Qualcomm

October 28, 2021

@QCOMResearch

Enabling on-device learning at scale

Joseph Soriaga

Sr. Director, Technology

Qualcomm Technologies, Inc.

Our presenter

Joseph Soriaga
Senior Director, Technology
Qualcomm Technologies, Inc.

Today's agenda

- What is on-device learning and why is it crucial for scaling intelligence?
- Our latest on-device learning research and results
- Conclusions and future directions
- Questions?

Smartphone

Smart homes

Video conferencing

Autonomous vehicles

Smart factories

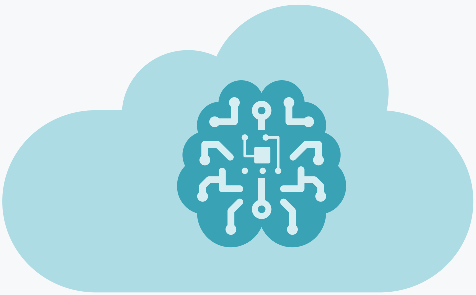
Extended reality

Smart cities

Video monitoring

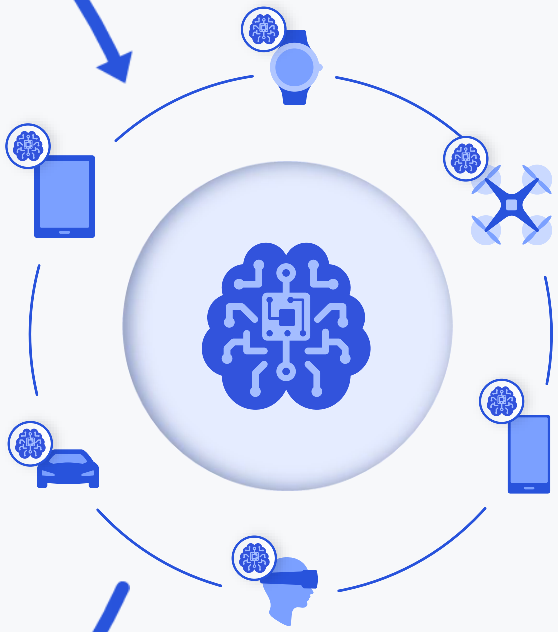
The need for intelligent, personalized experiences powered by AI is ever-growing

How do we maintain privacy and deal with all the data from edge devices?



On-device intelligence is paramount

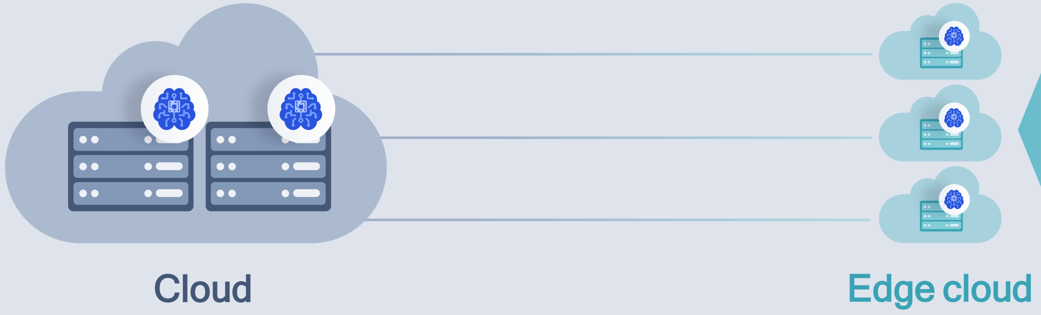
Process data closest to the source, complement the cloud

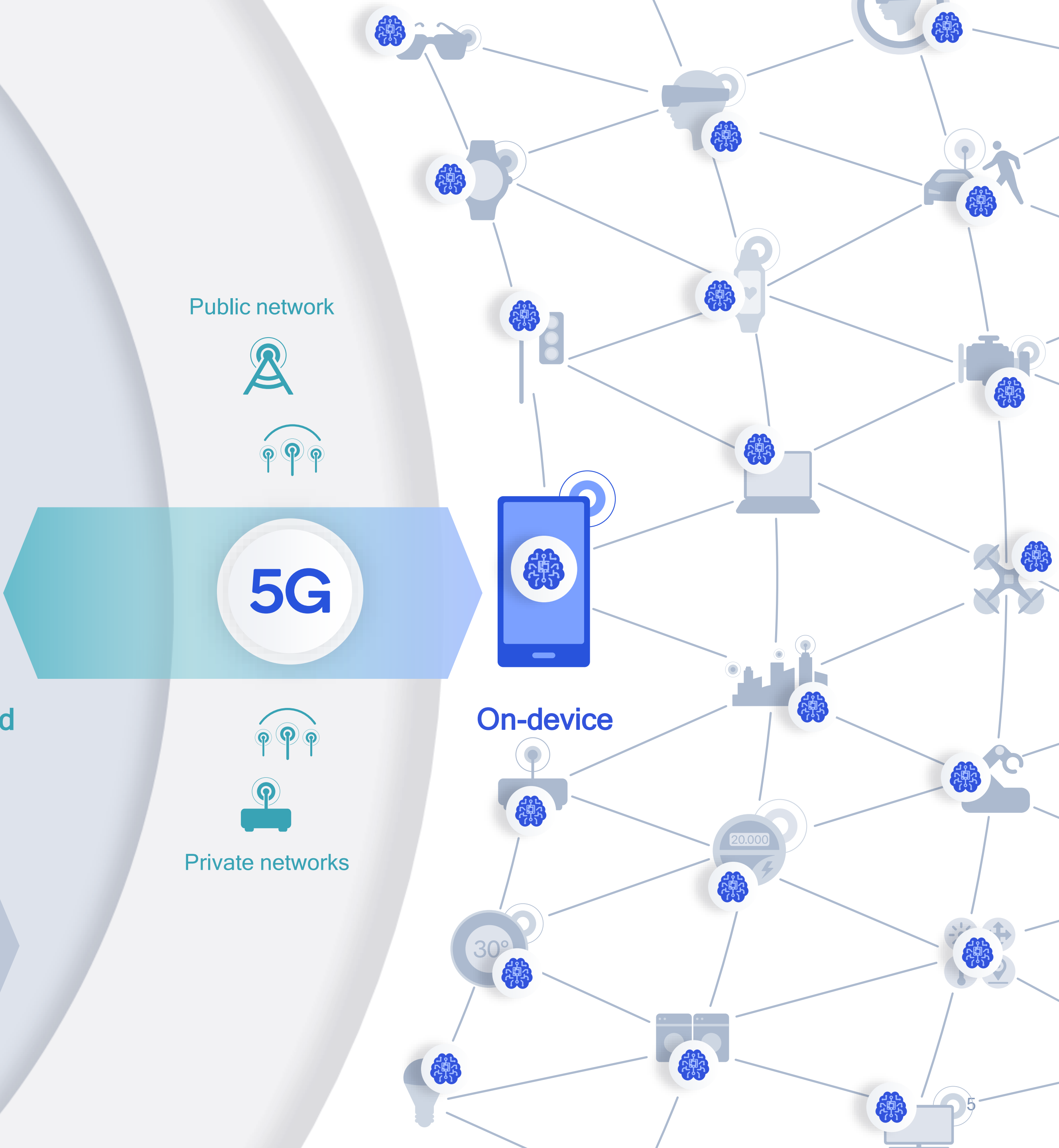


- Privacy
- Reliability
- Low latency
- Efficient use of network bandwidth

Transformation of the Connected Intelligent Edge has begun at scale

Processing data closer to devices at the edge derives new system values (e.g., lower latency, enhanced privacy)





Past
Cloud-centric AI
AI training and inference in the central cloud

Today
Partially-distributed AI
Power-efficient on-device AI inference

Future
Fully-distributed AI
With lifelong on-device learning

- Local network analytics
- Low-latency interactive content
- Boundless XR
- On-demand computing
- Industrial automation and control
- Enterprise data

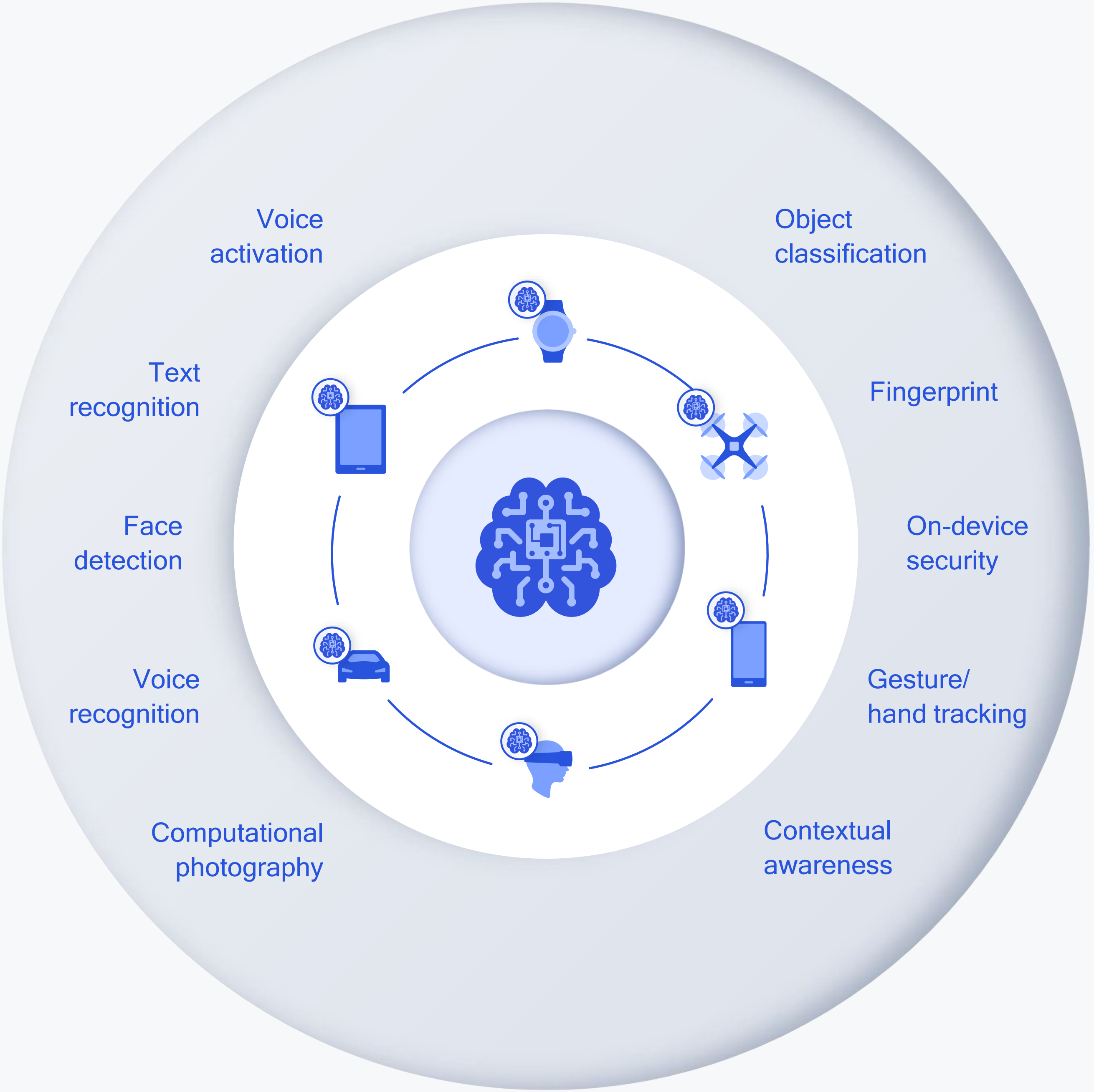
Connected Intelligent Edge

brings new and enhanced services

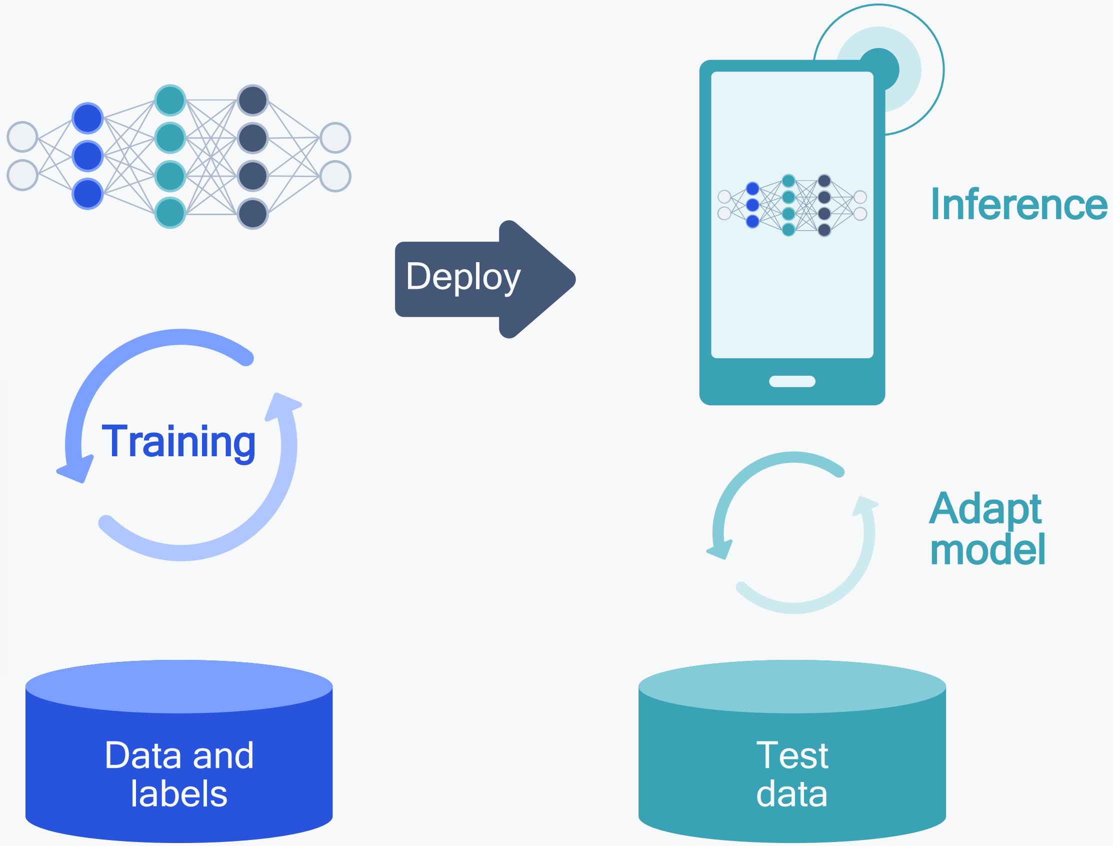
Edge cloud AI

+

On-device AI



What is on-device learning?

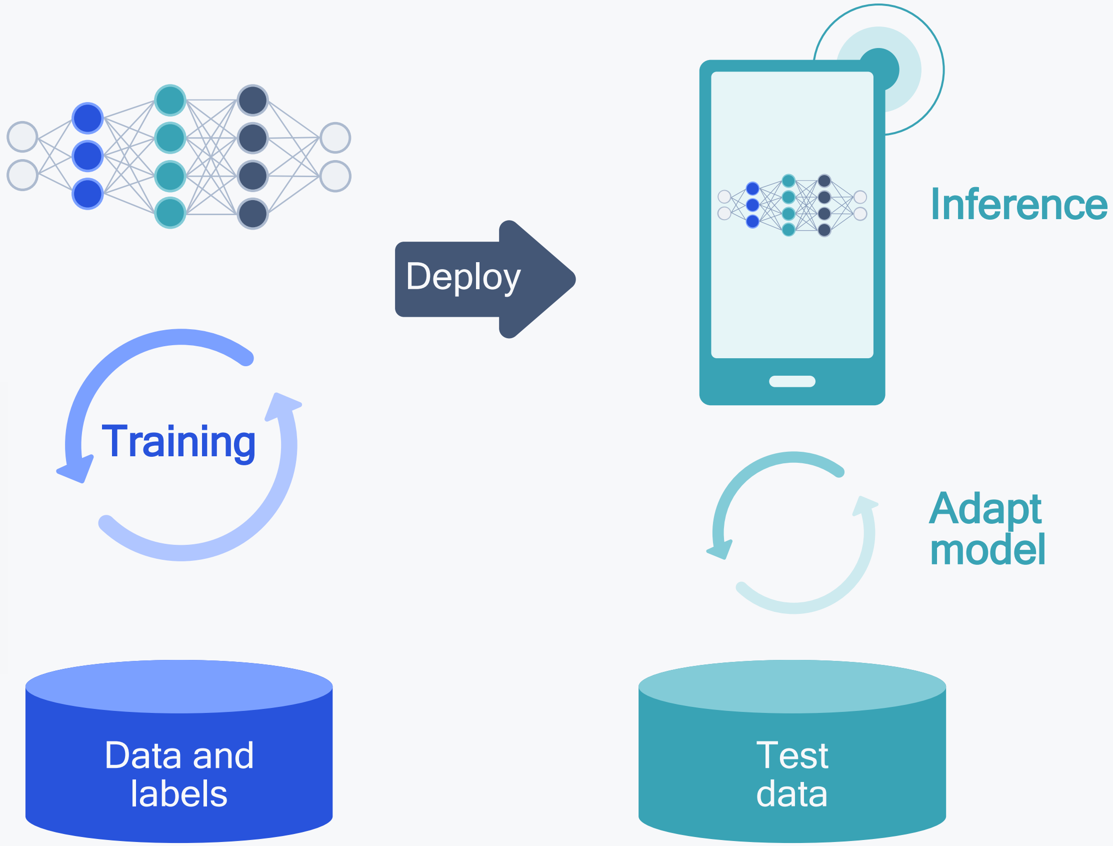


Offline training
A model is trained in the cloud with data reflecting the target application

On-device learning
Modifying model after deployment based on the test environment

On-device learning offers several benefits

- Continuous learning
- Personalization
- Data privacy
- Scale



With offline training, the test data can differ from training data (domain shift, distribution shift, anomalies) and may even change continuously

On-device learning can help to improve and maintain accuracy when original pre-trained model cannot generalize well

Overcoming challenges to achieve on-device ML benefits

Important considerations for on-device learning to achieve benefits for different use cases

Benefits

- Better examples than training dataset
- Ability to run with smaller models that adapt to the target data
- Preservation of privacy during model development

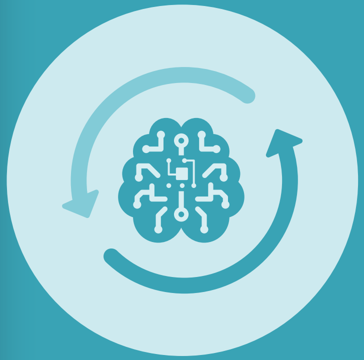
Challenges

- Local data can be limited, e.g., noisy labels and class imbalance
- Overfitting or catastrophic forgetting
- Limited compute, storage, and/or power
- Adversarial attacks to training
- Federated learning communication overhead

Our AI research areas address the key deployment challenges of on-device learning

Few-shot learning

How to adapt the model to a few labeled samples



Continuous learning with unlabeled data

How to use unlabeled data to do unsupervised learning

Federated learning for global adaptation

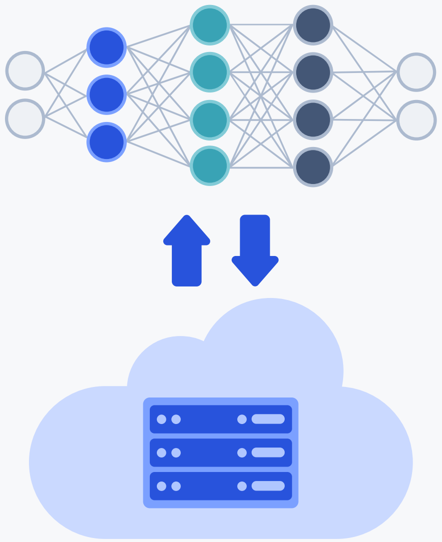
How to implement federate learning at scale and address deployment challenges

Low-complexity on-device learning

How to implement on-device learning to improve efficiency

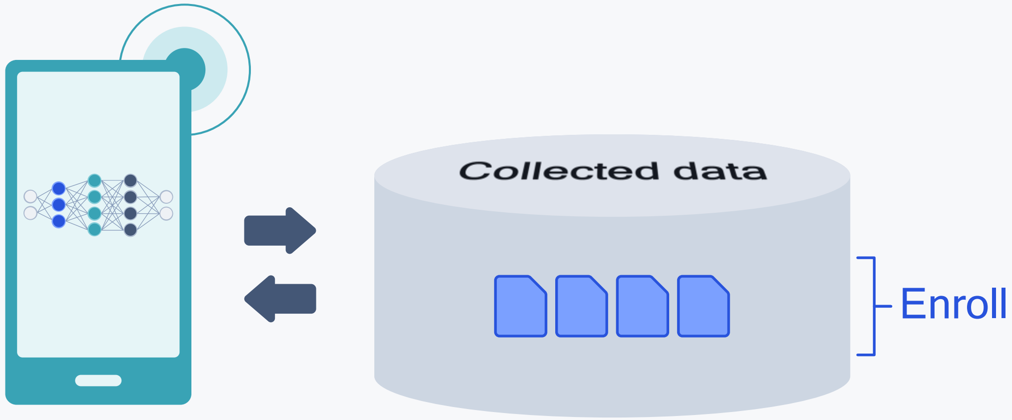
Learning from limited labeled data is crucial

Offline learning



Cloud data
and training

On-device learning



Target user labeled data (e.g., from enrollment)

Limited samples with user-assisted labeling

May be from cleaner environment

Few-shot learning

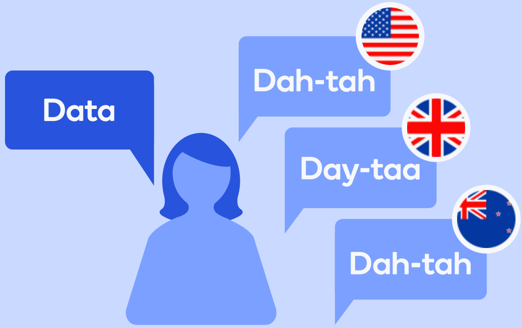
Improve the target user’s model using the initial collected data, such as enrollment

Few-shot learning for increased personalization

Improving keyword spotting (KWS) performance of outlier users through on-device learning

Keyword spotting

Identify when a keyword is spoken using always-on ML



Keyword spotting challenge

- In practice, it is hard to collect all types of accented utterance
- The KWS model may not be sensitive to users' accents and have poor performance for outliers

Keyword spotting solution

- Locally adapt the model to user enrollments
- Personalize the model at enrollment time

Detection rate for outlier users is over 30% worse, on average

Qualcomm AI Research internal results .

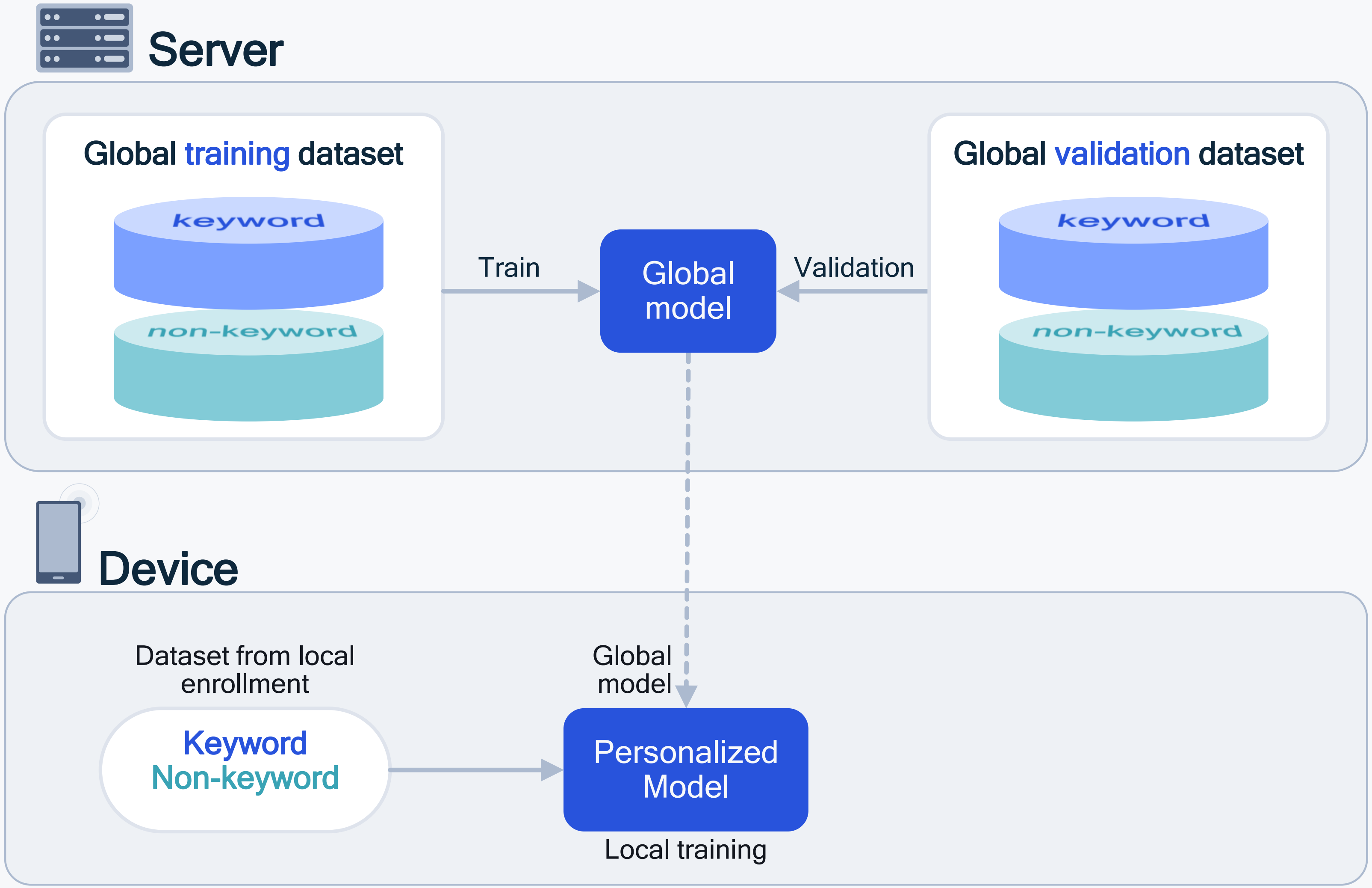
How to locally adapt keyword spotting for personalization

Train a global KWS model

- Global train/validation dataset

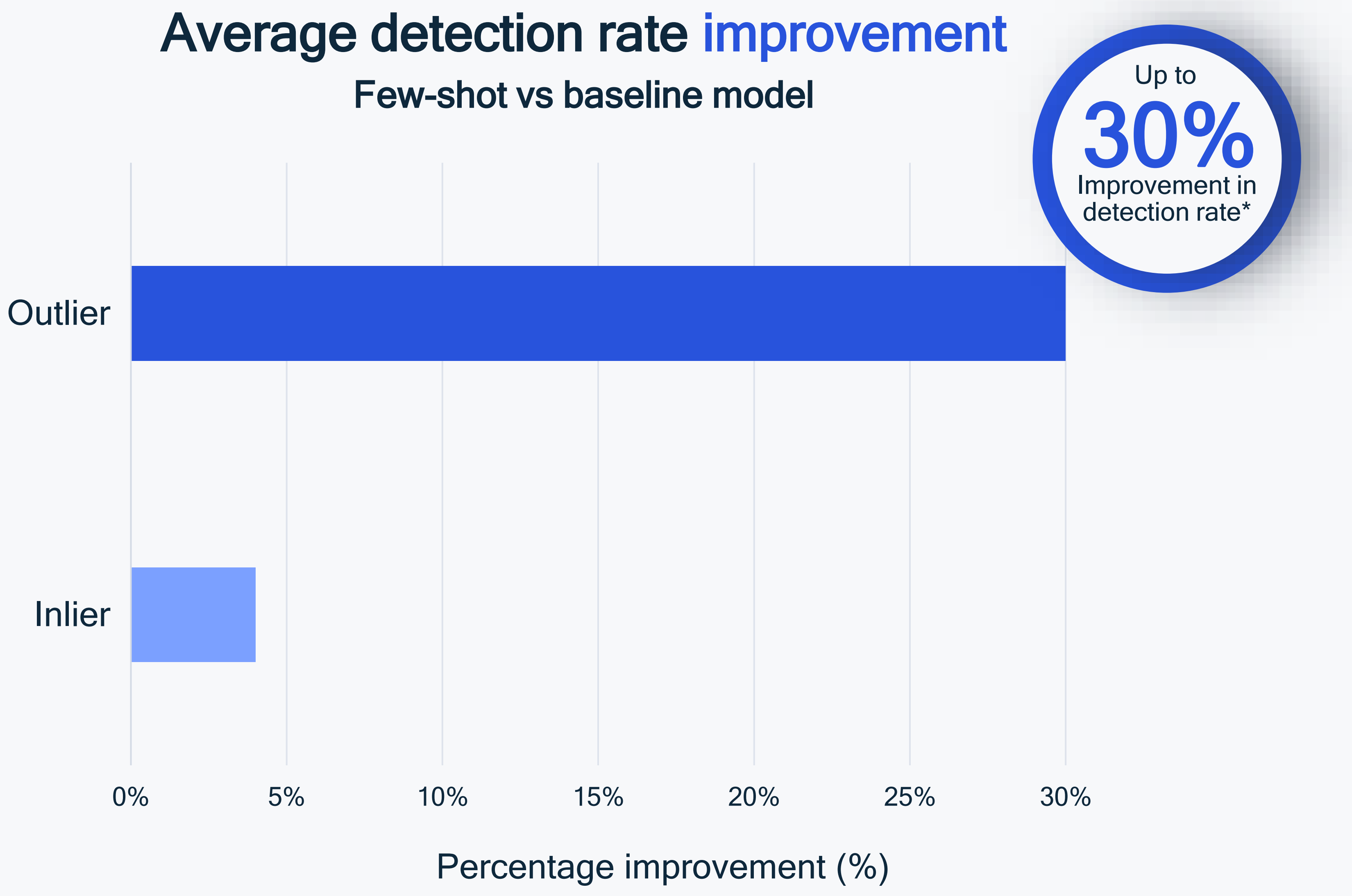
Local adaptation

- Collect enrollment data from target user
- Adapt the global model on local data



Few-shot learning for KWS improves performance

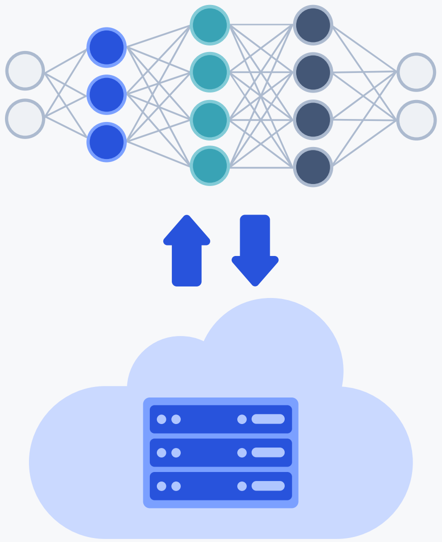
Personalization improvements across the board but particularly for outliers



Qualcomm AI Research internal results. For the few-shot results, performance is the average of all the locally adapted models.

Leveraging user data throughout deployment

Offline learning



Cloud data
and training

On-device learning

Target user data through continuous operation

Plentiful samples of unlabeled user data

Can be from various environment: clean, noisy

Continuous learning

Improve the target user's model based on data from continuous operation, often unlabeled data

Solving the challenges for continuous learning

Employ pseudo labeling and regularization to reduce impact from forgetting

Unlabeled collected data

Challenge

Training data are collected on the device without labels

Solution

Assign pseudo labels to training data through the verification process

Overfitting to small data

Challenge

Number of collected data is small

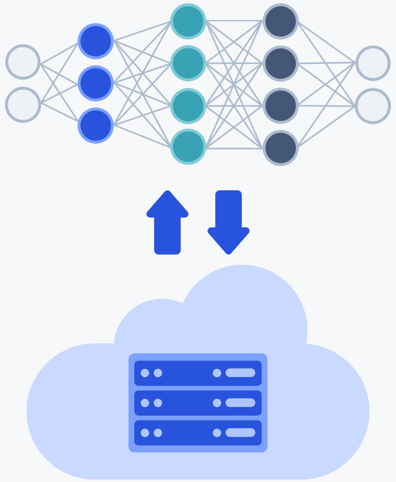
Solution

Exploit regularization loss that maintains some metrics from pre-trained model

Federated learning brings on-device learning to new level

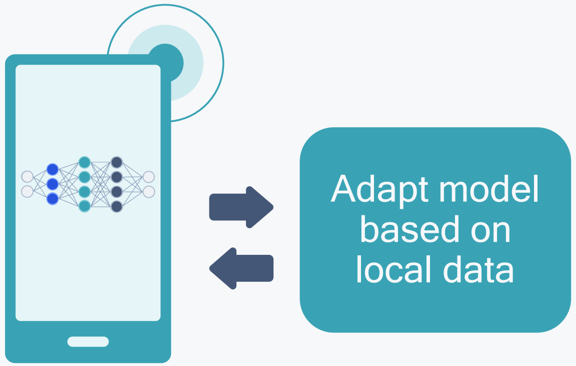
Adaptation on the device, once or continuously, locally and/or globally for continuous model enhancement

Offline learning



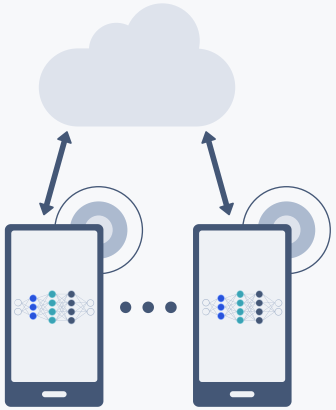
Data

On-device learning



Locally adapt once to a few samples (e.g., few shot learning) or continuously (e.g., unsupervised learning)

Federated learning



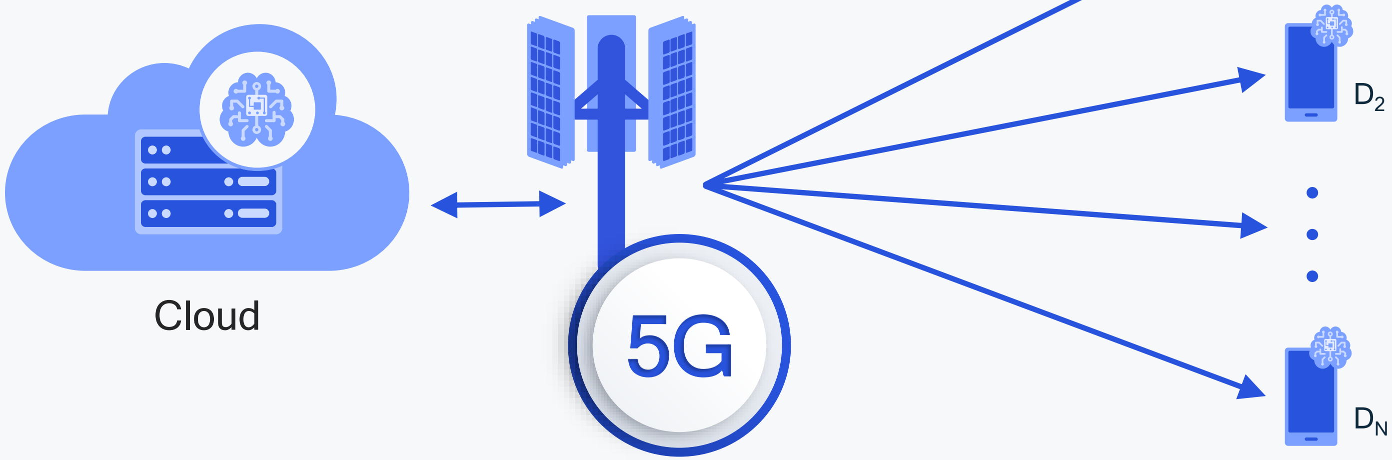
Aggregate model updates across multiple users to globally improve model from more diverse data

Offline training prior to deployment → Local adaptation → Global adaptation

Federated learning for global adaptation

1

A state-of-art global AI model is distributed to devices



2

Each device collects personal data and performs on-device training

Scale
Processing is spread over many devices

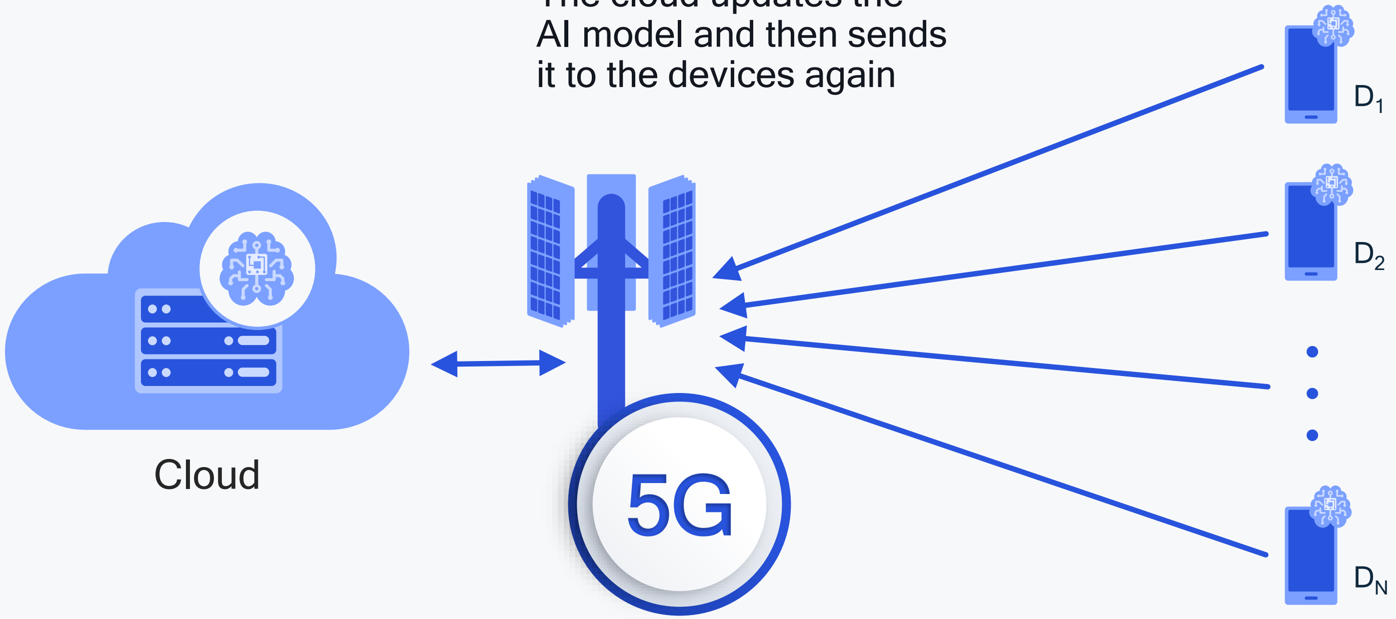
Personalization
Model customized based on your personal data

Privacy
Raw data stays on the device

Federated learning over 5G is the way to scale intelligence

4

The cloud updates the
AI model and then sends
it to the devices again



3

Each device:

1. Adds perturbation to the parameters of the AI model
2. Compresses the parameters
3. Encrypts them
4. Sends the update to the cloud

Scale
Network bandwidth is conserved

Privacy
Only noisy and encrypted weights sent to the cloud

Federated learning over 5G is the way to scale intelligence

User verification

The authentication problem needs big data to get a powerful verification model

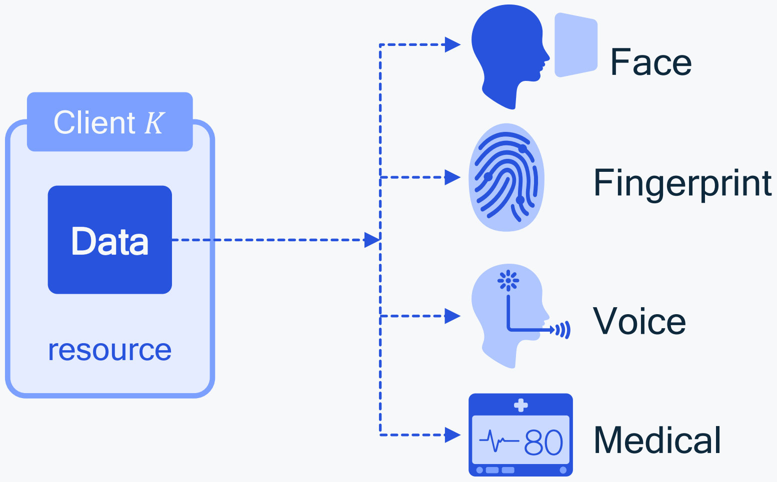
E.g., typical speaker verification system needs data from more than 600k different speakers

Challenge

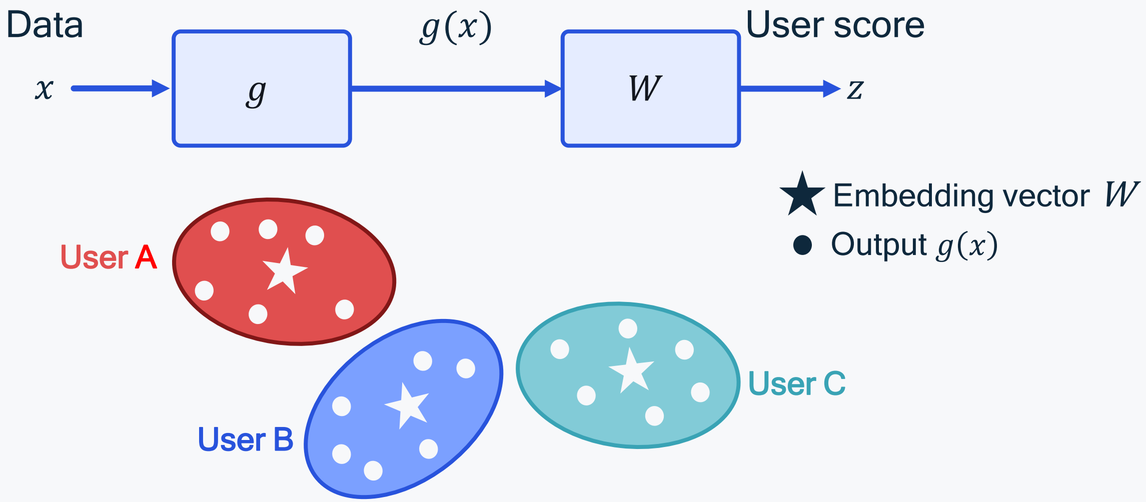
How can we learn this model while keeping all data private?

We do not want to compromise the sensitive biometric data of training participants

Personal data available for authentication



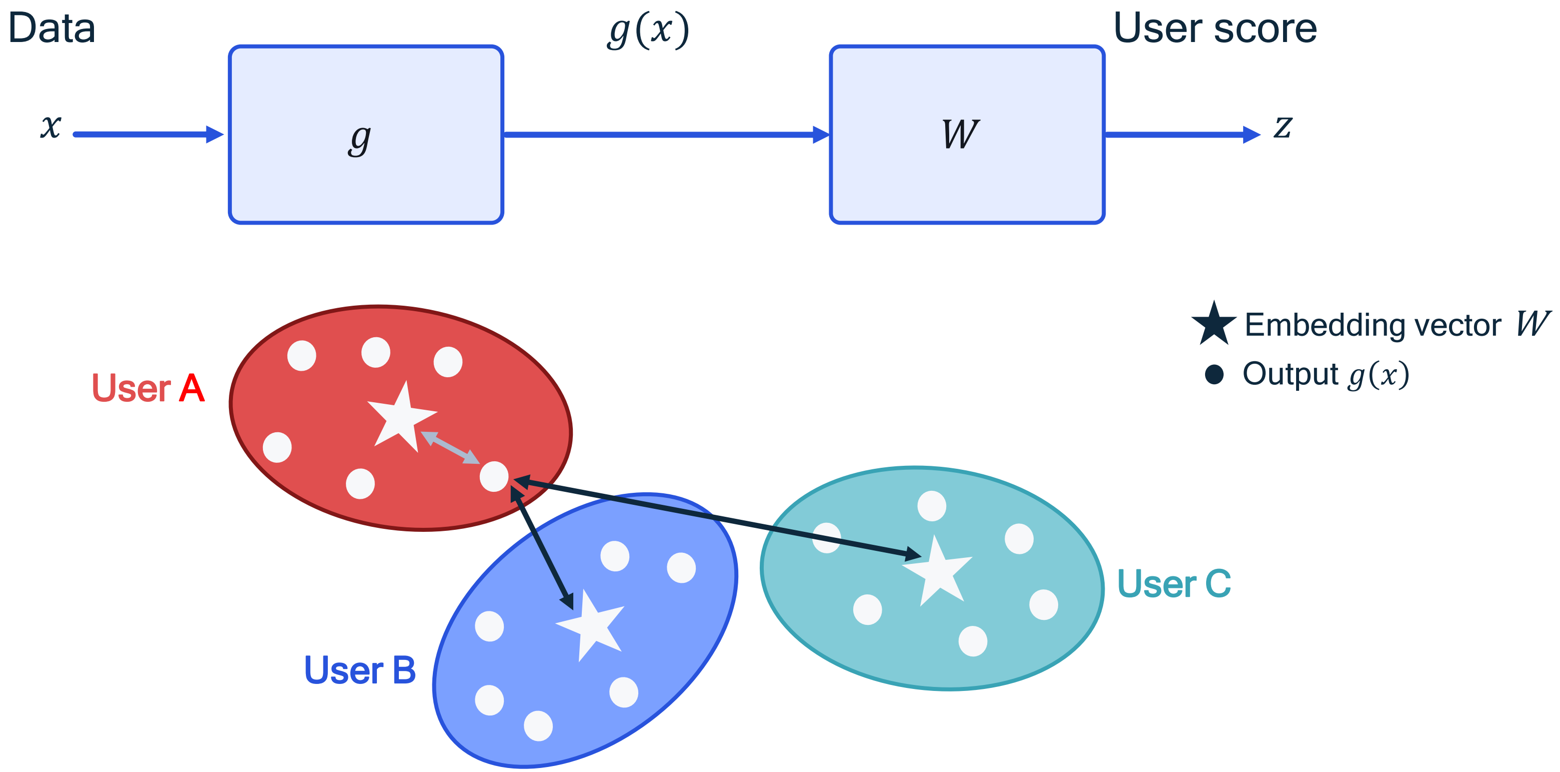
Deep learning approach for authentication



Federated learning can be a powerful tool for user verification

Traditional design of neural networks for user verification do not preserve privacy

User embeddings need to be shared for training



For user verification, neural network $g(x)$ should be trained to:

Minimize the loss to the target user
A smaller loss means a higher user score

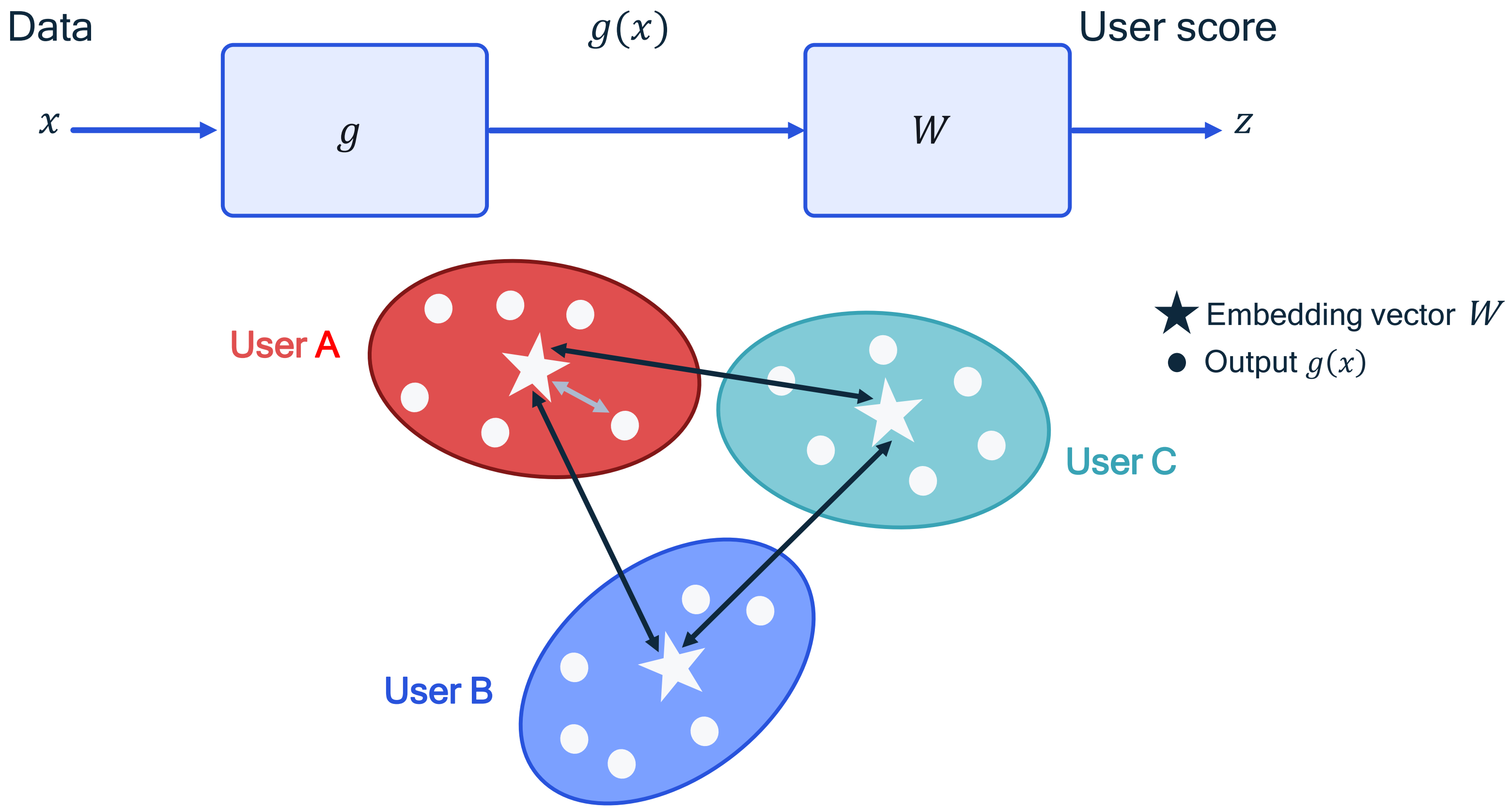
Maximize the loss to the other users
In traditional (one-hot) approaches, users share embeddings to calculate this loss (not private)

"Federated Learning of User verification Models Without Sharing Embeddings" (ICML 2021)

We enable federated learning for user verification without users sharing their embeddings

Generate user embeddings using error-correcting codes (ECC)

“Federated Learning of User verification Models Without Sharing Embeddings” (ICML 2021)

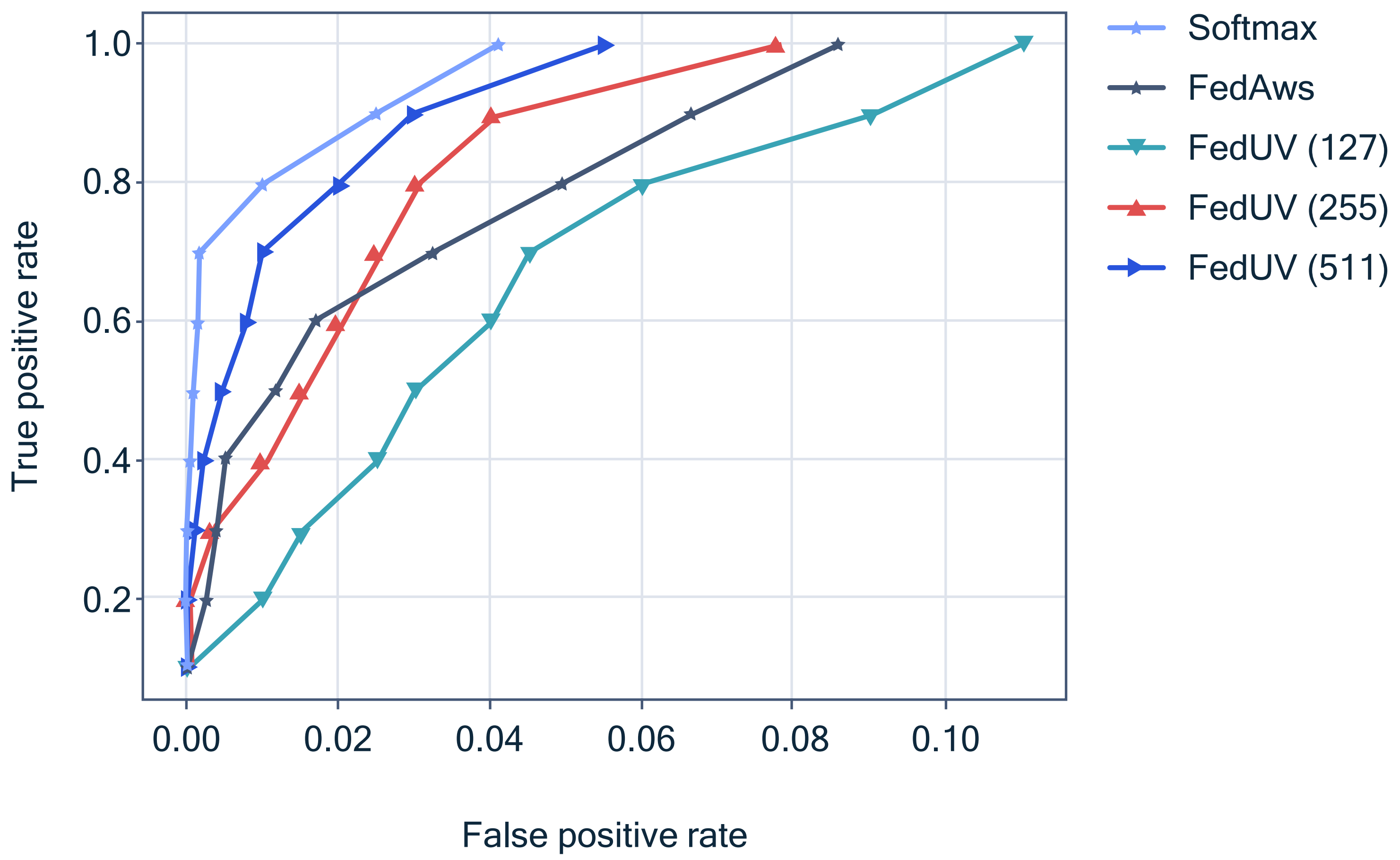


Our method (FedUV) accomplishes this by using embeddings that are codewords of error correcting codes (ECC) and optimizes network $g(x)$ using only positive loss function

Each user minimizes their own loss

ECC ensures user embeddings are maximally spaced to reduce score to other users

FedUV achieves state-of-the-art verification performance without users sharing their embeddings

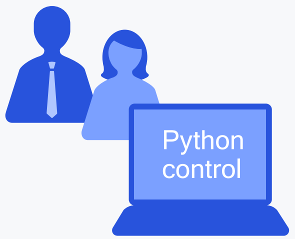


FedUV is comparable to the best method, which shares user embeddings (softmax)

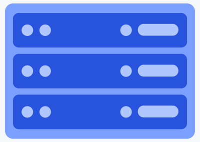
FedUV is better than existing method, which does not share user embeddings (FedAWS)

"Federated Learning of User verification Models Without Sharing Embeddings" (ICML 2021)

FL framework for research and application development on mobile

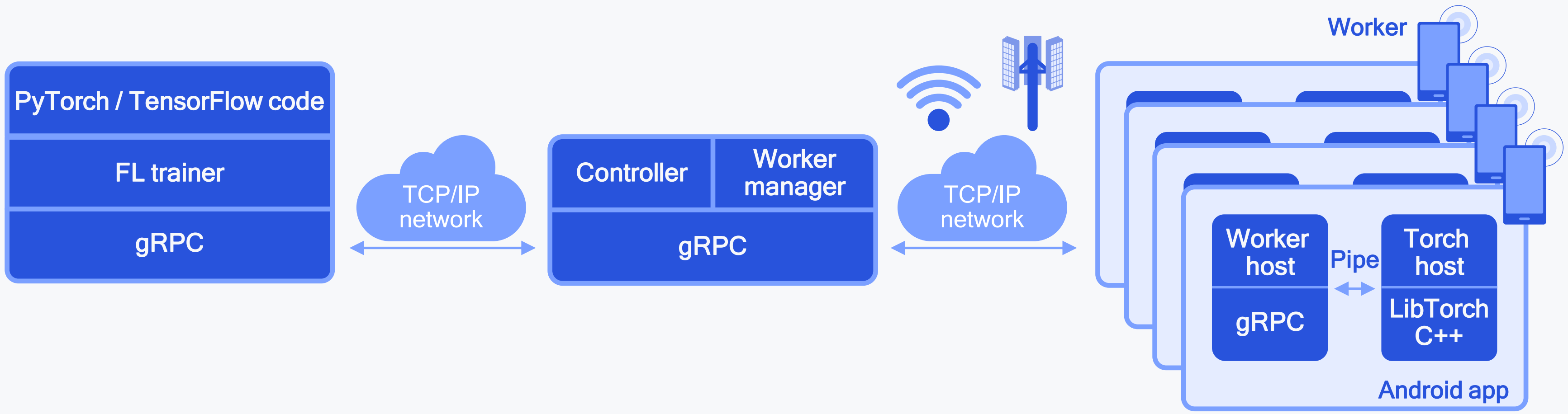


ML experts



Coordinator server

Mobile devices



Samsung Galaxy S21 device powered by Snapdragon® 888 Platform

FL demo of speaker verification

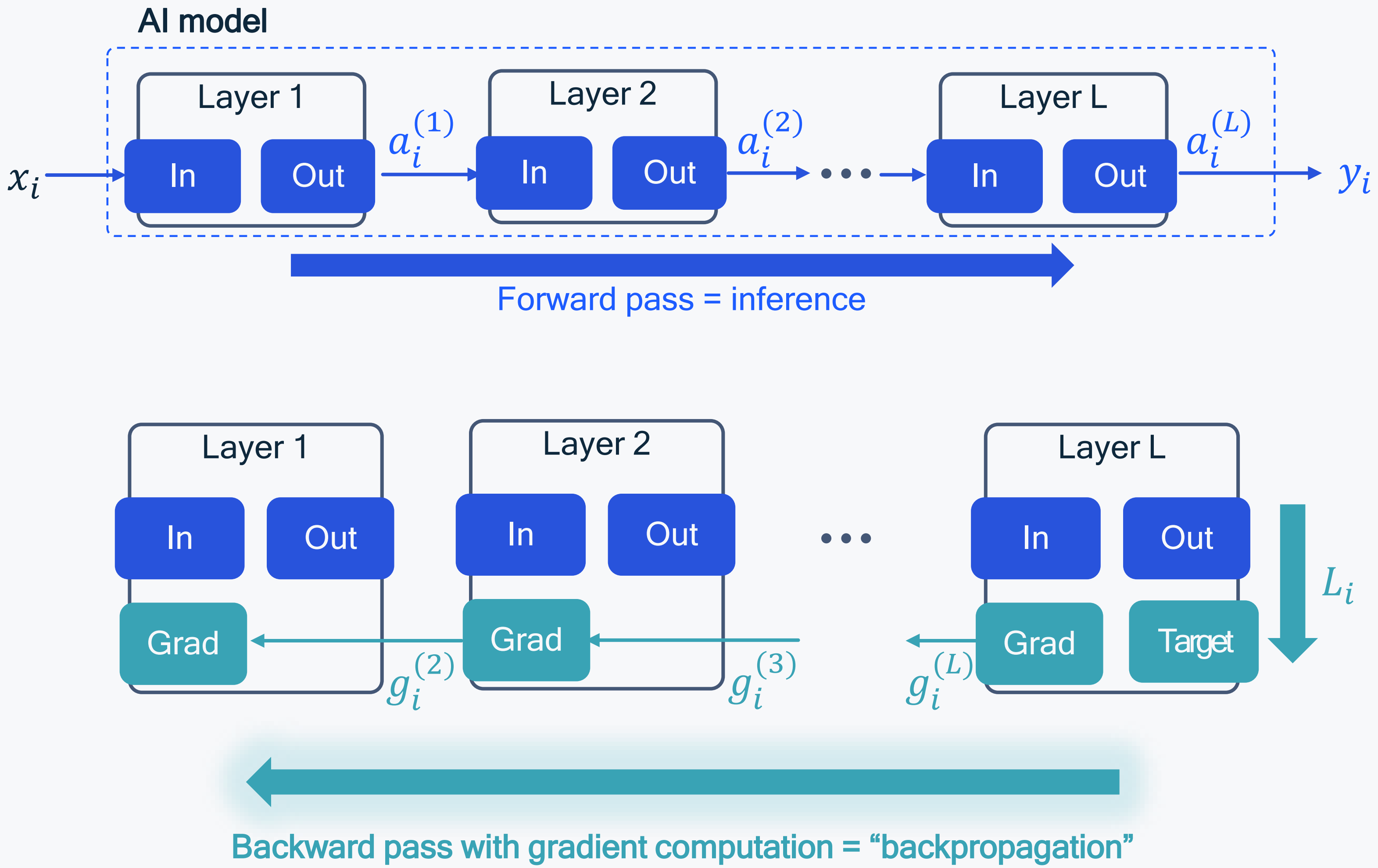
- Enrollment from 1000 clients
- Leverage PyTorch model & training pipeline from research framework

RPC: remote procedure call.

Low-complexity on-device learning

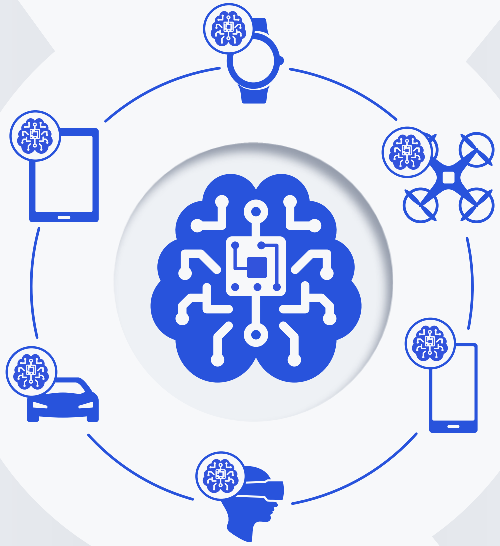
Learning with backprop is computationally demanding

Updating the model weights using backprop can be expensive, especially on power-constrained devices



Backprop training requirements

- Large memory
- Training runtime
- High precision
- Support for quantized inference



Adapt AI model on the device

1. Reduce complexity of backprop with quantized training
2. Efficient models for backpropagation
3. Adapt model using inference

Overcoming challenges to efficiently adapt a neural net on a device

Reduce backprop complexity with quantized training

NN quantization is very effective for NN inference: low energy with high accuracy

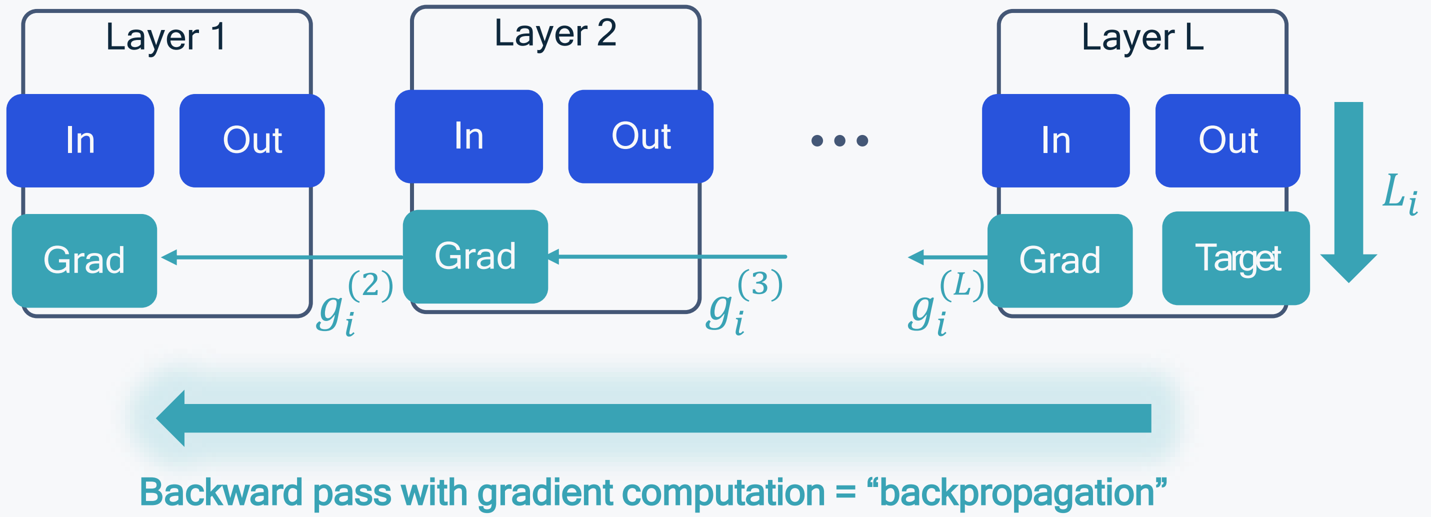
Can we use quantization in backpropagation to make NN training more efficient?

Challenge

Maintain accuracy and reduce compute and memory using quantized gradients and activations

Solution

Quantization with In-Hindsight Range Estimation

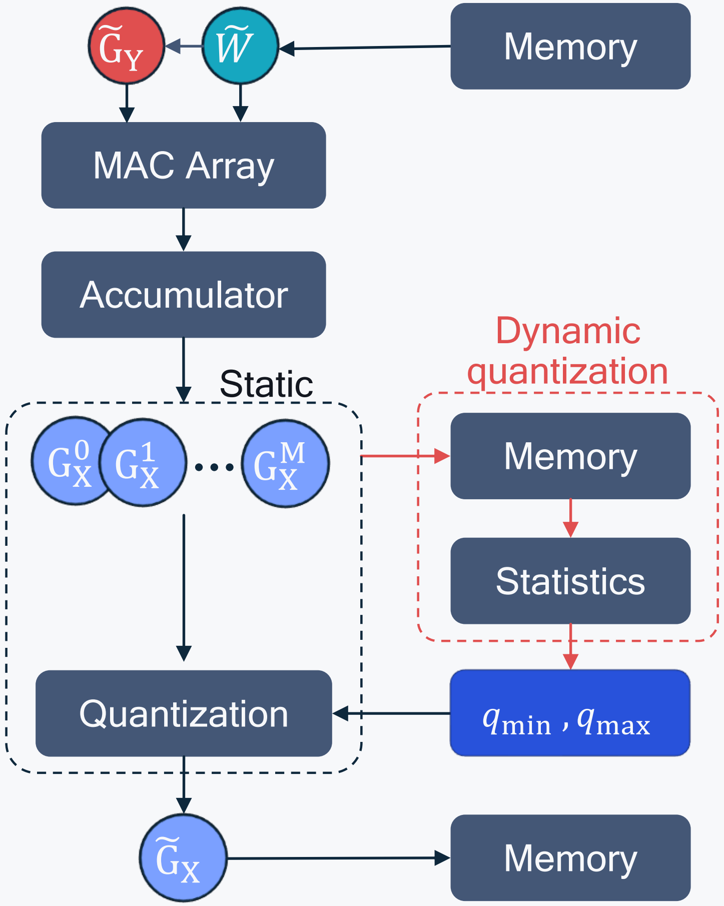


"In-Hindsight Quantization Range Estimation for Quantized Training" (CVPR ECV Workshop 2021)

Existing quantized training techniques are too complex

Estimating range with dynamic quantization

- Uses statistics from the current feature map to quantize it
- Requires writing the 32-bit feature map to memory before quantization
- Is expensive to implement due to high memory transfers



Gradient method	Activation method	ResNet18	Memory transfer
FP32	FP32	69.75	High
Current min-max	Current min-max	69.21 +/- 0.06	High
Running min-max	Running min-max	69.35 +/- 0.16	High

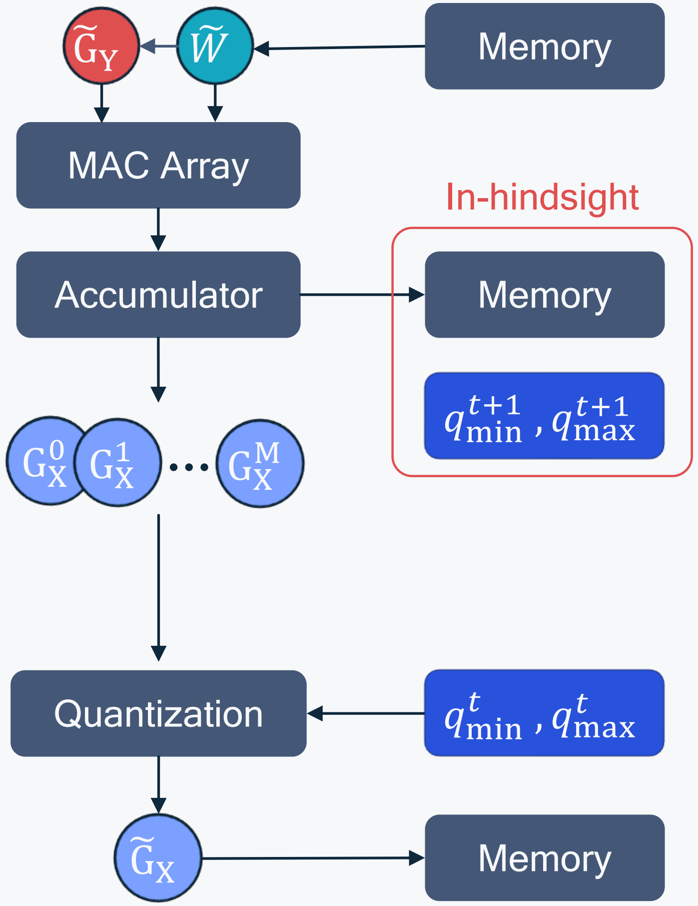
"In-Hindsight Quantization Range Estimation for Quantized Training" (CVPR ECV Workshop 2021)

In-Hindsight Range Estimation reduces quantize training complexity while maintaining accuracy

Use pre-computed quantization parameters to quantize current tensor

Extract statistics from current tensor for quantization parameters on next iteration

Much lower complexity and data movement



79%
Reduction in memory transfer*

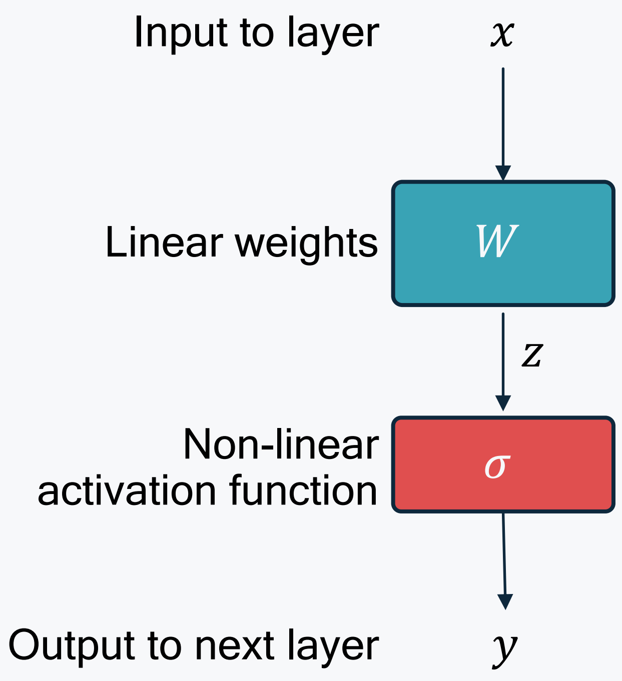
Gradient method	Activation method	ResNet18	Memory transfer
FP32	FP32	69.75	High
Current min-max	Current min-max	69.21 +/- 0.06	High
Running min-max	Running min-max	69.35 +/- 0.16	High
In-hindsight min-max	In-hindsight min-max	69.37 +/- 0.11	Low

Memory movement cost comparison between static and dynamic quantization

“In-Hindsight Quantization Range Estimation for Quantized Training” (CVPR ECV Workshop 2021)

Typically, the backpropagation calculation requires large memory to store activations

Typical layer of a neural network



ℓ is the loss function to be minimized

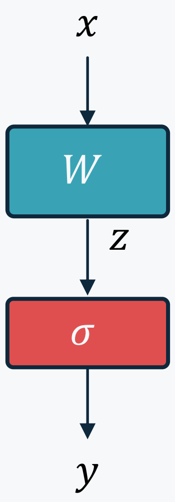
$$\frac{\partial \ell}{\partial z} = \frac{\partial \ell}{\partial y} \cdot \sigma'(z)$$

For backpropagation, **activations need to be stored at every layer** to calculate the gradient of the loss function

Using invertible layers reduces memory requirements for backpropagation

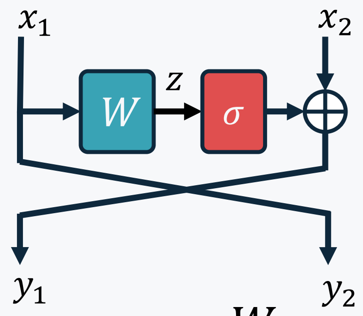
Activations of each layer can be reconstructed exactly from next layer

Typical layer of a neural network → **Invertible layer**



$$\frac{\partial \ell}{\partial z} = \frac{\partial \ell}{\partial y} \cdot \sigma'(z)$$

For backpropagation, **activations need to be stored at every layer** to calculate the gradient of the loss function



$$\begin{cases} x_1 = y_2 \\ x_2 = y_1 - F(y_2) \end{cases}$$

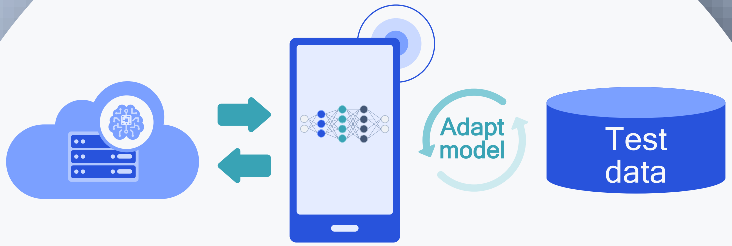
$$z = y_2 W$$

z can be computed for each layer in backpropagation, so **no need to store activations**

11x Reduction in activation memory

	#params (M) / #MACs (B)	Top 1 / Top 5	Activation mem. per image
MobileNet-V2	3.4 / 0.3	72.0 / 91.0	**43 MB**
Invertible network	3.24 / 0.3	72.5 / 90.7	**3.7 MB**

Qualcomm AI Research internal results.



Broad range of research directions for on-device learning

Personalization and labeling
Meta learning, active learning, learning with noisy labels

On-device training efficiency
Light-weight models, low complexity training, quantization

Tackling statistical heterogeneity in data
Smartly combining model updates from a broad distribution

Optimizing communication in FL
Compressing information sent on the uplink and downlink

Privacy, security, and robustness
Privacy guarantees, adversarial attack, anomaly detection

Advanced topologies for FL
Peer-to-peer, multi-cloud, and hierarchical privacy

Qualcomm

On-device learning is crucial for providing intelligent, personalized experiences without sacrificing privacy

We are conducting leading research and development in on-device learning

We are solving system and feasibility challenges to move from research to commercialization

Questions?

Connect with Us

www.qualcomm.com/ai

www.qualcomm.com/news/onq

@QCOMResearch

https://www.youtube.com/qualcomm?

http://www.slideshare.net/qualcommwirelessevolution

Thank you

Follow us on:

For more information, visit us at:

www.qualcomm.com & www.qualcomm.com/blog