit -> ciphertext: 2000+ bit

Key Size	Plaintext	Ciphertext	Encryption	Decryption
1024	6.87MB	287.64MB	216.87s	68.63s
2048	6.87MB	527.17MB	1152.98s	357.17s
3072	6.87MB	754.62MB	3111.14s	993.80

Paillier HE

Potential Solutions

- Accelerate HE operations
  - Limited parallelism: 3X with FPGA [9]
  - Communication stays the same
- Reduce encryption operations
  - One operation multiple data
  - “batching” gradient values
  - Compact plaintext, less inflation

    plaintext: 2000 bit -> ciphertext 2000bit

Challenge:

Maintain HE’s additively property

Decrypting the sum of 2 batched ciphertexts

=

Adding pairs separately

-0.3	0	2.6	-1.1

+

1.2	0.33	-4.2	-0.2

=

0.9	0.33	-1.6	-1.3

[9] San, Ismail, et al. "Efficient paillier cryptoprocessor for privacy-preserving data mining." Security and communication networks 9.11 (2016): 1535-1546..

Gradient Batching is non-trivial

All ciphertexts at aggregator: no *differentiation*, no *permutation*, no *shifting*

Only *bit-wise* additions on underlying plaintexts

					sign	exponent	mantissa
-0.3	0	2.6	-1.1	→	1	01111111	00011001100110011001101
+							
1.2	0.33	-4.2	-0.2	→	1	01111100	10011001100110011001101

Not addable

Gradients are floating numbers: exponent aligning is required for addition [9]

[9] San, Ismail, et al. "Efficient paillier cryptoprocessor for privacy-preserving data mining." Security and communication networks 9.11 (2016): 1535-1546..

Quantization for Batching

Floating gradient values cannot be batched -> quantization

A generic quantization method maps [-1, 1]
To [0, 255]

Quantization: 255 * (**-0.0079** - -1) / (1 - -1) = **126**

Dequantization: **127** * (1 - -1) / 255 + 2 * (-1) = **-1**

Batching with generic quantization

	0111 1110	1000 0001	…
	-0.0079 **126**	0.0079 **129**	
+	0000 0001	0111 1000	…
	-0.9921 **1**	-0.0551 **120**	
=	0111 1111	1111 1001	…
	-1 **127**	-0.0475 **249**	

original value

quantized value

Limitations

- Restrictive: client # is required
- Overflow easily: all positive integers
- No differentiation between positive and negative overflows

Our Quantization & Batching Solution

Desired quantization for aggregation

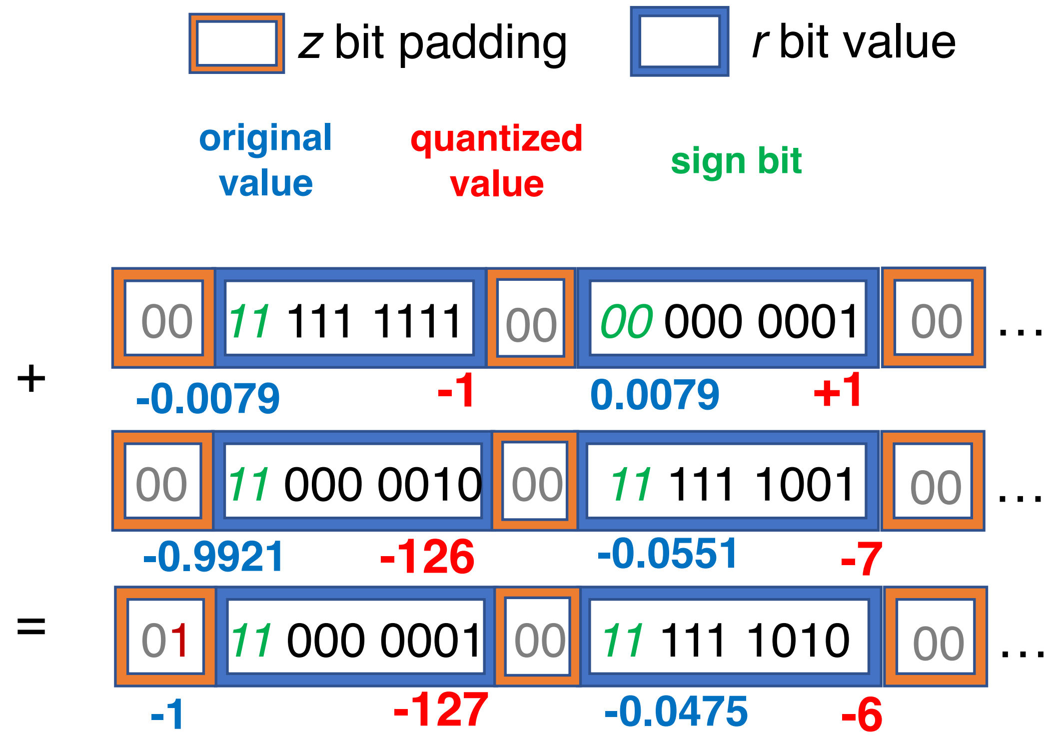
- *Flexible*
  - Aggregation results are unbatchable only with ciphertexts alone
- *Overflow-aware*
  - If overflow happens, we can tell the sign

Our Quantization & Batching Solution

Customized quantization for aggregation

- Distinguish overflow
  - *Signed* integer
- Positive and negative cancel out each other
  - *Symmetric* range
  - *Uniform* quantization

[-1, 1] is mapped to [-127, 127]



BatchCrypt

Our Quantization & Batching Solution

Customized quantization for aggregation

- *Signed* integer
- *Symmetric* range
- *Uniform* quantization

Challenges:

1. Differentiate overflows:

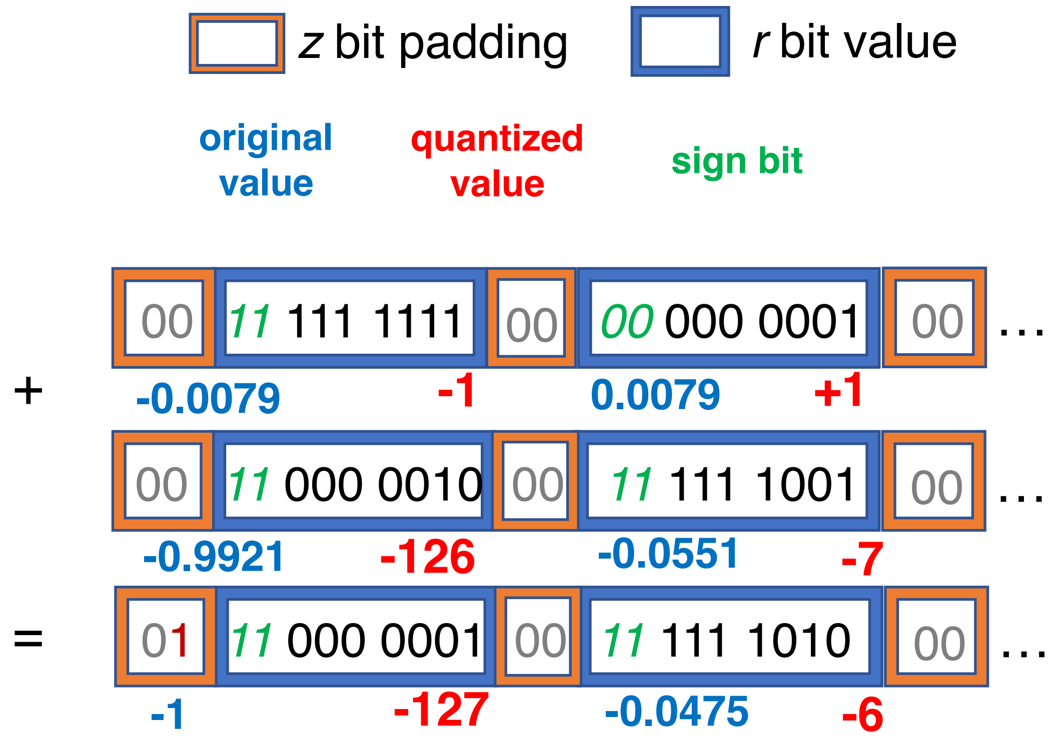
   two sign bits

2. Distinguish sign bits from value bits:

   two's compliment coding

3. Tolerate overflowing:

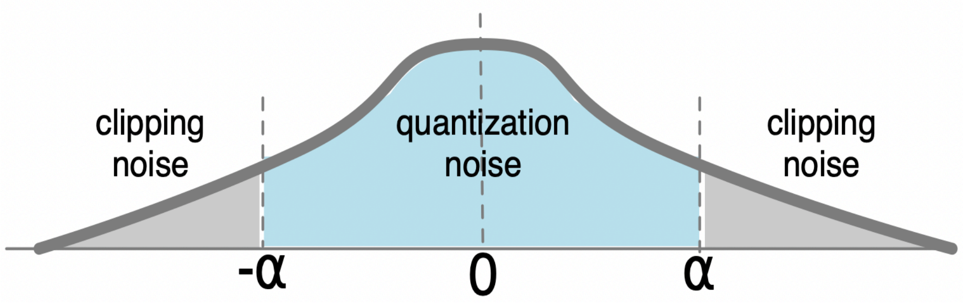
   padding zeros in between



BatchCrypt

Gradient Clipping

Gradients are *unbounded*
Quantization range is *bounded*
Clipping is required



Tradeoff:

Smaller α → Higher resolution within |α| 😀

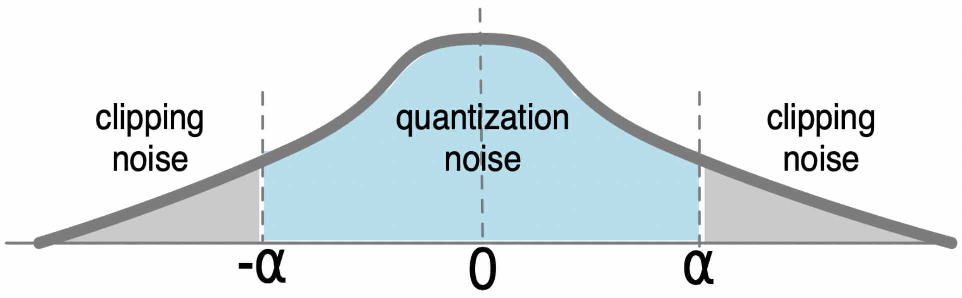
Smaller α → More diminished range information ☹️

Gradient Clipping

Gradients are *unbounded*

quantization range is *bounded*

Clipping is required

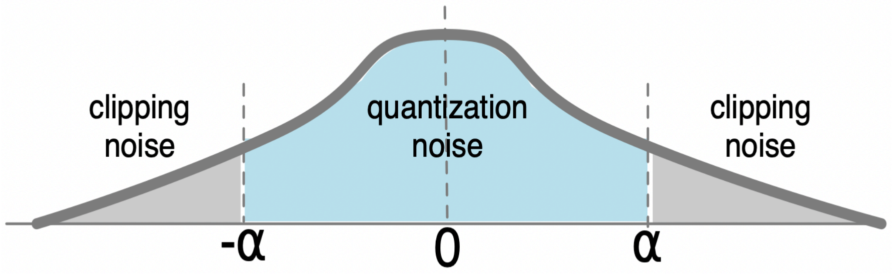


- ❑ Profiling quantization loss with a sample dataset [10]
  - FL has non-iid data
  - Gradients range diminishes during training: optimal shifts
- ❑ Analytical clipping with an online model
  - Model the noises with distribution fitting
  - Flexible & adaptable

[10] http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf

dACIQ: Analytical Gradient Clipping

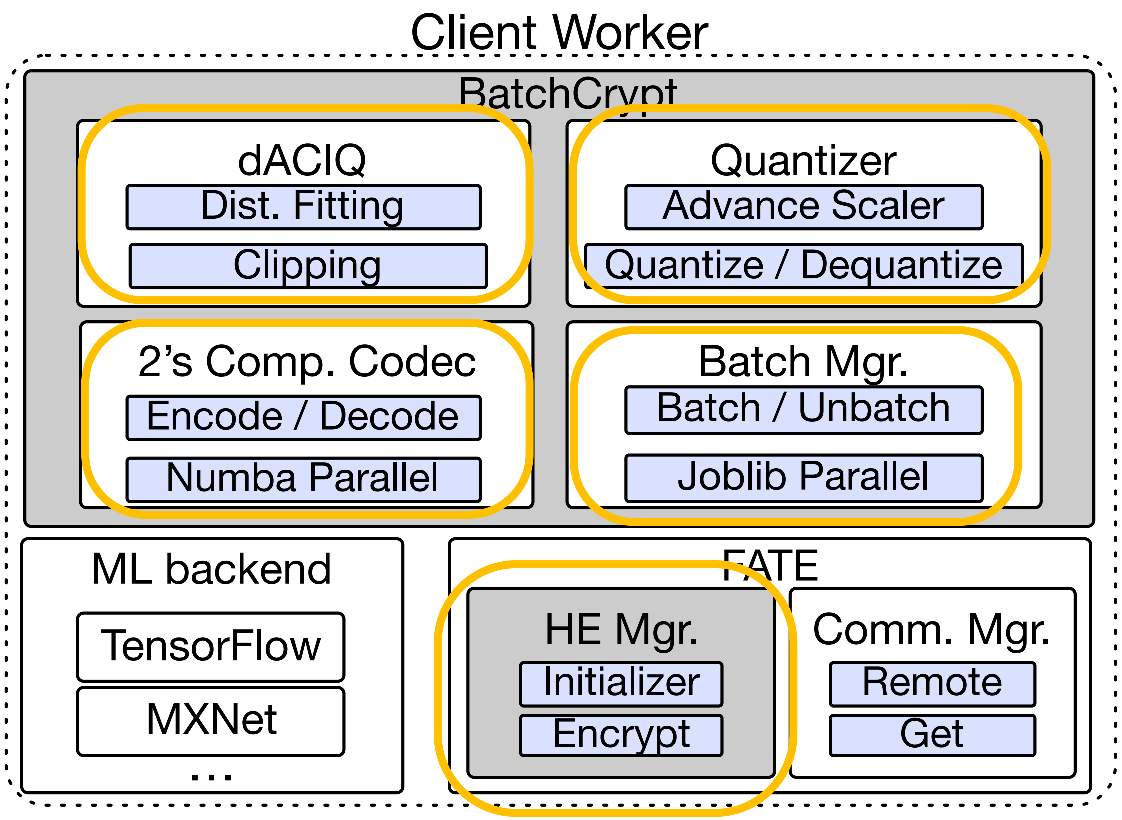
- Gradients distribution is bell-shaped: *Gaussian* like
- Conventional gaussian fitting: *MLE*, *BI*
  - ✓ Requires a lot of information
  - ✓ Computationally intensive



- *dACIQ* proposes a Gaussian Fitting method for distributed dataset
  - Only requires *max*, *min*, and *size*
  - Computationally efficient: online
  - *Stochastic* Rounding [11]
  - *Layer-wise* quantization

[11] Banner, Ron, Yury Nahshan, and Daniel Soudry. "Post training 4-bit quantization of convolutional networks for rapid-deployment." Advances in Neural Information Processing Systems. 2019.

Introducing BatchCrypt



BatchCrypt

- Built atop FATE v1.1
- Support TensorFlow, MXNet, and extendable to other frameworks
- Implemented in Python
- Utilize Joblib, Numba for maximum parallelism

Evaluations Setup

Test Models

Model	Type	Network	Weights
FMNIST	Image Classification	3-layer-FC	101.77K
CIFAR	Image Classification	AlexNet	1.25M
LSTM-ptb	Text Generation	LSTM	4.02M

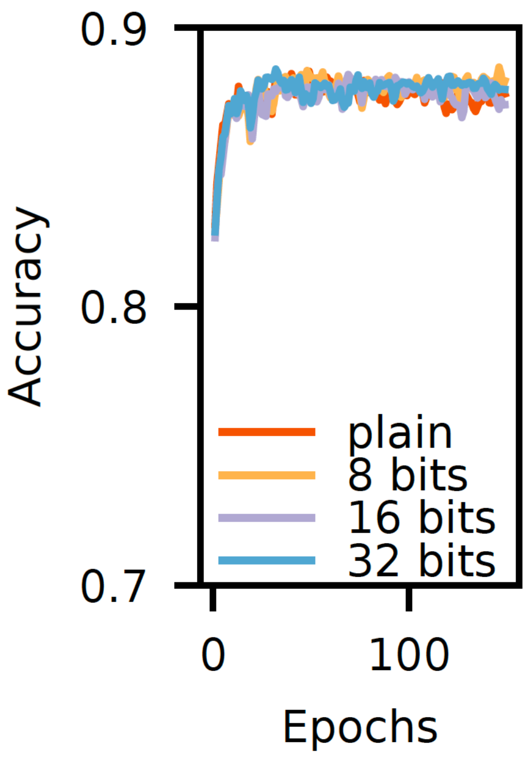
Test Bed

- AWS
- Cluster of 10, spanning 5 locations
- C5.4xlarge instances (16 vCPUs, 32 GB memory)

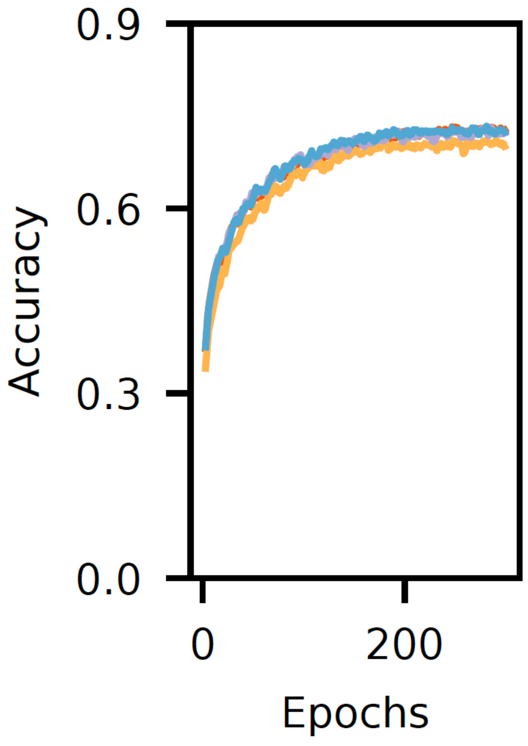
Region	US W.	Tokyo	US E.	London	HK
Up (Mbps)	9841	116	165	97	81
Down (Mbps)	9842	122	151	84	84

Bandwidth from clients to aggregator

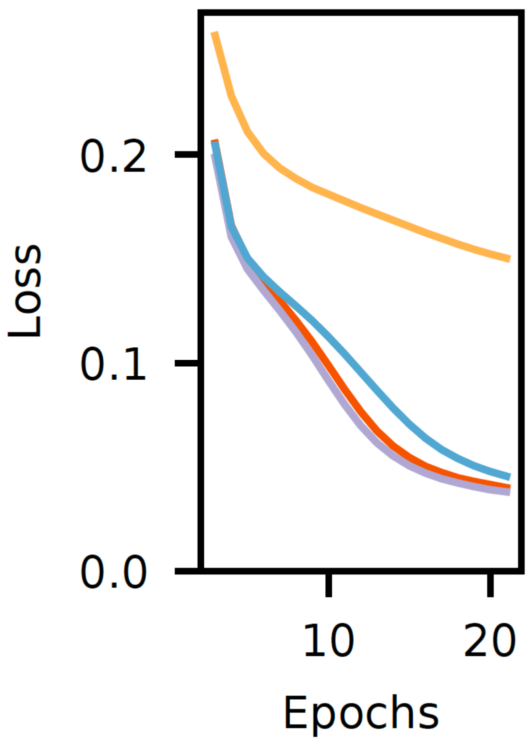
BatchCrypt’s Quantization Quality



FMNIST
test accuracy



CIFAR
test accuracy

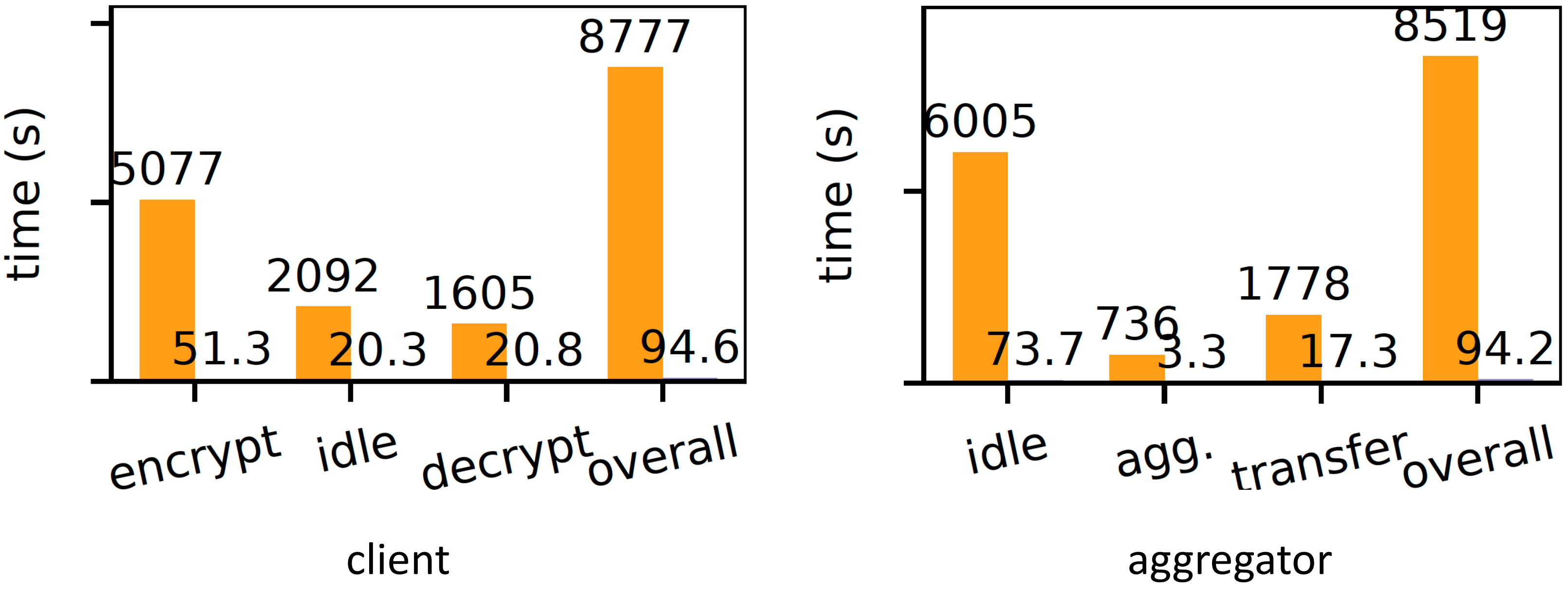


LSTM
loss

- Negligible loss
- Quantization sometimes outperforms plain: randomness adds regularization

BatchCrypt's Effectiveness: Computation

Iteration time breakdown of LSTM



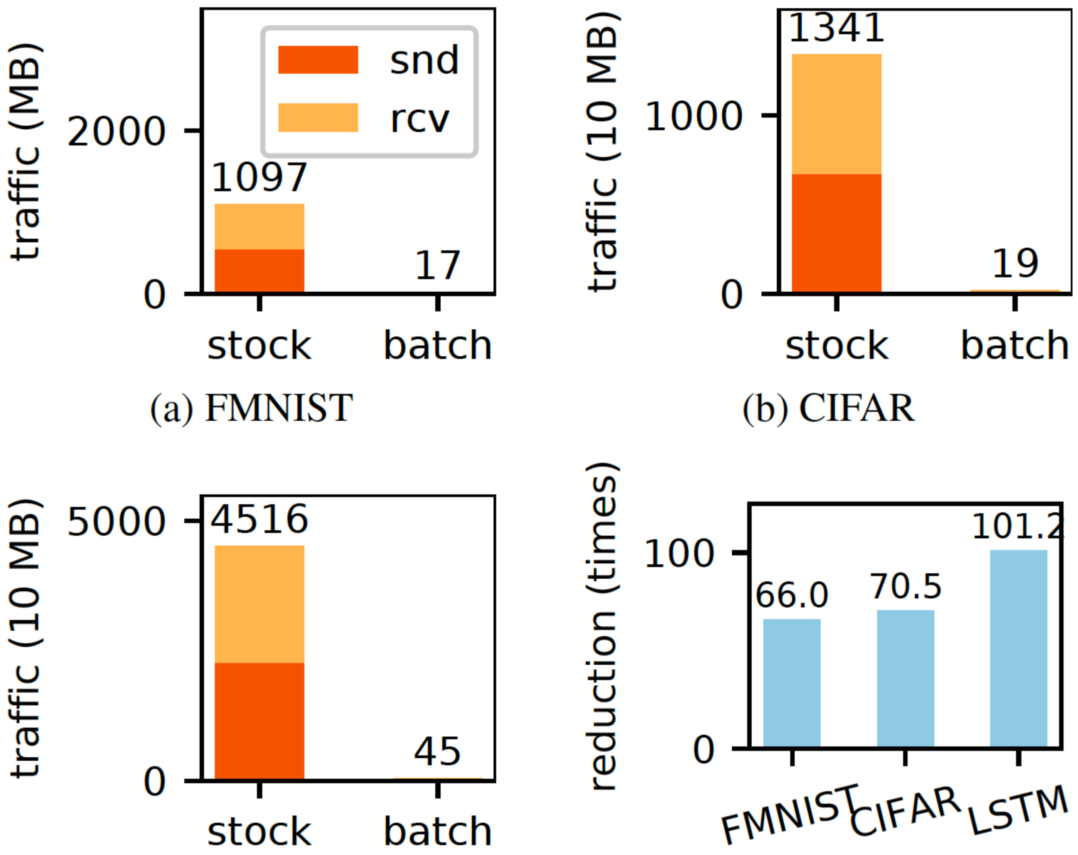
- Compared with stock FATE
- Batch size set to 100
- 16 bit quantization

- 23.3X for FMNIST
- 70.8X for CIFAR
- 92.8X for LSTM

Larger the model, better the results

BatchCrypt's Effectiveness: Communication

Network traffic consumed by communication per iteration



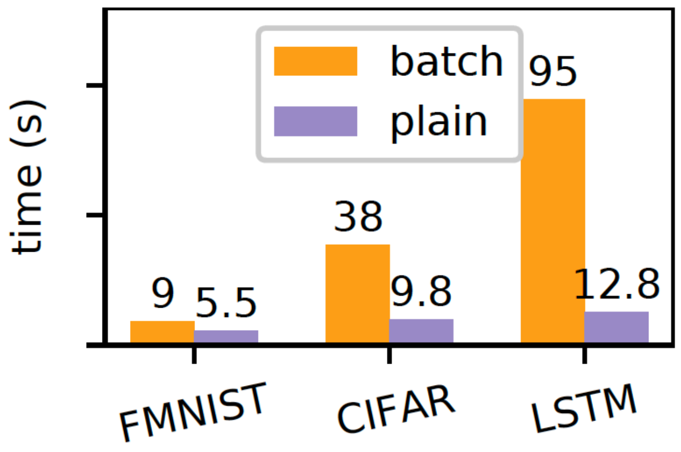
- Compared with stock FATE
- Batch size set to 100
- 16 bit quantization

- 66X for FMNIST
- 71X for CIFAR
- 101X for LSTM

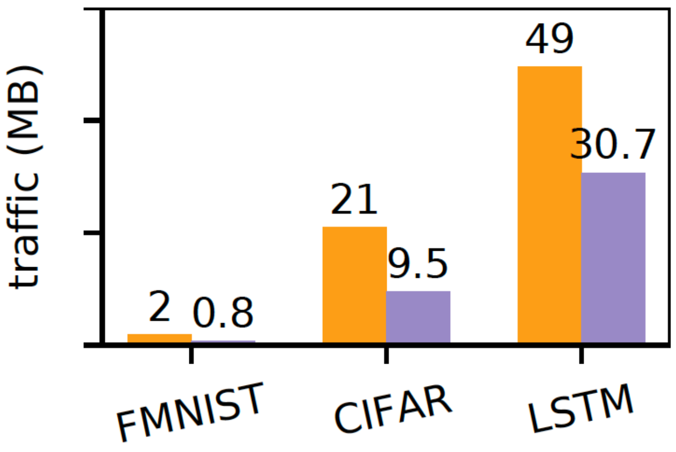
traffic

BatchCrypt's Overhead

Time and traffic per iteration



time



traffic

Feasible to train large models now

- Compared with plain distributed training without encryption
- Batch size set to 100
- 16 bit quantization

- Overhead significantly reduced
- Practical to deploy

BatchCrypt's Effectiveness: Convergence

Total time and communication until convergence

Model	Mode	Epochs	Acc. /Loss	Time (h)	Traffic (GB)
FMNIST	stock	40	88.62%	122.5	2228.3
	batch	68	88.37%	8.9	58.7
	plain	40	88.62%	3.2	11.17
CIFAR	stock	285	73.79%	9495.6	16422.0
	batch	279	74.04%	131.3	227.8
	plain	285	73.79%	34.2	11.39
LSTM	stock	20	0.0357	8484.4	15347.3
	batch	23	0.0335	105.2	175.9
	plain	20	0.0357	12.3	10.4

Conclusion

- Characterized HE enabled cross-silo FL
- Designed an efficient HE batching scheme BatchCrypt
    - Codesigning quantization, coding, & batching
    - Online analytical clipping dACIQ
- Implemented, and evaluated it on AWS
    - Up to 99% cost reduction

Thank you for coming!

BatchCrypt is open sourced at
https://github.com/marcoszh/BatchCrypt

Find me

https://marcoszh.github.io/

Graduating soon & seeking opportunities