On-Device Federated Learning of ASR Models

Tech Talk at UNC Charlotte

Yonghui Xiao
2024-04-24

Google Research

Agenda

About Me

Federated Pruning: Improving Neural Network Efficiency with Federated Learning

Online Model Compression for On-Device Federated Learning

[Optional] FedAQT: Accurate Quantized Training with Federated Learning

Q&A

Google Research

About Me

- Software Engineer at Google from 2017
- PhD from Emory University
- Research
  - ASR (Automatic Speech Recognition)
  - Federated Learning
  - Differential privacy
- 1000+ citations

Google Research

INTERSPEECH 2022
September 18 - 22 • Incheon Korea

Federated Pruning: Improving Neural Network Efficiency with Federated Learning

Rongmei Lin, Yonghui Xiao, Tien-Ju Yang, Ding Zhao,
Li Xiong, Giovanni Motta, Françoise Beaufays

Google Research

EMORY UNIVERSITY

Agenda

01 Introduction

02 Related Works

03 Methodology

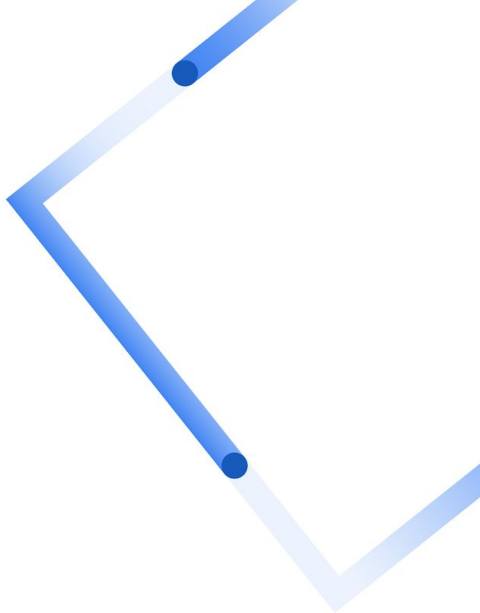
04 Experiment Results

05 Conclusion

Google Research

01

Introduction



Google Research

01 Introduction

- Background
  - Federated learning can be applied to ASR models to utilize client data.
  - Current SOTA ASR model is too large to train on the client devices.
  - How can we perform client model training efficiently? -- Network Pruning.
- Motivations
  - Traditional pruning methods consider centralized training on server model.
  - Leverage the property of federated learning and perform adaptive and customized network pruning on client models.
- Objective
  - Reduce the model size by applying adaptive pruning w/o quality degradation.

02

Related Works

02 Related works

- Lottery Ticket Hypothesis [1]
  - A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.

- Traditional Pruning Methods [2]
  - Three-stage pipeline: training, pruning and fine-tuning.
  - Used the parameter magnitude as the pruning criterion.

1. Frankle, Jonathan, and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks." arXiv preprint arXiv:1803.03635 (2018).
2. Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv preprint arXiv:1510.00149 (2015).

02 Related works

- Dynamic Sparse Neural Network Training
  - Prune-regrowth procedure that allows the pruned neurons to revive randomly. [3]
  - Dynamic parameter reallocation that changes the global pruning threshold. [4] (Coarse-grained adjustment: threshold get halved if the percentage of parameter pruned is too high or get doubled if percentage is too low)
  - Free reallocation of parameters between layers. [5]

3. Mocanu, Decebal Constantin, et al. "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science." Nature communications 9.1 (2018): 1-12.
4. Mostafa, Hesham, and Xin Wang. "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization." International Conference on Machine Learning. (2019).
5. Zhang, Chiyuan, Samy Bengio, and Yoram Singer. "Are all layers created equal?." arXiv preprint arXiv:1902.01996 (2019).

02 Comparison to related works

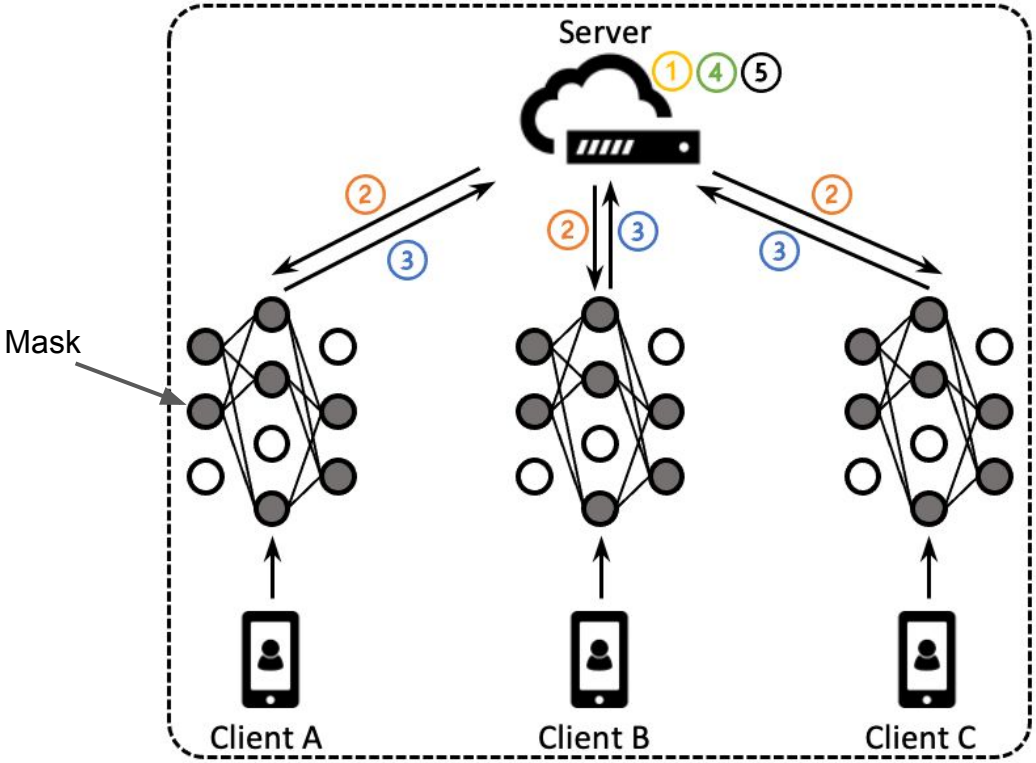
- Unlike these works focusing on centralized training, our work targets at federated learning (FL) and analyzes the impact of different pruning design decisions under this setting.
- A related work of model compression under the FL setting is Federated dropout [6]. Unlike our proposed method, federated dropout randomly generates reduced model and performs training on full model.
- Another preliminary work, PruneFL [7], also applies pruning with FL. It adopts sparse pruning instead of structural pruning as used in this work, so the resultant model will be less efficient when running on devices in practice.

6. D. Guliani, L. Zhou, C. Ryu, T.-J. Yang, H. Zhang, Y. Xiao, F. Beaufays, and G. Motta, “Enabling on-device training of speech recognition models with federated dropout,” arXiv:2110.03634, (2021).
7. Y. Jiang, S. Wang, B. J. Ko, W. Lee, and L. Tassiulas, “Model pruning enables efficient federated learning on edge devices,” arXiv:1909.12326, (2019).

03

Methodology

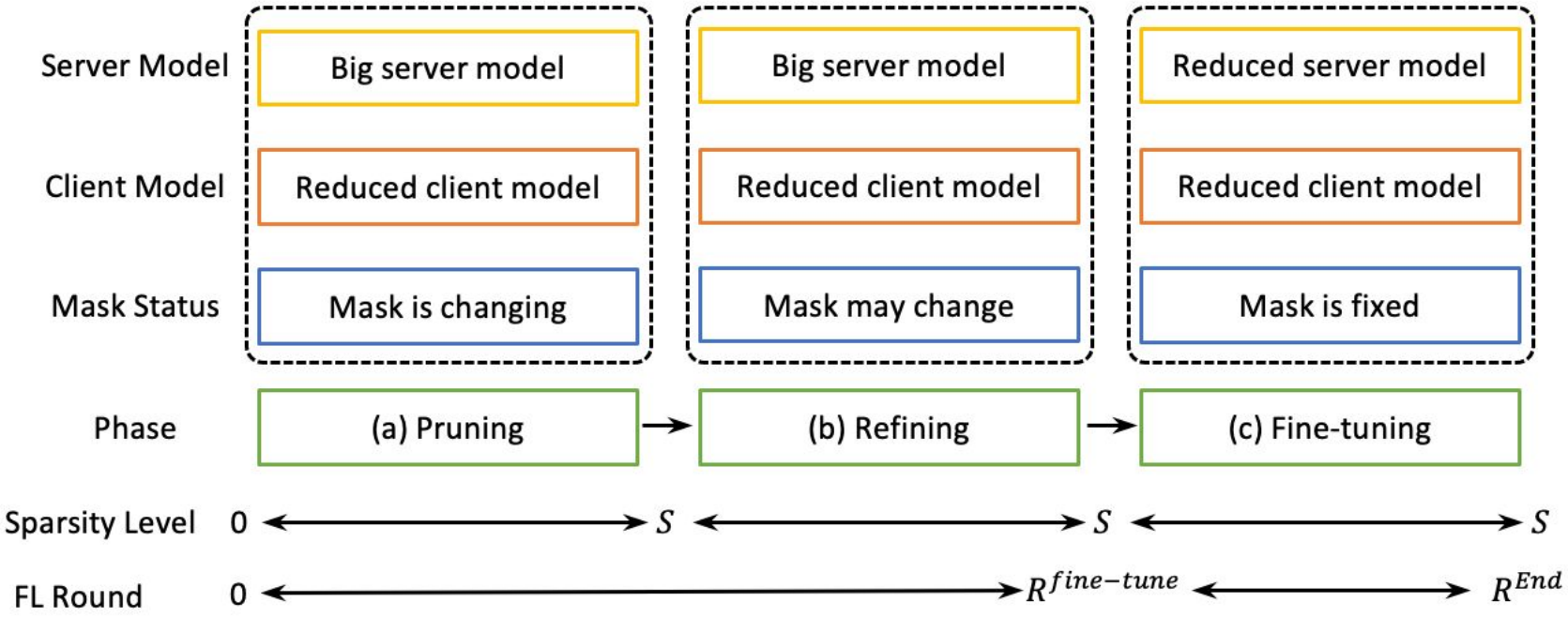
03 A *federated round* of proposed method



1. Pruning at server
2. Sending reduced model to clients
3. Sending reduced update to server
4. Mapping to original model
5. Federated averaging

Google Research

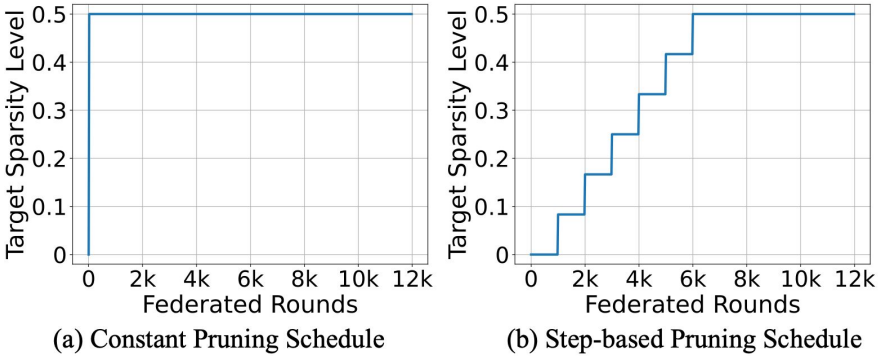
03 Three phases of Federated Pruning



Google Research

03 Pruning patterns & methods

- Pruning methods: the metric used to determine the salience of variable. We use three methods including (1) weight magnitude of variables, (2) momentum of gradient magnitude of the variables and the (3) multiplication of weight and gradient magnitude to measure the salience. To reach target level:
  - Constant level: prune to the target sparsity level ***S*** at the beginning.
  - Step-based level: gradually increase the sparsity level *w.r.t* current round ***r***.

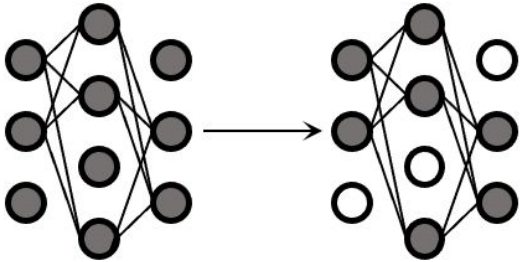


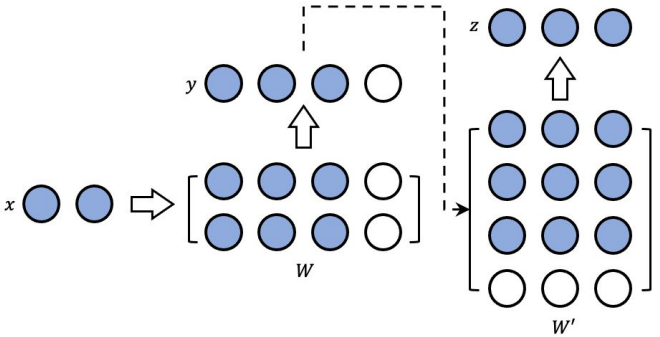
(a) Constant Pruning Schedule (b) Step-based Pruning Schedule

Google Research

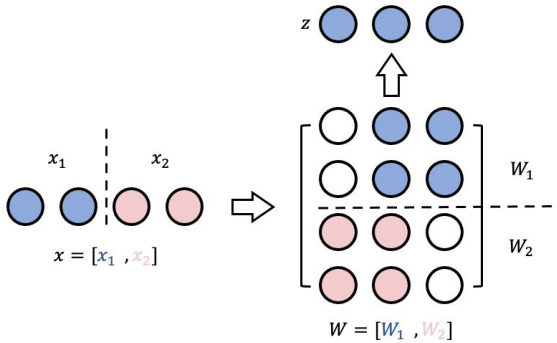
03 Pruning patterns & methods

- Structured pruning pattern: the structure of the pruned variables.
  - Whole row / column: prune the entire row or column of the two-dimensional weight matrices W.
  - Half row / column: evenly partition the two-dimensional variables W into [W1, W2] and prune each half of the row or column.





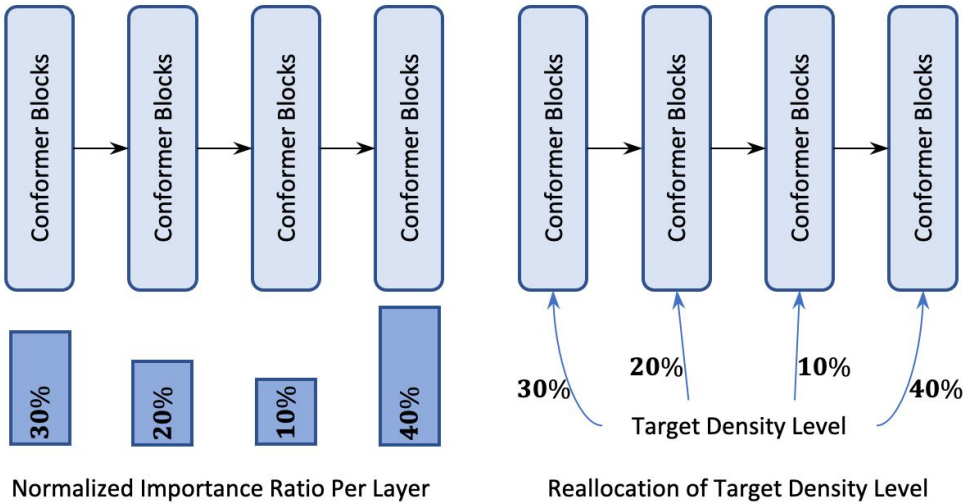
(a) Whole Column Pruning



(b) Half Column Pruning

Google Research

03 Sparsity reallocation process



$$LayerDensity = \frac{(1 - TargetSparsity) * ||w_i||}{\sum_{i=1}^{L} ||w_i||}$$

||w_i||: Averaged weight magnitude per layer.
L: the total L layers in the model.

Heuristic agent determine the layer-wise sparsity level:

- Step 1: use predefined rules to measure the importance of each layer (we use ||w_i||).
- Step 2: assign the weighted sparsity level to each layer using estimated importance score.

The less important layers get larger sparsity level.

(i.e. smaller density level)

Google Research

04

Experiment Results

04 Experiment settings

- We implemented the Federated Pruning in a distributed learning simulator.
- We use the SOTA ASR model Conformer-transducer as base model. (add citation)
- All the experiments share a same pre-trained baseline model as initialization.

04 Dataset

- The public LibriSpeech corpus.
  - Consists of 970 hours of labeled speech.
- Industry -scale data collected from different domains.
  - These multi-domain utterances contain 400k hours of speech and span domains of search, farfield, telephony and YouTube.
  - Our work abides by Google AI Principles, all datasets are anonymized and hand-transcribed.

04 Results on short-form multi-domain dataset

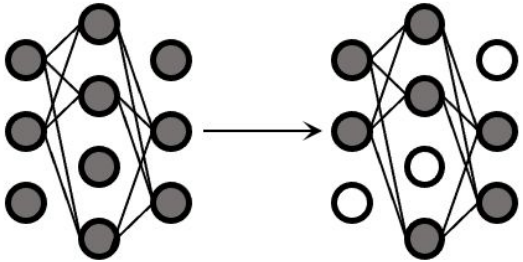
- Finally, we show the results on the large-scale short-form multidomain dataset. The reduced model is trained on our multidomain utterances and evaluated on the short-form dataset. Table 5 demonstrates the WERs on different sparsity levels. We conclude that our model can still achieve comparable performance to the baseline model (with sparsity level 0.0) on challenging dataset in the low sparsity level setting.

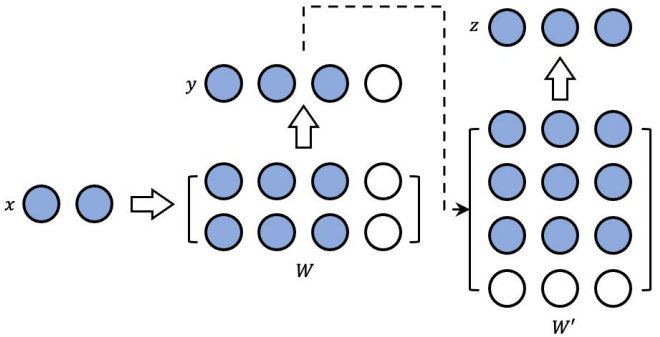
Table 5: *WERs of federated pruning on the voice search dataset.*

Sparsity Level	0.0	0.10	0.20	0.30	0.40	0.50
WER	6.4	6.7	7.0	7.4	7.9	8.9

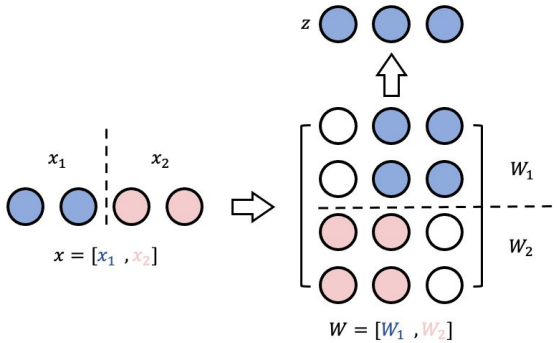
03 Pruning patterns & methods

- Structured pruning pattern: the structure of the pruned variables.
  - Whole row / column: prune the entire row or column of the two-dimensional weight matrices W.
  - Half row / column: evenly partition the two-dimensional variables W into [W1, W2] and prune each half of the row or column.





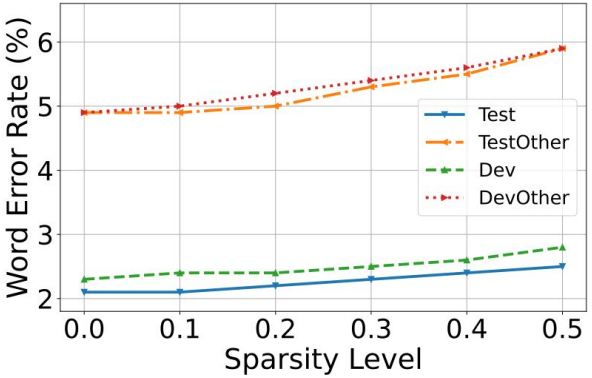
(a) Whole Column Pruning



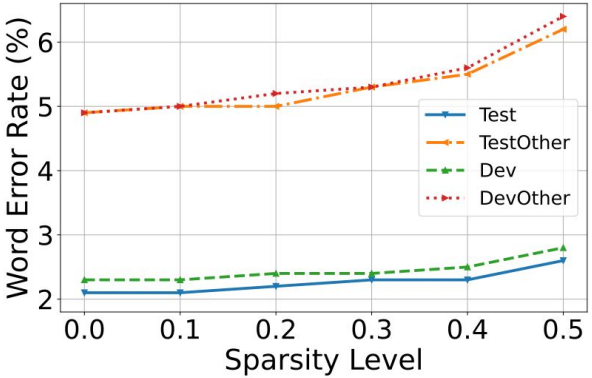
(b) Half Column Pruning

Google Research

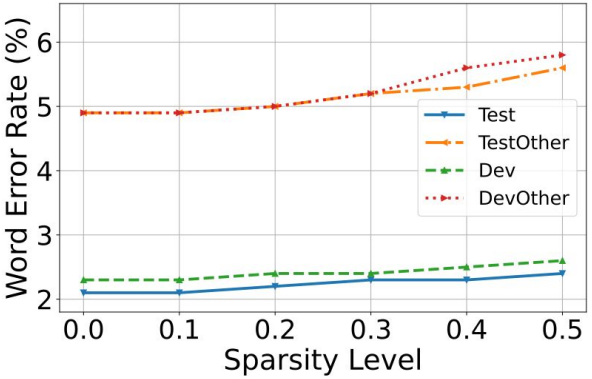
04 Federated pruning results



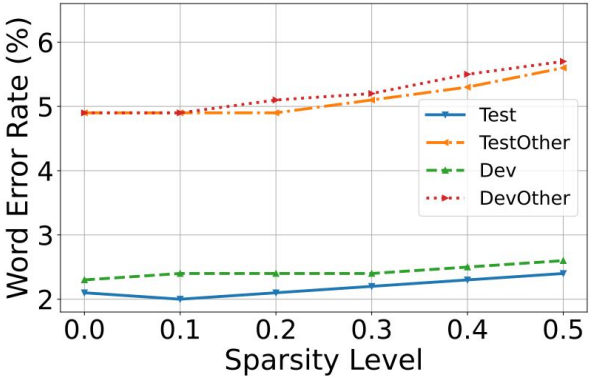
(a) Weight-based Column Pruning



(b) Weight-based Row Pruning



(c) Weight-based Half-column Pruning



(d) Reallocate Weight-based Column Pruning

- Quality degradation when the sparsity level > 30% in all pruning schemes.
- At sparsity level 50%: column pruning and especially the half-column pruning consistently outperforms the row-based pruning.

Google Research

04 Federated pruning results

- Different pruning methods can be used to estimate the importance of variables. We conduct ablation experiments of different measurements on the Librispeech dataset. Table 2 suggests the weight-based score achieves similar WER as other metrics, while it is also the most communication efficient and stable metric. Thus weight-based score is used as the importance metric.

Table 2: *WERs of different pruning methods*

Pruning Methods	**WER on Test with sparsity level**				
	0.10	*0.20*	*0.30*	*0.40*	*0.50*
weight based	2.1	2.2	2.3	2.4	2.5
gradient based	2.2	2.3	2.3	2.3	2.4
weight×gradient based	2.1	2.2	2.2	2.3	2.4

04 Adaptive per-layer sparsity results

- Table 3 demonstrates that, compared to federated pruning with unified sparsity, the adaptive sparsity achieves lower WERs on all evaluation sets with 50% sparsity level.

Table 3: *WERs of unified / adaptive sparsity at Sparsity 50%.*

Exp.	**WER**			
	Test	*TestOther*	*Dev*	*DevOther*
Baseline	2.1	4.9	2.3	4.9
Unified Sparsity	2.5	5.9	2.8	5.9
Adaptive Sparsity	2.4	5.6	2.6	5.7

04 With and without mask refinement

- The Mask Refinement phase maintains the original values for masked regions. The pruned variables are allowed to grow back, leading to higher flexibility and thus, lower WERs as shown in Table 4.

Table 4: *WERs of w/o and w/ Mask Refinement at Sparsity 40%.*

Exp.	**WER**			
	Test	*TestOther*	*Dev*	*DevOther*
w/o Mask Refine	2.3	5.5	2.6	5.7
w/ Mask Refine	2.3	5.3	2.5	5.3

05

Conclusion

05 Conclusion

- Improving the efficiency of federated learning: We propose Federated Pruning (FP) to leverage on-device data to effectively prune redundant parameters from models.
- Exploring different pruning design decisions: We explore and perform extensive ablation studies on two design decisions of pruning under federated learning: pruning patterns and pruning methods.
- Proposing a novel approach for adaptive sparsity: We propose a novel adaptive per-layer sparsity approach that dynamically allocates the target global sparsity level to each layer.
- Experimenting with production-grade environments: We evaluate the proposed Federated Pruning with production-grade models and datasets, which better reflects the real condition of deployment.

Google Research

Online Model Compression for On-Device Federated Learning

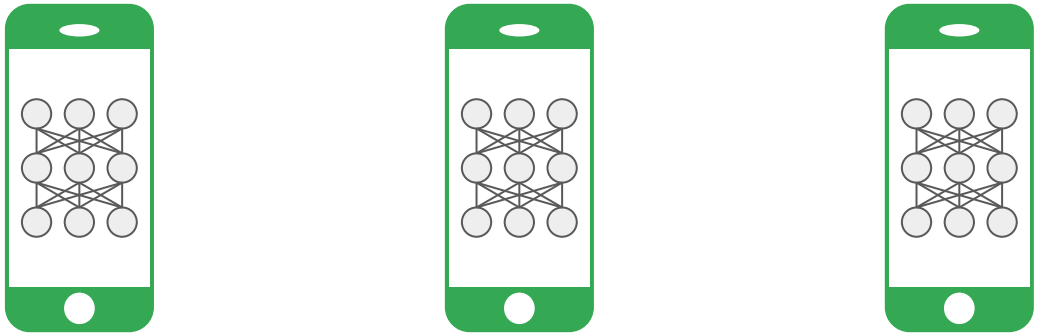
Tien-Ju Yang, Yonghui Xiao, Giovanni Motta, Françoise Beaufays,

Rajiv Mathews, Mingqing Chen

Google LLC

Google Research

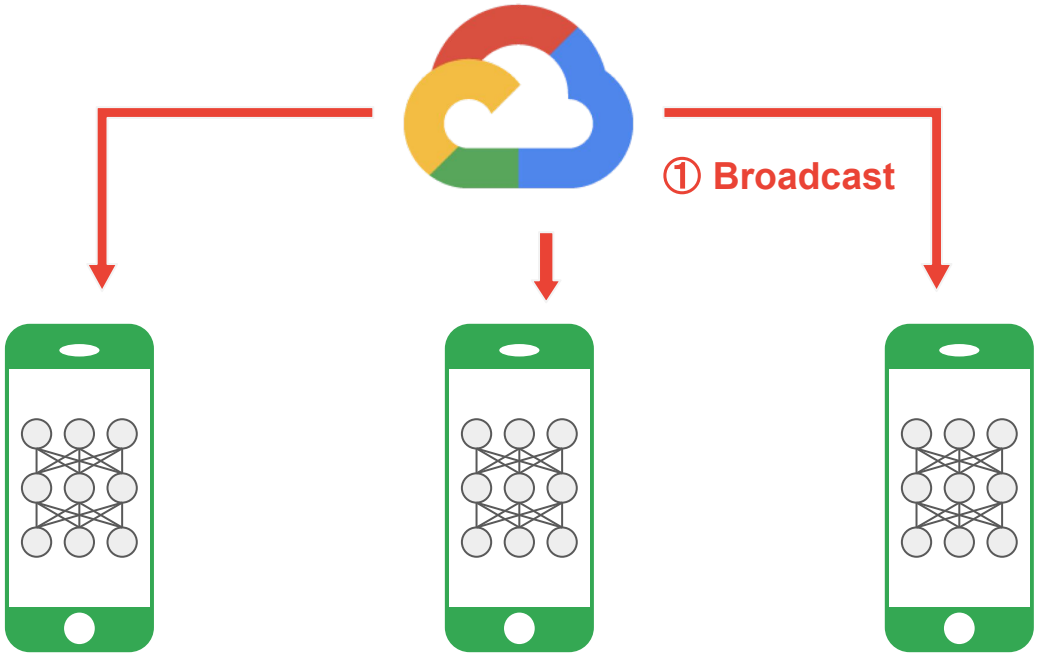
Federated Learning (FL)



FL trains neural network models on edge devices (clients) to preserve users' **privacy.**

Google Research

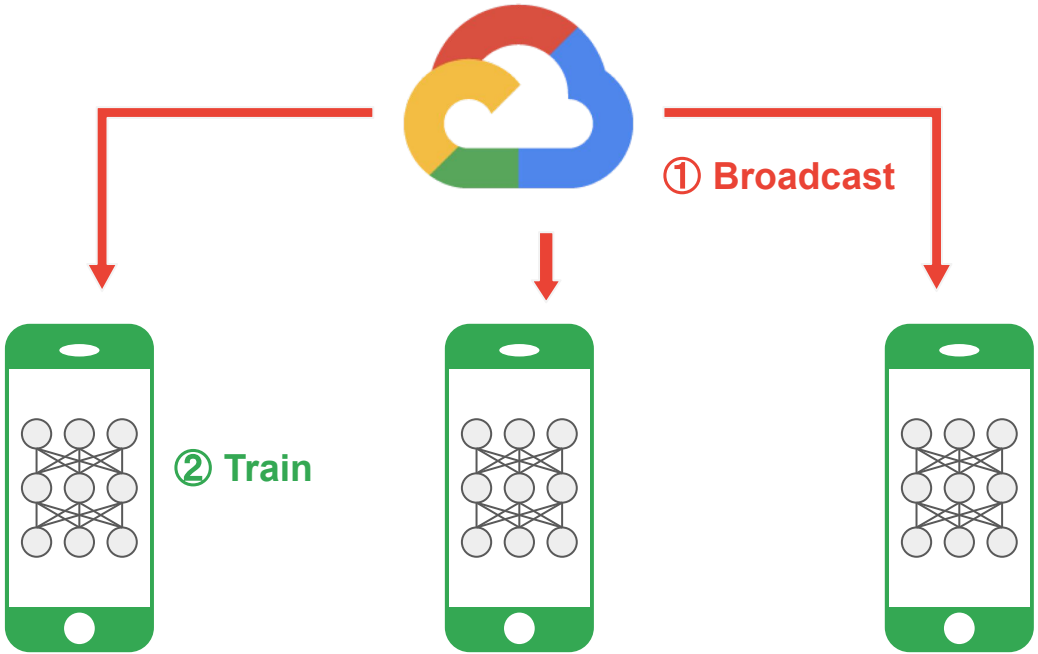
Federated Learning (FL)



FL trains neural network models on edge devices (clients) to preserve users' **privacy.**

Google Research

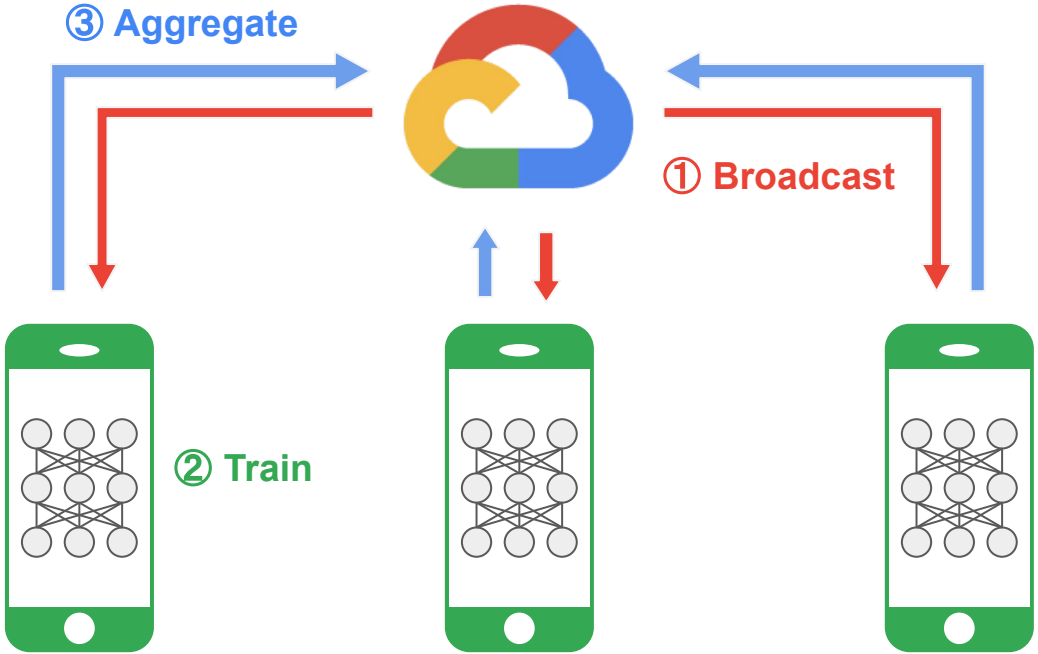
Federated Learning (FL)



FL trains neural network models on edge devices (clients) to preserve users' **privacy.**

Google Research

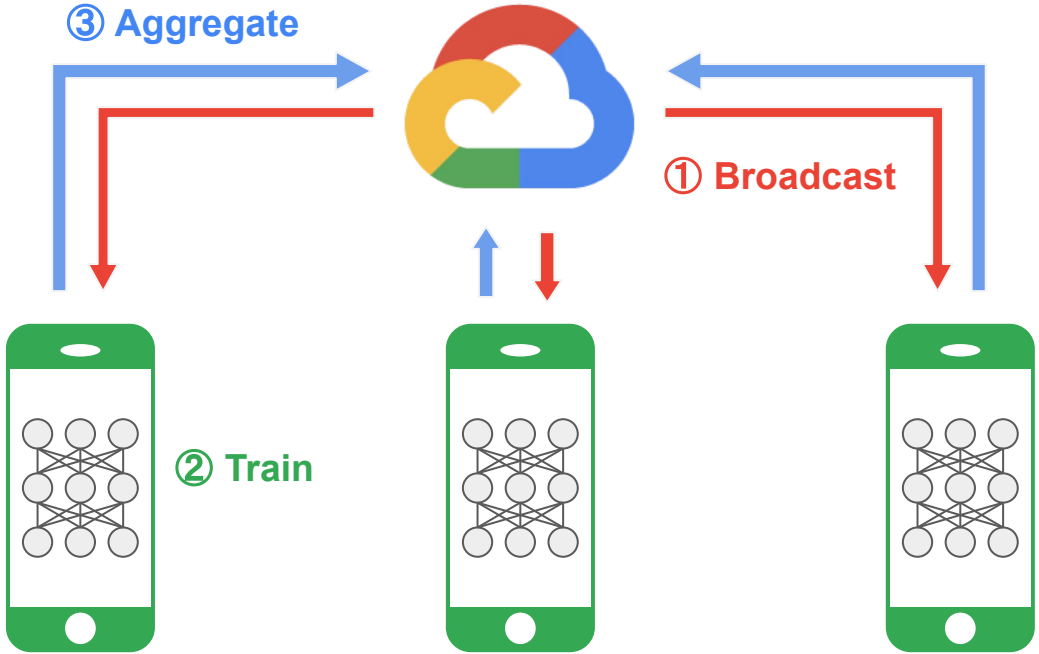
Federated Learning (FL)



FL trains neural network models on edge devices (clients) to preserve users' **privacy.**

Google Research

Federated Learning (FL)



Repeat Federated Round

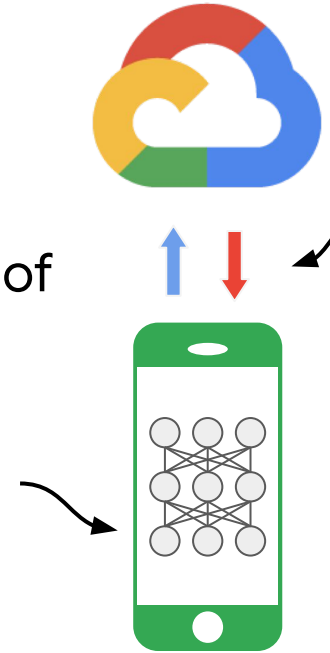
FL trains neural network models on edge devices (clients) to preserve users' **privacy.**

Google Research

Two Main Challenges

Limited on-device memory

Keeping hundreds of millions of parameters in full precision in memory can exceed the available memory.



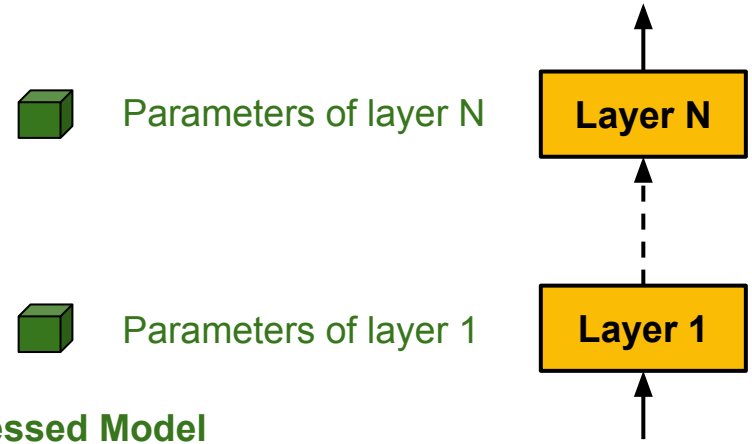
Costly communication

Transporting models in full precision burdens the communication network.

Google Research

Proposed Online Model Compression (OMC)

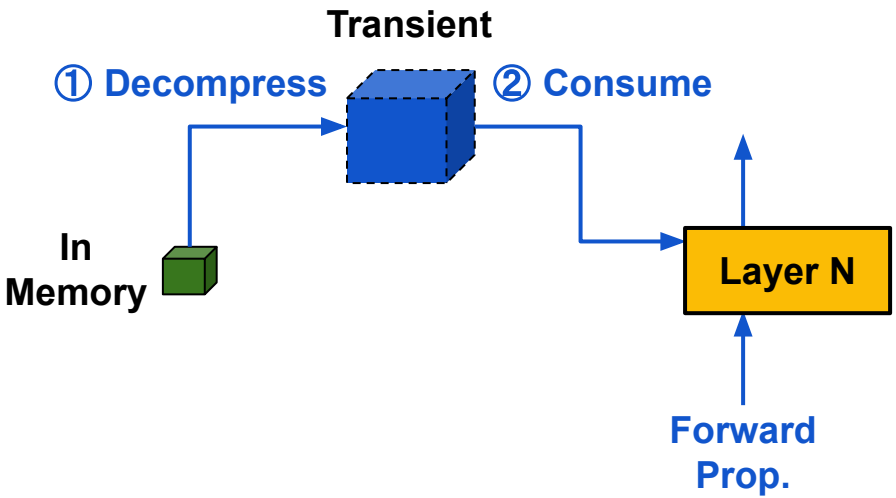
- **Idea:** store and transmit parameters in a compressed format and decompress them on the fly during training



Google Research

Forward Propagation of OMC

Compressed parameters are decompressed, consumed, and deallocated immediately

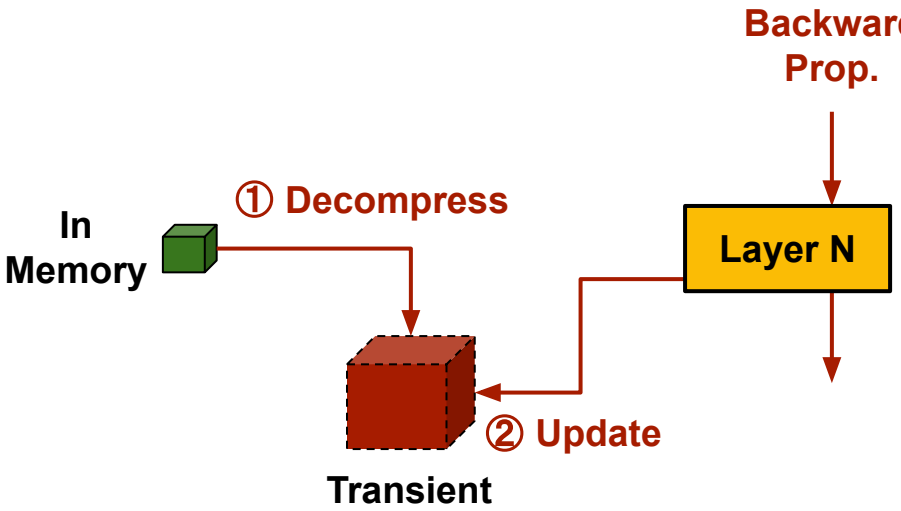


* The cubes with dashed borderlines are transient variables.

Google Research

Backward Propagation of OMC

Step 1, 2: Compressed parameters are decompressed and updated by gradients

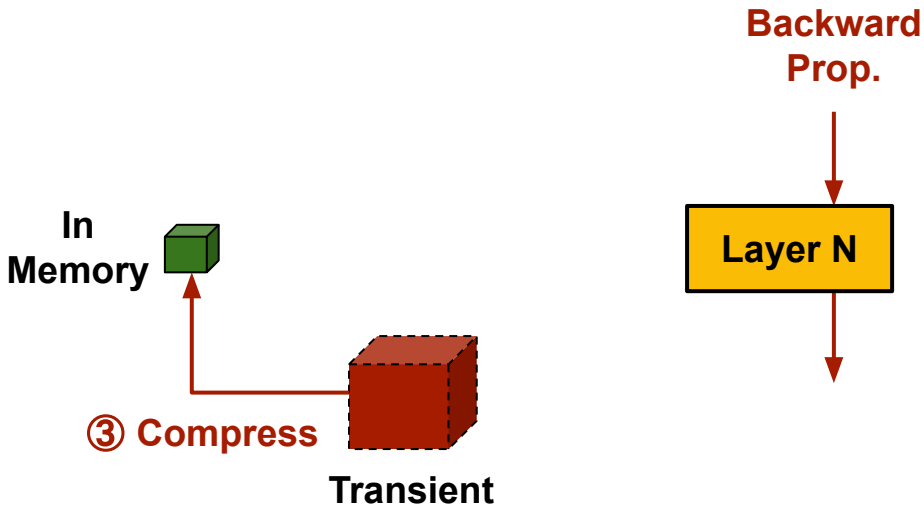


* The cubes with dashed borderlines are transient variables.

Google Research

Backward Propagation of OMC

Step 3: The updated decompressed parameters are compressed and deallocated

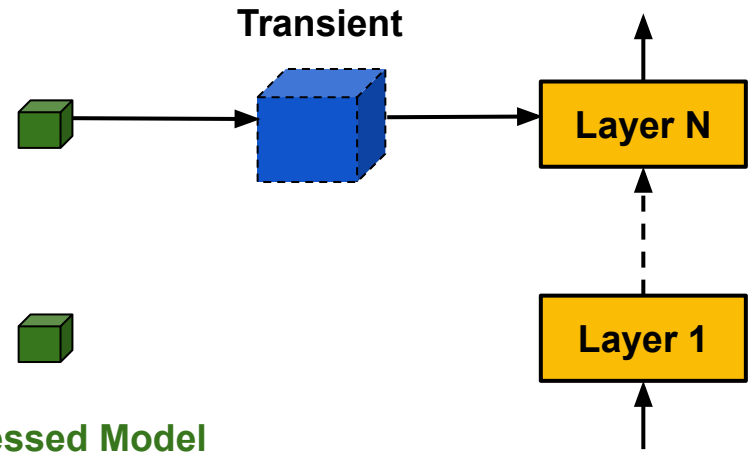


* The cubes with dashed borderlines are transient variables.

Google Research

Proposed Online Model Compression (OMC)

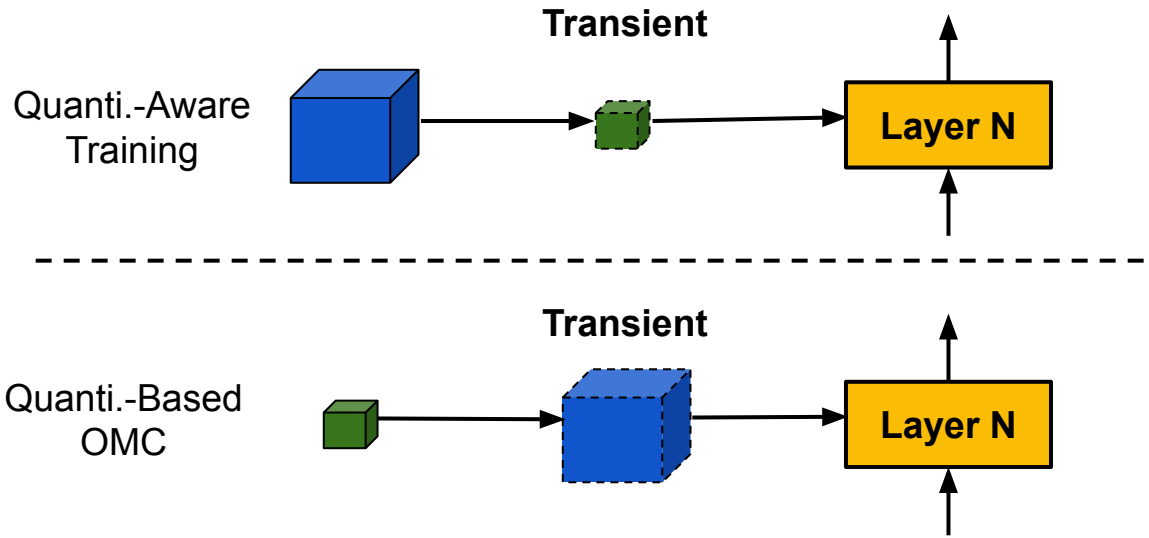
- OMC keeps only the compressed parameters and a small number of transient decompressed copies in the memory
- Decouple compression and hardware-support formats → lower memory/comm.
- Implementation of OMC should operate fast and can mitigate error accumulation

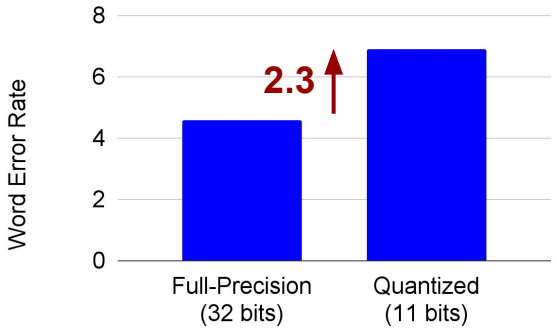


Google Research

Quantization-based OMC

- We adopt floating-point quantization as the compression format given its simplicity
- Different from quantization-aware training, quantization-based OMC stores quantized parameters ⟶ lower memory but faster error accumulation



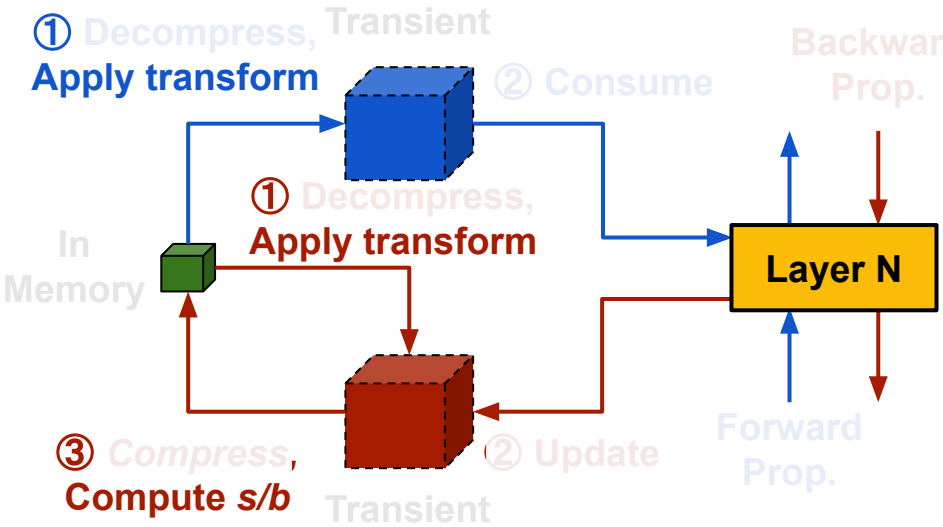


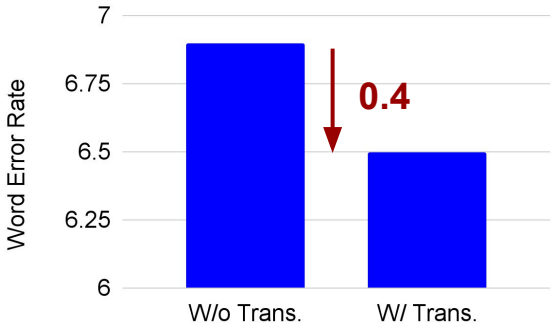
* Application: automatic speech recognition
* 130M Conformer on multi-domain dataset

Google Research

Per-Variable Transformation

- Apply per-variable transformation $\tilde{V}=sV+b$ after decompression to minimize errors
  - Storage overhead: 2 scalar values per-variable (scale and bias)
- The scale and bias are computed analytically during compression



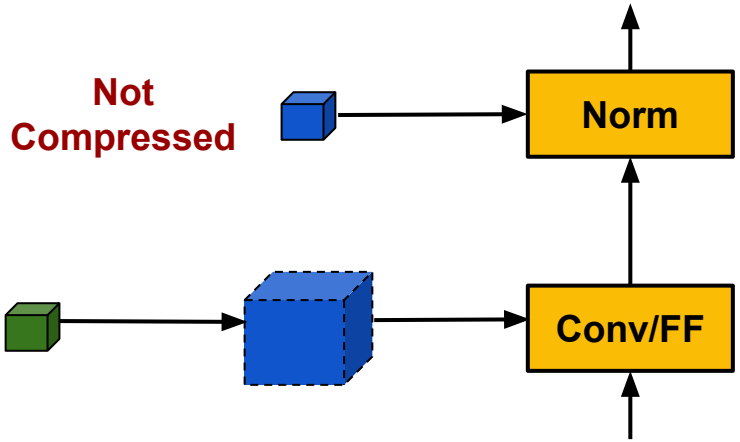


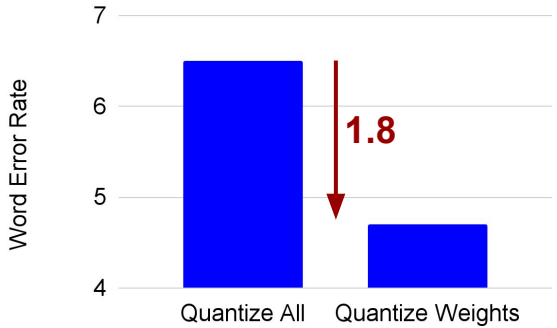
* Application: automatic speech recognition
* 130M Conformer on multi-domain dataset
* 11 bits

Google Research

Weight-Matrix-Only Quantization

- Observation: different parameter types have different sensitivity to quantization
  - Weights are less sensitive than the others but dominate model size
- Idea: quantize only weight matrices and keep remaining parameters in full precision



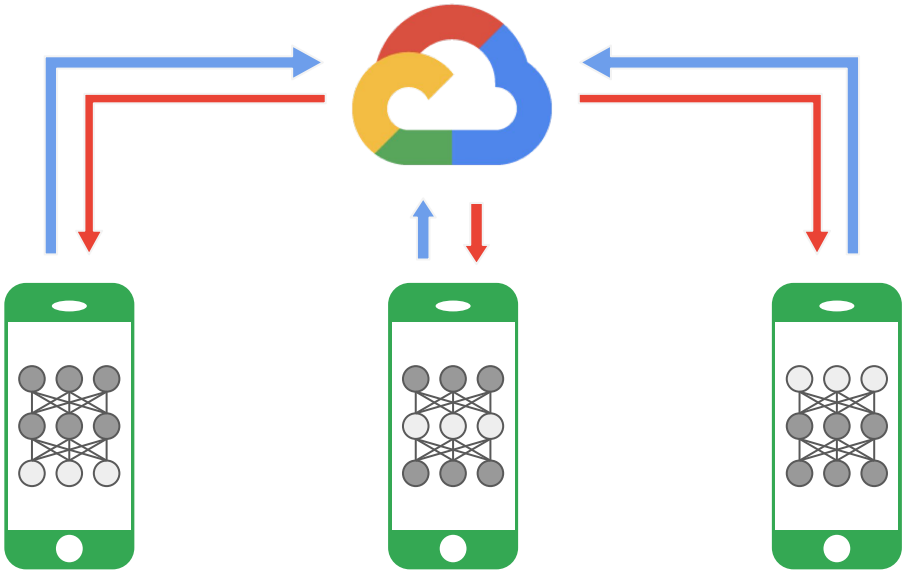


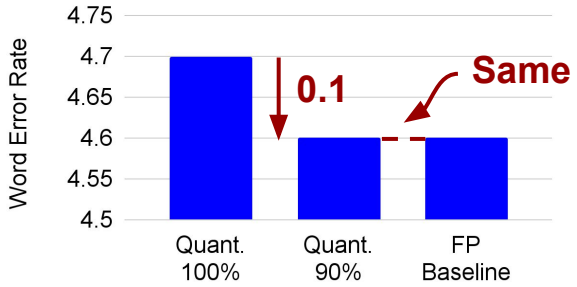
* Application: automatic speech recognition
* 130M Conformer on multi-domain dataset
* 11 bits

Google Research

Partial Variable Quantization

- Leverage the feature of FL: many clients train a model in parallel
- Idea: only quantize a subset of weight matrices (per-client)
  - Every weight matrix can obtain high-quality updates from some clients





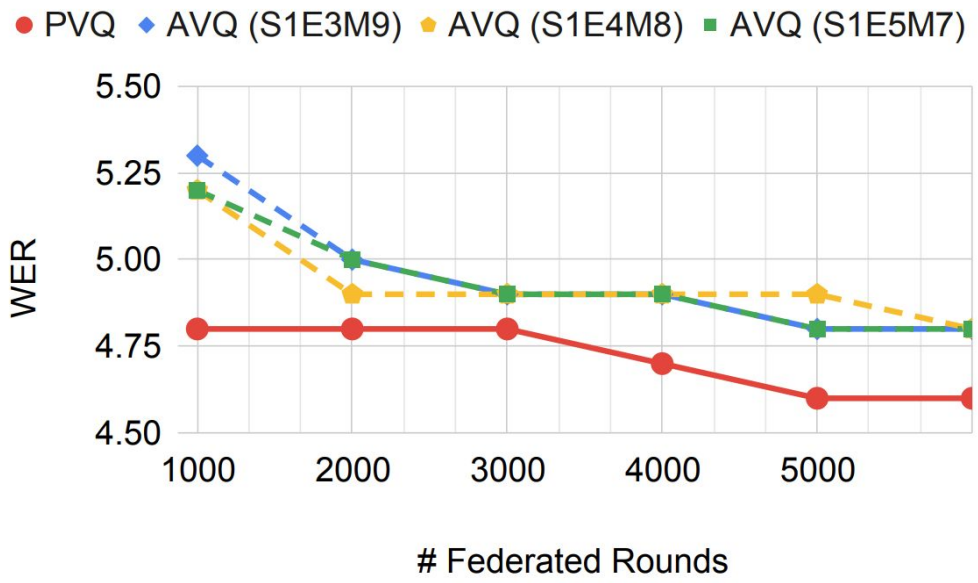
* Application: automatic speech recognition
* 130M Conformer on multi-domain dataset
* 11 bits

●: **quantized**
○: **full-precision**

Google Research

Partial Variable Quantization

- Partial variable quantization (PVQ) outperforms all variable quantization (AVQ, 100% quantization) with larger bit-widths



Example:

11-bit PVQ (90% quantized) outperforms 13-bit AVQ (100% quantized with various bit allocations) .

Google Research

Experimental Settings

- **Application:** automatic speech recognition
- **Models:**
  - Production-grade Conformers with 110M or 130M trainable parameters
- **Datasets:**
  - IID/non-IID LibriSpeech
  - Multi-Domain (MD) dataset contains ~400K hours of anonymized utterances from domains such as Search, Farfield, Telephony, and YouTube
- **Training scenarios:**
  - From-scratch training: IID/non-IID LibriSpeech
  - Transfer learning: first train on Non-MF (medium-form domain) from MD and then finetune on MF

Experimental Results

- From-scratch training 110M Conformer on IID/non-IID LibriSpeech
  - OMC (19 bits) achieves **similar WERs as FP32** for both IID and non-IID distributions
  - **64%** memory usage of parameters and communication cost
  - **Similar training speed**

Data Distribution		WER	Resource	
			Parameter Memory/ Communication	Speed (Rounds/Min)
IID	FP32 (32 bits)	2.1/4.6/2.2/4.8	474MB (100%)	29.5 (100%)
	OMC (19 bits)	**2.1/4.7/2.2/4.6**	**301MB (64%)**	**26.8 (91%)**
Non-IID	FP32 (32 bits)	2.0/4.7/2.2/4.9	474MB (100%)	29.5 (100%)
	OMC (19 bits)	**2.0/4.8/2.2/4.9**	**301MB (64%)**	**26.8 (91%)**

* WERs in dev/dev-other/test/test-other

Google Research

Detailed analysis in paper

Experimental Results

- Transfer learning of 130M Conformer on multi-domain dataset
  - OMC (11 bits) achieves **similar WERs** as FP32
  - OMC (6 bits) improves upon the before-adaptation model
  - **41% and 29%** memory usage of parameters and communication cost
  - **Similar training speed**

	WER	Resource	
		Parameter Memory/ Communication	Speed (Rounds/Min)
Before Adaptation	6.7	-	-
FP32 (32 bits)	4.6	548MB (100%)	11.9 (100%)
OMC (11 bits)	**4.6**	**224MB (41%)**	**11.1 (93%)**
OMC (6 bits)	**5.9**	**147MB (29%)**	**11.1 (93%)**

Google Research

Detailed analysis in paper

Experimental Results

- Measured memory usage on Pixel 4 phones
  - Implemented with Tensorflow Federated

- Results
  - 16-bit OMC achieves the same WERs as the FP32 baseline
  - Large Conformer: 197MB peak memory usage reduction (38% of model size)
  - Small Conformer: 84MB peak memory usage reduction (45% of model size)

Conclusion

- We proposed Online Model Compression (OMC) to enable training large models with federated learning
  - OMC allows to reduce memory usage and communication cost while maintaining training speed
- Our implementation of OMC includes
  - Floating point quantization, per-variable transformation, weight-matrix-only quantization, and partial variable quantization
- The experiments show that OMC can significantly improve FL efficiency with comparable accuracy as full-precision training

Google Research

FedAQT: Accurate Quantized Training with Federated Learning

Renkun Ni[1], Yonghui Xiao[2], Phoenix Meadowlark[2], Oleg Rybakov[2], Tom Goldstein[1], Ananda Theertha Suresh[2], Ignacio Lopez Moreno[2], Mingqing Chen[2], Rajiv Mathews[2]
[1]University of Maryland [2]Google LLC

Google Research

ICASSP 2024 KOREA

Overview

Federated learning (FL) has been widely used to train neural networks with the decentralized training procedure where data is only accessed on clients' devices for privacy preservation. However, the limited computation resources on clients' devices prevent FL of large models. To overcome the computational constraint and enable FL of the Conformer based ASR models, we propose FedAQT, an accurate quantized training framework under FL by training with quantized variables directly on clients' devices. We empirically show that our method can achieve comparable WER with only 60% memory of the full-precision model.

Google Research

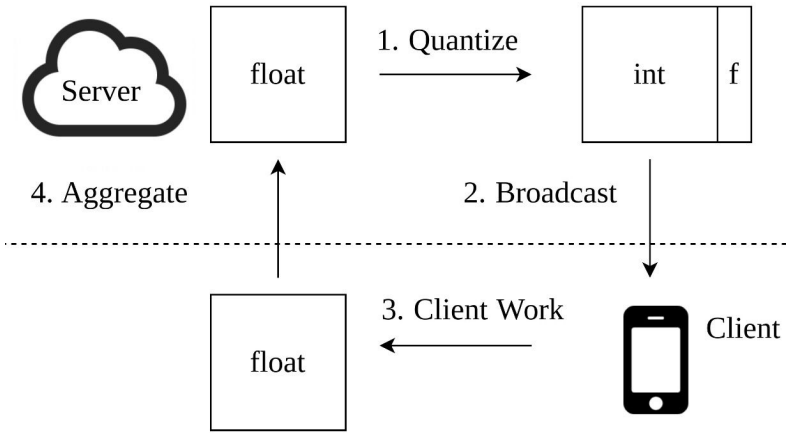
Federated Round of FedAQT

1. Quantize: We quantize the full-precision model to the target bit-width on the cloud. Both the full-precision and low-bit model is kept.
2. Broadcast: Only the low-bit model as well as float scalers will be distributed to the local devices.
3. Client work: To save memory during training, at the client training stage, each device trains the quantized model only with its own training data.
4. Aggregation and model update: After local training, we aggregate the float updates from the clients, and then updates the float model on the cloud.

Different from the typical quantization-aware training, where a full-precision is usually kept during training, we only keep the quantized model on the local devices, thus the memory will be saved when we load the quantized model.

Google Research

Federated Round



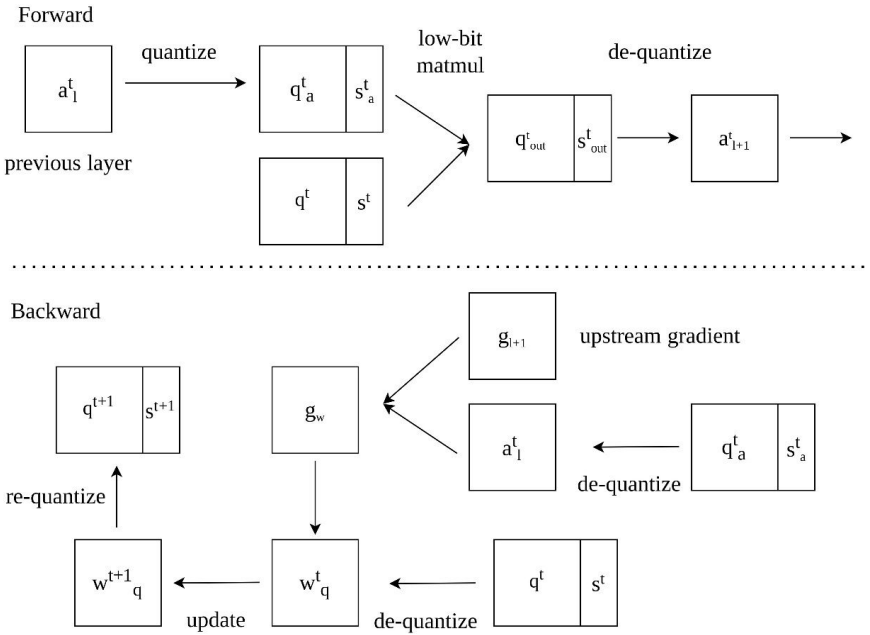
Google Research

Quantization Method

- Forward Pass: We quantize the activations from the previous layer and apply low-bit matrix multiplication with the quantized model weights.
- Backward Pass: To avoid performance drop, we still calculate the full-precision gradient, where we de-quantize the activation variable.
- FedSGD: Once the full-precision gradient is calculated, we can aggregate the local model updates to the server.
- FedAVG: Instead of updating the low-bit model weights, we de-quantize them and apply the gradients. Then, we re-quantize it again for the next iteration.

Google Research

Quantization in FL



Google Research

Experiments

For FedSGD, we follow the IID training regime and only use a small batch size of 2 to meet the memory requirement. We finetune the conformer-based model for 6000 steps. With our FedAQT framework, we show that similar WER can be achieved with various bit-width compared to the full-precision model.

Bit-width	test-clean	test-other	dev-clean	dev-other
Float32	4.8	10.2	4.7	10.3
int8	4.8	10.2	4.7	10.3
int4	4.9	10.3	4.7	10.3

Google Research

Experiments

In addition, we implement the real training process with fake gradients. On the broadcast stage, we distribute quantized variables, namely int8 variables and float scales, to the clients. During the client computation, instead of calculating the gradient for real int8 variables, we generate empty gradients as local updates and aggregate them to update the server model. Compared to full-precision model, we could save around 40% of the memory.

Variable Dtype	Memory (KB)	Saving (KB)
Float32	734188	-
Float16	568012	166176
Int8	468312	265876

Q&A

Thanks

Google Research