THE LASER WORKSHOP

Simulation on Differentially Private Federated Meta-learning Systems

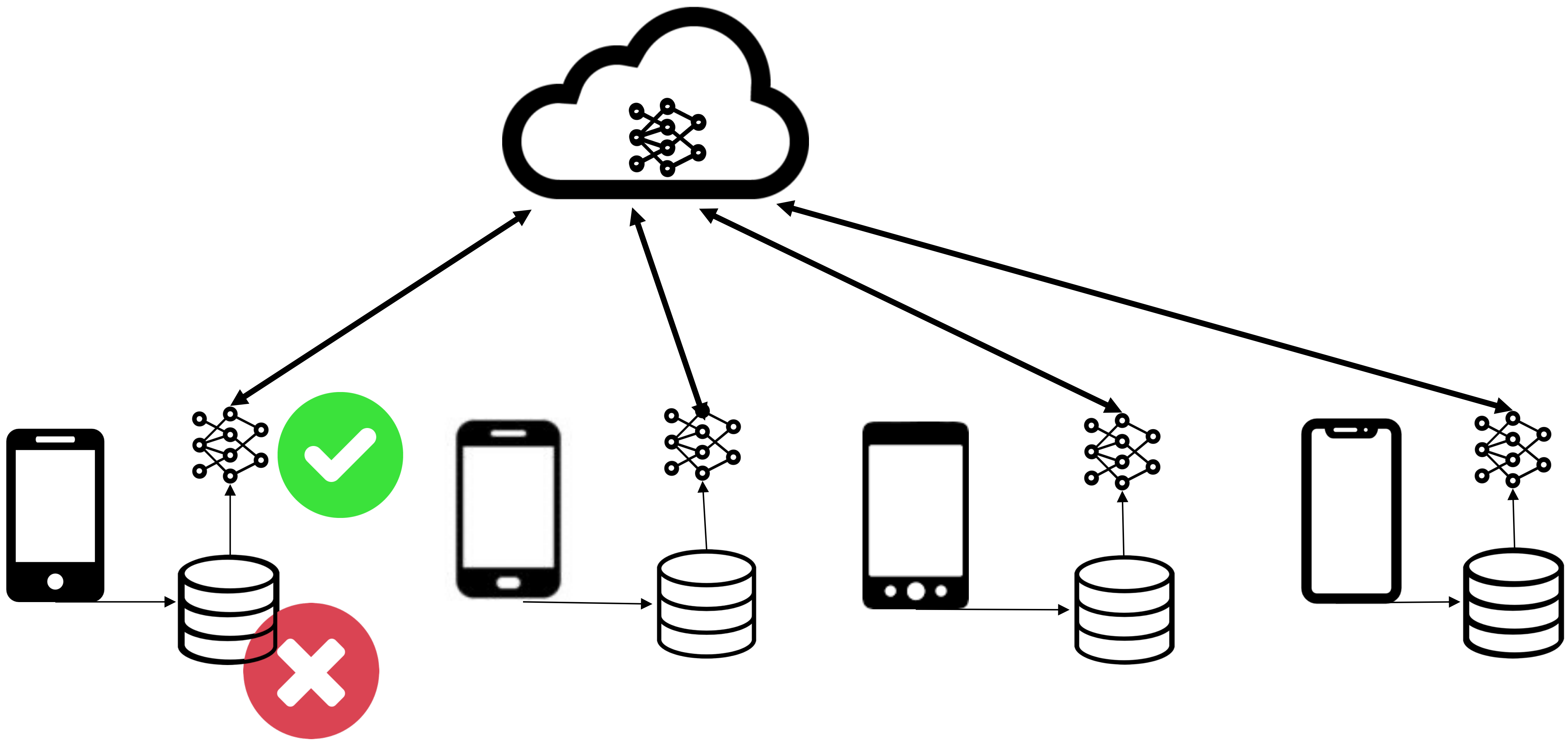
Authors: **Ning Wang**′, Yang Xiao†, Yimin Chen‡, Ning Zhang⋆, Wenjing Lou′, and Y. Thomas Hou′

Virginia Tech ′
University of Kentucky †
University of Massachusetts Lowell ‡
Washington University in St. Louis ⋆

Our ACSAC paper is entitled “Squeezing More Utility via Adaptive Clipping on Differentially Private Gradients in Federated Meta-Learning”.

Background: Federated Learning (FL)

- Key Characteristic: not requiring data sharing.
- Goal of FL: enable a central server to train a global model by aggregating model parameters from distributed intelligent end devices.

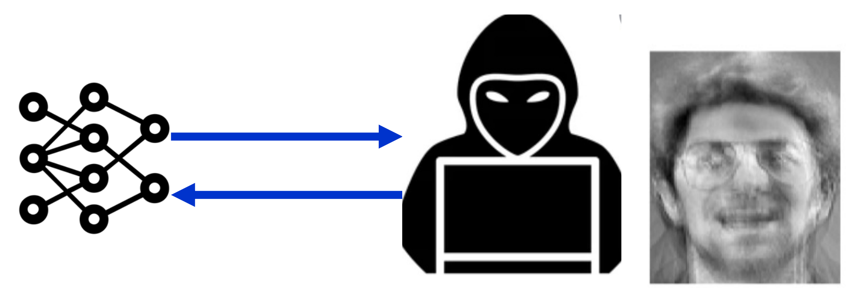


Research Problem and Goals

Privacy Attack against FL

- FL cannot guarantee the privacy of training data.
- State-of-the-art Inference Attack
  - Model Inversion Attack [3]
  - Membership inference attack [1,2]
  - Attribute Inference Attack [4]

Real training data



Inferred training data

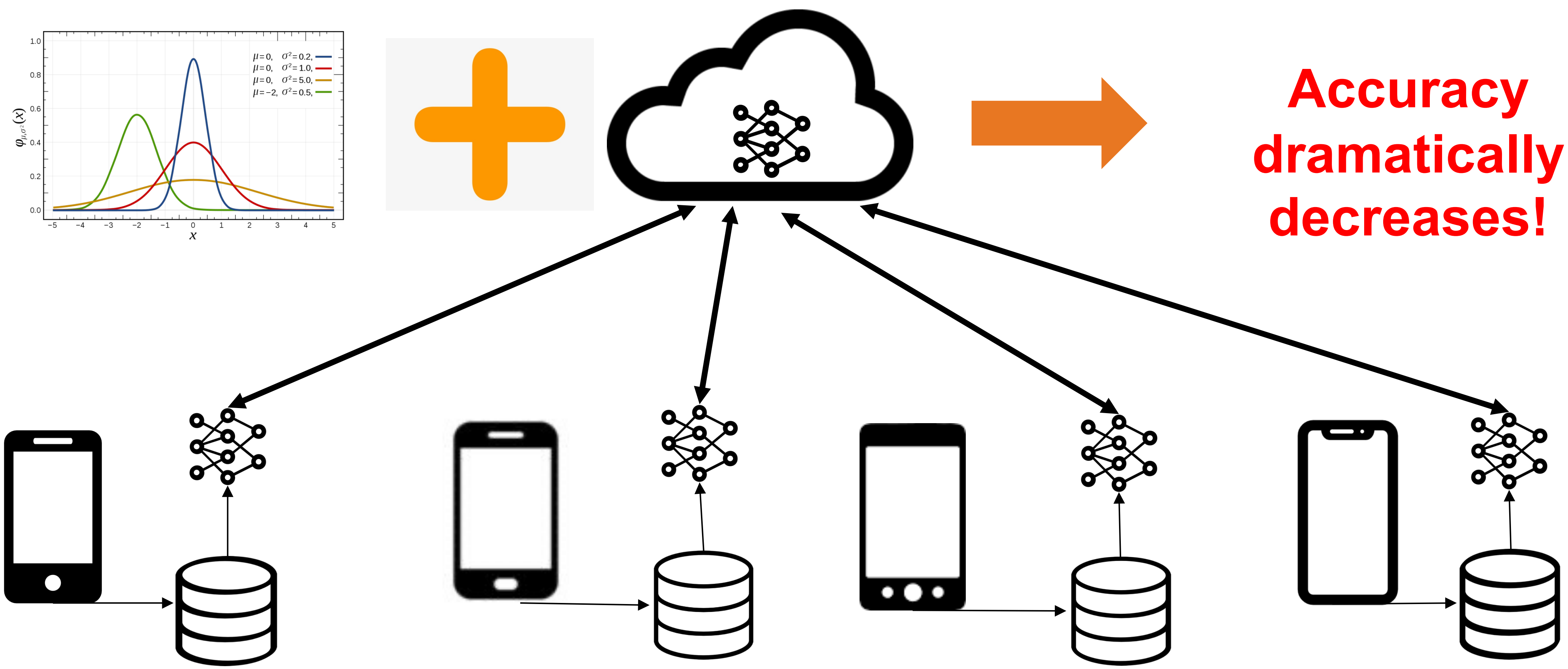
[1] Milad Nasr et al. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE Symposium on Security and Privacy (SP 19). IEEE, 739–753.
[2] Jingwen Zhang, Jiale Zhang, Junjun Chen, and Shui Yu. 2020. Gan enhanced membership inference: A passive local attack in federated learning. In ICC 20202020 IEEE International Conference on Communications (ICC). IEEE, 1–6.
[3] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In IEEE Conf. on Computer Communications (INFOCOM). IEEE, 2512–2520.
[4] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. 2009. Learning your identity and disease from research papers: information leaks in genome wide association study. In Proceedings of the 16th ACM conference on Computer and communications security (CCS). ACM, 534–544.

Problem and Goal of this Paper

- Problem
  - FL can protect data privacy to some extend.
  - Attackers are still capable to infer training data while knowing the model parameters.
  - Differential Privacy (DP) is a tool for privacy protection, but it harms the accuracy a lot.
  - We consider a scenario that local data is small.
- Goal
  - Provide rigorous privacy guarantee for users by incorporate DP.
  - Maintain a good trade-off between privacy and accuracy.

Differentially private federated Learning

- Problem: low accuracy

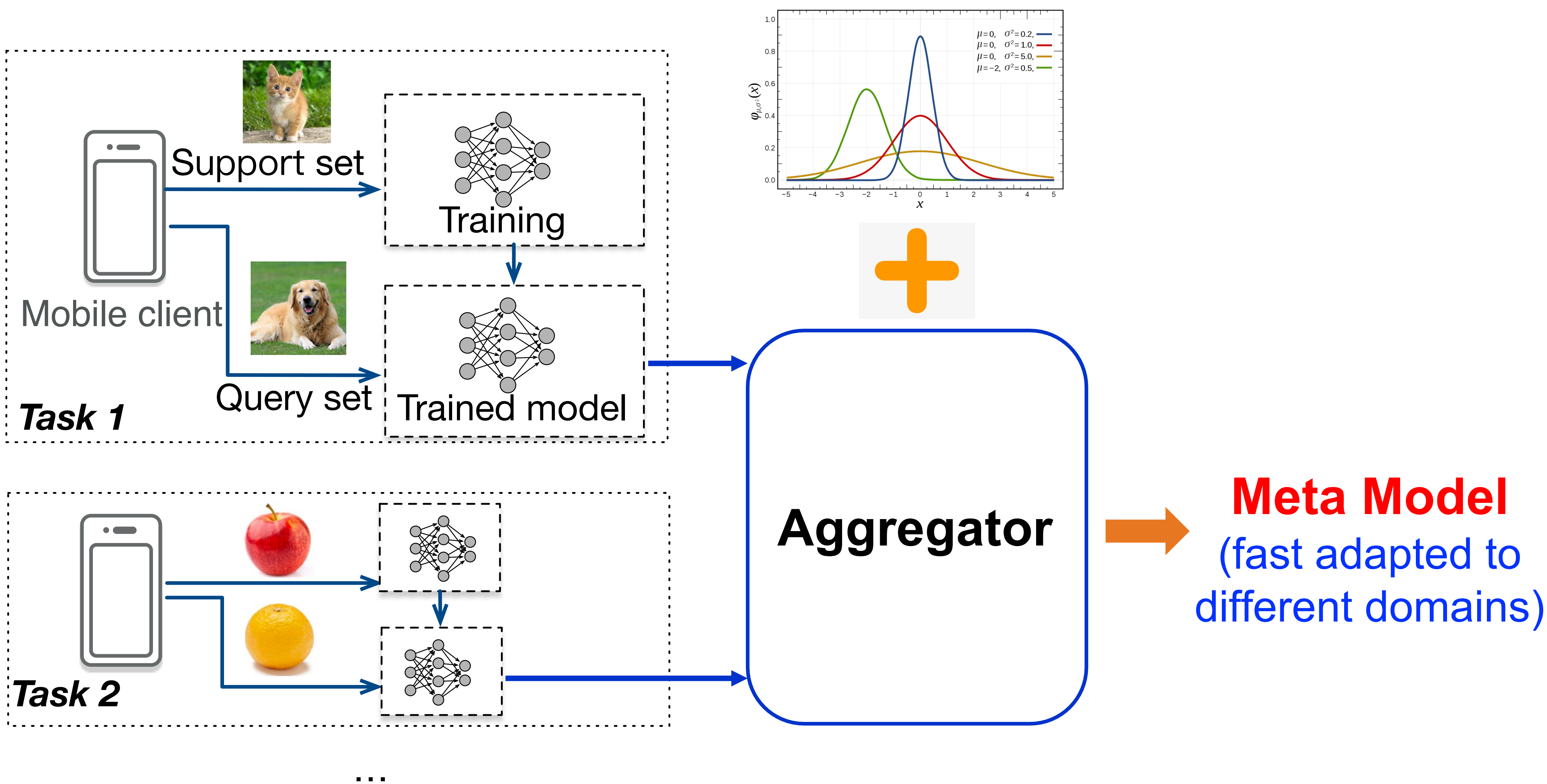


Continuous Training does help.

But it requires Enough Data & Training Power/Time

Differentially private federated Meta-Learning

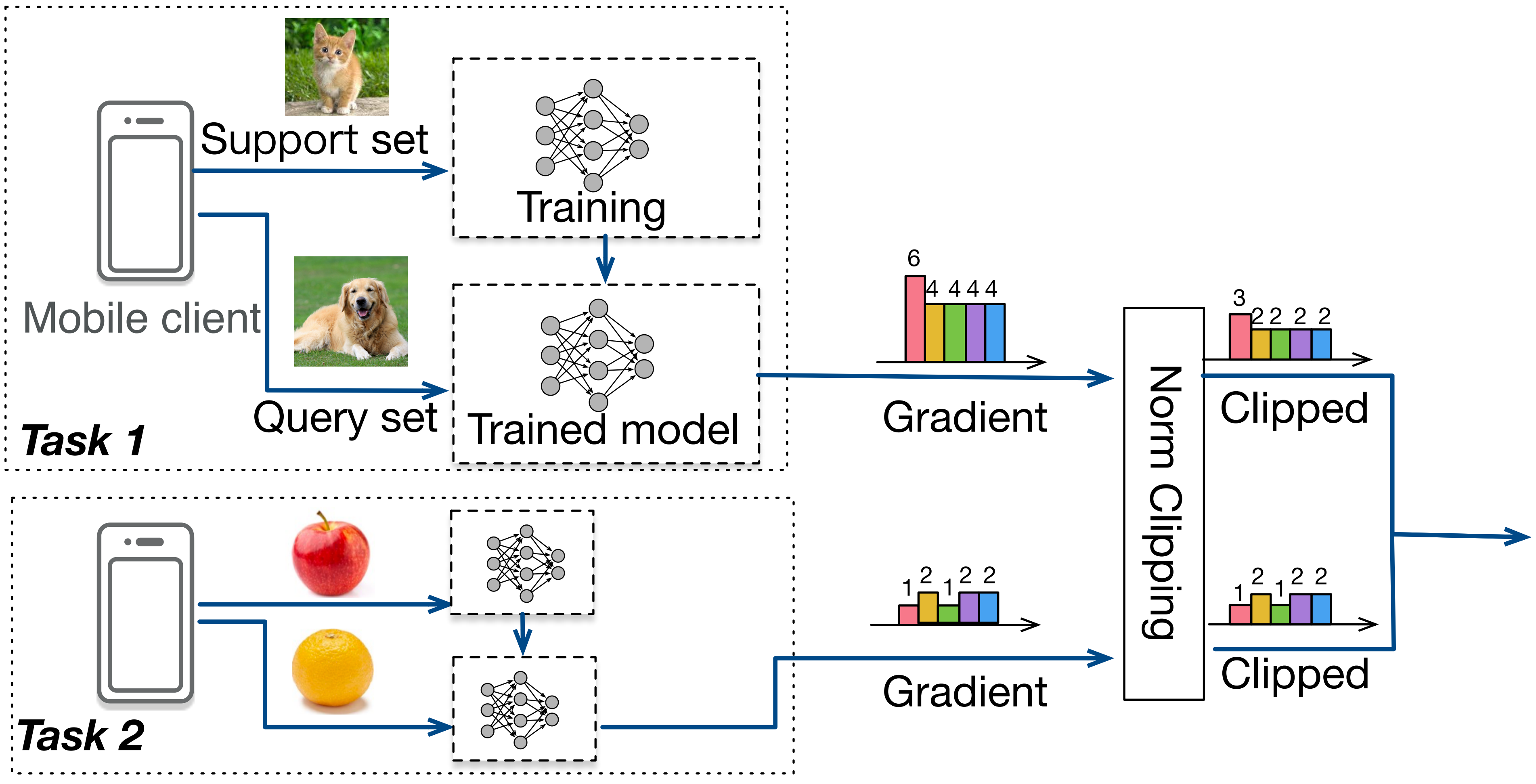
- Federated Learning → **Federated Meta-Learning**
  - Deal with few-shot problem.
  - Fast customization.



Proposed Workflow, Experiment and Evaluation

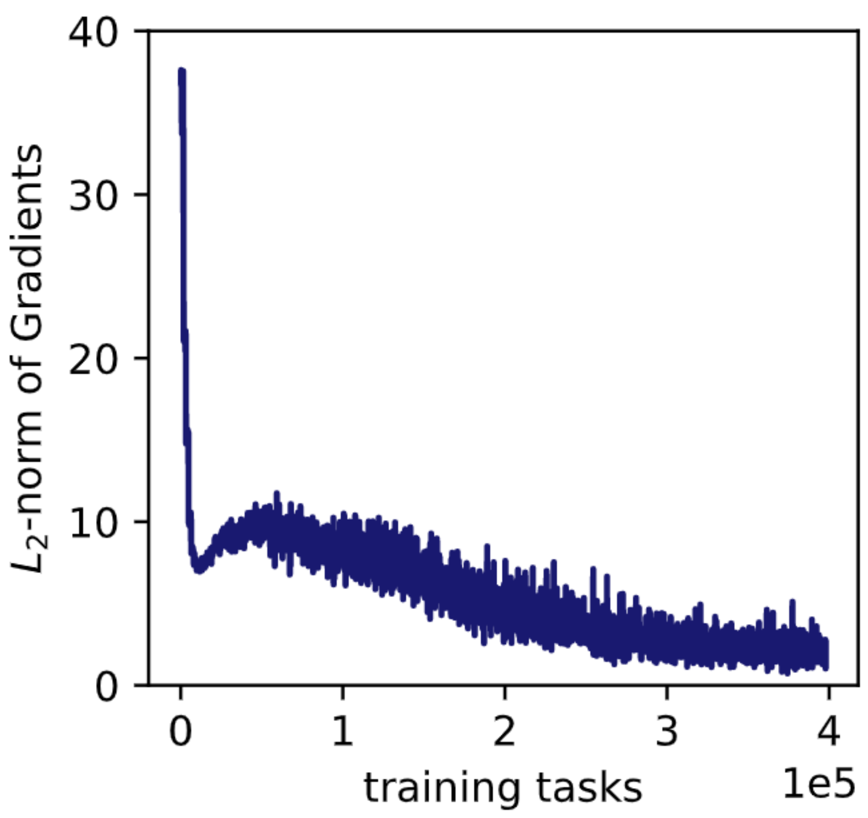
DP in Federated Meta-learning

- Adding noise
  - The noise should be proportional to the largest gradient.
  - To avoid too large noise, we should clip the gradient.



Our Proposal

- **Adaptive** Clipping
  - Naïve constant clipping maintain a fixed clipping threshold C. The noise will be: $k * C$.
  - Adaptive clipping: change the threshold C adaptively.

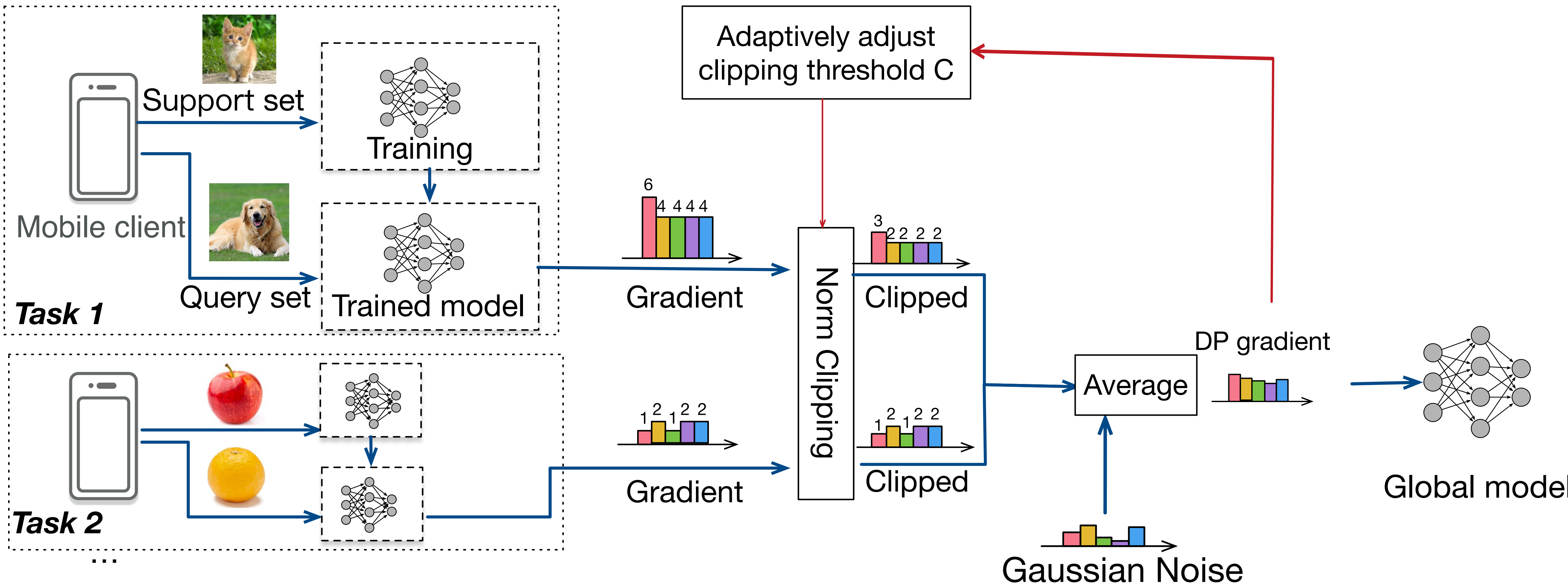


The gradients will decrease during the course of training.

We can change the threshold C according to the gradient change.

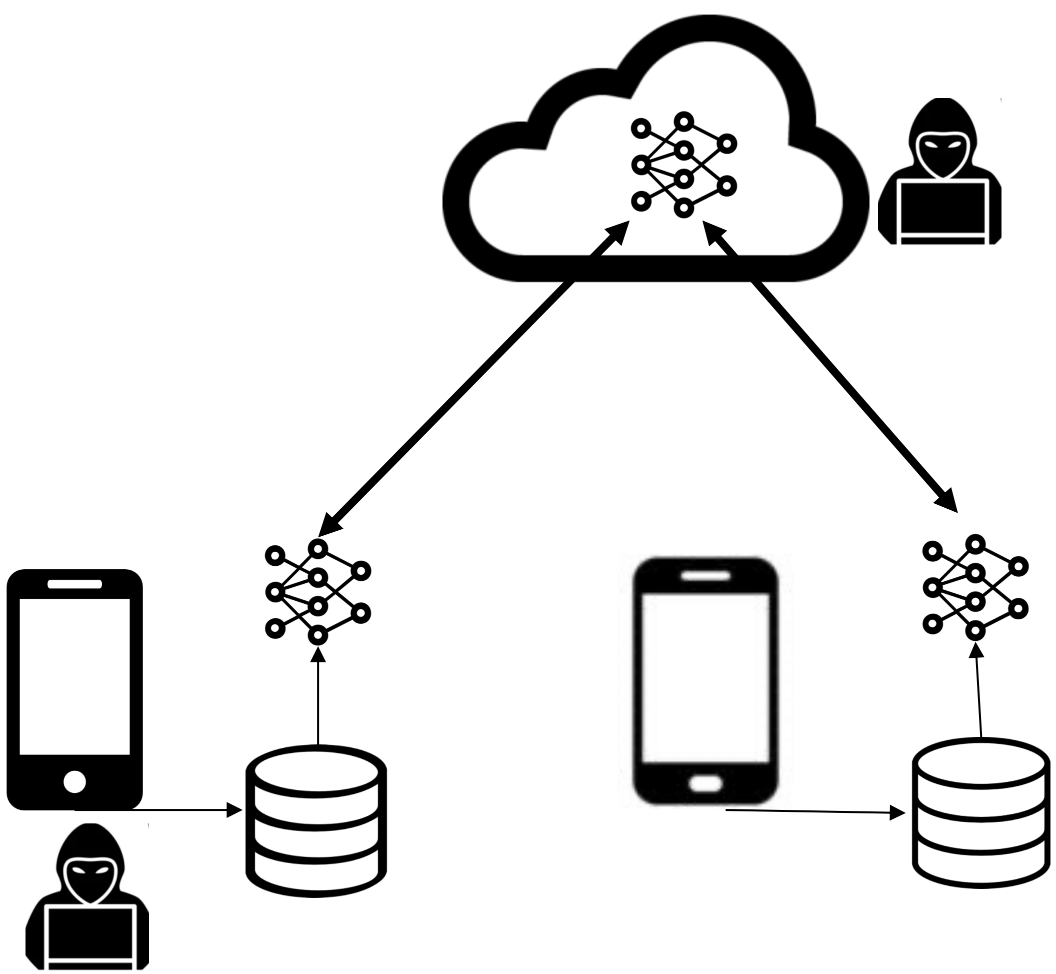
Differentially private Meta-learning

- The history of *Differentially Private* version gradients guides the current clipping.



Two Algorithms

- Two threat models
  - DP-AGR for threat model 1 where server is trusted, clients are honest-but-curious
  - DP-AGRLR for threat model 2 where the server is not trusted, and clients are honest-but-curious



Algorithm 3: DP-AGRLR (Client Side)

Input: Current global model Θ, local data $\mathcal{D}$, DP parameter (ϵ_0, δ_0), C_0, z_0

Output: gradient g

Function g = **Base-Model-Train**$(\Theta, \mathcal{D}^s, \mathcal{D}^q)$:

Initialize base-model: $\theta \leftarrow \Theta$;

Split local data $\mathcal{D}^s, \mathcal{D}^q \leftarrow \mathcal{D}$;

$z_0 \leftarrow$ compute_noise$(\epsilon_0, \delta_0, *args)$

for $(x_i, y_i) \in \mathcal{D}^s$ **do**

  record- level gradient: $g_i \leftarrow \nabla_\theta \mathcal{L}(\theta, x_i)$;

  clip gradient: $\hat{g}_i \leftarrow g_i * \min(1, \frac{C_0}{\|g_i\|})$;

$\tilde{g} \leftarrow \frac{1}{|\mathcal{D}^s|}\left(\sum_i \hat{g}_i + \mathcal{N}(0, (z_0 C_0)^2 \mathbf{I})\right)$;

update base-model: $\theta \leftarrow \theta - \eta_1 \tilde{g}$;

for $(x_i, y_i) \in \mathcal{D}^q$ **do**

  record-level gradient: $g_i \leftarrow \nabla_\theta \mathcal{L}(\theta, x_i)$;

  clip gradient: $\hat{g}_i \leftarrow g_i * \min(1, \frac{C_0}{\|g_i\|})$;

$g \leftarrow \frac{1}{|\mathcal{D}^q|}\left(\sum_i \hat{g}_i + \mathcal{N}(0, (z_0 C_0)^2 \mathbf{I})\right)$.

ing

VIRGINIA TECH

Experimental Setting

- Settings:
  - Image Datasets: Omniglot, CIFAR-100, mini-ImageNet
  - Client Number: 400,000
  - Clients in each learning round: 1500
  - Each client has 30 data records.
  - Meta-learning algorithm: MAML. (https://github.com/AntreasAntoniou/HowToTrainYourMAMLPytorch)

- Code:
  - Our code is available at https://github.com/ning-wang1/DPFedMeta.
  - Code Evaluated

Datasets (1/3)

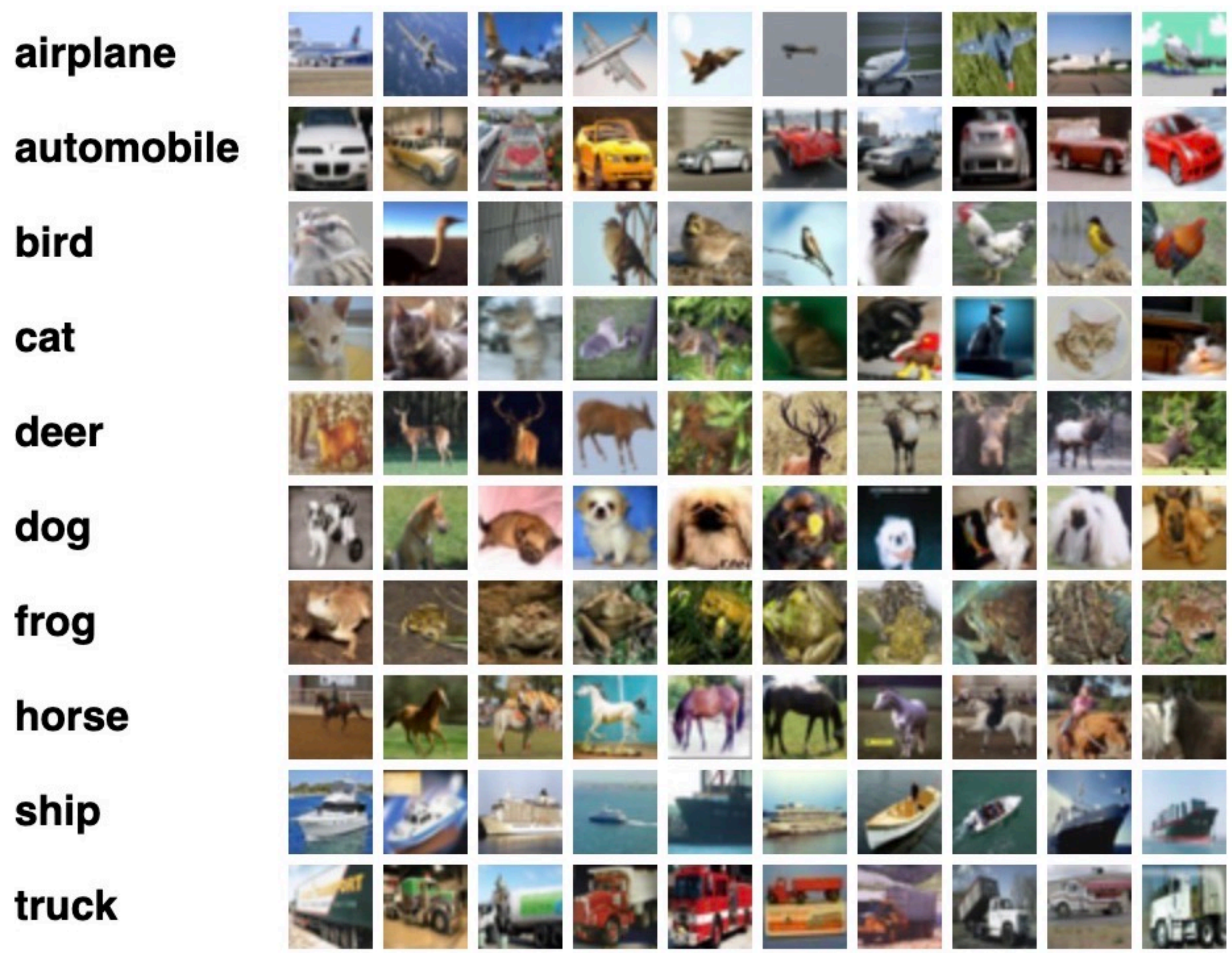
- Omniglot Dataset https://github.com/brendenlake/omniglot
- 1623 characters
- Each has 20 examples

Datasets (2/3)

- Mini-ImageNet Dataset https://github.com/yaoyao-liu/mini-imagenet-tools
- 100 classes
- Each has 600 examples

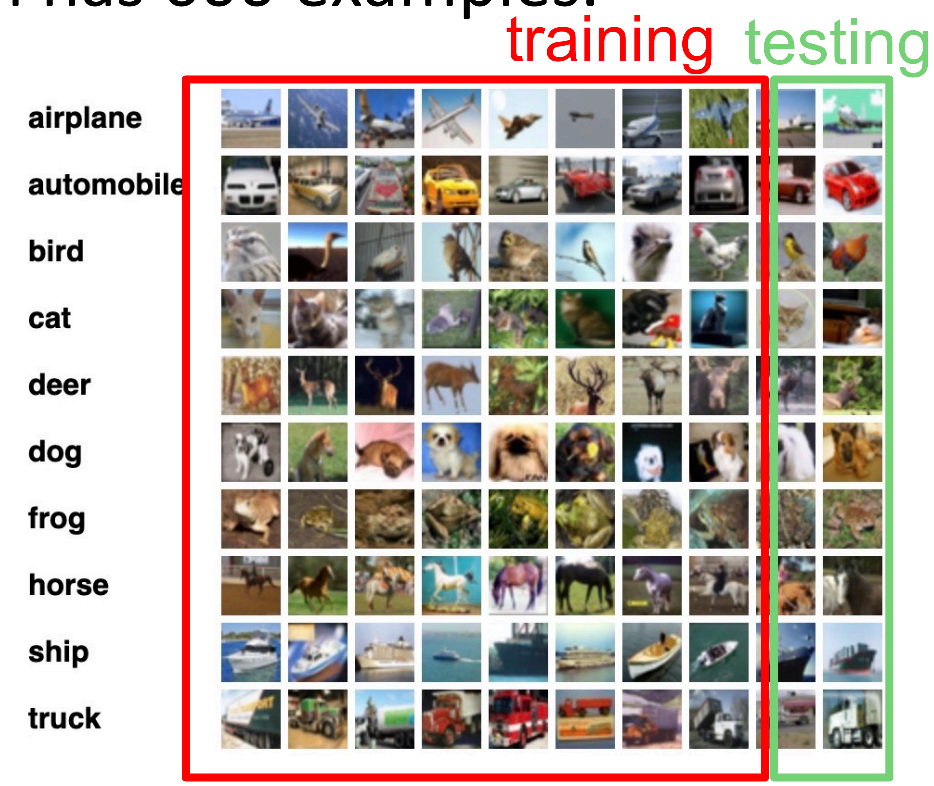
Datasets (3/3)

- CIFAR-FS Dataset https://github.com/bertinetto/r2d2
- 100 classes
- Each has 600 examples

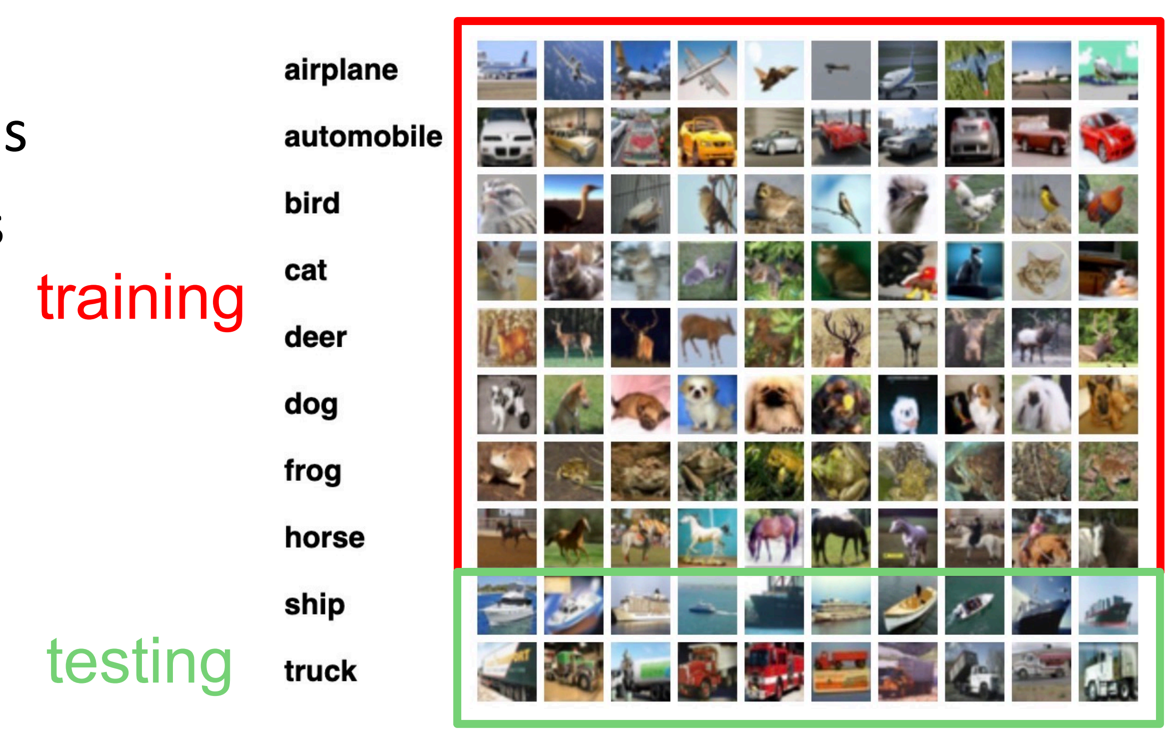


Data Split

- Use CIFAR dataset as example: 100 classes, each has 600 examples.
- A general image classification
  - Training: 100 classes, each has 500 examples
  - Testing: 100 classes, each has 100 examples



- Meta-learning
  - Training: 80 classes, each has 600 examples
  - Testing: 20 classes, each has 600 examples



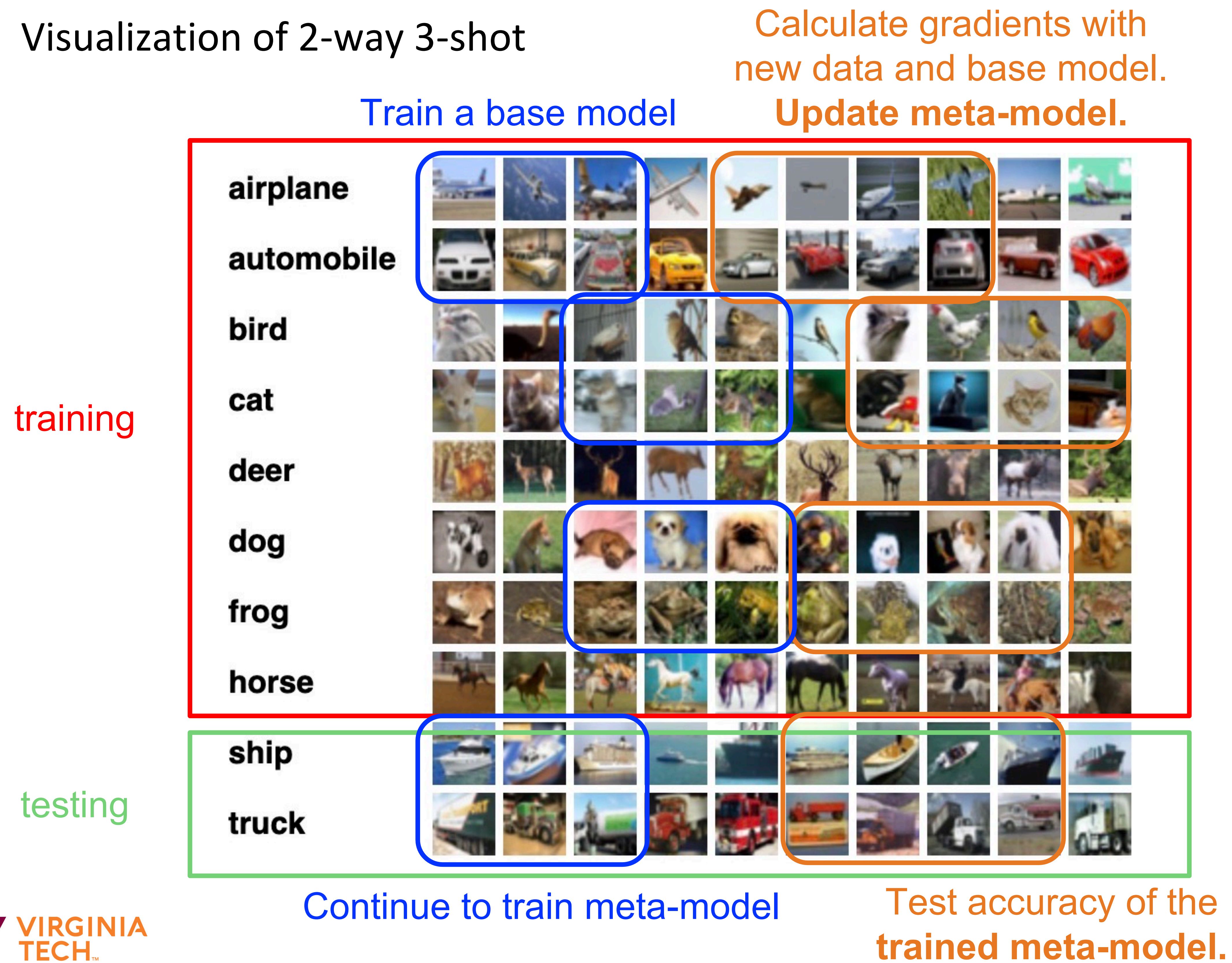
N-Way K-shot Task

- *N* denotes the number of classes.
- *K* represents the number of data records in each class.

- N-way K-shot meta-learning
  - Meta-Training:
    - Pick *N* classes, pick *K* records for each of the *N* classes, **learn a base model**.
    - Pick other records in the N classes to calculate gradients on the learned **base** model.
    - Use gradients to update **meta** model.
  - Meta-Testing
    - Pick *N* **unseen** classes. pick *K* records for each of the *N* classes. **Continuously train the meta-model** using these data.
    - Pick other records in the *N* classes to evaluate accuracy.

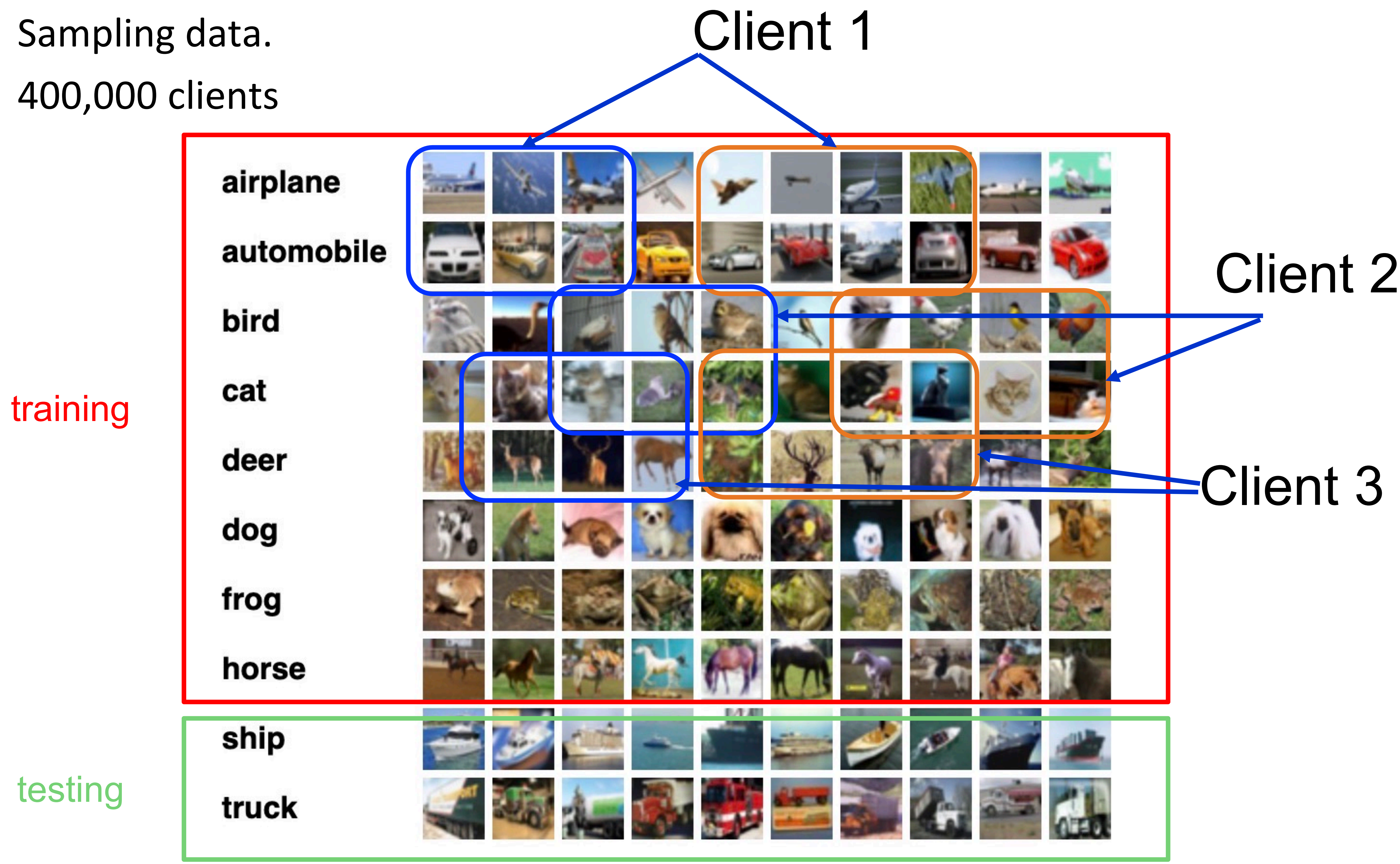
N-Way K-shot Task

- Visualization of 2-way 3-shot



Simulate Clients

- Sampling data.
- 400,000 clients



Deep learning Environment

- **Hardware:** a server equipped with a 3.3 GHz Intel Core i9-9820X CPU, three GeForce RTX 2080 Ti GPUs
- **Operating system:** Ubuntu 18.04.3 LTS (a different version of Ubuntu is ok if it support the Pytorch deep learning framework)
- **Deep learning framework:** Pytorch 1.4.0
- **Programming Language:** Python 3.6.10
- Other dependent library: https://github.com/ning-wang1/DPFedMeta/blob/main/environment.yml

Executable File | 179 lines (179 sloc) | 5.16 KB


```yaml
name: myenv
channels:
  - pytorch
  - conda-forge
  - defaults
dependencies:
  - _libgcc_mutex=0.1=main
  - _tflow_select=2.3.0=mkl
  - astor=0.8.1=py36h06a4308_0
  - blas=1.0=mkl
  - blinker=1.4=py36h06a4308_0
  - brotlipy=0.7.0=py36h27cfd23_1003
  - bzip2=1.0.8=h516909a_2
  - c-ares=1.17.1=h27cfd23_0
  - ca-certificates=2021.5.25=h06a4308_1
  - cachetools=3.1.1=py_0
  - cairo=1.16.0=h18b612c_1001
  - certifi=2021.5.30=py36h06a4308_0
  - cffi=1.13.2=py36h2e261b9_0
  - chardet=4.0.0=py36h06a4308_1003
  - click=8.0.1=pyhd3eb1b0_0
  - coverage=5.5=py36h27cfd23_2
  - cryptography=3.4.7=py36hd23ed53_0
  - cudatoolkit=10.1.243=h6bb024c_0
  - cycler=0.10.0=py36_0
  - cython=0.29.23=py36h2531618_0
  - dbus=1.13.12=h746ee38_0
  - expat=2.2.6=he6710b0_0
  - ffmpeg=4.0=hcdf2ecd_0
  - fontconfig=2.13.1=he4413a7_1000
  - freeglut=3.0.0=hf484d3e_1005
  - freetype=2.9.1=h8a8886c_1
  - gettext=0.19.8.1=hc5be6a0_1002
  - glib=2.58.3=py36h6f030ca_1002
  - google-api-python-client=1.7.11=py_0
  - google-auth=1.10.0=py_0
  - google-auth-httplib2=0.0.3=py_2
  - google-auth-oauthlib=0.4.4=pyhd3eb1b0_0
  - google-pasta=0.2.0=py_0
  - graphite2=1.3.13=hf484d3e_1000
```

Set up GPU for Deep Learning

- Goal
  - Enable Deep Learning libraries (e.g., Pytorch) talk to GPU.

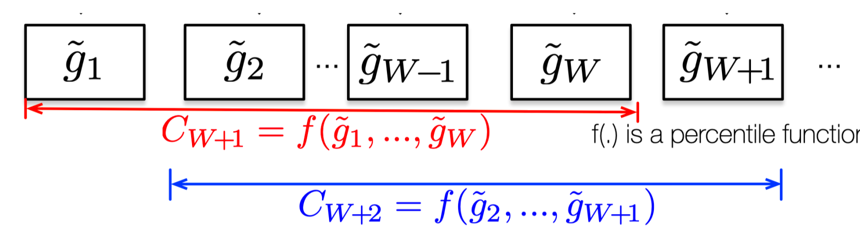
- Setting Up Steps:
  - NVIDIA Driver installation
  - CUDA installation
  - and CUDNN installation

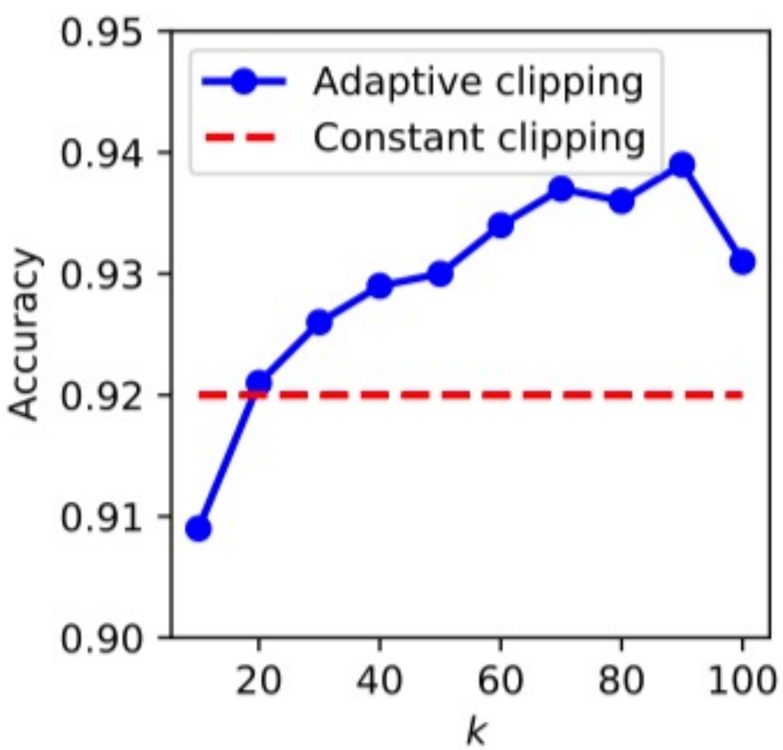
- Setting up guidelines are available on a blog
  https://towardsdatascience.com/deep-learning-gpu-installation-on-ubuntu-18-4-9b12230a1d31

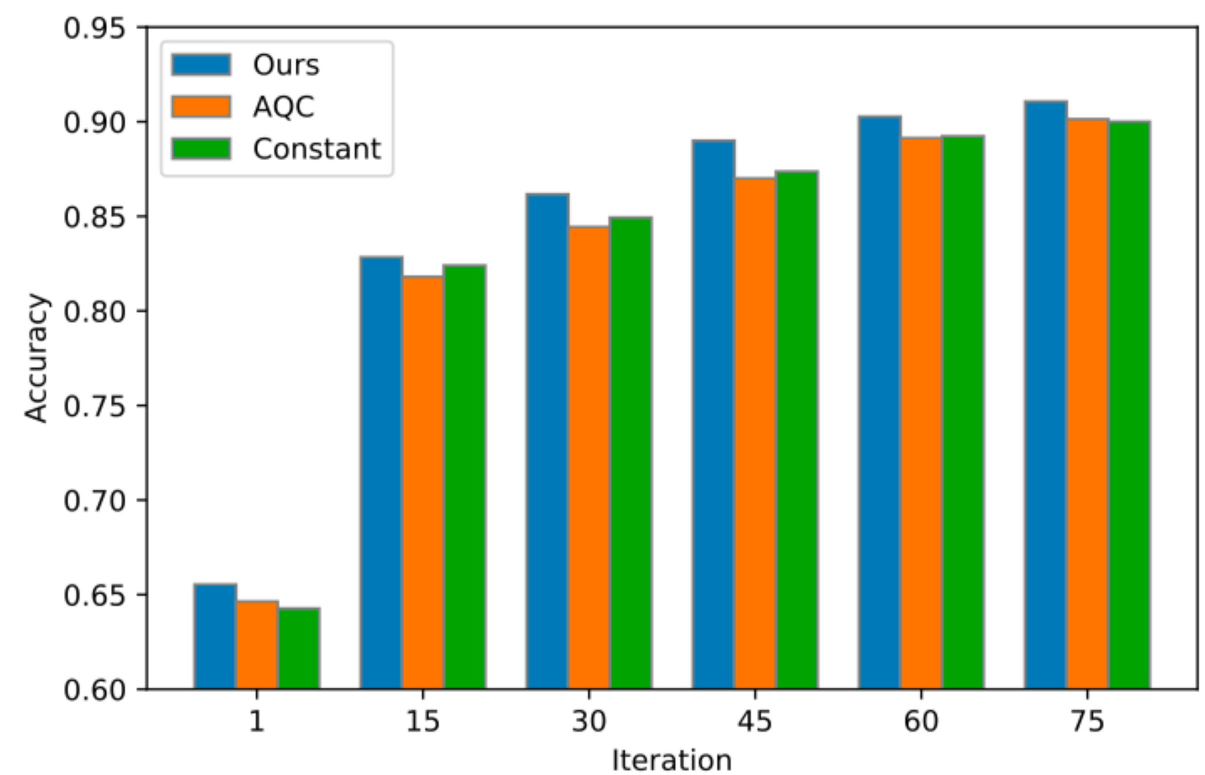
Adaptive Clipping Percentile k

- The adaptive clipping threshold at time step $t + 1$ is computed with a sequence of differentially private version of gradients before $t + 1$ (i.e. , $\tilde{g}_{t-W+1}, \tilde{g}_{t-W+2}, ..., \tilde{g}_t$) by

$$C_{t+1} := f(\{\tilde{g}_{t-W+1}, ..., \tilde{g}_t\}, k)$$



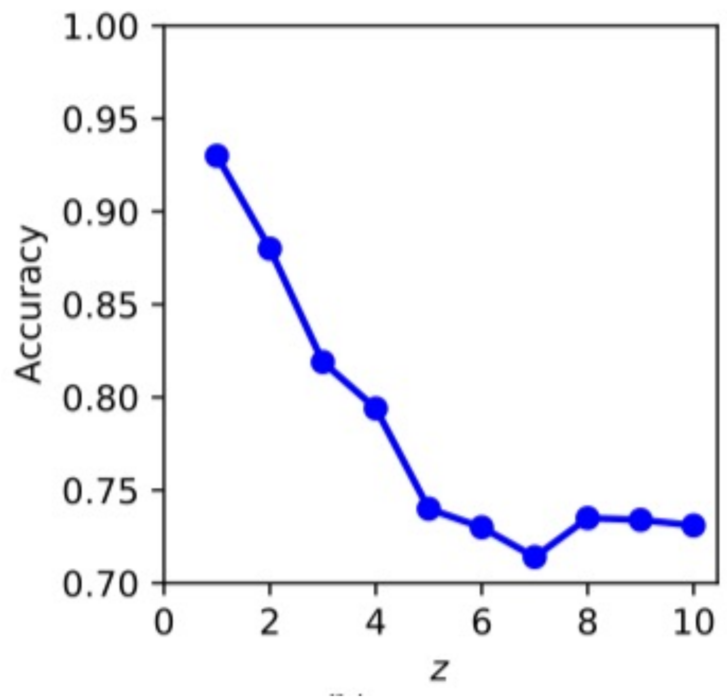




All other settings are the same, only change the clipping method.

Noise Level z

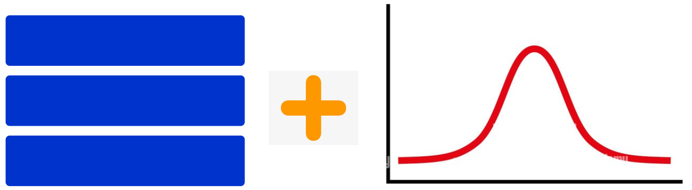
- Small Noise will consume the privacy budget quickly, so learning iterations will be limited.
- Larger Noise will cover useful gradients.
- Explore Trade-off.

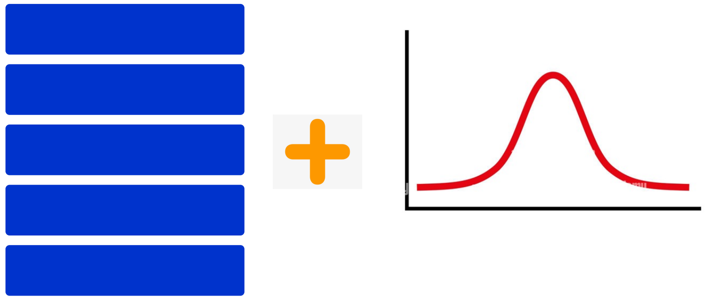


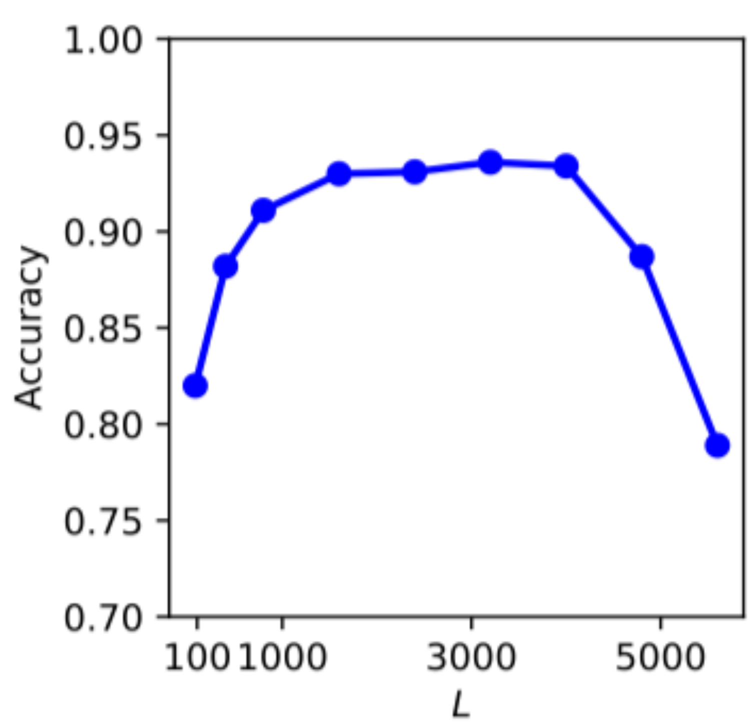
Noise=1 get the best accuracy.
It indicates learning iteration is not a key limitation on the used dataset.

Client Sampling Number L

- With small sampling number, noise may cover the gradients.
- With large sampling number, algorithm will reach the privacy leakage threshold quickly, so learning iterations will be limited.
- Explore Trade-off

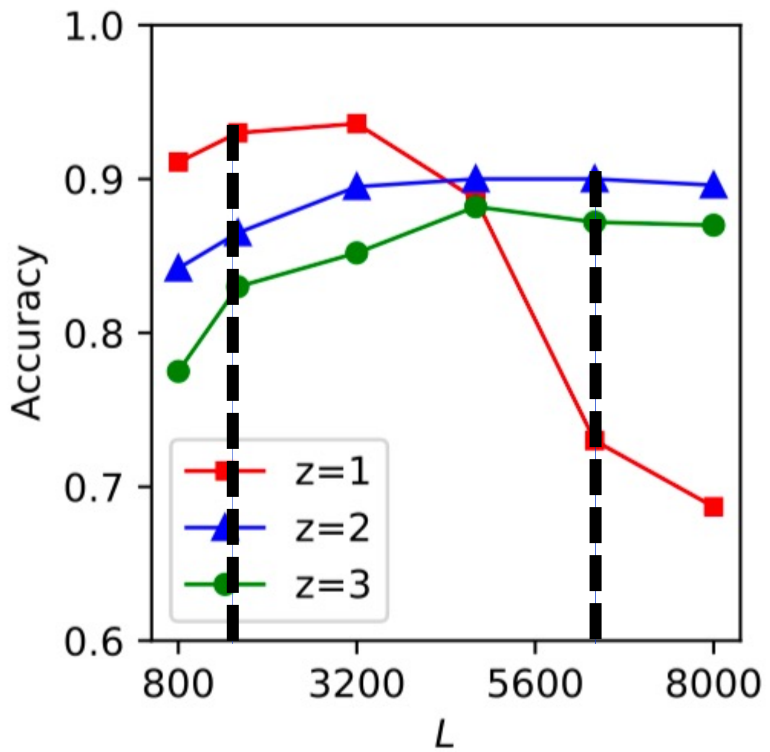






L=1600 get the best accuracy.

Explore Client Sampling Number L and Noise z Together



- When L is small,
  - The smaller noise get better accuracy.
  - Because larger noise may cover the gradients.

- When L is large enough,
  - The smaller noise get extremely low accuracy.
  - Because the combination of large L and small z will reach the privacy leakage threshold quickly, so learning iterations will be limited.

VIRGINIA TECH.

System Evaluations over Baselines

- privacy budget
  - DP-AGR achieves $(1.5, 10^{-6})$-DP;
  - DP-AGRLR achieves $(2.5, 10^{-5})$-DP for record-level privacy
  - Baseline achieves $(9.5, 10^{-3})$-DP
- Accuracy

Table 2: Meta-testing accuracy (%) with DP-AGR, DP-AGRLR and other baselines.

Dataset	N-way K-shot	Random initial	Non-private	Private Algorithm		
				DP-AGR	DP-AGRLR	GBML [14]
Omniglot	5-way 1-shot	49.2	99.4	93.9	72.4	44.6
	5-way 5-shot	61.0	99.8	96.8	89.7	75.0
CIFAR-FS	5-way 1-shot	33.8	61.0	47.1	39.0	32.2
	5-way 5-shot	45.4	78.6	58.2	49.2	48.6
Mini-ImageNet	5-way 1-shot	23.3	51.7	37.3	27.7	26.1
	5-way 5-shot	24.2	65.3	48.8	33.2	38.0

Hardware Evaluation Results

- We are running the code on a server equipped with a 3.3 GHz Intel Core i9-9820X CPU, and a GeForce RTX 2080 Ti GPU. The running time of DPAGR

Table 3: Per-task computation time

Dataset	MAML	DP-AGR	DP-AGRLR
Omniglot	39.9ms	54.7ms	0.52s
CIFAR-F	68.7ms	81.3ms	1.06s
Mini-ImageNet	112.3ms	102.7ms	1.17s

- DP-AGR achieves comparable computational performance with the original non-private MAML algorithm.
- DP-AGRLR is more time-consuming due to the need for computing per-record gradients.

Discussion & Meta Questions

Base Code

- A baseline Meta-learning algorithm: MAML. (https://github.com/AntreasAntoniou/HowToTrainYourMAMLPytorch). It’s a centralized non-private meta-learning algorithm.
- The privacy evaluation library, moments accountant: https://github.com/tensorflow/privacy

Reproduced Baselines

- We reproduced the results of Base code as one baseline.
  - MAML: https://github.com/AntreasAntoniou/HowToTrainYourMAMLPytorch.
  - Their results are reproduced.
- For the other baseline GBML [1], code is not published.
  - We reimplemented their methods.
  - Our produced results were different from their reported results, some with higher accuracy while some with lower accuracy.
  - We used two common datasets with GBML. For the two datasets, I ended up copying their results published in their paper.
  - For another dataset, we report the results evaluated by our reimplementation.

[1] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. 2020. Differentially Private Meta-Learning. In International Conference on Learning Representations.

Lessons Learned

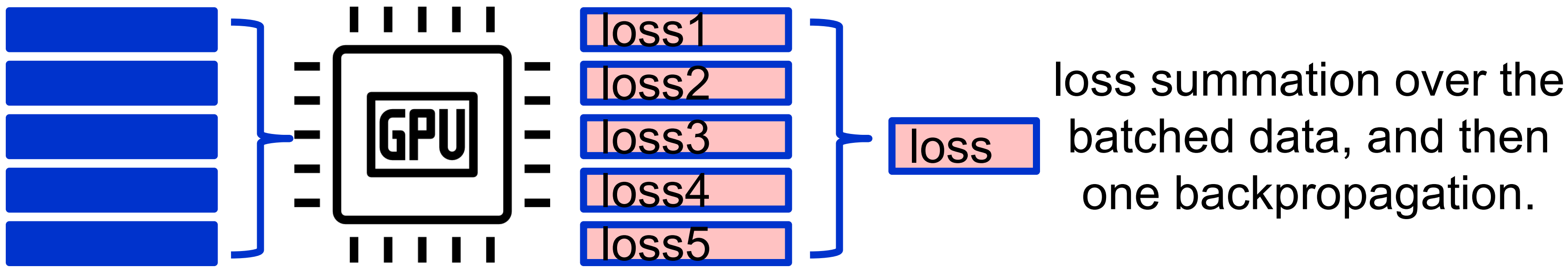
- Guidelines for k,z, and L
  - First, we recommend to start from a small noise multiplier z (e.g., 1) and increase z only when you can not guarantee convergence before using up the privacy budget.
  - Second, we recommend starting with a relatively large L especially when z is large.
  - We can decrease L only when you can not guarantee convergence before using up the privacy budget.
  - Compared with the non-private training, we need apply a larger learning rate since the training rounds are limited because of privacy concerns.
  - Finally, as privacy parameter ϵ is only determined by z and L, we can adjust other parameters, such as k, to boost the model accuracy.

Slow Training Problem

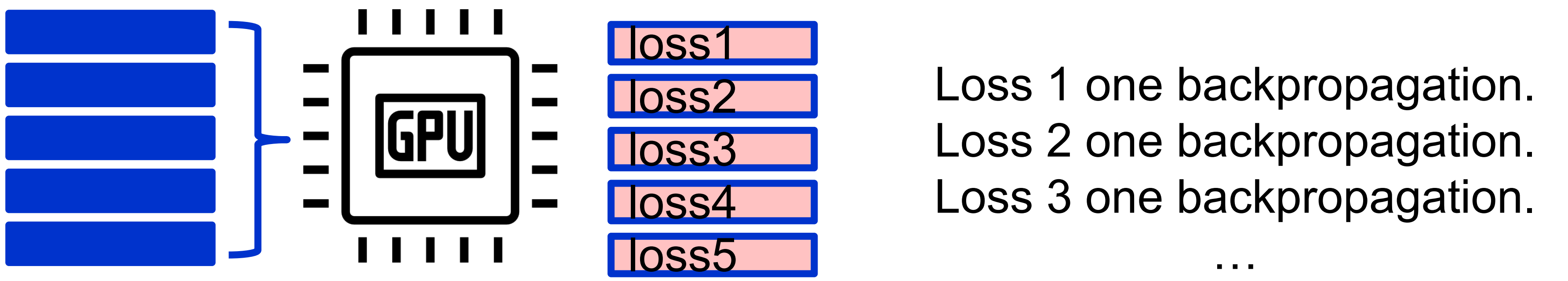
- Problem
  - The training time of DP-AGRLR for 400,000 tasks is over 30h.
- Reason
  - Per-record gradient calculation is time-consuming, and it’s an open problem.

Slow Training Problem

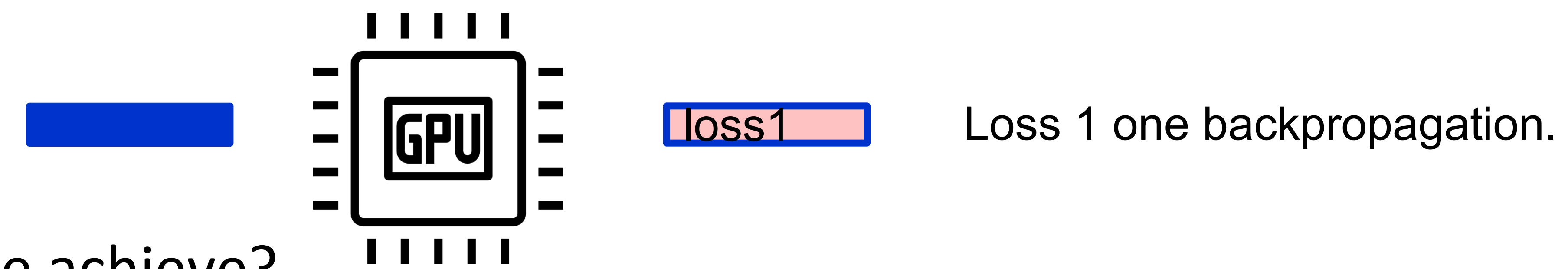
- How to deal?
  - GPU speeds up gradient calculation for **batched** data.



  - If we still load a batch of data, but calculate gradient one record by another. There will be much I/O between GPU and CPU.



  - Loading one record a time.



- What did we achieve?
  - Training time from 30h to 6h.

Wrap up discussion

VIRGINIA TECH

Summary

- Differentially private federated meta-learning architecture.
  - Parameter tunning is time-consuming
  - We should reason the tuning directions beforehand but not tune randomly.
  - Good trade-off can be found.
- Understand how GPU speeds up to avoid wrong configurations.

Future Direction

- Further improve the time efficiency by using state-of-the-art single-record gradient acceleration techniques [1].
- Differential Privacy is represented by two parameters (ε, δ).
  - It is not straightforward to understanding how good the privacy is.
  - We plan to implement privacy attack.
    - Membership inference attack [2,3]
    - Model Inversion Attack [4]
    - Attribute Inference Attack [5]
  - Attack on differentially private model and non-private model.
  - The attack success rate can be an indicator for the privacy protection level.

[1] Lee, Jaewoo, and Daniel Kifer. "Scaling up differentially private deep learning with fast per-example gradient clipping." *Proceedings on Privacy Enhancing Technologies* 2021.1 (2021)
[2] Milad Nasr et al. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE Symposium on Security and Privacy (SP 19). IEEE, 739–753.
[3] Jingwen Zhang, Jiale Zhang, Junjun Chen, and Shui Yu. 2020. Gan enhanced membership inference: A passive local attack in federated learning. In ICC 20202020 IEEE International Conference on Communications (ICC). IEEE, 1–6.
[4] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In IEEE Conf. on Computer Communications (INFOCOM). IEEE, 2512–2520.
[5] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. 2009. Learning your identity and disease from research papers: information leaks in genome wide association study. In Proceedings of the 16th ACM conference on Computer and communications security (CCS). ACM, 534–544.

Thank You!
Q&A