FEDERATED LEARNING

SUMMARY OF
MODERN FEDERATED LEARNING METHODS

REFERENCES

- Reddi, Sashank, et al.
  "Adaptive Federated Optimization."
  arXiv preprint arXiv:2003.00295 (2020).
- Wang, Hongyi, et al.
  "Federated learning with matched averaging."
  arXiv preprint arXiv:2002.06440 (2020).
- Xie, Cong, Oluwasanmi Koyejo, and Indranil Gupta.
  "Generalized byzantine-tolerant sgd."
  arXiv preprint arXiv:1802.10116 (2018).

REFERENCES

- Zhang, Jingzhao, et al.
"Why ADAM beats SGD for attention models."
arXiv preprint arXiv:1912.03194 (2019).
- Yurochkin, Mikhail, et al.
"Bayesian nonparametric federated learning of neural networks."
arXiv preprint arXiv:1905.12022 (2019).
- Wang, Yushi.
"Co-op: Cooperative machine learning from mobile devices."
(2017).

REFERENCES

- Xie, Cong, Sanmi Koyejo, and Indranil Gupta.
”Zeno++: Robust Fully Asynchronous SGD."
arXiv preprint arXiv:1903.07020 (2019).
- Lalitha, Anusha, et al.
"Peer-to-peer federated learning on graphs."
arXiv preprint arXiv:1901.11173 (2019).
- Li, Daliang, and Junpu Wang.
"Fedmd: Heterogenous federated learning via model distillation."
arXiv preprint arXiv:1910.03581 (2019).

CONTENTS

- What is
  - Federated Learning
  - FedAvg
- Problems on Federated Learning
- Solutions
  - Adaptive Optimizer
  - Matched Averaging
  - Byzantine-tolerant
- TODO

FEDERATED LEARNING

FEDERATED LEARNING

- 연합 학습(Federated Learning, FL)
  - 분산 머신 러닝 패러다임
  - 많은 수의 클라이언트들이 중앙 서버에게
  - 자신의 훈련 데이터를 공유하지 않고 학습에 협력

FEDERATED LEARNING

- 클라이언트들이 모델을 업데이트
  - 임의의 클라이언트 최적화기를 이용해
  - 로컬 데이터의 손실(loss)을 줄이기 위해

FEDERATED LEARNING

- 서버는 글로벌 모델을 업데이트
  - 클라이언트가 송신한 모델 업데이트의 평균으로부터
    - 로컬 모델의 파라미터들을 element-wise하게 평균
  - 클라이언트들에 걸친 손실을 최소화하기 위해

FEDERATED LEARNING

- 연합 학습(Federated Learning, FL)
    - 중앙 서버의 조율 하에서
    - 많은 수의 클라이언트가 모델 학습에 협력하는
        - 엣지(edge) 디바이스
        - 여러 기관 등
    - 머신 러닝 패러다임

FEDERATED LEARNING

- FL의 핵심은
  - 클라이언트의 원(raw) 데이터가
  - 절대로 서버나 다른 클라이언트에게
  - 공유되지 않는다는 것

FEDAVG

FEDAVG

- FEDAVG 기법
  - 각 라운드에서 클라이언트들의 서브셋이 선택되고 (주로 무작위로)
  - 서버가 그들에게 글로벌 모델을 브로드캐스트
  - 병렬적으로 클라이언트들이 고유의 손실 함수를 통해 SGD를 수행
  - 서버에게 모델을 전송
  - 서버는 로컬 모델들의 평균을 구해 글로벌 모델을 업데이트

FEDAVG

- FEDAVG 기법

$$x_i^t = \textsc{Sgd}_K(x_t, \eta_l, F_i)$$

  - 그라디언트 $\nabla f_i(x, z)$에 대해
  - 로컬 학습률 η_l 을 통해
  - x_t 에서 시작해
  - K 스텝만큼 SGD

FEDAVG

- FEDAVG 기법

Algorithm 1 **Simplified FEDAVG**

Initialization: x_0

for $t = 0, \cdots, T-1$ **do**

  Sample subset $\mathcal{S}$ of clients

  $x_{i,0}^t = x_t$

  for each client $i \in \mathcal{S}$ **in parallel do**

    $x_i^t = \text{SGD}_K(x_t, \eta_l, F_i)$

  $x_{t+1} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} x_i^t$

FEDAVG

- FEDAVG 기법의 재작성
  - 각 라운드에서 클라이언트들의 서브셋이 선택되고 (주로 무작위로)
  - 서버가 그들에게 글로벌 모델을 브로드캐스트
  - 병렬적으로 클라이언트들이 고유의 손실 함수를 통해 SGD를 수행
  - 서버에게 델타(모델의 업데이트 정보)를 전송
  - 서버는 델타들의 평균을 구해 글로벌 모델을 업데이트

FEDAVG

- FEDAVG 기법의 재작성

$$x_{t+1} = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} x_{i,K}^{t} = x_t - \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\left(x_t - x_{i,K}^{t}\right)$$

- $\Delta_i^t := x_{i,K}^t - x_t$
- $\Delta^t = \frac{1}{|S|}\Sigma\Delta_i^t$

FEDAVG

- 서버는
  - 평균을 통한 수도-그라디언트(pseudo-gradient)를
  - 학습률 1로 SGD함과 동일

FEDAVG

- FEDAVG 기법의 재작성
- 수도-그라디언트 관점에서 일반화

Algorithm 2 **Generalized** FEDAVG

Initialization: x_0

for $t = 0, \cdots, T-1$ **do**

  Sample subset $\mathcal{S}$ of clients

  $x_{i,0}^t = x_t$

  for each client $i \in \mathcal{S}$ **in parallel do**

    for $k = 0, \cdots, K-1$ **do**

      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$

      $x_{i,k+1}^t = \textsc{ClientOpt}(x_{i,k}^t, g_{i,k}^t, \eta_l, t)$

    $\Delta_i^t = x_{i,K}^t - x_t$

  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$

  $x_{t+1} = \textsc{ServerOpt}(x_t, -\Delta_t, \eta, t)$

FEDAVG

- 일부에서는 가중 평균(weighted average)를 사용
  - 클라이언트의 훈련 샘플의 수에 따른 가중 평균
  - 클라이언트의 데이터에 대한 메타데이터에 해당
  - Uniform한 평균 대비 일반적으로 좋은 성능

PROBLEMS

PROBLEMS ON FEDERATED LEARNING

- FL의 문제점
    - 이종적인 데이터
    - 높은 통신 비용
    - 클라이언트 드리프트
    - 적응형 학습률의 부재

PROBLEMS ON FEDERATED LEARNING

- 기존의 분산 최적화와는 달리
  - 연산의 병렬화에 초점
  - 데이터는 할당됨
- FL은 이종적인 데이터를 다룬다는 점이 문제이자 쟁점
  - 클라이언트마다의 데이터 편향이 극심

PROBLEMS ON FEDERATED LEARNING

- 미니배치(mini-batch)를 사용하는 표준적인 접근법은
  - 한 번 (배치 혹은 에폭) 학습하고 통신하고
- 높은 통신 비용을 초래하므로
- 많은 FL 환경에 적합하지 않음

PROBLEMS ON FEDERATED LEARNING

- 이에 FL을 위한 많은 최적화 기법은
  - 로컬(local) 클라이언트 업데이트를 활용
  - 클라이언트가 통신 전에 로컬 모델을 수 차례 업데이트
- FL에 널리 쓰이는 로컬 최적화 기법 중 하나인
  - FEDAVG

PROBLEMS ON FEDERATED LEARNING

- FEDAVG의 각 라운드에서
  - 클라이언트의 일부는 병렬적으로 SGD의 몇 에폭을 수행
  - 클라이언트 드리프트(client drift) 발생
- 클라이언트는 모델 업데이트 정보를 서버와 통신
  - 서버는 평균을 통해 새 글로벌 모델을 계산
  - 적응적 학습률(adaptive learning rate)의 부재

CLIENT DRIFT

- 클라이언트 드리프트 (client drift)
  - 이종적 환경에서 여러 로컬 SGD 에폭의 수행은
  - 글로벌한 최적 모델로부터 멀리 떨어지게(drift) 만듦

ADAPTIVE LEARNING RATE

- 적응적 학습률 (adaptive learning rate)
  - 학습 과정 중의 Heavy-tail 노이즈 분포 등
  - 필연적으로 적응적 학습률이 필요한 환경들이 있음
  - 이는 과거의 반복(iteration)들로부터 정보에 기반한
  - 그래디언트 기반의 최적화를 수행하도록 함
- 정보 공유를 하지 않는 FL의 기본 속성에 따르면
  - 적응형 학습률은 도전적인 문제임

ADAPTIVE OPTIMIZER

ADAPTIVE OPTIMIZER

- 컨트리뷰션
  - 클라이언트와 서버가 다른 학습률을 활용할 수 있도록 함
  - FL을 위한 적응형 최적화 기술들을 개발

ADAPTIVE OPTIMIZER

- 클라이언트와 서버의 학습률을 분리하는 자연스러운 방법을 제시
  - 즉, FEDAVG의 일반화
  - FEDAVG는 클라이언트와 서버가 SGD를 쓰고, 서버의 학습률이 1인 경우
- 클라이언트와 서버가 다른 학습률을 활용할 수 있도록 함
  - 로컬 업데이트의 억제 & 강화가 가능
  - 클라이언트 드리프트를 해결할 방법을 제공

ADAPTIVE OPTIMIZER

- 적응형 방법들인
  - ADAGRAD, ADAM, YOGI 등은 머신 러닝 커뮤니티에서 유명한 기법들
  - 학습률을 튜닝할 수고를 덜고
  - Heavy-tail 노이즈 분포 등을 잘 다룸
- 이들 프레임워크 위에서
  - FL을 위한 적응형 최적화 기술들을 개발
  - 최초의 FL을 위한 적응형 최적화 기법

FEDADAGRAD

- FEDADAGRAD

Algorithm 3 FEDADAGRAD

Initialization: x_0, $\tau > 0$ and $v_{-1} \geq \tau^2$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} + \Delta_t^2$
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

FEDADAGRAD

- FEDADAGRAD

Algorithm 3 FEDADAGRAD

Initialization: x_0, $\tau > 0$ and $v_{-1} \geq \tau^2$

for $t = 0, \cdots, T-1$ **do**

Sample subset $\mathcal{S}$ of clients

$x_{i,0}^t = x_t$

for each client $i \in \mathcal{S}$ **in parallel do**

for $k = 0, \cdots, K-1$ **do**

Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$

$\boxed{x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t}$ CLIENTOPT = SGD

$\Delta_i^t = x_{i,K}^t - x_t$

$\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$

$v_t = v_{t-1} + \Delta_t^2$

$x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

FEDADAGRAD

- FEDADAGRAD

Algorithm 3 FEDADAGRAD

Initialization: x_0, $\tau > 0$ and $v_{-1} \geq \tau^2$

for $t = 0, \cdots, T-1$ **do**

Sample subset $\mathcal{S}$ of clients

$x_{i,0}^t = x_t$

for each client $i \in \mathcal{S}$ **in parallel do**

for $k = 0, \cdots, K-1$ **do**

Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$

$x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$

$\Delta_i^t = x_{i,K}^t - x_t$

$\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$

$v_t = v_{t-1} + \Delta_t^2$

$x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

통합 후 평균

FEDADAGRAD

- FEDADAGRAD

Algorithm 3 FEDADAGRAD

Initialization: x_0, $\tau > 0$ and $v_{-1} \geq \tau^2$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} + \Delta_t^2$
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

ADAGRAD

FEDYOGI AND FEDADAM

- FEDYOGI & FEDADAM

Algorithm 4 **FedYogi** (and **FedAdam**)

Initialization: $x_0, v_{-1} \geq \tau^2$, decay $\beta_2 \in (0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_{t-1}$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} - (1-\beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FedYogi)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$ **(FedAdam)**
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

FEDYOGI AND FEDADAM

- FEDYOGI & FEDADAM

Algorithm 4 FEDYOGI (and **FEDADAM**)

Initialization: $x_0, v_{-1} \geq \tau^2$, decay $\beta_2 \in (0,1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\boxed{\Delta_i^t = x_{i,K}^t - x_{t-1}}$ CLIENTOPT = SGD
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} - (1-\beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t}+\tau}$

FEDYOGI AND FEDADAM

- FEDYOGI & FEDADAM

Algorithm 4 **FEDYOGI** (and **FEDADAM**)

Initialization: $x_0, v_{-1} \geq \tau^2$, decay $\beta_2 \in (0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x^t_{i,0} = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g^t_{i,k}$ of $\nabla F_i(x^t_{i,k})$
      $x^t_{i,k+1} = x^t_{i,k} - \eta_l g^t_{i,k}$
    $\Delta^t_i = x^t_{i,K} - x_{t-1}$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta^t_i$
  $v_t = v_{t-1} - (1-\beta_2)\Delta^2_t \operatorname{sign}(v_{t-1} - \Delta^2_t)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta^2_t$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

통합 후 평균

FEDYOGI AND FEDADAM

- FEDYOGI & FEDADAM

Algorithm 4 **FedYogi** (and **FedAdam**)

Initialization: $x_0, v_{-1} \geq \tau^2$, decay $\beta_2 \in (0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x^t_{i,0} = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g^t_{i,k}$ of $\nabla F_i(x^t_{i,k})$
      $x^t_{i,k+1} = x^t_{i,k} - \eta_l g^t_{i,k}$
    $\Delta^t_i = x^t_{i,K} - x_{t-1}$
  $\Delta_t = \frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} \Delta^t_i$
  $v_t = v_{t-1} - (1-\beta_2)\Delta^2_t \operatorname{sign}(v_{t-1} - \Delta^2_t)$ **(FedYogi)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta^2_t$ **(FedAdam)**
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t}+\tau}$

YOGI 또는 ADAM

ADAPTIVE LEARNING RATE

- FEDAVG 대비 FEDADAGRAD, FEDADAM, FEDYOGI를 비교
  - 서버의 학습률이 조정된다는 특징
  - 학습률이 1 고정이 아님
- 서버 최적화기가
  - 조정된 학습률과
  - 모멘텀 파라미터 0.9를 가지는 SGD와도 비교
  - FEDAVGM으로 표기

FEDAVGM

- 서버 최적화기가
  - 조정된 학습률과
  - 모멘텀 파라미터 0.9를 가지는 SGD
  - FEDAVGM으로 표기

EVALUATION

- 적응형 최적화기만 사용해도 FL의 성능이 크게 오름

FED...	ADAGRAD	ADAM	YOGI	AVGM	AVG
CIFAR-100	23.9	**42.3**	41.6	37.3	26.5
EMNIST CR	**85.7**	**86.0**	**86.1**	**86.3**	85.9
SHAKESPEARE	57.1	**57.4**	**57.6**	**57.5**	57.0
STACKOV... NWP	11.3	**22.1**	**22.2**	13.7	9.5
STACKOV... LR	**0.68**	0.62	0.64	0.22	0.19
EMNIST AE	7.29	16.99	**0.98**	**1.21**	2.63

MATCHED AVERAGING

MATCHED AVERAGING

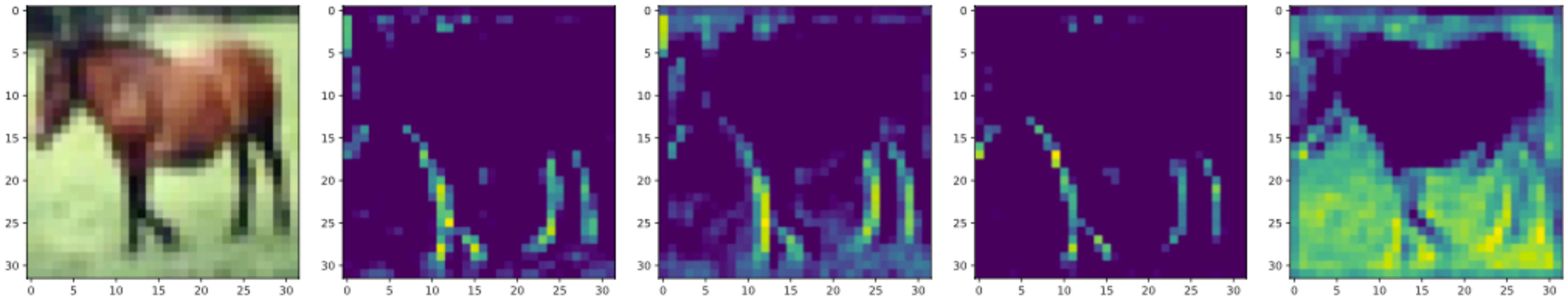
- FedAvg에서 coordinate-wise averaging
  - 여러 문제가 생길 수 있음
  - 이는 신경망 파라미터의 치환 불변성 때문

PERMUTATION INVARIANCE

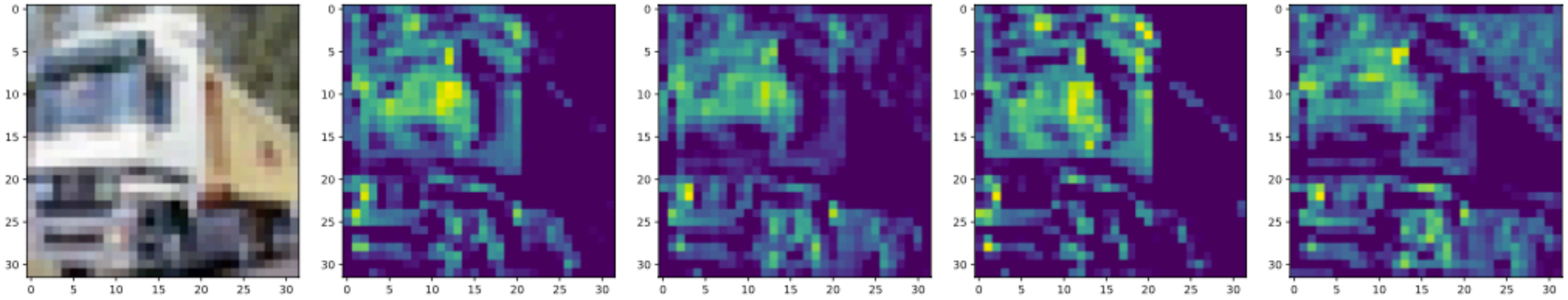
- 치환 분별성(permutation invariance)
  - 주어진 신경망 파라미터의 순서만 바꿔서 여러 변형을 만들 수 있음
- $\hat{y} = \sigma(xW_1)W_2$ 과 $\hat{y} = \sigma(xW_1\Pi)\Pi^T W_2$

PERMUTATION INVARIANCE

- 필터의 매칭을 시각화해서 살펴보면 더 직관적:



(a) Raw input1 (b) FedMA filter 0 (c) Client1, filter 0 (d) Client2, filter 23 (e) FedAvg filter 0



(f) Raw input2 (g) FedMA filter 0 (h) Client 1, filter0 (i) Client 2, filter23 (j) FedAvg filter0

PERMUTATION INVARIANCE

- 최적의 가중치가 $\{W_1, W_2\}$ 이라 가정,
- 두 이종적 데이터셋 $X_j, X_{j'}$ 에 대한 훈련된 가중치는
  - $\{W_1\Pi_j, \Pi_j^T W_2\}$ 와 $\{W_1\Pi_{j'}, \Pi_{j'}^T W_2\}$
- 높은 확률로 $\Pi_j \neq \Pi_{j'}$ 이므로
- 어떠한 Π에 대해서도 $(W_1\Pi_j + W_1\Pi_{j'})/2 \neq W_1\Pi$

PERMUTATION INVARIANCE

- 의미있는 평균을 위해서는 치환을 풀어야 함
  - $(W_1\Pi_j\Pi_j^T + W_1\Pi_{j'}\Pi_{j'}^T)/2 = W_1$

PROBABILITY FEDERATED NEURAL MATCHING

- 확률적(Probability) Federated Neural Matching (PFNM)
  - Averaging 하기 전
  - 클라이언트 신경망의 뉴런들을 매칭함으로써 문제를 다룸

PROBABILITY FEDERATED NEURAL MATCHING

- PFNM이 FedAvg 대비
  - 우수한 성능과 효율적인 커뮤니케이션 비용을 보임
  - 그러나 PFNM은 단순한 신경망에서만 동작
    - 가령, 전연결 feedforward 네트워크

PROBABILITY FEDERATED NEURAL MATCHING

- PFNM을 CNN과 LSTM에 적용
  - 그러나 가중치 평균 방법 대비 매우 적은 수준의 향상만 있었음
- PFNM을 CNNs와 LSTMs로 확장할 수 있지만
  - 깊은 구조들에서 효과를 보지 못함
  - 반복적인 matched averaging이
  - 전체적으로 좋지 못한 결과를 야기한 것으로 추정

MATCHED AVERAGING

- Federated Matched Averaging (FedMA) 방법을 제안
  - 최신 CNN이나 LSTM을 위한 새로운 layer-wise 연합 학습 알고리즘
  - 커뮤니케이션 비용을 줄이고, SOTA의 성능을 보임

MATCHED AVERAGING

- Layer-wise한 매칭 스킴을 제안
  - 데이터 센터(서버)가 클라이언트로부터 레이어의 첫 층에 해당하는 가중치를 수집
  - 한 레이어에 해당하는 매칭을 수행
  - 연합 모델의 가중치를 클라이언트들에게 브로드캐스트
  - 클라이언트는 반영 후 연속되는 모든 레이어를 고유 데이터셋으로 학습
    - 이 때 연합 레이어는 동결함
  - 마지막 레이어에 이르기까지 반복

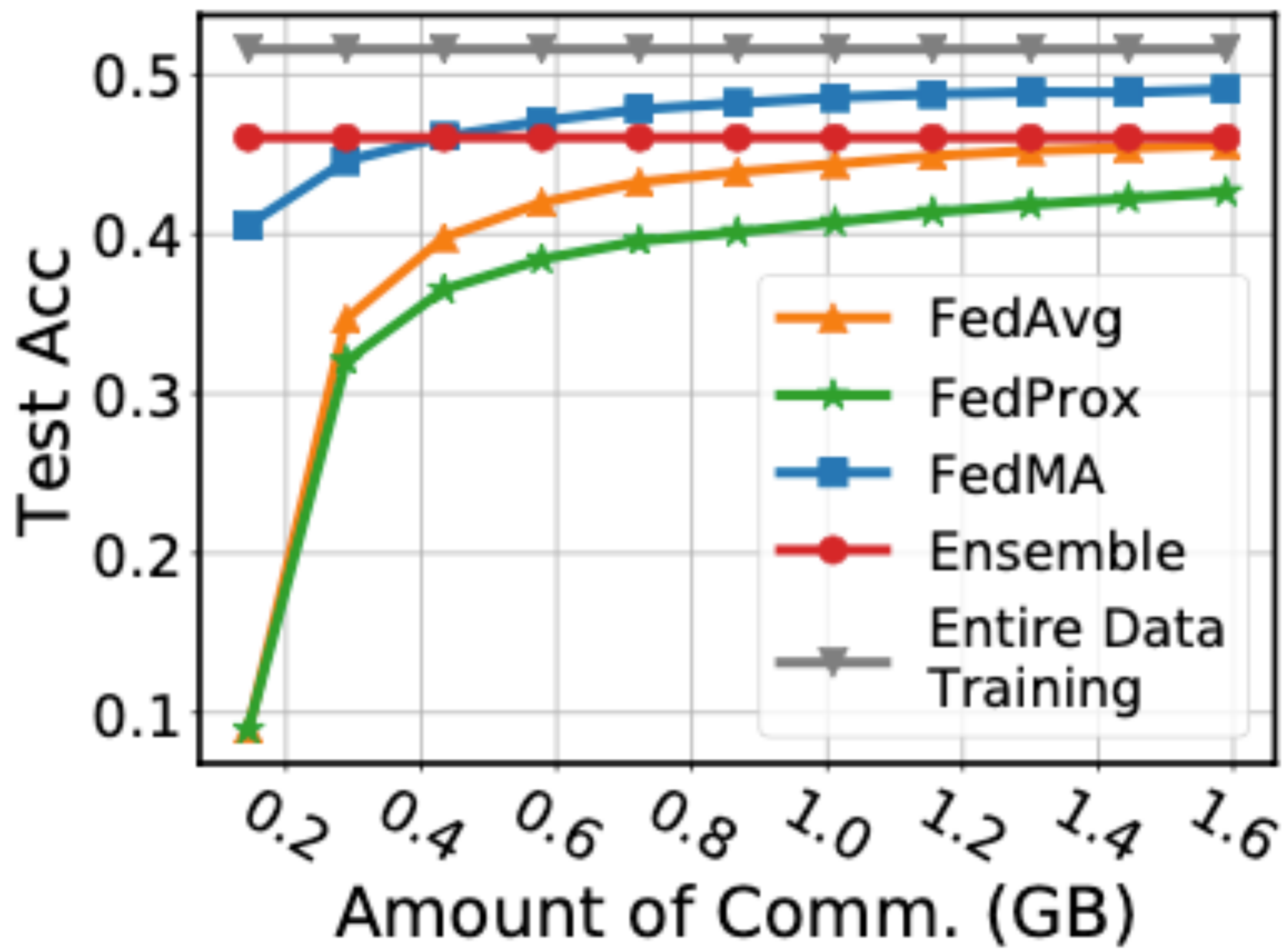
MATCHED AVERAGING

- **Algorithm 1:** Federated Matched Averaging (FedMA)

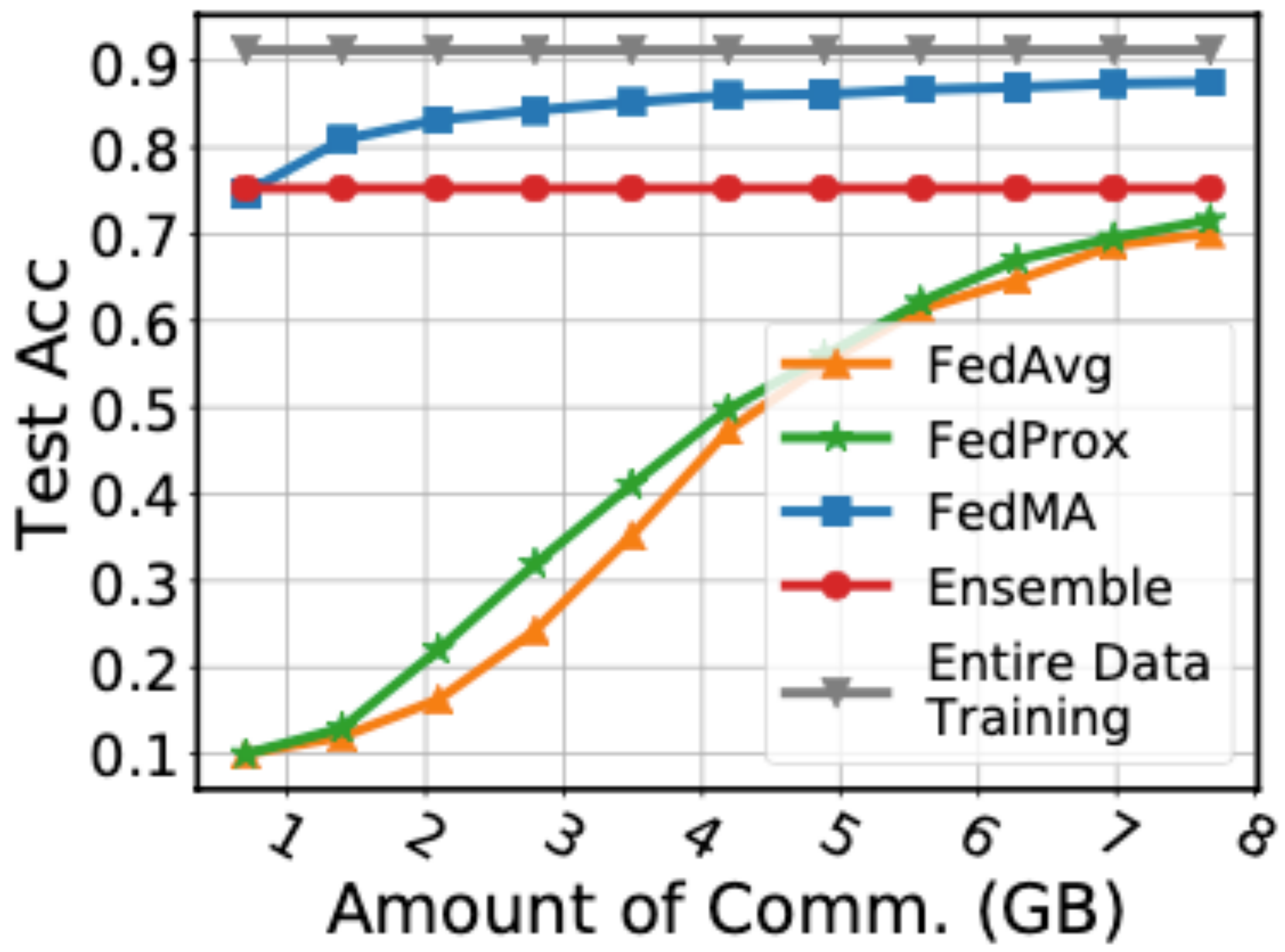
Input : local weights of N-layer architectures $\{W_{j,1}, \ldots, W_{j,N}\}_{j=1}^{J}$ from J clients
Output: global weights $\{W_1, \ldots, W_N\}$
$n = 1$;
while $n \leq N$ **do**
  if $n < N$ **then**
    $\{\Pi_j\}_{j=1}^{J}$ = BBP-MAP$\left(\{W_{j,n}\}_{j=1}^{J}\right)$; `// call BBP-MAP to solve Eq. 2`
    $W_n = \frac{1}{J} \sum_j W_{j,n} \Pi_j^T$;
  else
    $W_n = \sum_{k=1}^{K} \sum_j p_{jk} W_{jl,n}$ where p_k is fraction of data points with label k on worker j;
  end
  for $j \in \{1, \ldots, J\}$ **do**
    $W_{j,n+1} \leftarrow \Pi_j W_{j,n+1}$; `// permutate the next-layer weights`
    Train $\{W_{j,n+1}, \ldots, W_{j,L}\}$ with W_n frozen;
  end
  $n = n + 1$;
end

EVALUATION

- FedMA가 FedAvg와 FedProx 성능을 상회



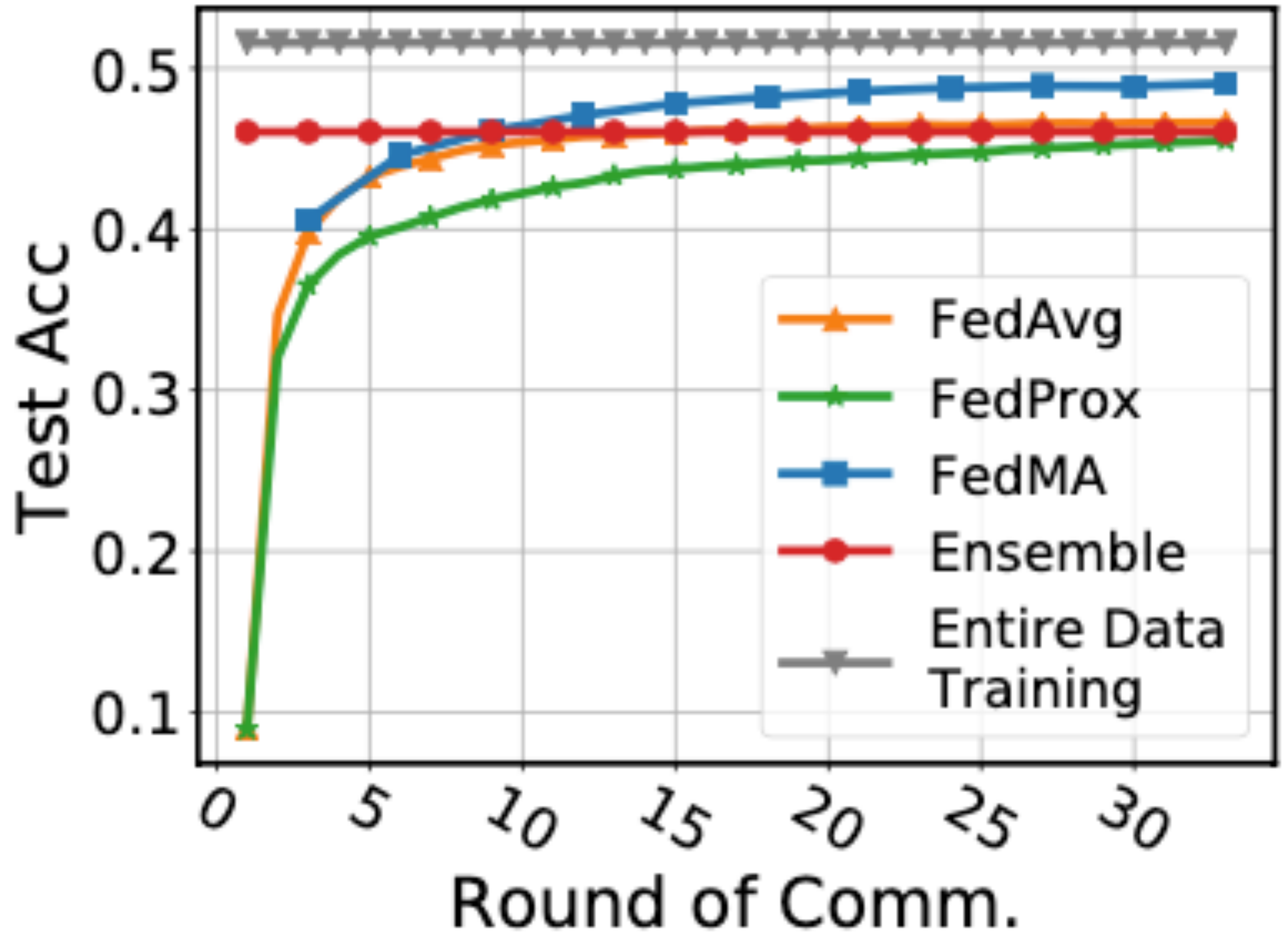
(a) LSTM, Shakespeare; message size



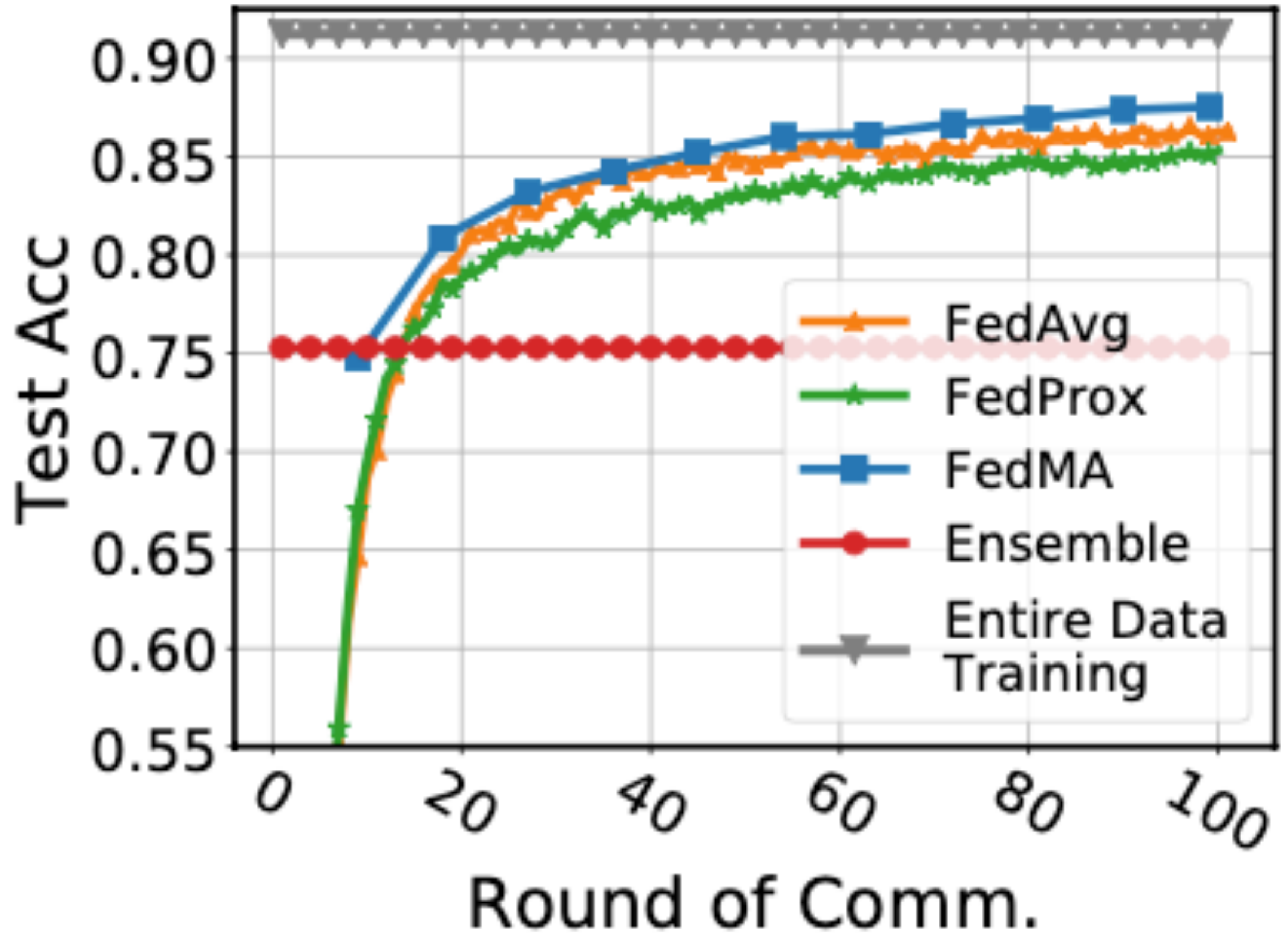
(c) VGG-9, CIFAR-10; message size

EVALUATION

- FedMA가 FedAvg와 FedProx 성능을 상회



(b) LSTM, Shakespeare; rounds



(d) VGG-9, CIFAR-10; rounds

EVALUATION

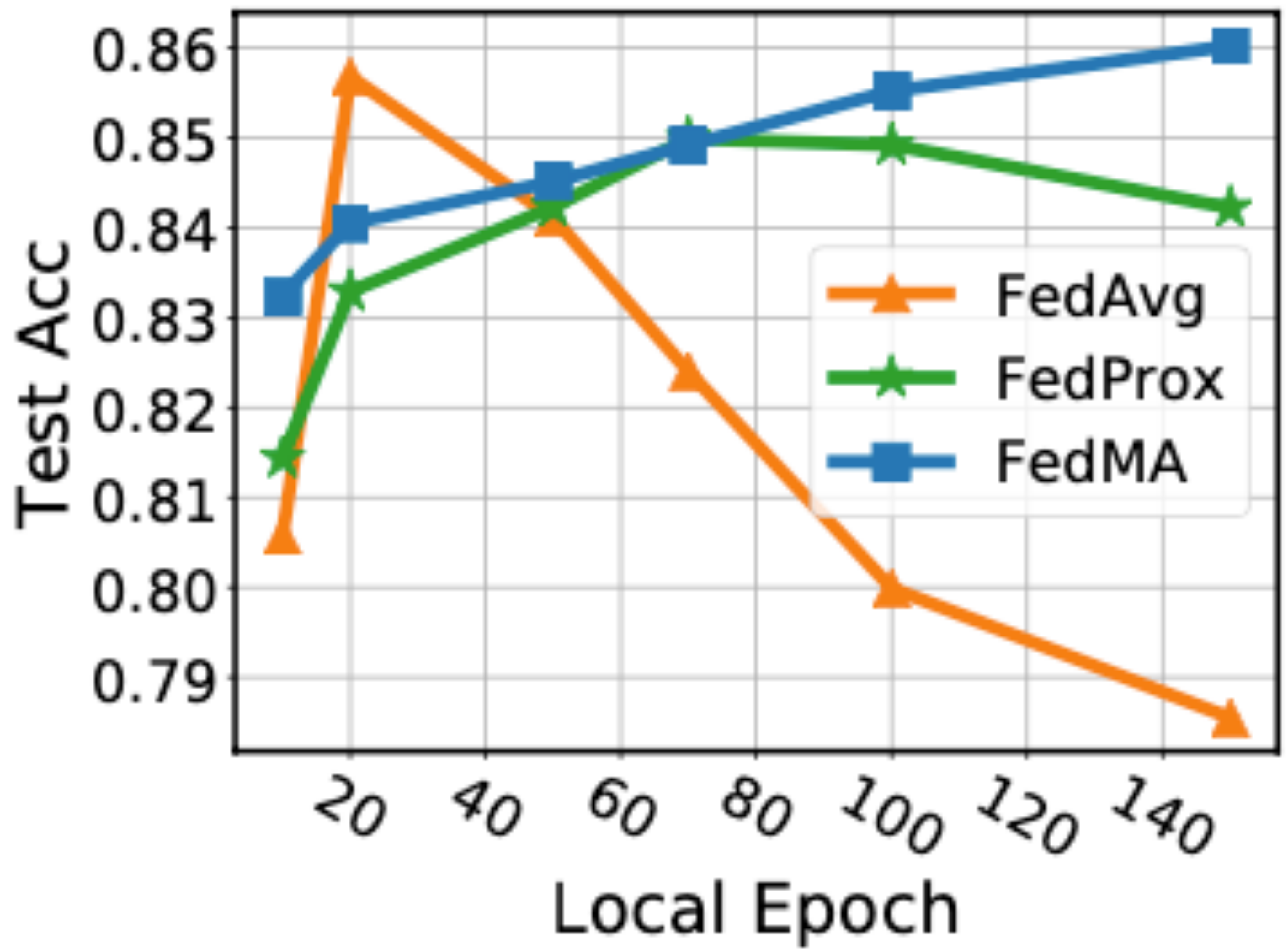
- 사전 연구들에 따르면
- 로컬(local) 학습에서의 에폭 E에 따라
    - FedAvg의 성능에 영향을 끼침은 물론
    - 때로는 발산하는 결과를 야기

EVALUATION

- 그러나 FedMA에서는
  - Local Epoch이 클 수록 유리
  - 로컬 클라이언트가 원하는 만큼 학습할 수 있는 오직 유일한 방법
- FedAvg는 성능 저하를 야기
- FedProx는 부분적으로만 문제를 경감
  - 더 큰 에폭에서는 성능 저하

EVALUATION

- FedMA는 로컬 클라이언트가 원하는 만큼 학습할 수 있는 오직 유일한 방법



BYZANTINE TOLERANT

GENERALIZED BYZANTINE-TOLERANT SGD

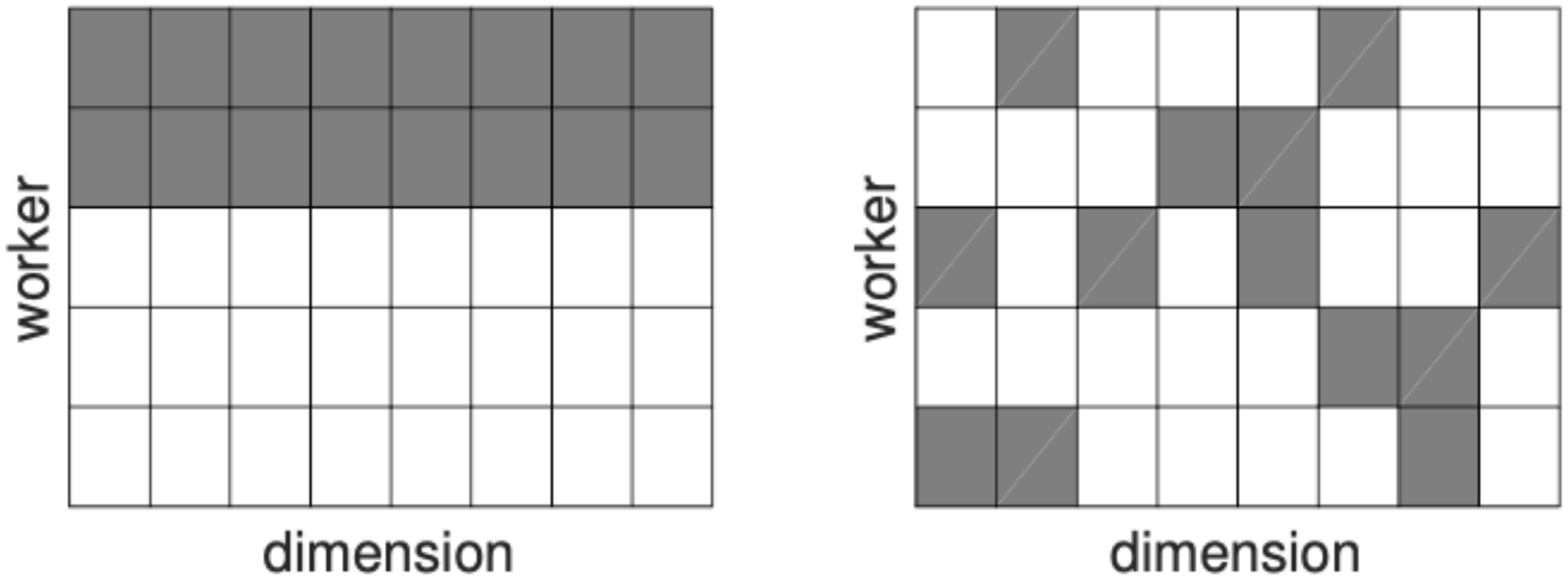
- 컨트리뷰션
  - 분산 학습에서 비잔틴 모델의 일반화
  - 일반화된 비잔틴이 있는 환경에서 강건한 통합 방법을 제안

GENERALIZED BYZANTINE

- 분산 학습에서, 여러 종류의 공격 유형이 있음
- 일반적으로 공격자는 모델 학습을 방해하고자 함
    - SGD 수렴을 느리게 만들거나
    - 나쁜 솔루션을 향하게 함

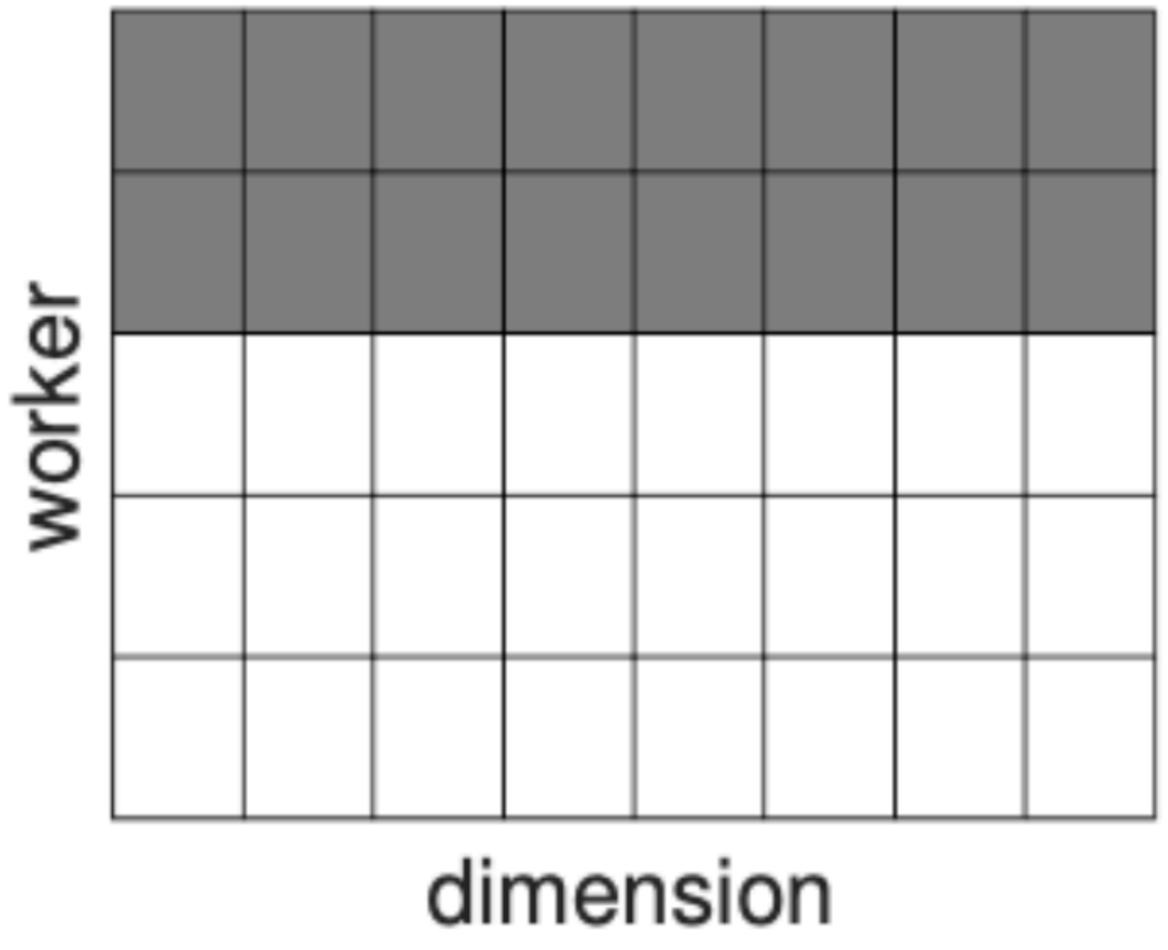
GENERALIZED BYZANTINE

- 실패 모델은 $n \times d$ 행렬로 표현 가능
    - 5명의 워커, 8차원의 그레디언트
    - (a)는 (b)의 특별한 경우에 해당



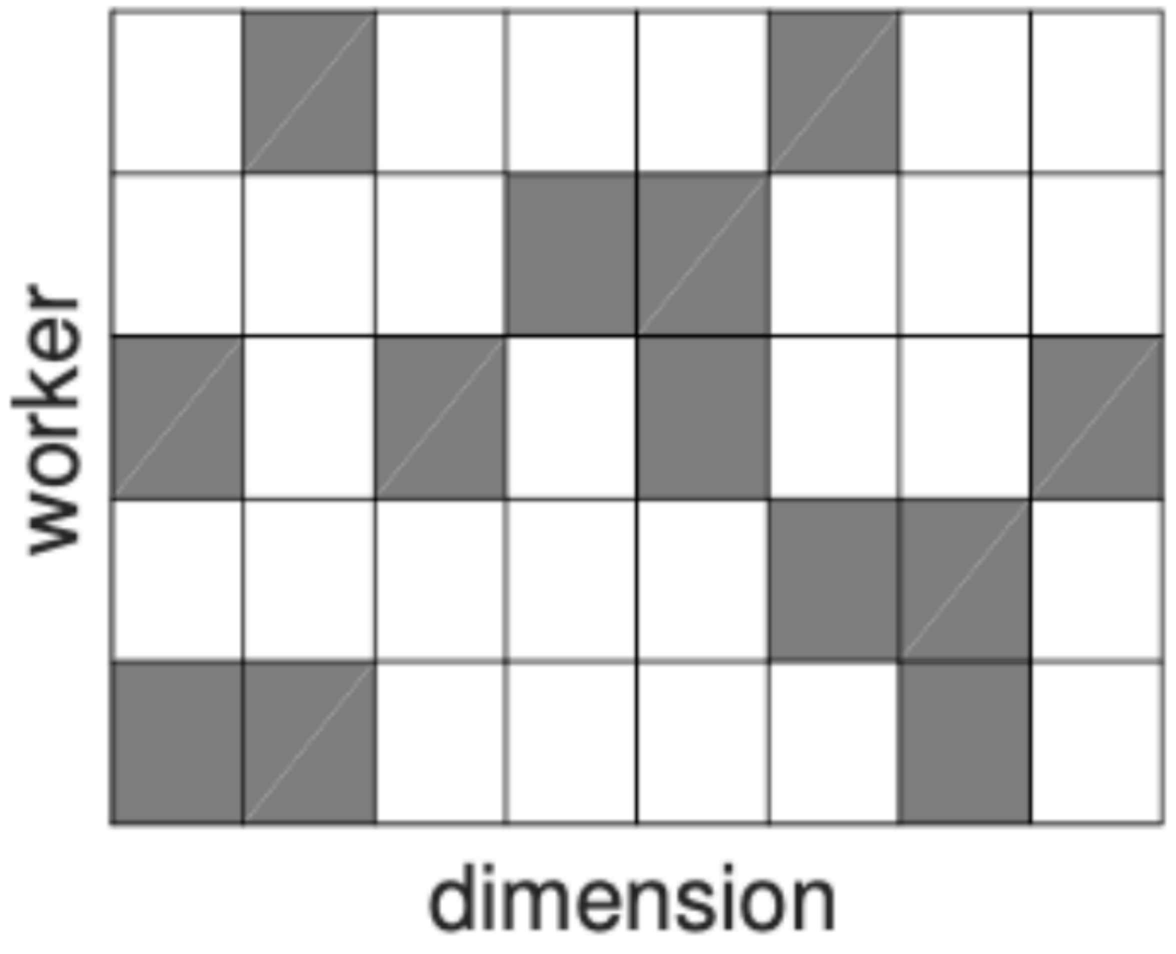
(a) Classic Byzantine (b) Generalized Byzantine

GENERALIZED BYZANTINE

- 전통적인 비잔틴 모델:
- $$\tilde{v}_i = \begin{cases} v_i, \textit{ if the ith worker is correct,} \\ arbitrary, \textit{ if the ith worker is Byzantine.} \end{cases}$$
- 

(a) Classic Byzantine

GENERALIZED BYZANTINE

- 일반화된 비잔틴 모델:
- $$(\tilde{v}_i)_j = \begin{cases} (v_i)_j, \textit{if the the } j\textit{th dimension of } v_i \textit{ is correct,} \\ arbitrary, otherwise, \end{cases}$$
- 

(b) Generalized Byzantine

GENERALIZED BYZANTINE

- 각 차원에 대해 비잔틴 값의 수는 절반 미만으로 가정
  - 흔한 가정
  - Dimensional Byzantine 탄력성이라 칭함

MEDIAN-BASED AGGREGATION

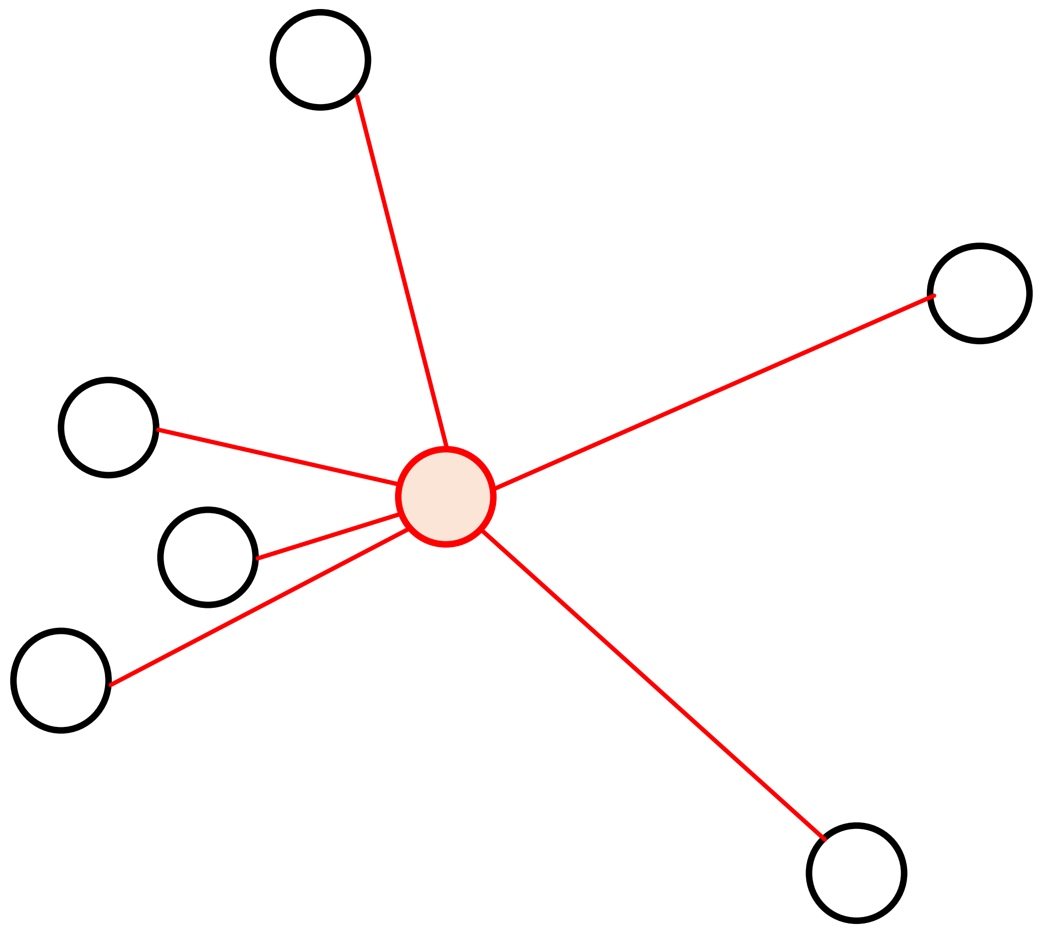
- 3개의 중앙값 기반 통합 규칙을 제안
  - Geometric Median
  - Marginal Median
  - Beyond Median

MEDIAN-BASED AGGREGATION

- Geometric Median (기하중앙값)
- 평균에 대한 강건한 추정량으로 사용
  - 최대, 데이터의 절반이 부정해도
  - 부정하지 않은 데이터에 대한 추정을 제공

MEDIAN-BASED AGGREGATION

- Geometric Median (기하중앙값)
- $$\lambda = GeoMed(\{\tilde{v}_i : i \in [n]\}) = \underset{v \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} \|v - \tilde{v}_i\|$$
  - 거리의 합의 최소



MEDIAN-BASED AGGREGATION

- Marginal Median
- $\mu = MarMed(\{\tilde{v}_i : i \in [n]\})$
- μ의 j번째 차원은 $\mu_j = median(\{(\tilde{v}_1)_j, \ldots, (\tilde{v}_n)_j\})$
    - $median(\cdot)$은 1차원 중앙값

MEDIAN-BASED AGGREGATION

- Marginal Median
- μ의 j번째 차원은 $\mu_j = median(\{(\tilde{v}_1)_j, \ldots, (\tilde{v}_n)_j\})$

worker

1	3	1	5	9	2	5
4	5	1	4	5	1	9
6	4	3	2	2	3	1
9	8	5	1	5	2	2
1	6	4	8	4	2	3

dimension

Sorting

1	3	1	1	2	1	1
1	4	1	2	4	2	2
4	5	3	4	5	2	3
6	6	4	5	5	2	5
9	8	5	8	9	3	9

MEDIAN-BASED AGGREGATION

- Beyond Median
- 비잔틴의 수 q를 쉽게 추정할 수 있다면
- 중앙값에 가까운 $n - q$ 개의 값의 평균을 활용할 수 있을 것
  - “mean around median”

MEDIAN-BASED AGGREGATION

- Beyond Median
- $\rho = MeaMed(\{\tilde{v}_i : i \in [n]\})$
- ρ의 j번째 차원은 $\rho_j = \dfrac{1}{n-q}\Sigma_{\mu_j \to i}(\tilde{v}_i)j$
- $\mu_j \to i$는 중앙값 μ_j에 가장 가까운 top-$(n-q)$ 값들

EVALUATION

- 예상대로, 평균(mean) 방법은 비잔틴 탄력성이 없음

EVALUATION

- GeoMed
  - 전통적인 비잔틴 탄력성은 있으나
  - Dimensional 비잔틴 탄력성은 없음

EVALUATION

- MarMed와 MeaMed는 Dimensional 비잔틴 탄력성이 있음
  - 그러나 Omniscient 공격에서 MarMed는 수렴이 늦음
- Omniscient
  - 모든 워커들이 전송한 그레디언트를 알고 있음
  - 그레디언트의 총합에 매우 큰 음수를 곱함
  - 목표는 SGD가 원래의 반대 방향으로 크게 이동하도록 하는 것

EVALUATION

- 동기 SGD를 위한 3가지의 중앙값 기반 통합 규칙을 제안
  - 낮은 시간 복잡도를 가짐
    - 평균적으로 $O(dn)$
  - 실제로 좋은 성능을 보임

TODO

TODO

- 네트워크의 일반화
  - 클라이언트-서버 구조가 아닌
  - P2P 등 Decentralized & Asynchronous 한 환경
- Co-op
  - Cooperative Learning
- Zeno++

TODO

- 전이학습의 응용
- FedMD
  - 모델까지 Private하게 관리
  - Consensus를 통한 모델 업데이트

FEDERATED LEARNING

SUMMARY OF
MODERN FEDERATED LEARNING METHODS