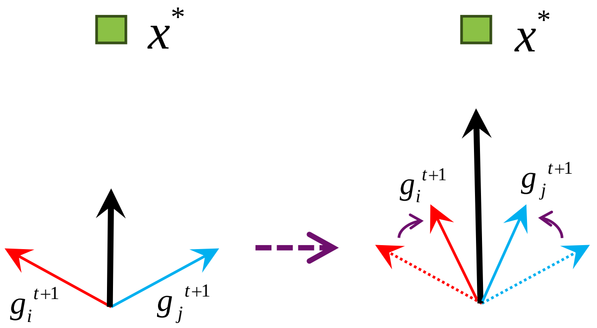
FedGH: Gradient Harmonization in Federated Learning

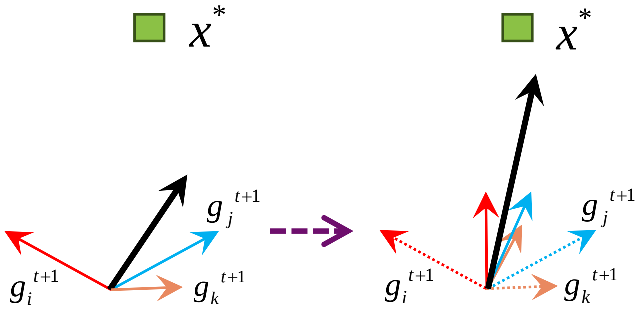
Algorithm 1: Federated Averaging with Gradient Harmonization (`FedGH`)

Input: $K, T, \eta, E, w^0, N, p_k, k = 1, ..., N$
for $T = 0, ..., T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  Server sends w^t to all chosen devices
  Each device $k \in S_t$ updates w^t for E epochs of SGD on F_k with step-size η to obtain w_k^{t+1}
  Each device $k \in S_t$ sends w_k^{t+1} back to the server
  Server recalculates gradient $g_k^{t+1}, k \in S_t$
  for $i \in S_t$ **do**
    for $j \in S_t \backslash i$ **in random order do**
      if $g_i^{t+1} \cdot g_j^{t+1} < 0$ **then**
        $g_i^{t+1} := g_i^{t+1} - \frac{g_i^{t+1} \cdot g_j^{t+1}}{\|g_j^{t+1}\|^2} g_j^{t+1}$
  Server updates w_k^{t+1} **using** $g_k^{t+1}, k \in S_t$
  Server aggregates the w's as $w^{t+1} = \frac{1}{K} \sum_{k \in S_t} w_k^{t+1}$
end

1. Faster convergence rate



2. Mitigating local drifts



3. A parameter-free **distal term**

Easily integrates with other federated learning frameworks as a plug-and-play module that does not require any parameter tuning.

Related works: non-IID federated learning

1.Mitigating local drifts during local training

Algorithm 2 FedProx (Proposed Framework)

Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
Server sends w^t to all chosen devices
Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w h_k(w; w^t) = F_k(w) + \boxed{\frac{\mu}{2}\|w - w^t\|^2}$
Each device $k \in S_t$ sends w_k^{t+1} back to the server
Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

FedProx [1]:

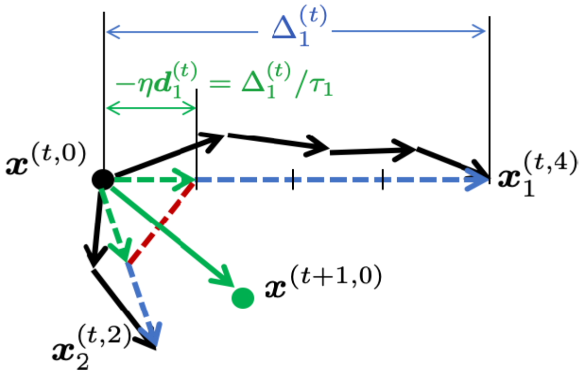
Proximal term with hyperparameter μ

FedGH(ours):

Distal term without any hyperparameters

FedProx + GH(ours)

2.Mitigating local drifts on server aggregation



Novel Generalized Update Rule

$$x^{(t+1,0)} = x^{(t,0)} - \boxed{\tau_{\text{eff}} \boxed{\sum_{i=1}^{m}} w_i} \eta d_i^{(t)}$$

Optimizes $\widetilde{F}(x) = \sum_{i=1}^{m} w_i F_i(x)$

$$x^{(t+1,0)} - x^{(t,0)} = \sum_{i=1}^{m} p_i \Delta_i^{(t)} = -\sum_{i=1}^{m} p_i \|a_i\|_1 \cdot \frac{\eta G_i^{(t)} a_i}{\|a_i\|_1}$$

$$= -\underbrace{\left(\sum_{i=1}^{m} p_i \|a_i\|_1\right)}_{\tau_{\text{eff}}:\ \text{effective local steps}} \sum_{i=1}^{m} \eta \underbrace{\left(\frac{p_i \|a_i\|_1}{\sum_{i=1}^{m} p_i \|a_i\|_1}\right)}_{w_i:\ \text{weight}} \underbrace{\left(\frac{G_i^{(t)} a_i}{\|a_i\|_1}\right)}_{d_i:\ \text{normalized gradient}}$$

We use the vanilla SGD to ensure a fair comparison with other baselines

$$x^{(t+1,0)} - x^{(t,0)} = \left(\sum_{i=1}^{m} p_i \tau_i^{(t)}\right) \sum_{i=1}^{m} \frac{p_i \Delta_i^{(t)}}{\tau_i^{(t)}}$$

FedNova [2]:

Server weighted aggregates based on the number of iterations

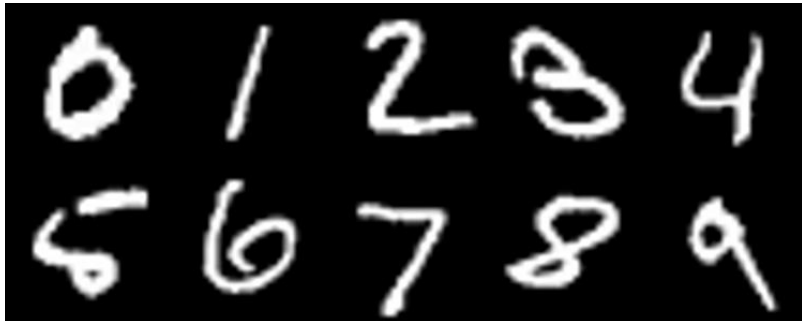
FedGH(ours):

Server aggregates after gradient deconfliction

FedNova + GH(ours)

[1] Federated Optimization in Heterogeneous Networks, MLSys, 2020
[2] Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization, NeurIPS, 2020

Experiments on MNIST



MNIST

10 categories
Training samples: 70000
Testing samples: 10000
Sample size: (28, 28)

Layer (type)	Output Shape	Param #
Linear-1	[-1, 512]	401,920
ReLU-2	[-1, 512]	0
Linear-3	[-1, 256]	131,328
ReLU-4	[-1, 256]	0
Linear-5	[-1, 10]	2,570

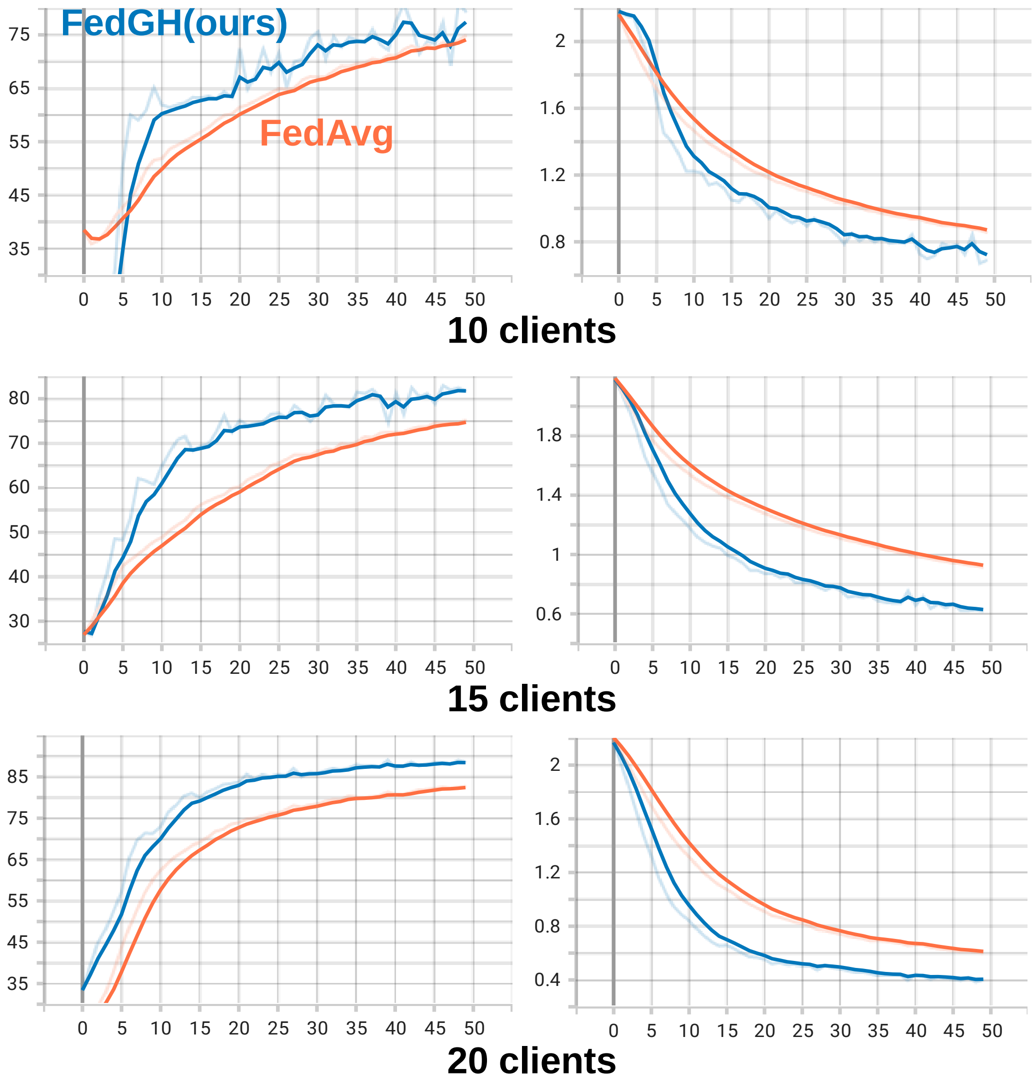
2NN Model
(Params: 0.536M, FLOPs: 0.535M)

Default hyperparameters:

E = 1	K = 20
B = 128	C = 1.0
lr = 0.01	Rounds = 50
Optimizer = vanilla SGD	α = 0.01
	μ = 0.1

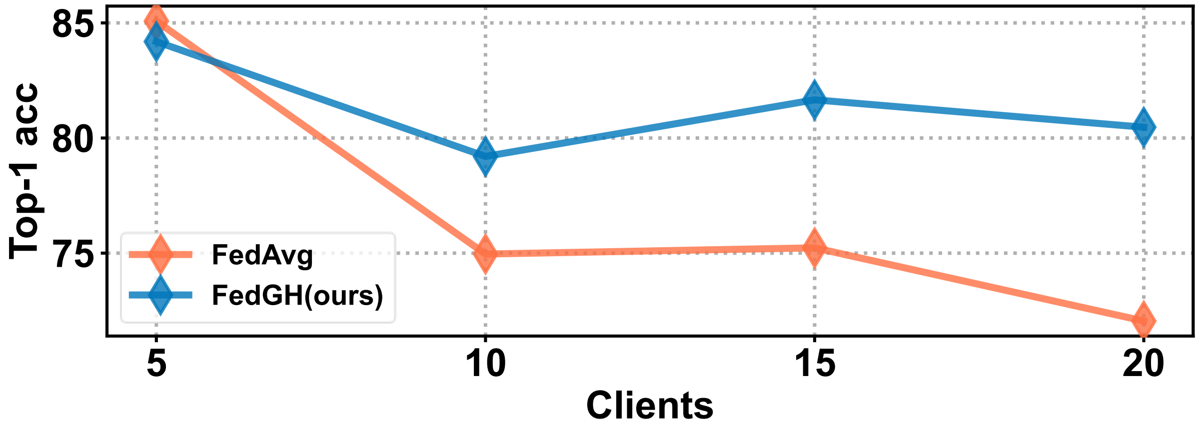
Environment: NVIDIA GeForce RTX 3090, Pytorch 2.0.0

Experiments on MNIST



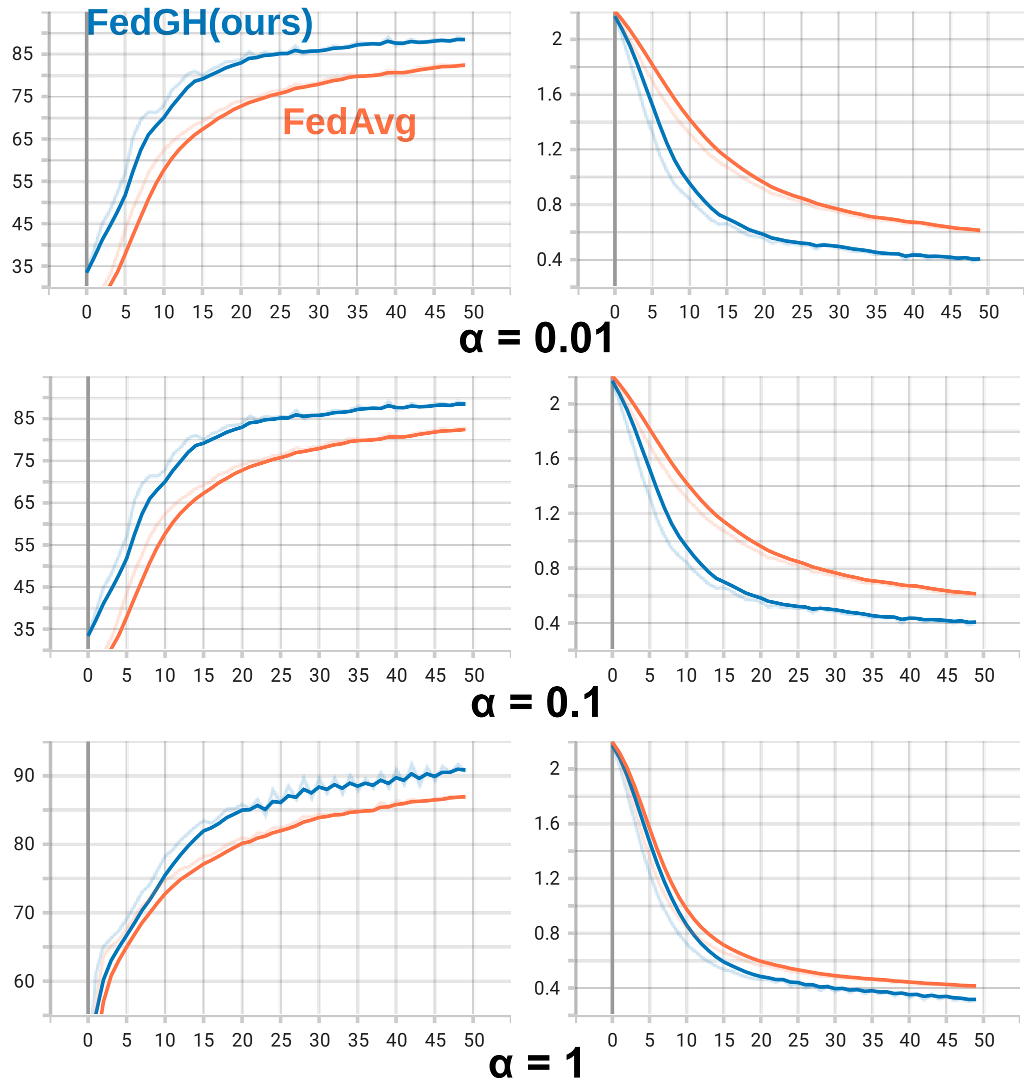
Clients	Method	Top-1 acc.	Top-3 acc.
5	FedAvg [1]	**85.08**	98.17
	FedAvg [1] + GH (ours)	84.19 (-0.89)	**98.64 (+0.47)**
10	FedAvg [1]	74.96	95.43
	FedAvg [1] + GH (ours)	**79.20 (+4.24)**	**96.31 (+0.88)**
15	FedAvg [1]	75.23	94.87
	FedAvg [1] + GH (ours)	**81.66 (+6.43)**	**96.53 (+1.66)**
20	FedAvg [1]	72.05	93.32
	FedAvg [1] + GH (ours)	**80.47 (+8.42)**	**95.65 (+2.33)**

Table 1. Effect of the number of clients on the MNIST 2NN with C = 1.0, $\alpha = 0.01$.



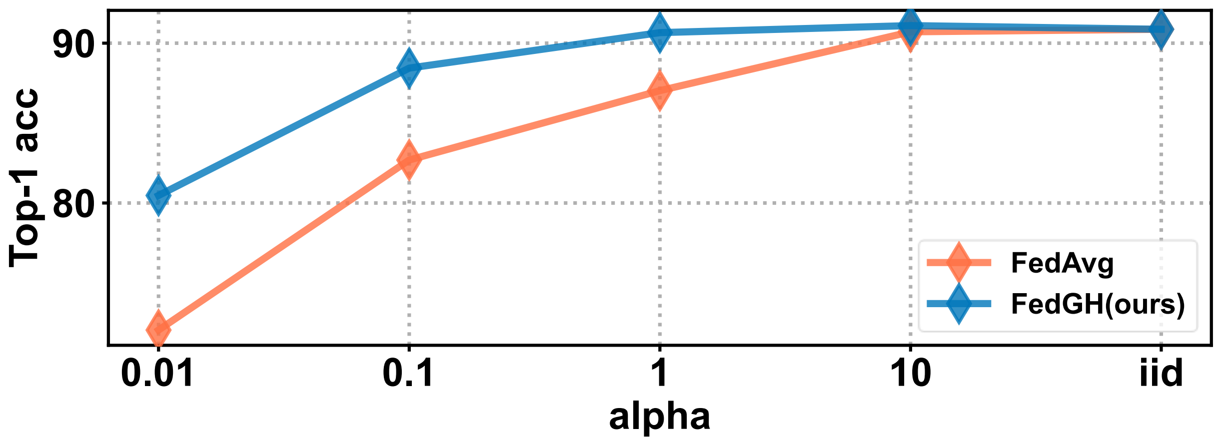
[1] Communication-Efficient Learning of Deep Networks from Decentralized Data, PMLR, 2017

Experiments on MNIST



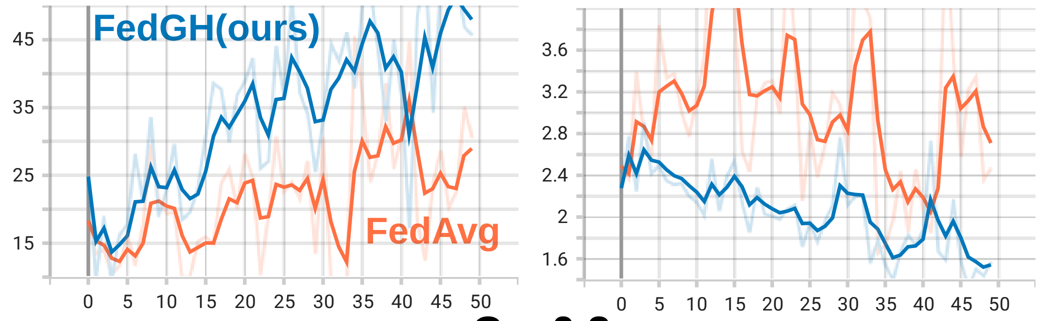
α	Method	Top-1 acc.	Top-3 acc.
0.01	FedAvg [1]	72.05	93.32
	FedAvg [1] + GH (ours)	**80.47** (+8.42)	**95.65** (+2.33)
0.1	FedAvg [1]	82.69	96.19
	FedAvg [1] + GH (ours)	**88.44** (+5.75)	**97.69** (+1.50)
1	FedAvg [1]	87.03	97.81
	FedAvg [1] + GH (ours)	**90.66** (+3.63)	**98.51** (+0.70)
10	FedAvg [1]	90.71	98.06
	FedAvg [1] + GH (ours)	**91.10** (+0.39)	**98.23** (+0.17)
∞ (iid)	FedAvg [1]	**90.87**	**98.05**
	FedAvg [1] + GH (ours)	**90.87** (+0.00)	**98.05** (+0.00)

Table 2. Effect of the heterogeneity α on the MNIST 2NN with C = 1.0, 20 clients.

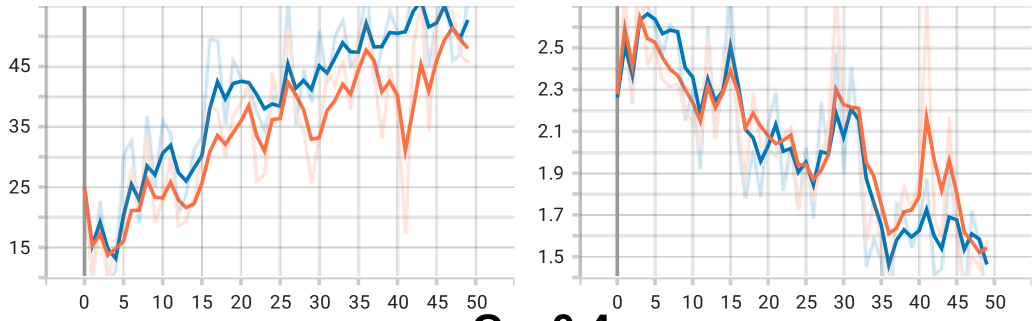


[1] Communication-Efficient Learning of Deep Networks from Decentralized Data, PMLR, 2017

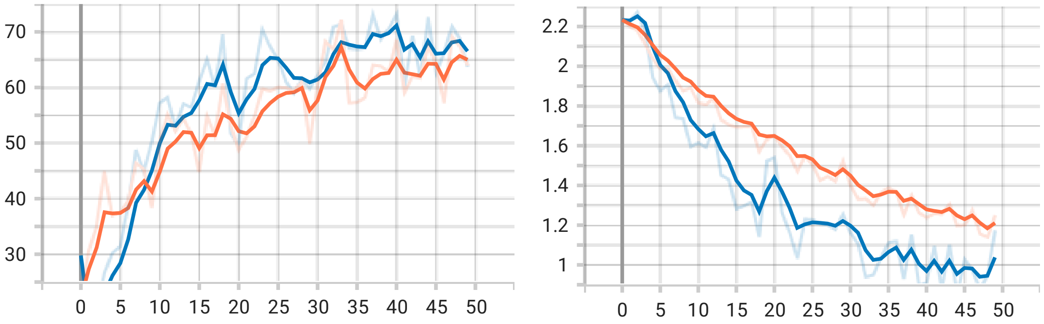
Experiments on MNIST



C = 0.2



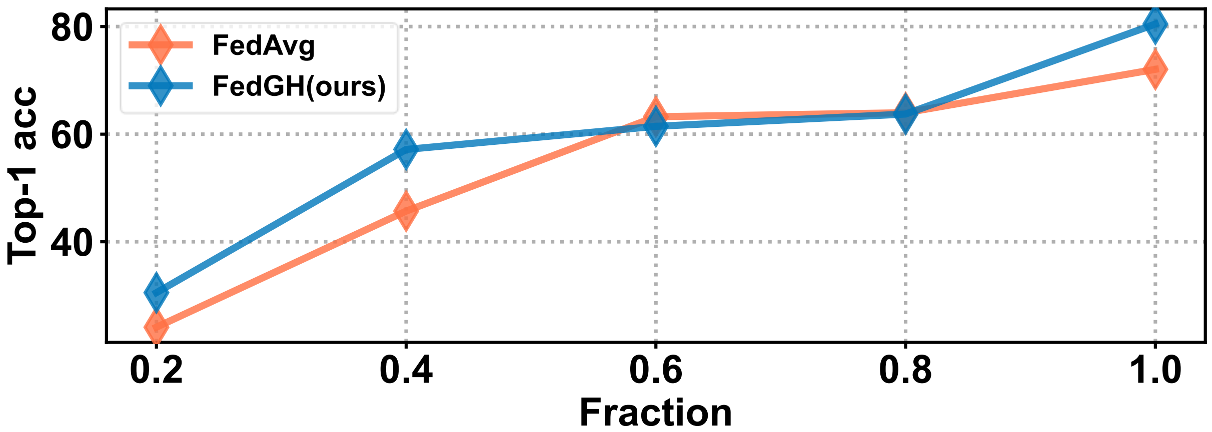
C = 0.4



C = 0.8

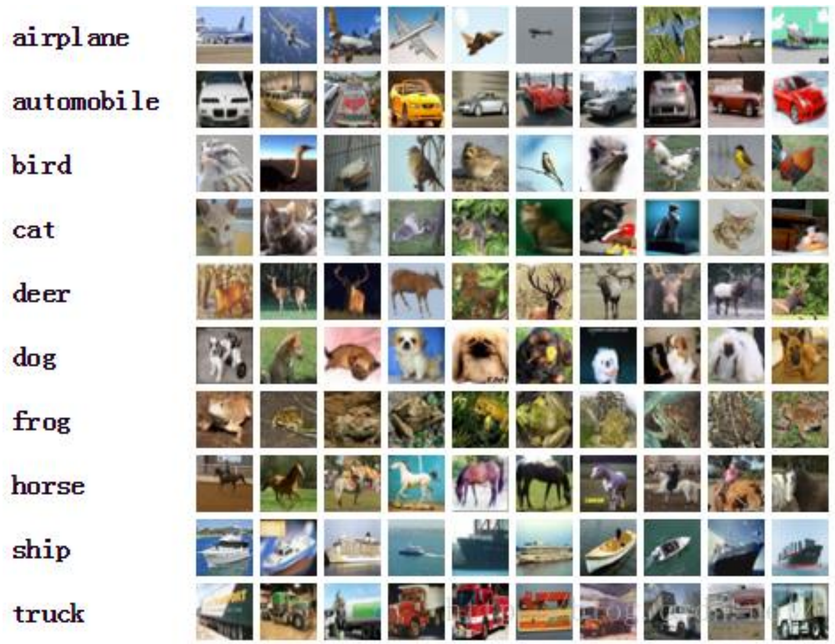
Fraction	Method	Top-1 acc.	Top-3 acc.
0.2	FedAvg [1]	24.12	62.83
	FedAvg [1] + GH (ours)	**30.54** (+6.42)	**67.96** (+5.13)
0.4	FedAvg [1]	45.70	69.70
	FedAvg [1] + GH (ours)	**57.16** (+11.46)	**81.87** (+12.17)
0.6	FedAvg [1]	**63.24**	**81.53**
	FedAvg [1] + GH (ours)	61.45 (-1.79)	81.20 (-0.33)
0.8	FedAvg [1]	**63.98**	77.55
	FedAvg [1] + GH (ours)	63.74 (-0.39)	**78.92** (+0.17)
1.0	FedAvg [1]	72.05	93.32
	FedAvg [1] + GH (ours)	**80.47** (+8.42)	**95.65** (+2.33)

Table 3. Effect of the client fraction C on the MNIST 2NN with $\alpha = 0.01$, 20 clients.



[1] Communication-Efficient Learning of Deep Networks from Decentralized Data, PMLR, 2017

Experiments on CIFAR-10



CIFAR-10

10 categories
Training samples: 50000
Testing samples: 10000
Sample size: (32, 32, 3)

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 28, 28]	2,432
ReLU-2	[-1, 32, 28, 28]	0
Conv2d-3	[-1, 64, 24, 24]	51,264
ReLU-4	[-1, 64, 24, 24]	0
MaxPool2d-5	[-1, 64, 12, 12]	0
Dropout-6	[-1, 64, 12, 12]	0
Linear-7	[-1, 128]	1,179,776
ReLU-8	[-1, 128]	0
Dropout-9	[-1, 128]	0
Linear-10	[-1, 10]	1,290
ReLU-11	[-1, 10]	0

CNN Model
(Params: 1.235M, FLOPs: 32.554M)

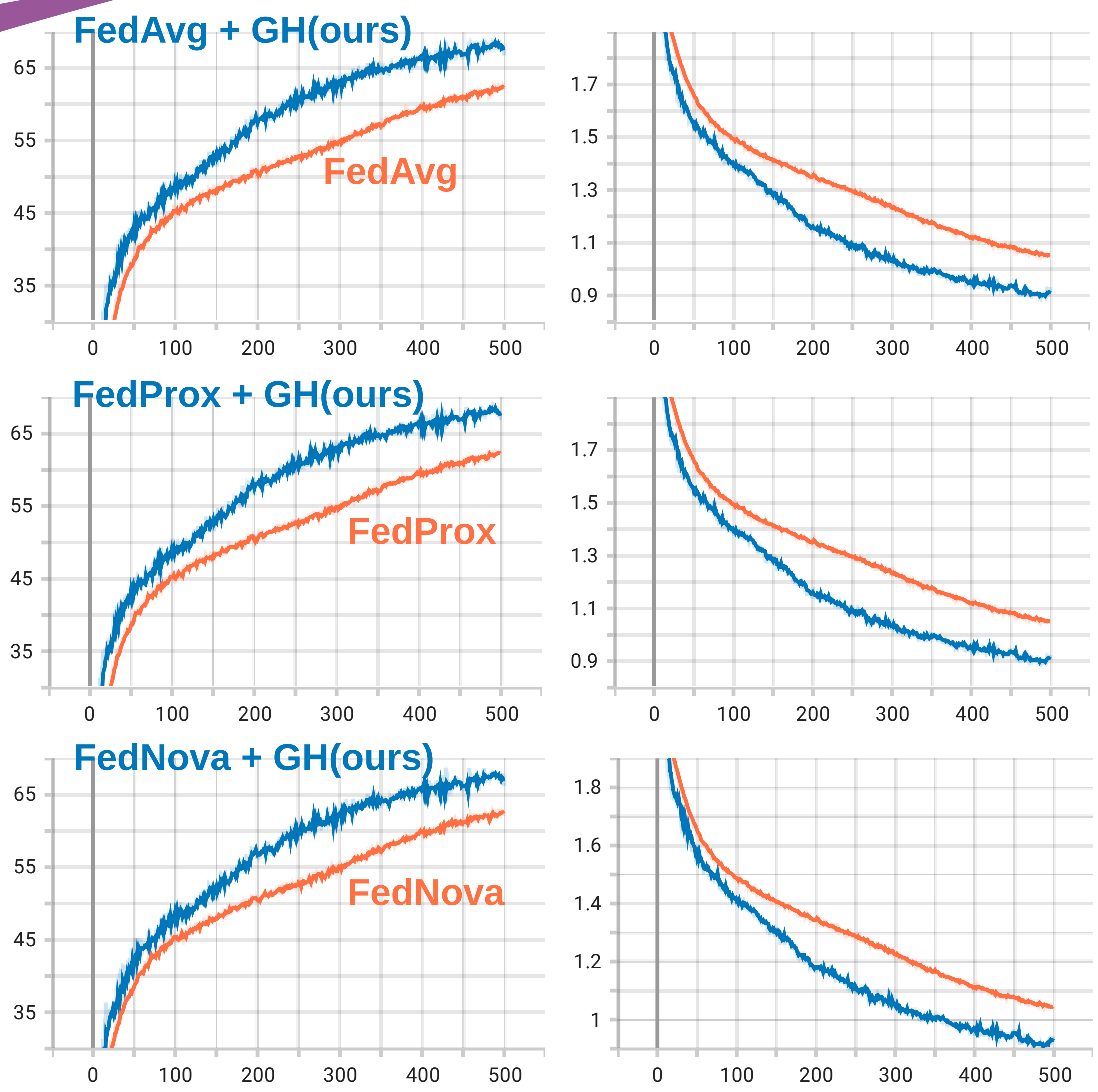
Default hyperparameters:

E = 1
B = 128
lr = 0.01
Optimizer = vanilla SGD

K = 20
C = 1.0
Rounds = 500
α = 0.5
μ = 0.1

Environment: NVIDIA GeForce RTX 3090, Pytorch 2.0.0

Experiments on CIFAR-10



Rounds	Method	Top-1 acc.	Top-3 acc.
100	FedAvg [1]	45.31	80.11
	FedAvg [1] + GH (ours)	**47.78** (+2.47)	**81.85** (+1.74)
200	FedAvg [1]	50.65	83.48
	FedAvg [1] + GH (ours)	**57.04** (+6.39)	**87.85** (+4.37)
500	FedAvg [1]	62.09	90.29
	FedAvg [1] + GH (ours)	**66.75** (+4.66)	**91.88** (+1.59)
100	FedProx [2]	45.29	80.11
	FedProx [2] + GH (ours)	**47.69** (+2.40)	**81.86** (+1.75)
200	FedProx [2]	50.67	83.52
	FedProx [2] + GH (ours)	**57.03** (+6.36)	**87.92** (+4.40)
500	FedProx [2]	62.13	90.30
	FedProx [2] + GH (ours)	**66.91** (+4.78)	**91.97** (+1.67)
100	FedNova [3]	45.44	80.21
	FedNova [3] + GH (ours)	**47.16** (+1.72)	**81.47** (+1.26)
200	FedNova [3]	50.44	83.57
	FedNova [3] + GH (ours)	**56.21** (+5.77)	**87.38** (+3.81)
500	FedNova [3]	62.14	90.22
	FedNova [3] + GH (ours)	**66.16** (+4.02)	**91.59** (+1.37)

Table 4. Results on the CIFAR-10 CNN with $\alpha = 0.5$, C = 1.0, 20 clients.

[1] Communication-Efficient Learning of Deep Networks from Decentralized Data, PMLR, 2017
[2] Federated Optimization in Heterogeneous Networks, MLSys, 2020
[3] Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization, NeurIPS, 2020

Experiments on Tiny-ImageNet

Tiny-ImageNet

200 categories

Training samples: 100000

Testing samples: 10000

Sample size: (64, 64, 3)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×8
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23	[1×1, 256 3×3, 256 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet18

(Params: 11.27M, FLOPs: 148.951M)

Default hyperparameters:

E = 5

B = 64

lr = 0.01

Optimizer = vanilla SGD

K = 20

C = 1.0

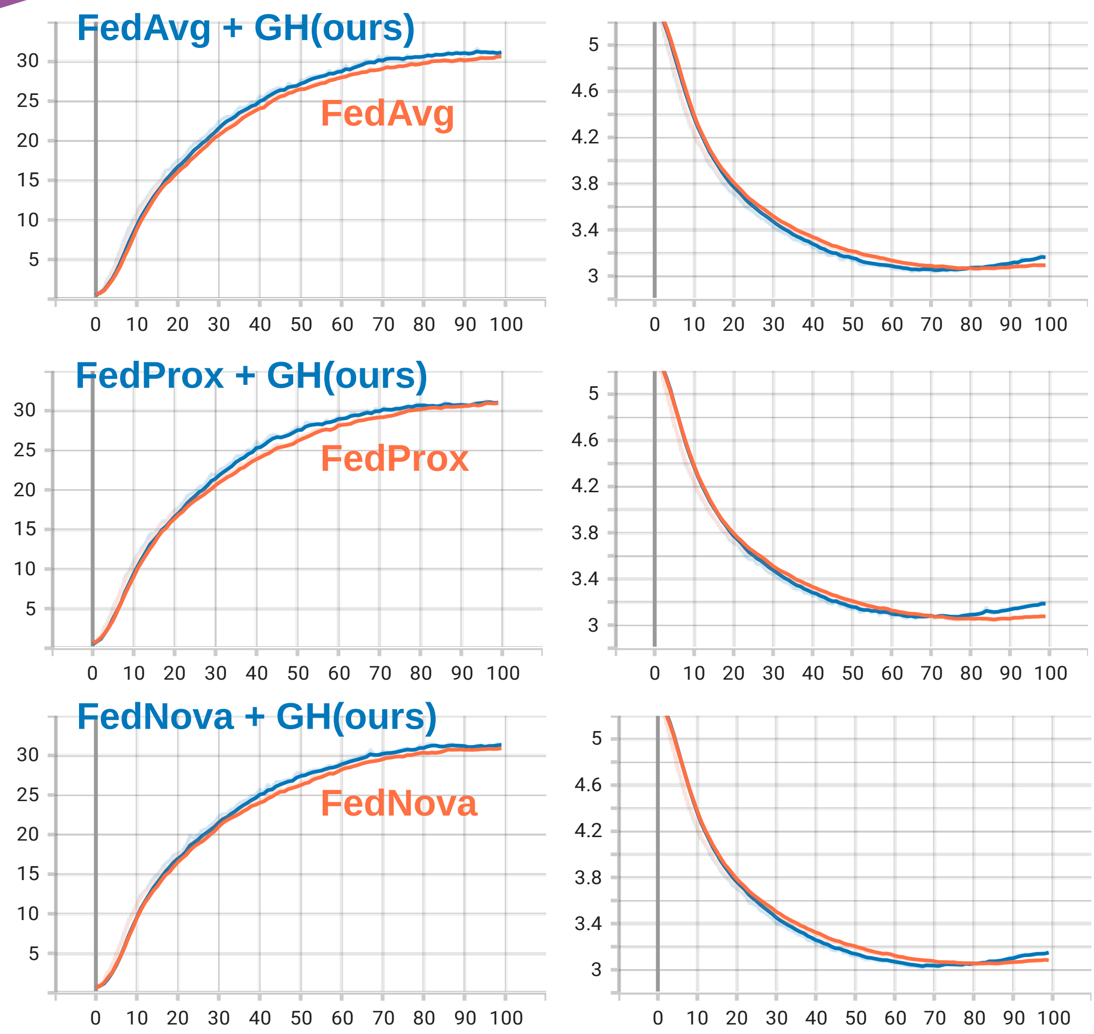
Rounds = 100

α = 0.1

μ = 0.1

Environment: NVIDIA GeForce RTX 3090, Pytorch 2.0.0

Experiments on Tiny-ImageNet



Rounds	Method	Top-1 acc.	Top-3 acc.
70	FedAvg [1]	29.05	47.08
	FedAvg [1] + GH (ours)	**30.63** (**+1.58**)	**48.58** (**+1.50**)
100	FedAvg [1]	30.62	49.21
	FedAvg [1] + GH (ours)	**31.30** (**+0.68**)	**49.30** (**+0.09**)
70	FedProx [2]	29.20	47.24
	FedProx [2] + GH (ours)	**30.26** (**+1.06**)	**47.93** (**+0.69**)
100	FedProx [2]	31.02	49.09
	FedProx [2] + GH (ours)	**31.06** (**+0.04**)	**49.71** (**+0.62**)
70	FedNova [3]	29.50	47.61
	FedNova [3] + GH (ours)	**30.07** (**+0.57**)	**48.74** (**+1.13**)
100	FedNova [3]	31.01	49.29
	FedNova [3] + GH (ours)	**31.41** (**+0.40**)	**50.04** (**+0.75**)

Table 5. Results on the Tiny-ImageNet ResNet18 [4] with $\alpha = 0.1$, C = 1.0, 20 clients.

[4] Deep residual learning for image recognition, CVPR, 2016
[1] Communication-Efficient Learning of Deep Networks from Decentralized Data, PMLR, 2017
[2] Federated Optimization in Heterogeneous Networks, MLSys, 2020
[3] Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization, NeurIPS, 2020

Limitations

01 Performance

FedGH only works if the gradient conflict (non-iid) occurs. It is powerless in the small number of clients or homogeneous data scenarios. Empirically, **FedGH** brings limited performance gains on over-parameterized models.

02 Methodology

FedGH can be viewed as an adaptive global learning rate scheduler based on clients' consensus. However, a larger learning rate may lead to training oscillations and a faster transition from underfitting to overfitting. Thus, Introducing EMA [1] or plugging **FedGH** with other existing methods will perform better.

03 Theory

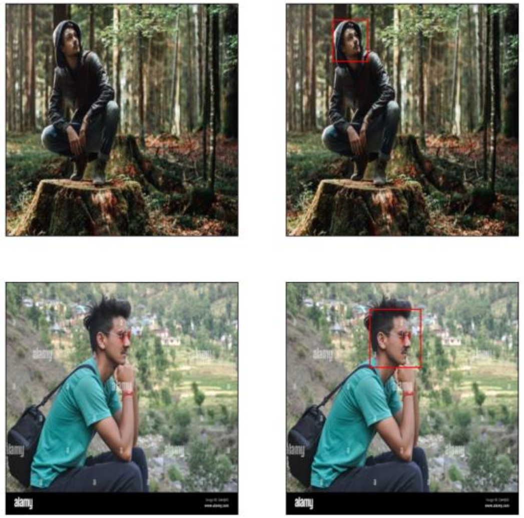
FedGH is derived from intuitive ideas and observations of toy experiments. Convergence guarantee and principled understanding should be done further.

04 Experiment

Further validation of **FedGH** on more challenging tasks (semantic segmentation, language modeling, etc.) and more SOTA models is also needed.

[1] Generalized Federated Learning via Sharpness Aware Minimization, ICML, 2022

Experiments on YOLOv8n



Human Faces
(Object Detection)

Training samples: 1983

Testing samples: 221

Sample size: (640, 640, 3)

```
                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    751507  ultralytics.nn.modules.head.Detect           [1, [64, 128, 256]]
YOLOv8n summary: 225 layers, 3011043 parameters, 3011027 gradients
```

YOLOv8n
(225 layers, 3011043 parameters, 3011027 gradients)

Default hyperparameters:

num_clients = 4

rounds = 30

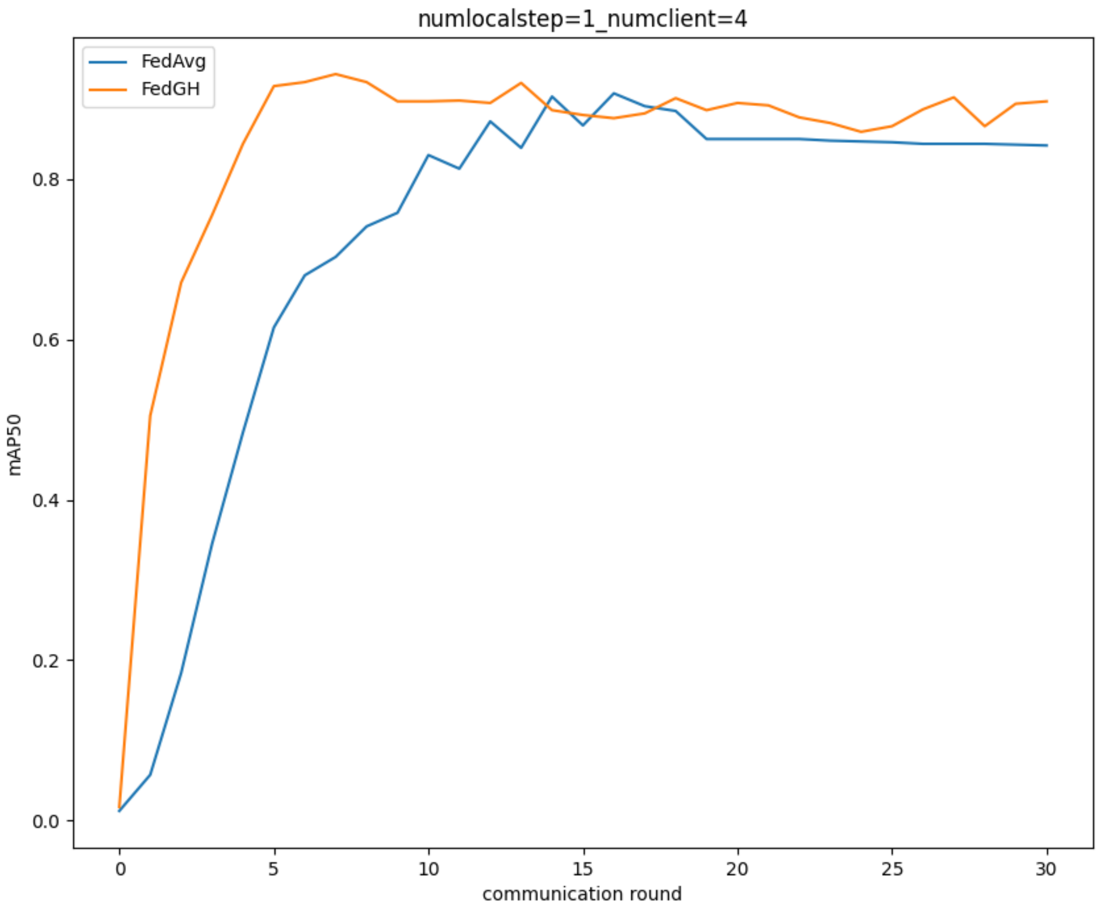
batch_size = 32

local_steps = 1.0

learning_rate = 0.0000001

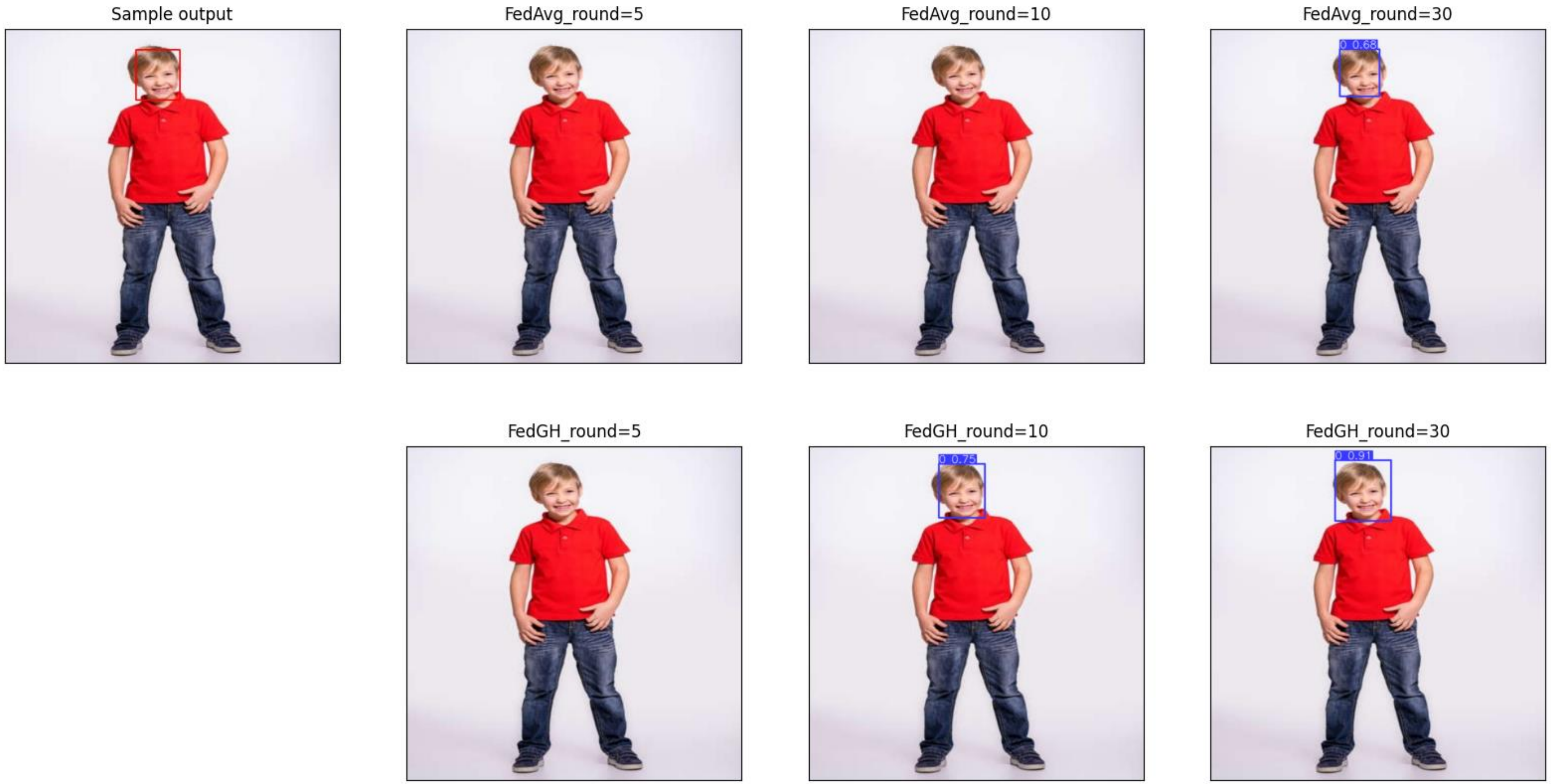
Environment: NVIDIA GeForce RTX 3090, Pytorch 2.0.1

Experiments on YOLOv8n



Round	Method	mAP50
5	FedAvg	0.615
5	FedAvg + GH(ours)	0.916 (+0.301)
10	FedAvg	0.83
10	FedAvg + GH(ours)	0.897 (+0.067)
30	FedAvg	0.842
30	FedAvg + GH(ours)	0.897 (+0.055)

Experiments on YOLOv8n



Experiments on YOLOv8n