Mobile & Ubiquitous Computing, 2020 Spring

FedEx: Scalable Framework for Mobile User-Adaptive Deep Learning Training

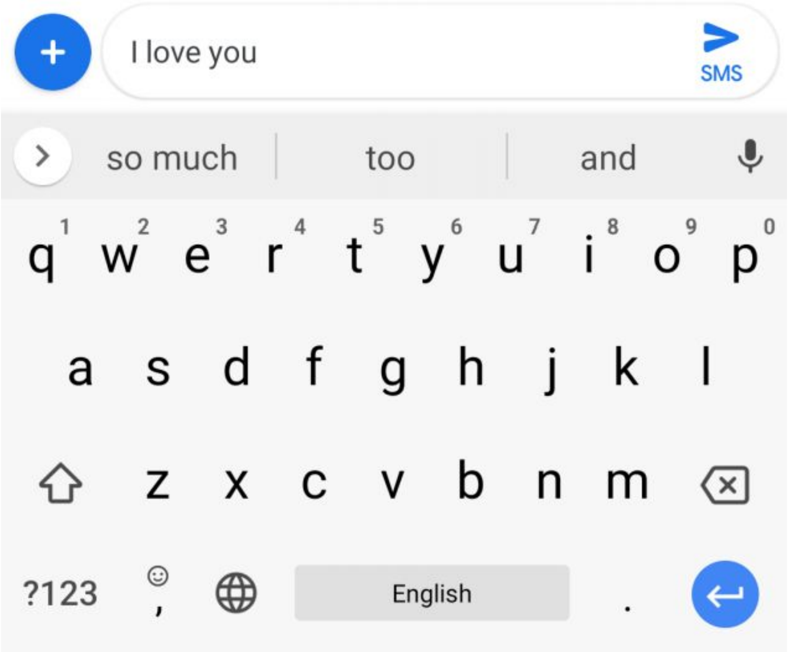
{Ahnjae Shin, Kyunggeun Lee, Heeseung Yun, Taebum Kim} @ Team 1

Two misconceptions on

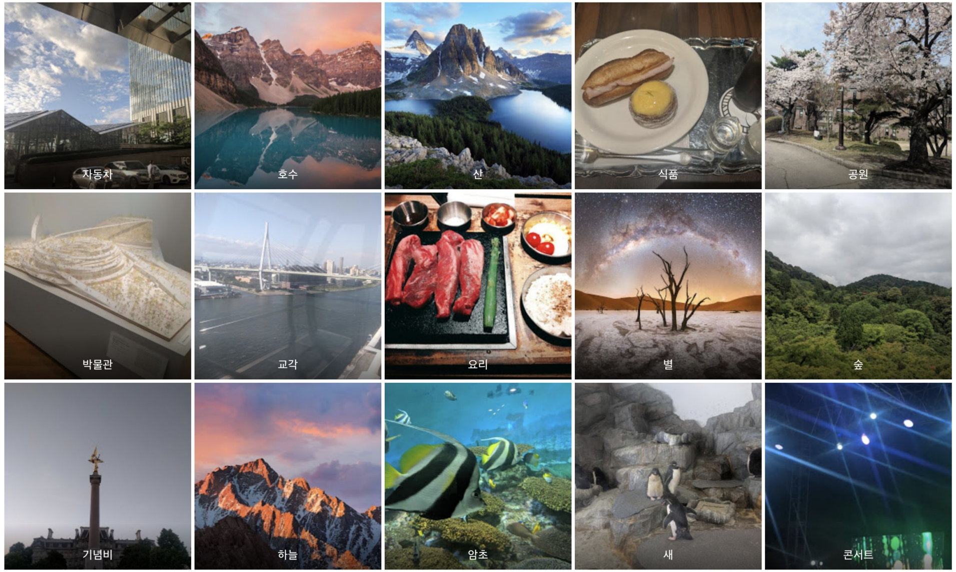
ML driven mobile apps

Misconception 1:

Mobile ML apps are **static**



gboard



google photos

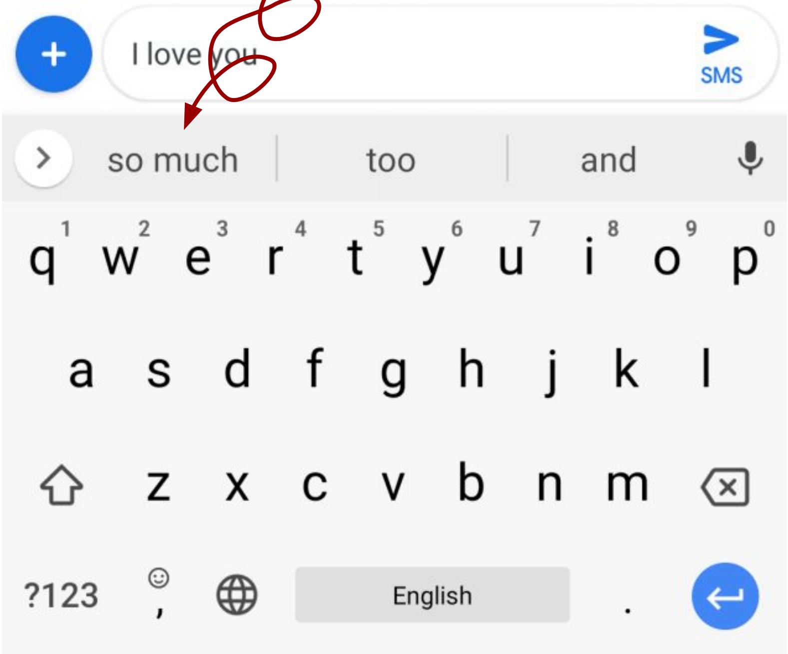
Truth 1:

Mobile ML apps are **dynamic**

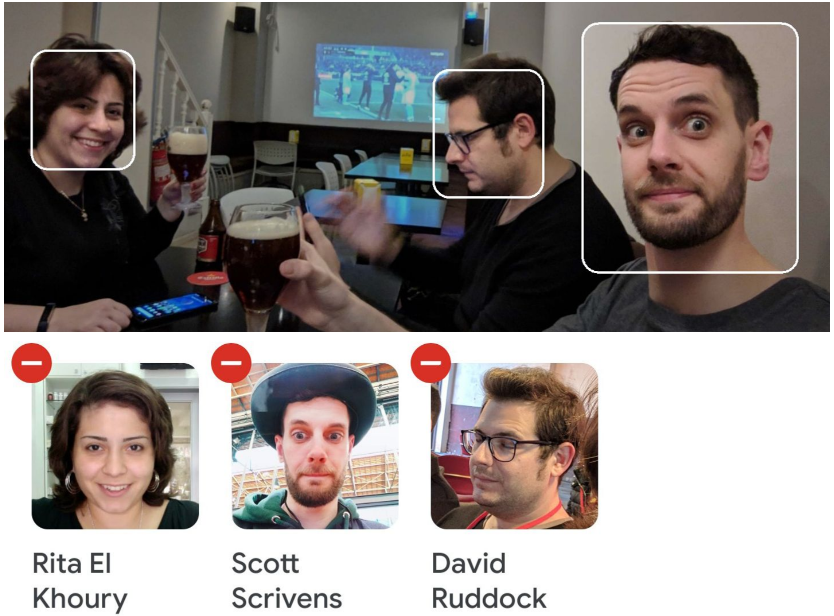
Misconception 2:

Users do not provide **annotated** data

click which one?



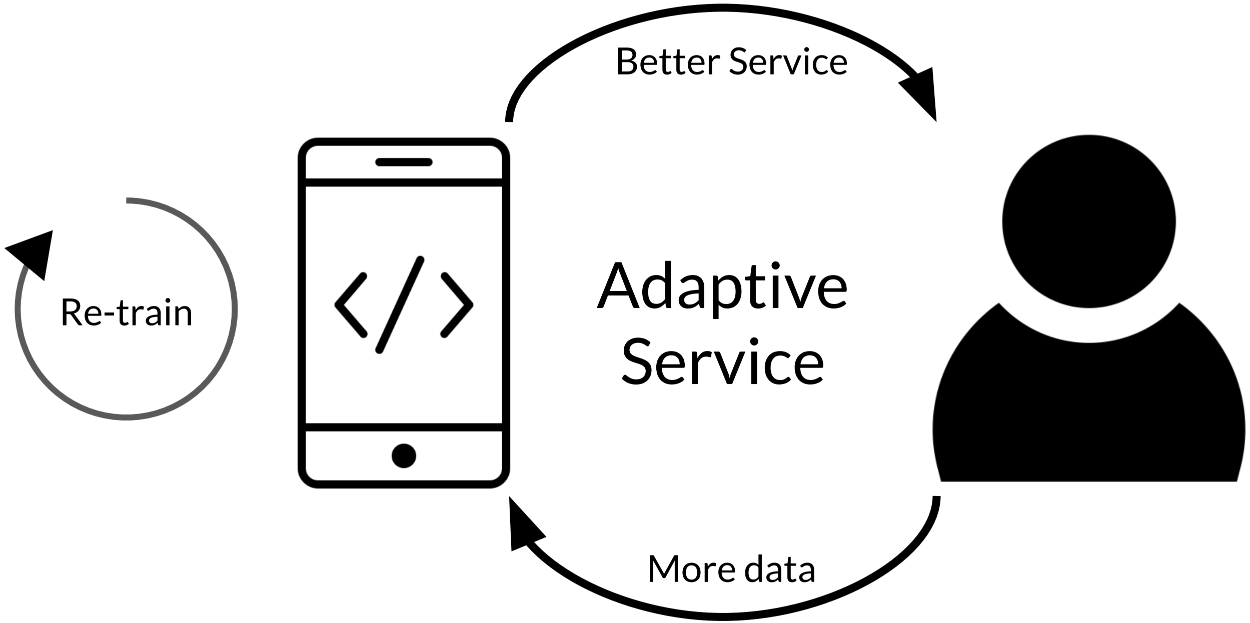
Preference

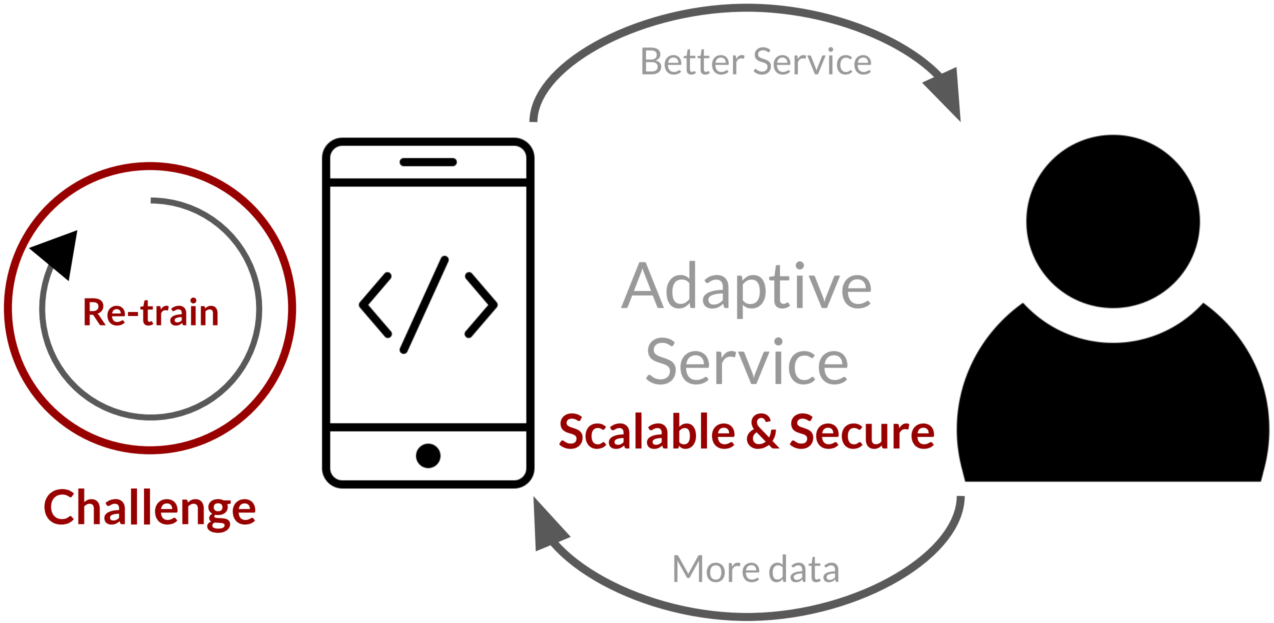


Correction

Truth 2:

User feedback is a data to **train**


Better Service
Re-train
Adaptive
Service
More data


Better Service
Re-train
Adaptive
Service
Scalable & Secure
Challenge
More data

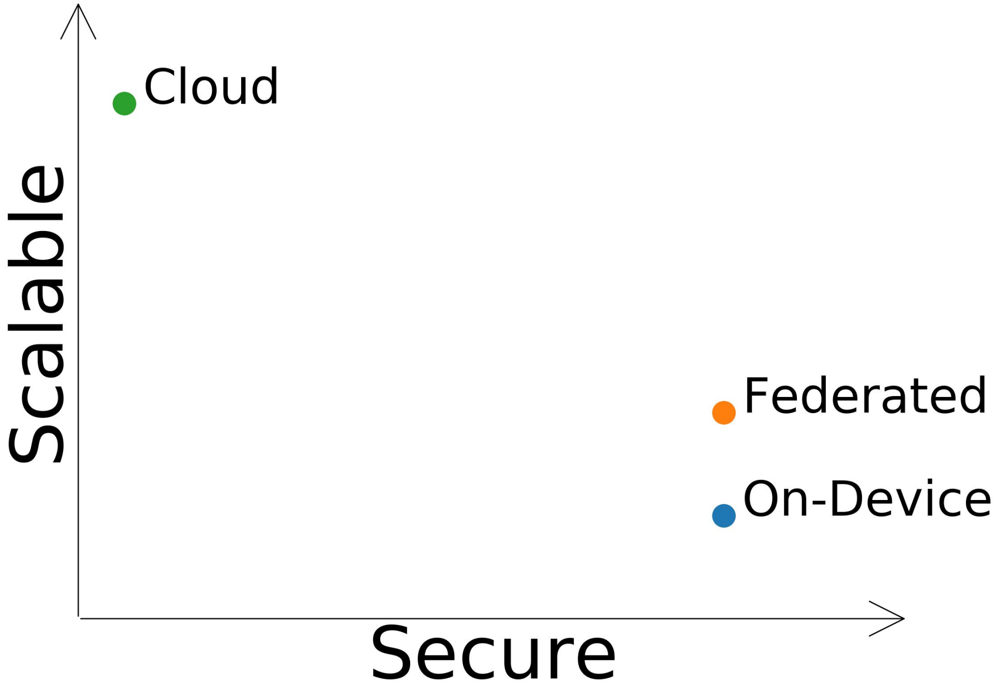
Scalable Able to scale to large models

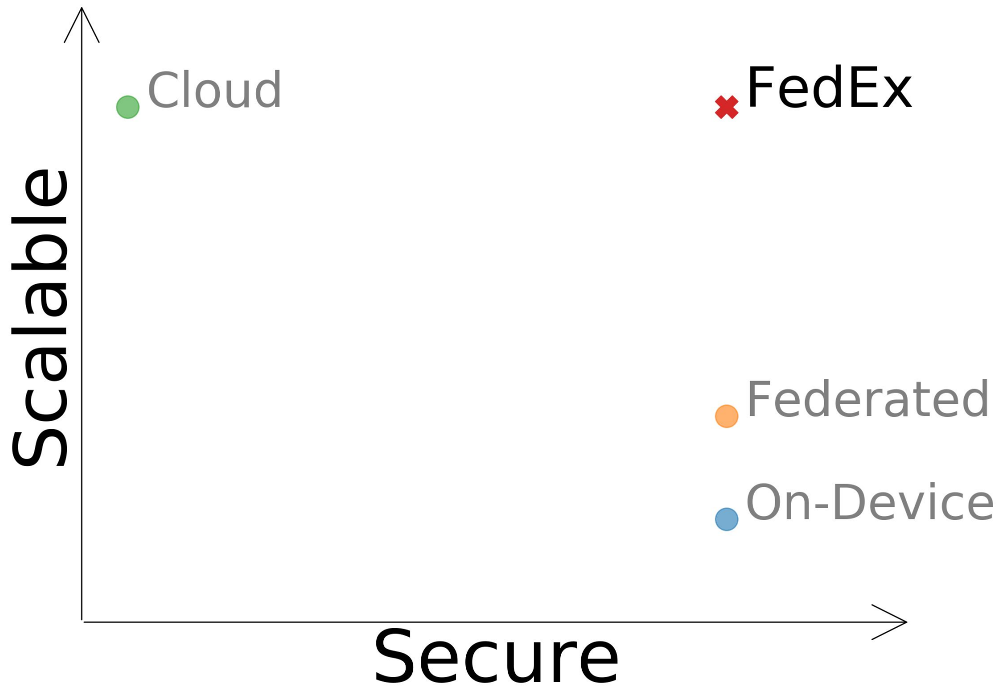
Secure Able to train without raw data

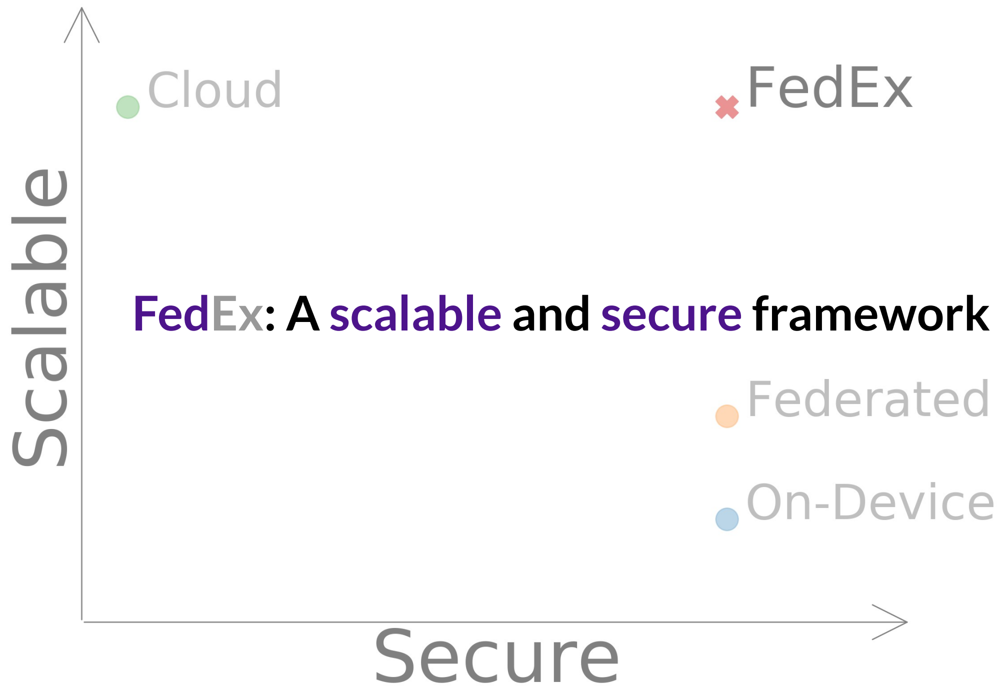
Previous Solutions

1. On-device learning (Kang et al.)
    a. Use of local resource to train model
    b. Big pressure of energy usage
    c. Limited to small models
2. Federated learning (McMahan et al.)
    a. Aggregation of gradients via network
    b. Limited computation and network resource
    c. Limited to small models (e.g. MNIST + CNN with only two 5x5 conv layers)
3. Cloud learning (Lee et al.)
    a. Use cloud resource to train model
    b. User data should be uploaded to cloud
    c. Has security risk

Previous Solutions

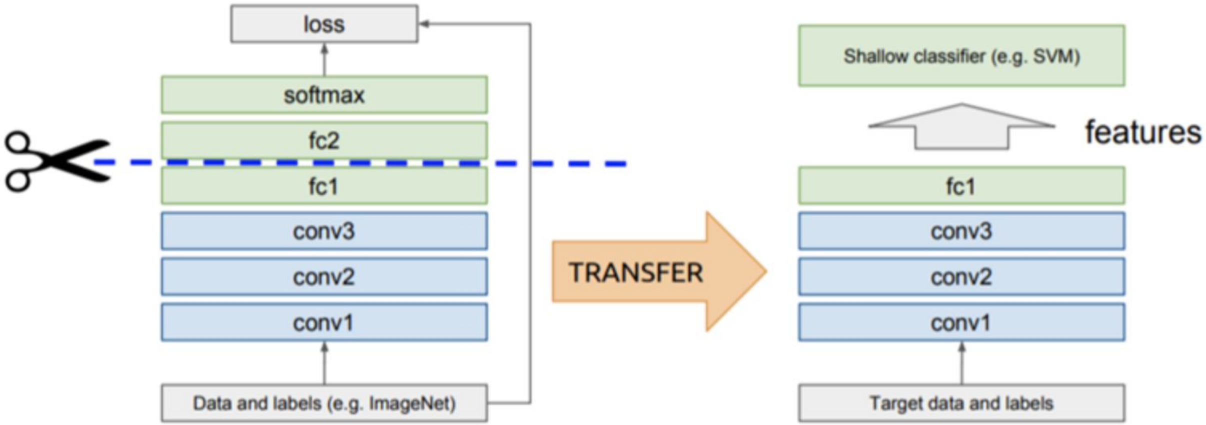



Cloud
FedEx
Federated
On-Device
Scalable
Secure


Cloud
FedEx
FedEx: A scalable and secure framework
Federated
On-Device
Scalable
Secure

Key Insight

- When we use user data (small) to update original ML model (large data), we use transfer learning paradigm
  - Freeze the model up to certain point, and do not update weights of frozen layers
- Not every model weight is updated every time.
  ⇒ Divide updating model weights into *whole update* and *partial update*



Solution

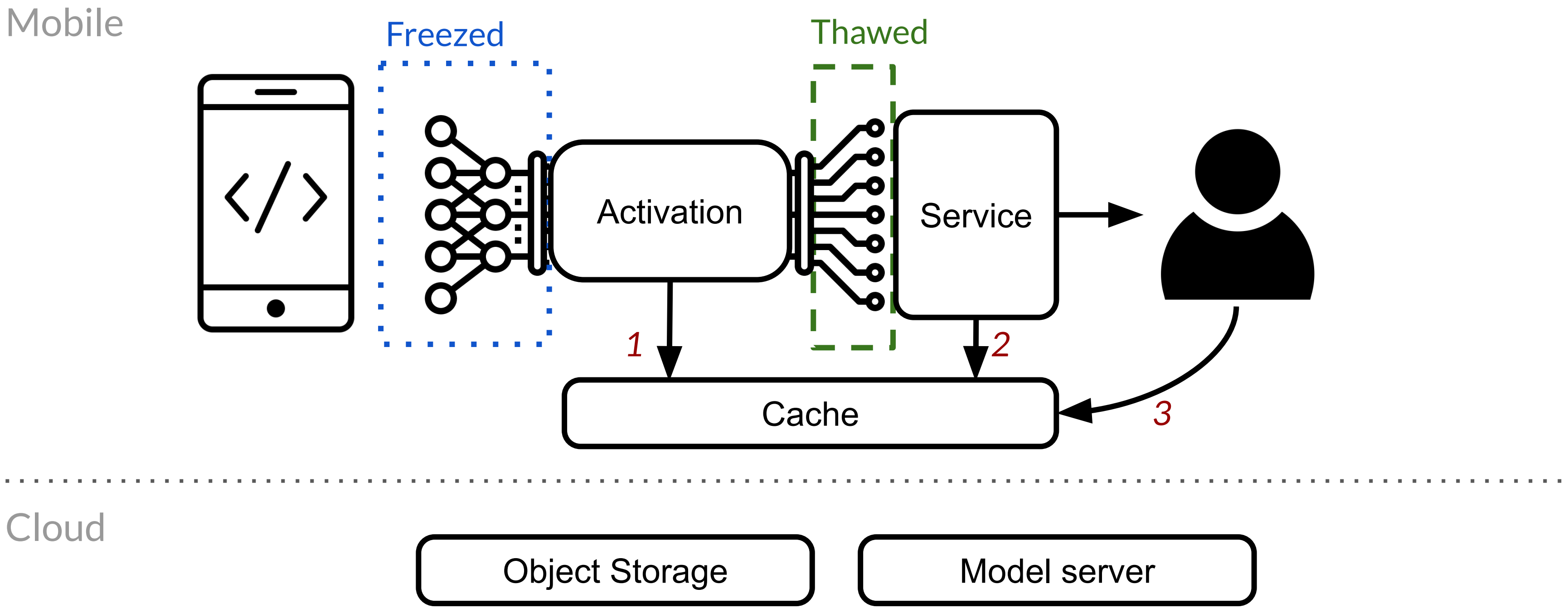
FedEx: A framework that provide updates via transfer learning

- Cache activation vector
  - Each activation have timestamp
  - Evict entry if activation timestamp is before current model's timestamp
  - Evict entry if user annotation is not provided, and app is terminated
- Transparently update model
  - Mobile device periodically pull model. It compare weight hash with previous one.
- Sync # of freezed layers (chosen by cloud)
  - Cloud store activation values and compute new activation if possible, or broadcast mobile devices notification
  - Evict entries that have different hyper-parameters
  - Mobile device uses activation checkpoints to avoid recomputation

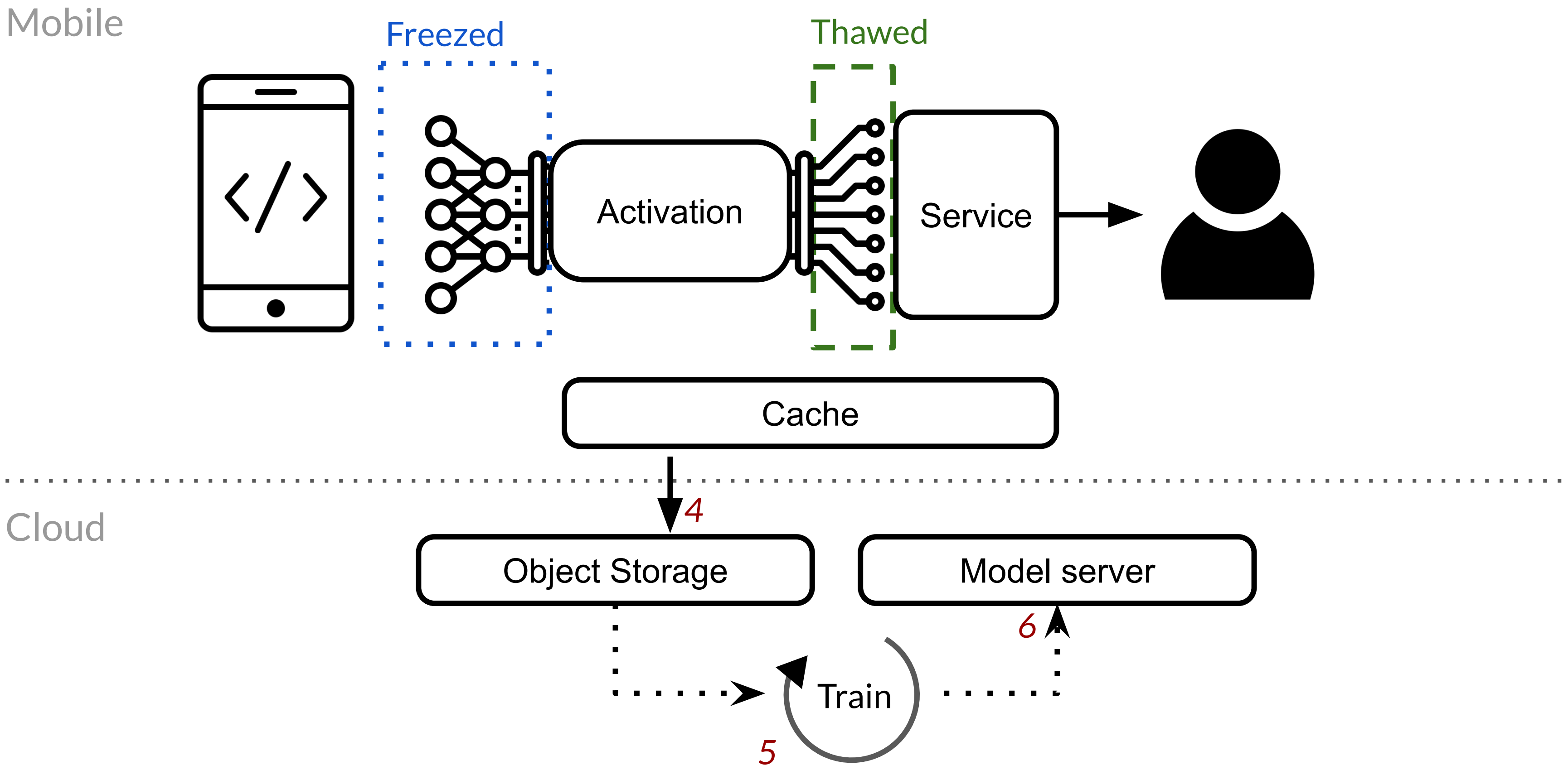
Optimization

1. **Performance**
   a. Retain cached activation value for frequent predictions
   b. FedEx does not change former part of model
2. **Energy**
   a. Only communicate with cloud if battery levels are okay

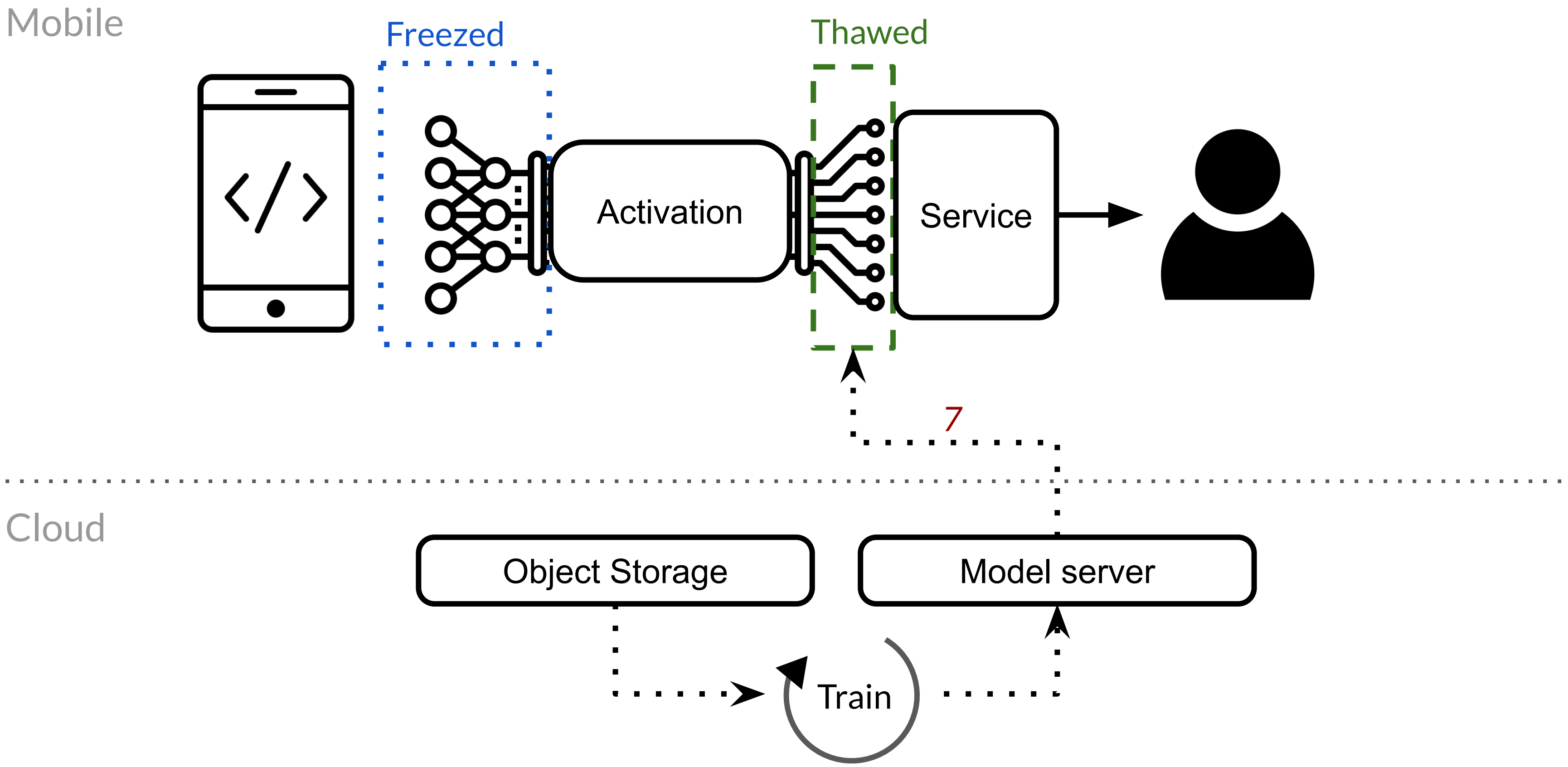
System Architecture



System Architecture



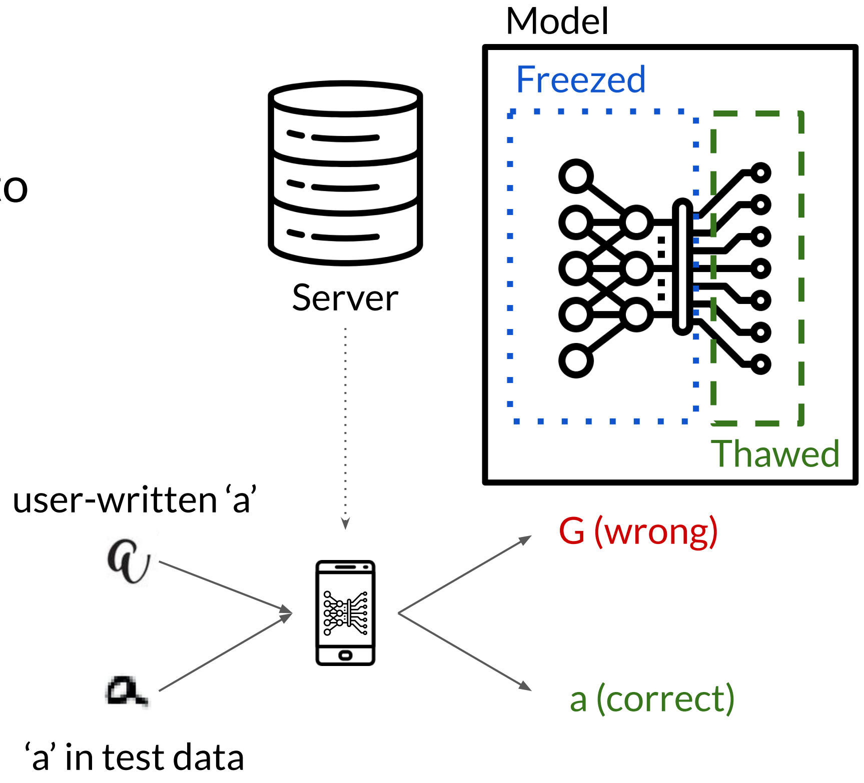
System Architecture



Example app: Handwriting Recognition

Initial Deployment

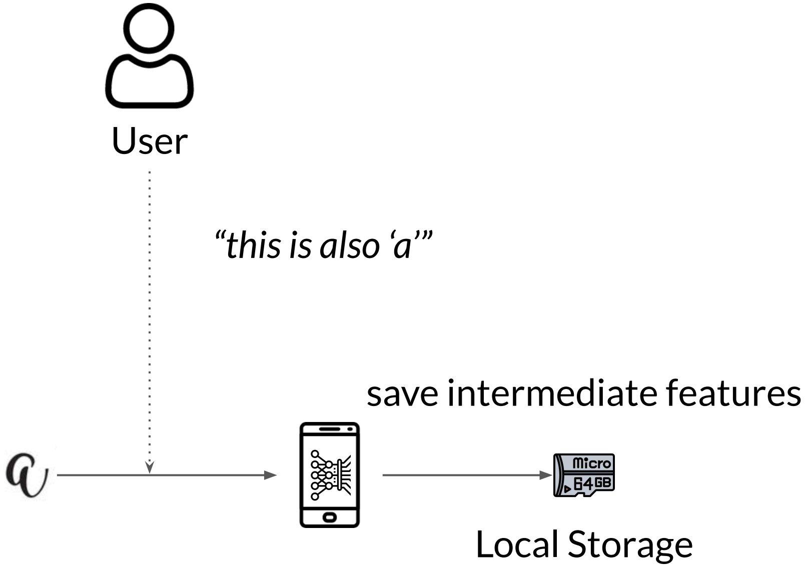
- Generally trained model is deployed into mobile device
- Network structure
  - freezed backbone part (non-trainable)
  - non-freezed (thawed) part (trainable)
- Model performance may not good enough



Example app: Handwriting Recognition

User Specific Data Collection

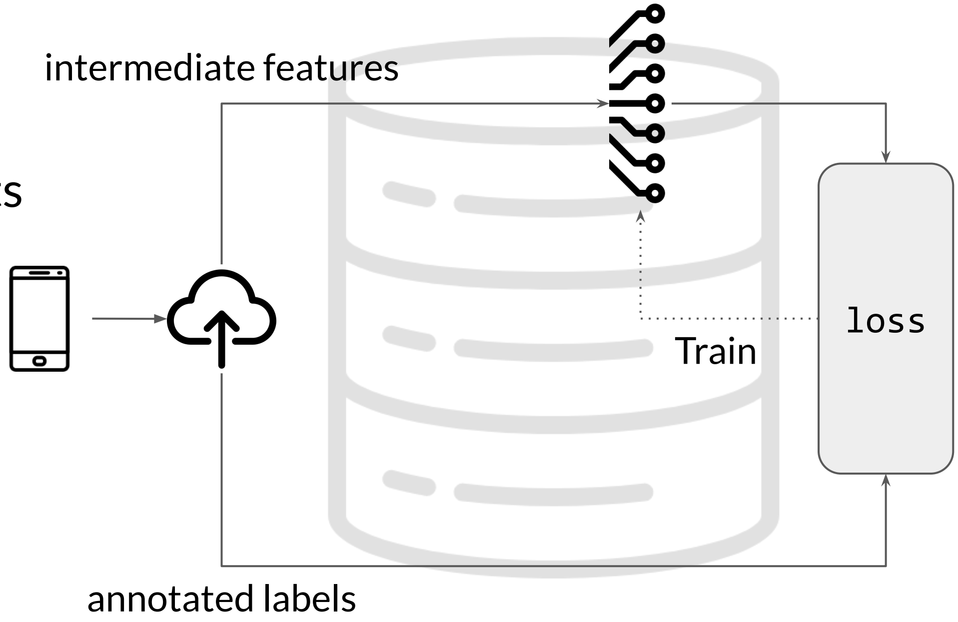
- User can annotate their own data
- Each data is fed into freezed backbone
- Save corresponding intermediate features in local storage



Example app: Handwriting Recognition

Additional Training at Server

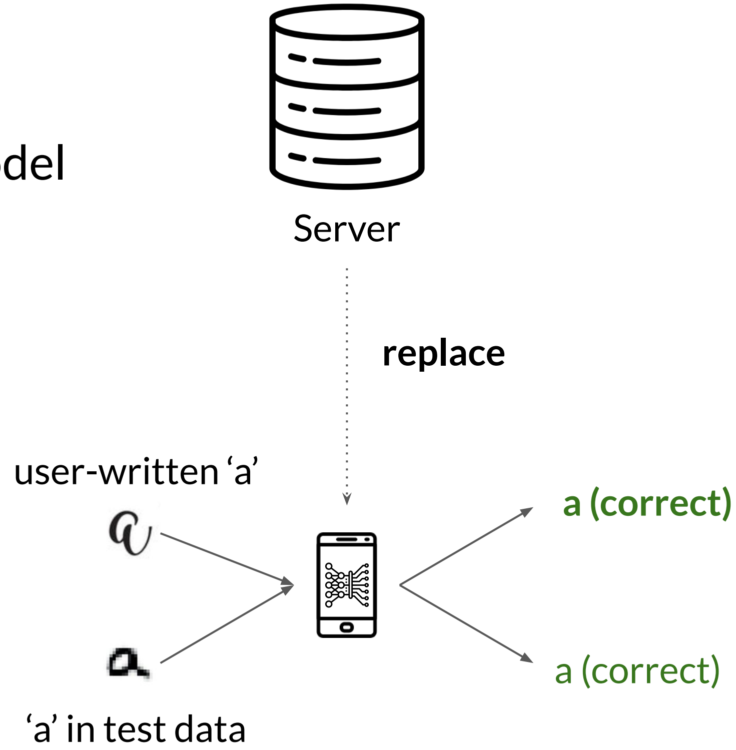
- Periodically, flush collected data to the server (cloud)
- Using collected data, server conducts additional training



Example app: Handwriting Recognition

Model Replacement

- after additional training, replace previous model to the new model
- model performance becomes better

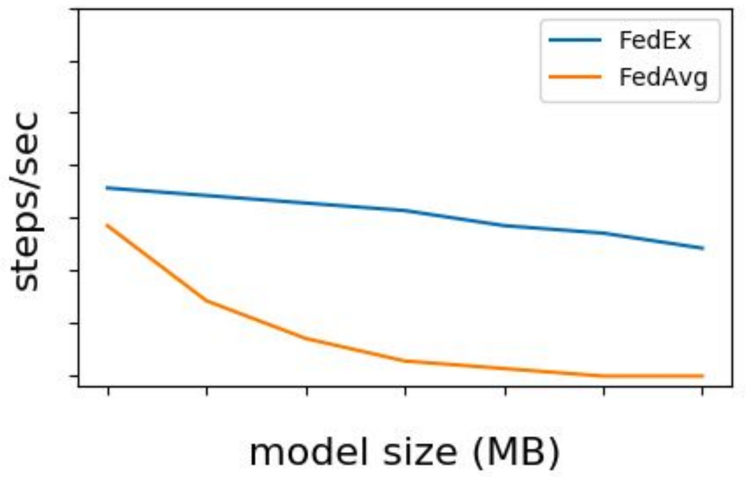


Evaluation Strategy

- **Scalable**: FedEx training scales with large models
- **Efficient**: FedEx uses less energy

Evaluation 1: Scalability

- Hardware setup
  - A server equipped with high-end GPUs
  - 1 android device
  - Demo handwriting recognition app
- Baseline: FedAvg (McMahan et al.)
- Metric: Training throughput (steps / sec)
- Expected results
  - Higher training throughput, especially as the model size gets larger.
  - Similar scalability to user



Evaluation 2: Efficiency

- Hardware setup
  - A server equipped with high-end GPUs
  - 1 android device
  - Demo handwriting recognition app
- Baseline: FedAvg (McMahan et al.)
- Metric: Total power needed to train model (mW)
- Expected results
  - FedEx uses less energy

Micro-benchmarks

- Scalability w.r.t. multiple clients
  - Simulate multiple devices on cloud.
  - Evaluate throughput w.r.t. number of clients
- Performance breakdown
  - How much energy is used in communication

Overall Plan

#Iter.	Objective	Duration	Misc.
1	**Ideation & Design** Brainstorming, Literature Review, Proposal Feedback	4/1 - 4/14	4/6: Proposal
2	**System & Architecture** Cloud Setup, System Design, Client Mockup	4/15 - 4/28	
3	**Model & Algorithm** Data Preparation, Model Implementation, Integration	4/29 - 5/12	5/11: Project Demo
4	**Optimization & Experiment** Ablation Studies, Alpha Deployment, Load Testing	5/13 - 5/26	
5	**Deployment & Deliverable**	5/27 - 6/8	6/8: Final Presentation

Deliverable

- Midterm deliverable
  - Proof of concept via cloud
  - Mobile device will be simulated on cloud as a rpc server
- Final Deliverable
  - Handwriting recognition app
  - Technical report regarding evaluations
- Success Criteria
  - Our method succeeds to train larger model faster than baseline method
  - Our method succeeds to use less energy than baseline method

We ship fresh models daily

Thank you

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data”, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017*.

[2] D. Kang, E. Kim, I. Bae, B. Egger and S. Ha, "C-GOOD: C-code Generation Framework for Optimized On-device Deep Learning," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2018.*

[3] K. Lee, K. Lee H. Kim, C. Suh and K. Ramchandran, “SGD on Random Mixtures: Private Machine Learning under Data Breach Threats”, *International Conference on Learning Representations (ICLR) 2018*.

backup slides

Challenge & Solution

Applying transfer learning to mobile-cloud hybrid setting has challenges

1. Sync between mobile <-> cloud

   Needs to update model & predictions transparently to user & app

   a. [model] Pull based. Hash weights and compare with previous weights

2. Cloud change freeze & thaw layers.

   When freeze layer changes, mobile device needs recomputation.

   a. Each sample has label (originated from which activation)
   b. [Cloud] shootdown wrong labels & broadcast to mobile devices with wrong labels
   c. [Mobile] Caching via activation checkpointing (don’t recompute everything)

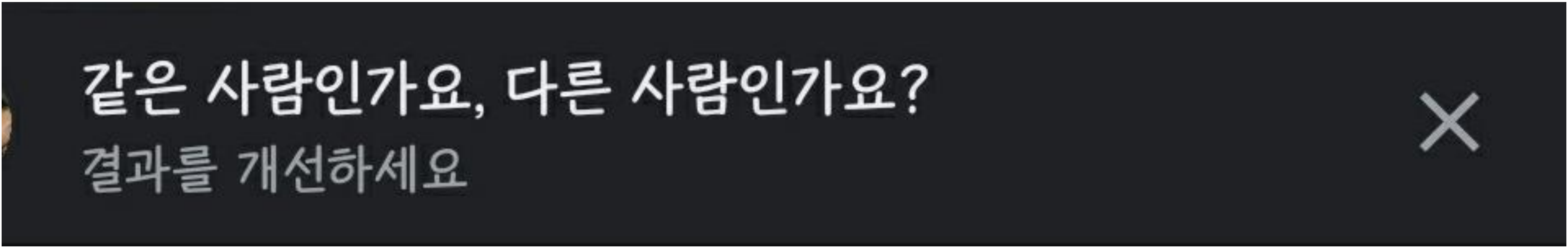
Optimizations

1. Cache previous results (Model does not change in freezed layer)
   Not possible in previous frameworks
2. Watch battery level and sync only when high level

Adaptive app framework via federated learning in transfer learning settings

- purpose:"mobile device에 deploy된 general 한 모델을 user-specific training을 통해 성능 향상 / personalization"
- background
  - general ai는 personalize가 어려움.
  - 하지만 각 유저 데이터를 다 받아서 학습은 어려움 & network & privacy 문제 & unlabeled data!
- opportunity
  - google photo / iphone photo 에는 이미 user annotation을 받는 기능이 있다. (이 얼굴은 누구가 맞습니까? 등) 이런 데이터를 응용하면 좋겠다. 하지만 network & privacy!
- challenge:
  - 어떻게 유저 데이터를 추가 학습하여 personalize된 모델 만듬?
  - 어떻게 모델을 업데이트 함?

User annotation 예시 (Google Photo)

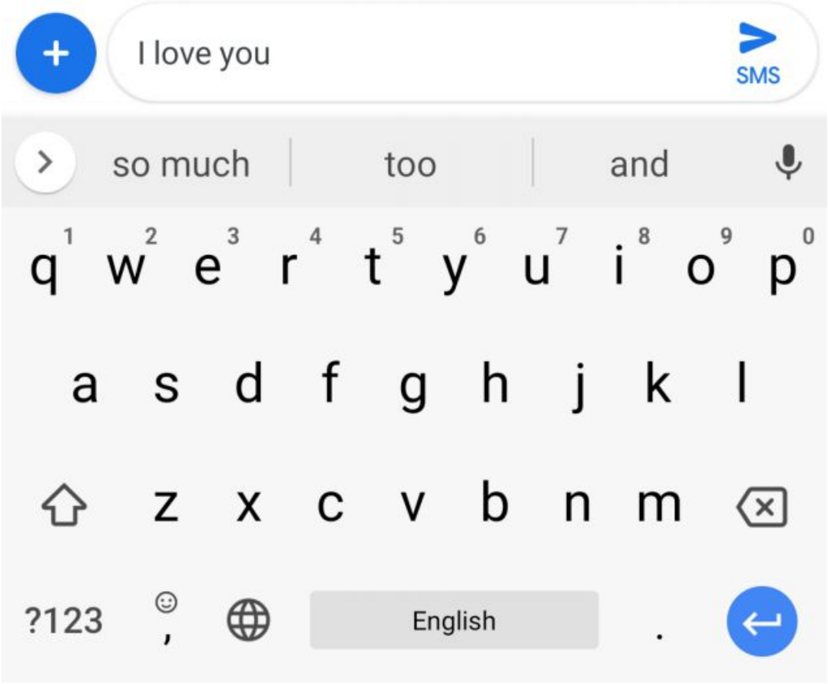


Data Privacy in Federated Learning

- Secure Aggregation (Bonawitz et al.)
  - 서버는 개별 user의 answer를 볼 수 없고, 전체 user들의 aggregated answer만 볼 수 있도록
- Differential Privacy
  - Local device는 일정 확률로 random answer를 서버에 보냄 (noisy data)
    - Server asks: 하루에 5시간 이상 스마트폰을 사용했니?
    - Local device answers:
      - 50% 확률로: 진짜 대답 (yes or no)
      - 25% 확률로: yes
      - 25% 확률로: no
  - Noisy data를 사용하므로 accuracy에 영향이 있지만, 데이터가 매우 많으면 영향이 tolerable
- 하지만, raw data가 아닌 gradient update를 보낼 때에도 이것들이 꼭 필요한가?
  - Our solution: raw data가 아닌 feature map을 보냄

Application: Mobile Keyboard AutoComplete

- Federated learning of RNN for keyboard prediction (Hard et al.)
- Client-side training (i.e. *Federated Averaging*) shows better precision & recall than server-oriented training

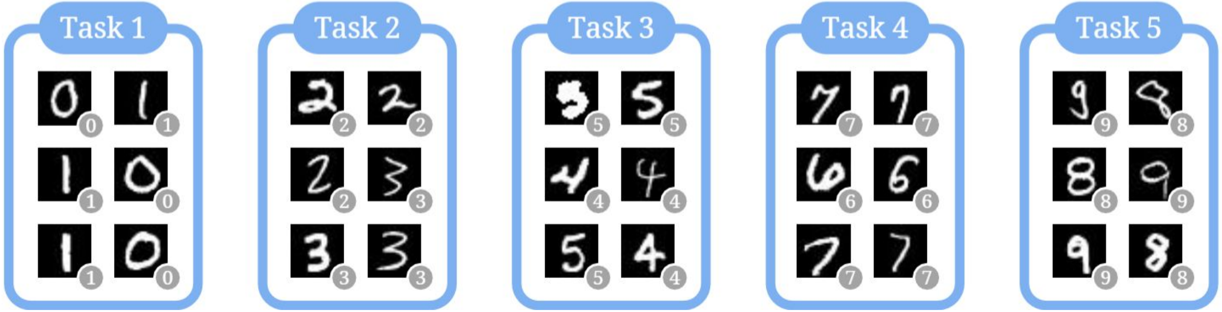


Continual Learning Perspective

- Federated learning scenarios share many common traits with continual learning
  - Model should handle multiple tasks in an efficient manner
  - Data stream from each tasks is not i.i.d.
  - Model should avoid catastrophic forgetting
  - etc.
- There has been a few attempts to unite them into a single architecture
  - Partitioned Variational Inference (Bui et al.)
  - Fed-APC (Yoon et al.)

Continual Learning Perspective

- Experiment scenarios in continual learning can be adopted for federated learning system
  - Split-MNIST, Split-CIFAR10/100 (Lee et al.)



  - NonIID-50: Multiple tasks from MNIST, FashionMNIST, NotMNIST, SVHN, CIFAR10/100, etc. (Yoon et al.)

Federated Continual Learning (Yoon et al.)

- Network decomposition
  - Global shared parameter
  - Sparse task-specific parameter
- Clients with non-i.i.d data stream are respectively optimized
- With LeNet as backbone, Fed-APC alleviates forgetting issues while transferring knowledge base from one task (i.e. client) to another.