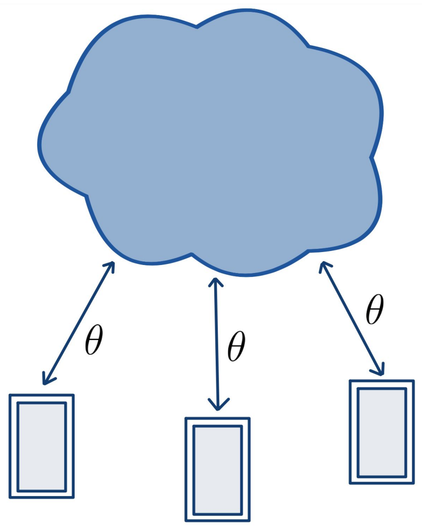
Politecnico di Torino

UNDERSTANDING THE MAIN CHALLENGES OF FEDERATED LEARNING

Giuseppe Galilei S295620
Pietro Cagnasso S300801
Nicolò Vergaro S295633

MLDL
A.A 2021/2022

Basic FL Problem



Main Challenges

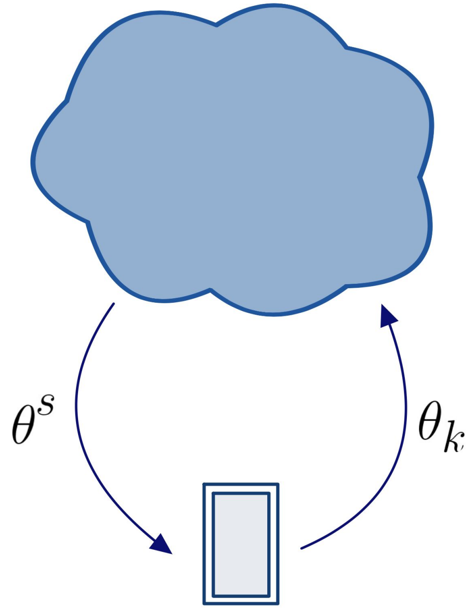
- Statistical heterogeneity
- System heterogeneity
- Privacy preservation

Related Works

- FedAVG
- FedGKT - Systems heterogeneity
- FedDyn - Statistical heterogeneity
- Gradient inversion attack - Privacy preservation

Models - Federated Averaging [1]

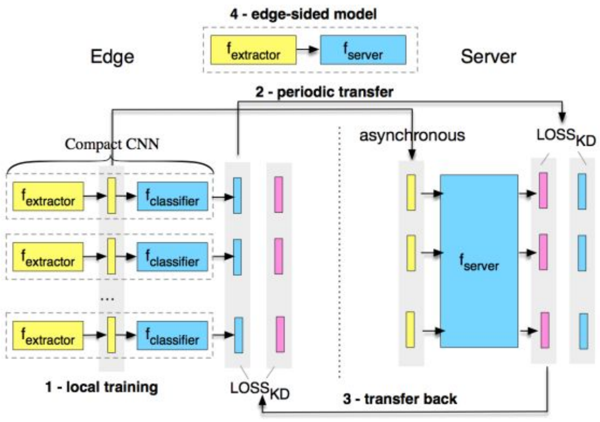
1. Each active client receives the global model
2. Each trains on its own data with multiple SGD steps
3. Updates are sent to the server
4. The server aggregates all the updates with a weighted average
5. The average is sent back to the active clients



[1] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics, 2017.

Models - Group Knowledge Transfer [2]

- Addresses systems heterogeneity
- Train small CNNs on resource-constrained edge devices
  - This CNN consists of a feature extractor and a classifier
- Train large CNNs on more powerful servers
  - This CNN lacks the first feature extraction layer.
- Advantages compared to FedAVG:
  - reduced demand for edge computation
  - lower communication bandwidth
  - asynchronous training



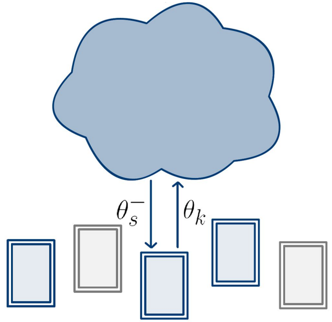
[2] Chaoyang He et al. Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge, Jul. 2020

Models - Dynamic Regularization [3]

- Addresses statistical heterogeneity and communication optimization
- Uses internal states
- Modifies the client's loss function so the local optimization is coherent with the global one
- Adds two terms to the standard loss:
  - Linear penalty term, debiases the effect of local losses
  - Quadratic penalty term, ensures convergence in the long run

$$h = h^- - \alpha \frac{1}{m}\Big(\sum_{k \in \mathcal{P}} \theta_k - \theta_s^-\Big)$$

$$\theta_s = \Big(\frac{1}{|\mathcal{P}|}\sum_{k \in \mathcal{P}} \theta_k\Big) - \frac{1}{\alpha} h$$



$$\theta_k = \text{argmin}_\theta \; L_k(\theta) - \langle \nabla L_k(\theta_k^-),\; \theta \rangle + \frac{\alpha}{2}\|\theta - \theta_s^-\|^2$$

$$\nabla L_k(\theta_k) = \nabla L_k(\theta_k^-) - \alpha(\theta_k - \theta_s^-)$$

[3] Durmus Alp Emre Acar et al. Federated Learning Based on Dynamic Regularization, Sep. 2020

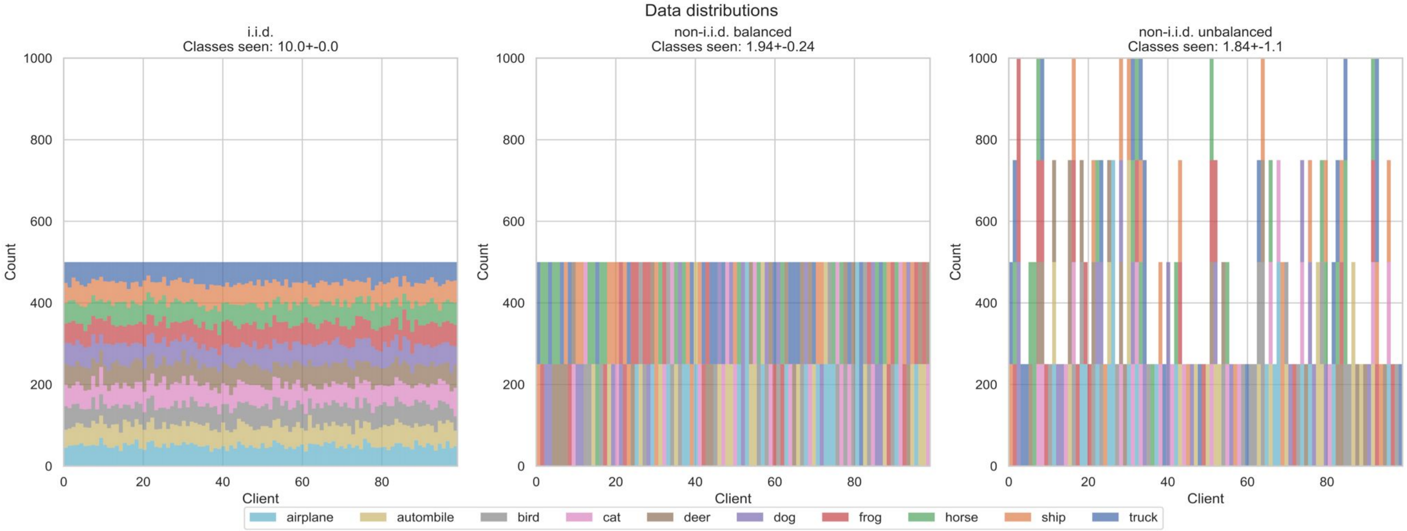
Results - Experiment Settings

1. Run the experiments for **50 epochs/rounds**
2. Use **three seeds** (0, 128, 479) and average the results
3. Use **CIFAR10** as dataset
4. Run all **three typical cases** of the federated scenario
5. Federated experiments with **100 available clients, 10% active**
6. Train a **ResNet50** [4] from scratch with **Batch Normalization** [5] or **Group Normalization** [6] layers.

[4] Kaiming He et al. Deep Residual Learning for Image Recognition, Dec. 2015
[5] Sergey Ioffe et al. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Feb. 2015
[6] Yuxin Wu et al. Group Normalization, Mar. 2018

Results - Sampling Strategy

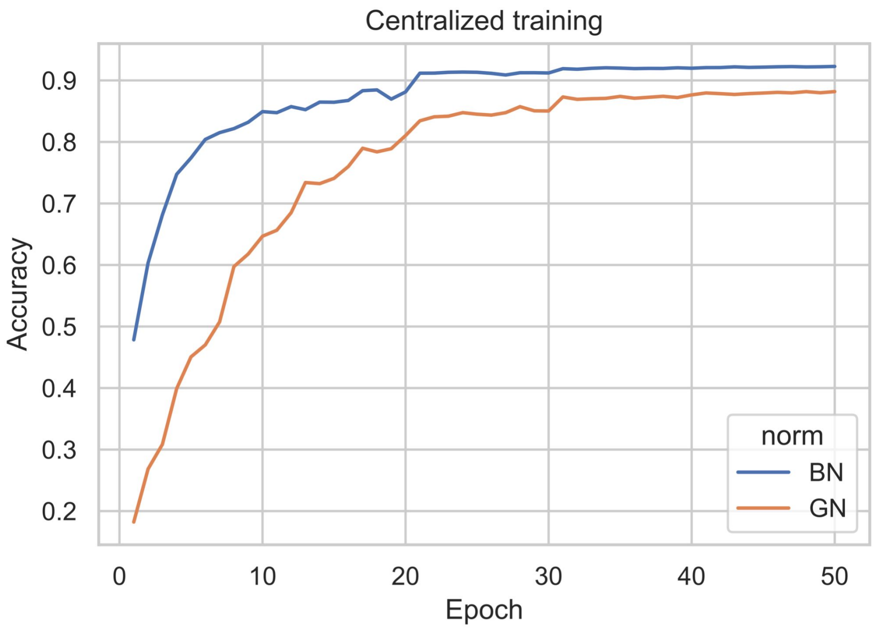
- Proposed by McMahan et al. [1]
- i.i.d. : **shuffle** the dataset and split into equal parts
- Non-i.i.d. : **sort** the dataset based on its label and **split into shards** of equal size
    - **Balanced:** same number of shards to all clients
    - **Unbalanced:** random number of shards to each client (minimum one)



Results - Centralized Scenario

- Upper bound for federated scenario
- Hyperparameters tuning, fixing:
    - Batch size: 128
    - Optimizer: SGD

Normalization	Accuracy
BN	92.28%
GN	88.18%

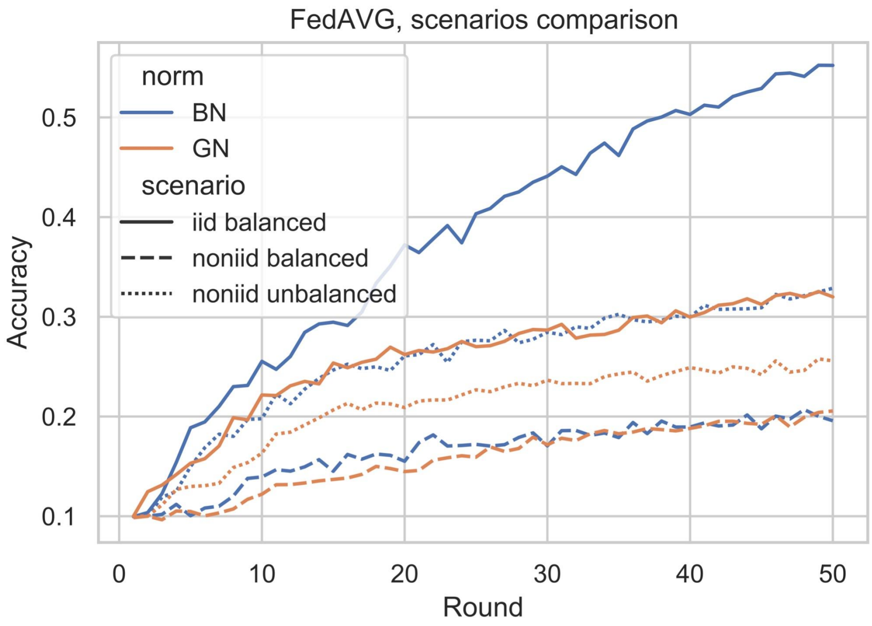


lr=1e-2, weight_decay=1e-5, momentum=0.9
Multi-step scheduler at 20,30,40 x0.33

Results - Federated Baseline (FedAVG)

- One local epoch in each client
- Different hyperparameters
- Results:
  - In line with the paper
  - BN always better than GN
  - Non-i.i.d. unbalanced over balanced

Normalization	i.i.d	non-i.i.d balanced	non-i.i.d unbalanced
BN	55.217%	19.577%	32.867%
GN	32.000%	20.557%	25.587%

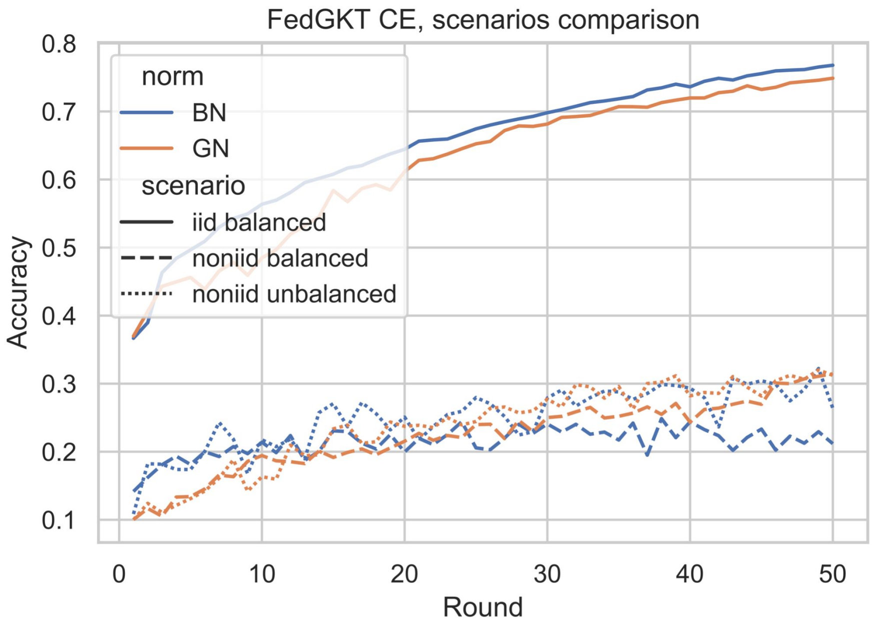


lr=1e-2, weight_decay=0, momentum=0.5

Results - System Heterogeneity (FedGKT)

- Server: Resnet49, 10 epochs
- Client: Resnet8, 1 local epoch
- Two different losses scenarios: CE, CE+KD
- Results:
  - Up to 40% improvement over baseline in i.i.d.
  - Non-i.i.d. similar to the baseline

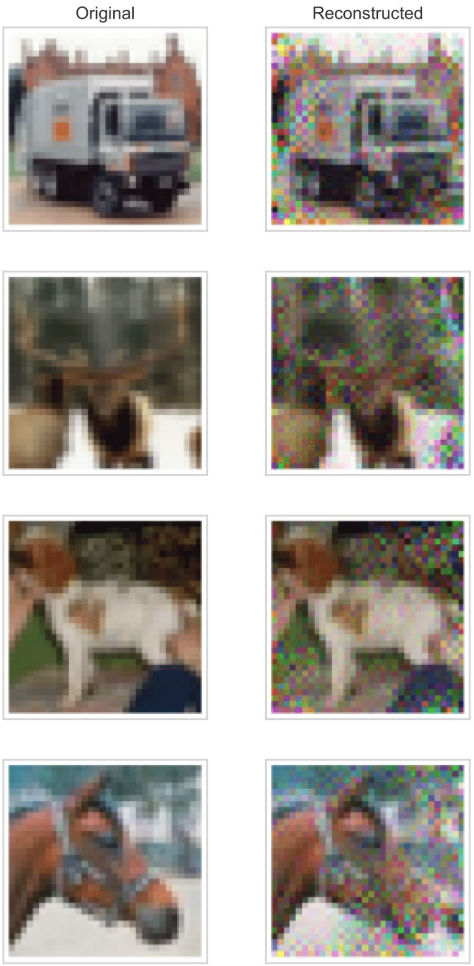
Loss	Normalization	i.i.d	non-i.i.d balanced	non-i.i.d unbalanced
CE	BN	76.789%	21.180%	26.392%
CE	GN	74.883%	31.455%	31.224%
CE+KD	BN	51.950%	28.283%	26.351%
CE+KD	GN	51.053%	23.650%	26.824%



lr=1e-2, weight_decay=0, momentum=0.5
α=0.5

Results - Gradient Inversion Attack [7]

- Honest but curious server that knows:
  - BN statistics
  - target labels
- Gradients from a pre-trained ResNet50
- Results, in 3000 iterations:
  - recognizable subjects

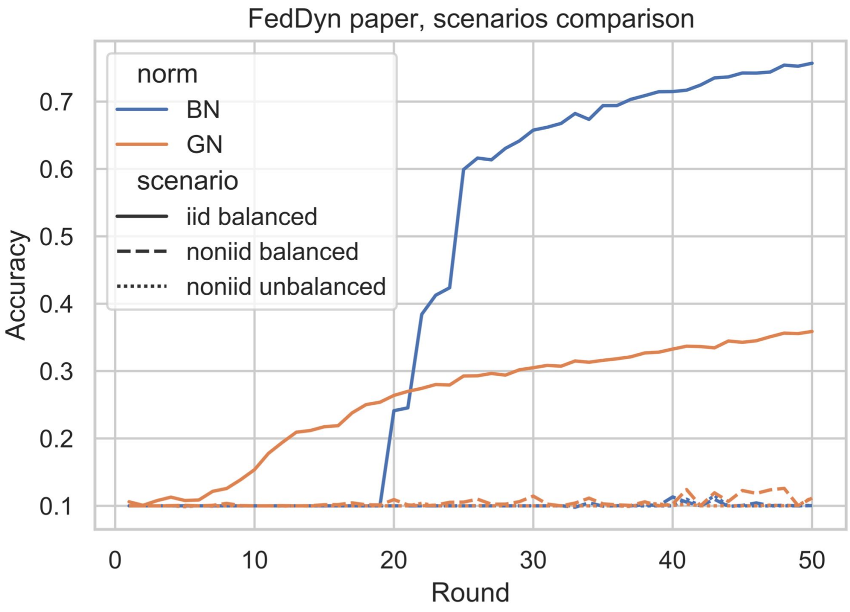


[7] Jonas Geiping et al. Inverting Gradients - How easy is it to break privacy in federated learning?, Mar. 2020

Results - FedDyn

- Five local epochs in each client
- ResNet50 with paper's hyperparameters
- Results:
  - BN produces NaN loss values
  - GN has no NaN loss values, but a worse accuracy
  - Non-i.i.d. flat

Normalization	i.i.d	non-i.i.d balanced	non-i.i.d unbalanced
BN	75.693%	10.033%	10.080%
GN	35.880%	11.157%	11.227%

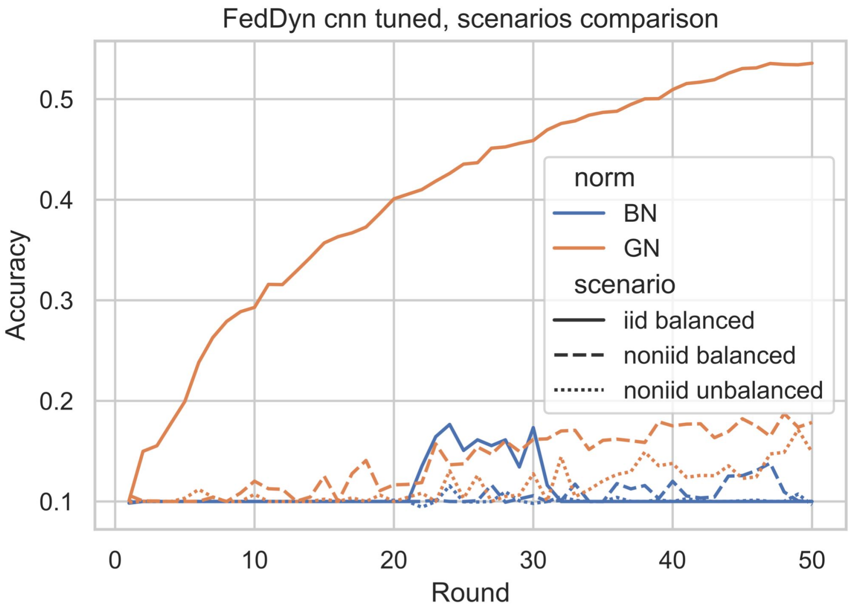


α=0.01, lr=0.1, weight_decay=1e-3, momentum=0

Results - FedDyn (tuned hyperparameters)

- Tuning on model used in the paper
- Five local epochs in each client
- Results:
    - BN worse than before
    - GN improves both in i.i.d. and non-i.i.d.

Normalization	i.i.d	non-i.i.d balanced	non-i.i.d unbalanced
BN	10.000%	10.001%	9.663%
GN	53.570%	17.860%	14.897%

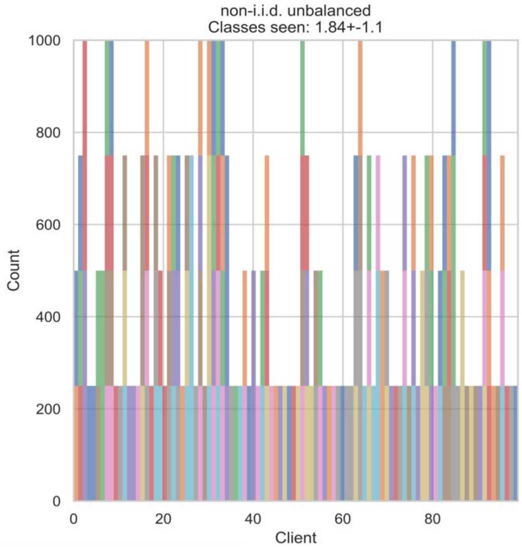


α=0.1, lr=1e-2, weight_decay=5e-3, momentum=0.9

Consideration 1/5

Why non-i.i.d. unbalanced results are better than balanced in FedAVG?

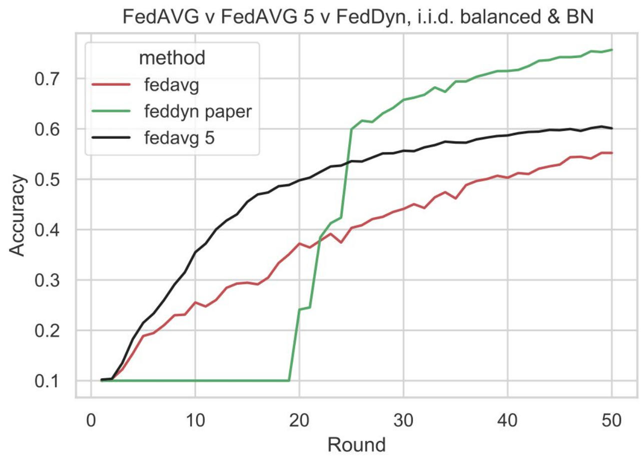
- Result already highlighted in the original paper
- Explained by the dataset splitting strategy:
    - In the unbalanced setting, clients with more shards see more classes.
    - These clients can, on average, generalize better
    - FedAVG aggregates results by a weighted average, so the contribution of these clients is rewarded more



Consideration 2/5

Letting FedAVG locally optimize more: closer to FedDyn?

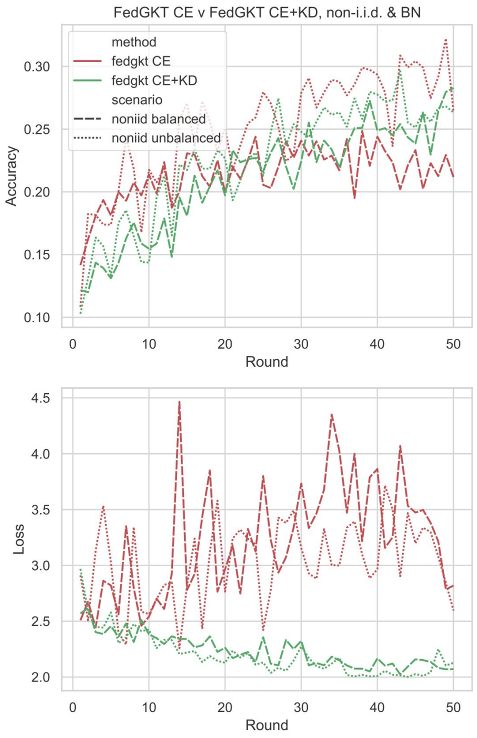
- More local epochs could:
  - speed up learning in i.i.d
  - risk overfitting in non-i.i.d
- We experimented:
  - i.i.d. case
  - Batch Normalization
  - 5 local epochs
- 4.883% increase compared to the results with one local epoch
- Still more than 10% below FedDyn



Consideration 3/5

Which loss should we use in FedGKT?

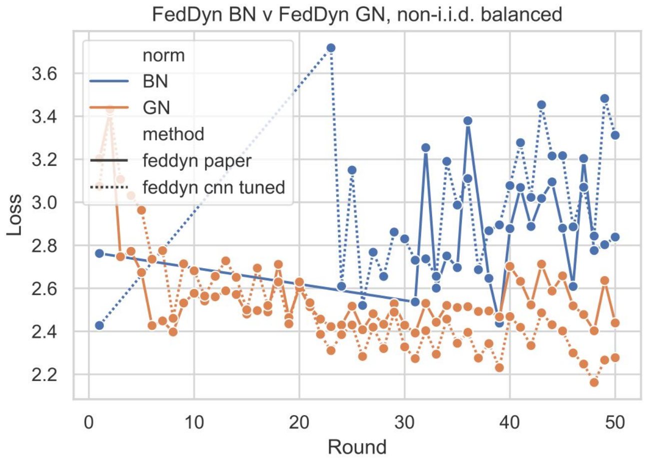
- Authors found that, in the server:
    - CE loss only performed better on smaller datasets, such as CIFAR-10
    - CE+KD loss performed better on more difficult datasets, such as CIFAR-100.
- We found that:
    - CE loss: accuracy improvements but losses are much less "smooth" than CE+KD
    - Group Norm outperforms Batch Norm only in the non-i.i.d. balanced case with CE loss
    - CE+KD loss outperforms CE loss only in the non-i.i.d balanced case with Batch Norm



Consideration 4/5

Is there a Group Normalization advantage in FedDyn?

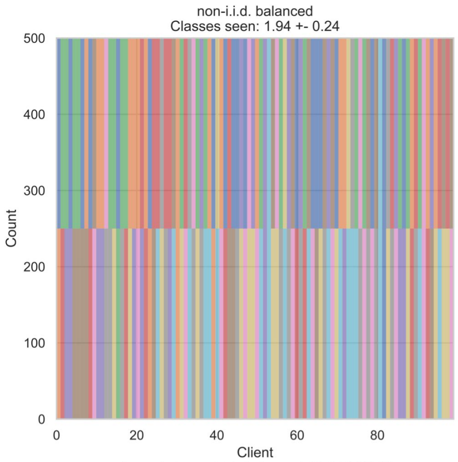
- ResNets with Batch Norm layers seem to be unstable and result in many loss equal to NaN
- ResNets with Group Norm layers seem much more stable and did not result in any loss equal to NaN
- Using Group Norm layers allegedly allows performing tuning on simpler networks that translate on the ResNets themselves

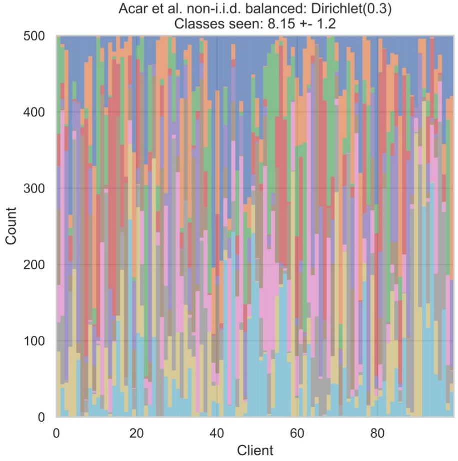


Consideration 5/5

FedDyn: why non-i.i.d. cases systematically perform worse than results reported in the paper?

- We **could not** reproduce the original paper's results
- We tested the worst non-i.i.d. case sampling used in the original paper: Dirichlet(0.3).
- Each client sees on average 8.15 classes, 4 times more classes than each of our clients sees in the worst non-i.i.d case
- Clients in the original paper can generalize much more efficiently





Conclusions

- FedDyn vs FedGKT:
  - They obtain comparable results in the i.i.d case. Further analysis would be interesting.
- Standard sampling
  - An easier sampling leads to better results. We think the community should settle on a standard sampling.
- With great Resnets comes great responsibility
  - Is it possible to tune hyperarameters on small networks and use them on bigger networks with optimal performances? We could save time and resources.
- Optimistic gradient inversion attack
  - Knowing BN statistics and the original labels is not always feasible.
  - Newly proposed algorithms should consider introducing security measures or perform analysis of possible threats (e.g FedDyn doesn’t)

Thanks for your attention