Systematic Generation and Mitigation of Distribution Shifts in Federated Learning

Ryan Mehenni[1], Lucca Guerin Saturnino[1], and Gabriel Marival[2]

[1]ETH Zürich (ETHZ)
[2]École Polytechnique Fédérale de Lausanne (EPFL)

***Abstract*—Federated Learning (FL) enables decentralized model training while preserving data privacy, but struggles with non-IID data across clients. This paper studies two common distribution shifts—feature and label—and evaluates methods to address them. We simulate feature shift using writer-based data splits and Gaussian noise, and apply MADE to reweight client losses. Label shift is induced via Dirichlet sampling, with MOON used to align local and global representations. Experiments show that MADE improves performance under feature shift, while MOON yields limited gains under label skew, highlighting the need for tailored FL strategies.**

I. Introduction

Federated Learning (FL) is a decentralized paradigm where multiple clients collaboratively train a shared model without exchanging their raw data. This setup is particularly well-suited for privacy-sensitive or distributed environments, but it introduces new challenges, most notably data heterogeneity. In practice, client data distributions often diverge due to various forms of non-independent and identically distributed (non-IID) characteristics. Understanding and addressing this deviation is critical for achieving robust and generalizable performance in federated settings.

Several works [1] base their definition of distribution shift on Bayesian theory; let $P_i(x, y)$ and $P_j(x, y)$ denote the data distributions of two distinct clients i and j.
When $P_i(x, y) \neq P_j(x, y)$, the statistical discrepancy through Bayes' theorem yields:

$$P(x, y) = P(y \mid x)P(x) = P(x \mid y)P(y)$$

From this decomposition, two common types of distribution shifts emerge in federated learning:

1) **Feature Distribution Shift**: The marginal distribution $P(x)$ differs across clients, while the conditional $P(y \mid x)$ remains consistent. For example, in a digit classification task, clients may write the same numbers differently, resulting in visual variation across datasets.
2) **Label Distribution Shift**: The marginal distribution $P(y)$ varies, but $P(x \mid y)$ is consistent. This could occur if certain clients are more likely to generate specific labels—for instance, some users may write more 0s and 1s than other digits.

These distribution shifts pose challenges for standard federated averaging (FedAvg), which assumes IID data across clients.

This project aims to explore how feature distribution shift and label distribution shift can be systematically generated in experimental settings, and to survey the range of methods developed to mitigate their impact. By understanding how to simulate these scenarios and evaluating available strategies to overcome them, we seek to provide insights into designing more robust and adaptive federated learning systems capable of handling real-world heterogeneity.

II. Models and Methods

A. Baseline Federated Learning Algorithm

The accepted federated training process baseline [1] follows the standard FedAvg algorithm, which averages with equal weight the result of each K client SGD steps for a total of T rounds, as outlined in the pseudocode A-A.

B. Baseline Model

We employ a simple fully connected neural network, `SimpleNN`, as the base model. The architecture consists of two layers: an input layer that flattens the 28×28 image into a 784-dimensional vector, followed by a fully connected layer with 200 ReLU-activated hidden units, and an output layer with 10 units corresponding to class logits.

C. Dataset

We use the EMNIST-Digits dataset [2] for our experiments , a digit classification benchmark derived from the Extended MNIST (EMNIST) dataset. EMNIST-Digits contains 280,000 grayscale images of handwritten digits (0–9), each of size 28×28 pixels. The dataset is split into 240,000 training samples and 40,000 test samples, with class labels evenly distributed. Compared to the original MNIST dataset, EMNIST offers a larger and more diverse set of handwriting samples collected from a broader pool of writers, making it well-suited for simulating real-world federated learning scenarios.

D. Feature Distribution Shift

To model the feature distribution shift, we leverage the natural shift of the EMNIST dataset. Since each image is associated with a specific writer and each writer has a unique handwriting, assigning different writers to different clients generates *a real-world feature imbalance* as described in [3]. We do this by dividing the writers into disjoint groups of size $\frac{N_{writers}}{N_{clients}}$,where $N_{writers}$ is the total number of writers and

$N_{clients}$ is the number of clients. We then distribute a different group to each client. To accentuate the feature distribution shift we also decided to add Gaussian noise to generate *noise-based feature imbalance* which is also described in [3]. We do this by adding noise $e_i \sim \mathcal{N}(0, \sigma \frac{i}{N_{clients}})$ to each image of the party P_i, we then clip the values of each pixel to be in $[0, 1]$.

In order to improve the results with feature distribution shift, we decide to implement two mask autoencoder for distribution estimation (MADE).

a) local MADE ($q_i(x)$): Each client train a MADE from scratch on it's own local data to generate a local log-density estimator $u_l(x) = \log q_i(x)$.

b) global MADE ($p(x)$): which is trained similarly to FedAvg: broadcast the current global MADE weights, take a few local gradient steps, send updates back, and re-aggregate weighting the results with $\frac{N_i}{N}$, where N_i is the sample size of client i and N is the total sample size.

Then instead of directly computing the weights as $\alpha_i(x) = \frac{p(x)}{q_i(x)}$. We instead train a classifier $h(u)$ to predict if a MADE feature-vector u comes from the global or local model, so $h(u) \simeq P(global|u)$. We then estimates the weights as:

$$\alpha_i(x) \simeq \frac{h(u)}{1 - h(u)}$$

Finally each client minimizes the re-weighted objective within the FedAvg framework

$$\min_{\theta} \sum_{(x,y) \in D_i} \alpha_i(x) \mathcal{L}(f_\theta(x), y)$$

[4].

E. Label Distribution Shift

To model Label Distribution Shift, we use the Dirichlet distribution, a multivariate generalization of the Beta distribution, which is well-suited for generating probability vectors over K classes. Specifically, for n clients and K classes, we draw client-specific class proportions $\boldsymbol{\pi}^{(c)} \sim \text{Dir}(\alpha)$, where $\alpha \in \mathbb{R}^K_{>0}$ is the concentration parameter. For a symmetric Dirichlet with $\alpha_k = \beta$ for all k, the sampled $\boldsymbol{\pi}^{(c)}$ determines what fraction of class-k data is assigned to client c.

This approach allows fine control over the degree of non-IID-ness through the hyperparameter β. When $\beta \ll 1$, the Dirichlet distribution tends to produce sparse vectors, meaning each client will concentrate on a small subset of classes, simulating strong label shift. As $\beta \to \infty$, the distributions become more uniform, approximating the IID case. This makes the Dirichlet-based partitioning both flexible and statistically principled to simulate varying heterogeneity conditions.

In order to mitigate the issue of label distribution shift, we implemented MOON (Model-Contrastive Federated Learning, [5]),an algorithm which builds on top of the standard Federated Averaging algorithm, by adding a constrastive regularizer. The idea behind MOON lies in the fact that a model trained on a whole dataset often learns better representation than a model trained on a skewed dataset. We should therefore try control this model drift and bridge the gap between the representations learned by the local model and the global model.

Concretely, at each client, we aim to minimize :

$$\mathcal{L}_{MOON} = \mathcal{L}_{sup}(w_i^t; (x, y)) + \mu \mathcal{L}_{cont}(w_i^t, w_i^{t-1}, w^t, x)$$

where w^t is the current global model weight w_i^t represents the weights associated to client i, at the current step t, w_i^{t-1} the weights associated to client at the previous step $t-1$, $\mathcal{L}_{sup}$ is the usual supervised loss $\mathcal{L}_{cont}$ is the contrastive loss, and μ is a hyper- parameter to control the strength of regularization.

This additional loss encourages the local model to stay closer to the global model and further from the previous-round model.

More precisely, let us describe the expression of this constrastive loss. We denote by $F_w()$ the whole network and $R_w()$ to denote the network before the output layer (i.e., $R_w(x)$ is the mapped representation vector of input x). For every input x, the goal is to reduce the distance from $z = R_{w^t}(x)$ and $z_{glob} = R_{w^t}$ and increase the distance between $z = R_{w^t}(x)$ and $z_{prev} = R_{w_i^{t-1}}$. We define the model-constrastive loss as :

$$\mathcal{L}_{cont} = -\log \left(\frac{\exp\left(sim(z, z_{glob})/\tau\right)}{\exp\left(sim(z, z_{glob})/\tau\right) + \exp\left(sim(z, z_{prev})/\tau\right)} \right)$$

where $sim(.,.)$ is the cosine similarity measure.

III. Results

We conduct experiments on the MNIST-Digits dataset [2]. Unless mentioned otherwise, we set the number of communication rounds T = 5, the number of updates K = 10 and the number of clients to 5.

A. Baseline results with FedAvg

Two baseline configurations were considered:

1) **FedAvg with a single client:** All data is held by one client, effectively turning the setup into a centralized learning scenario.
2) **FedAvg with five clients (IID):** The dataset is randomly partitioned in an IID manner among five clients, with each client receiving an equal share (i.e., the entire dataset is split in half and then evenly distributed).

In both configurations, the resulting accuracy falls within the range of 84% to 85%.

B. Feature Distribution Shift

We applied both methods previously discussed to generate the shift. First we do *real-world feature imbalance* with 30 clients, then add a gaussian noise with $\sigma = 0.5$ and clip the values between $[0, 1]$.

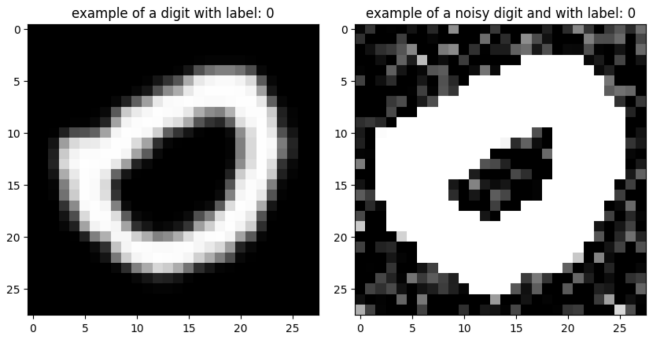

Fig. 1: clean and noisy digit

From 1 We can clearly see the effects of the noise on the digits, the digit gets wider and the there is not a clear fade from the digit to the background anymore.
We trained both models with 5 client gradient steps and 200 rounds and we plotted for both methods a graph of the training loss over the communication rounds and it's Log-Log version.

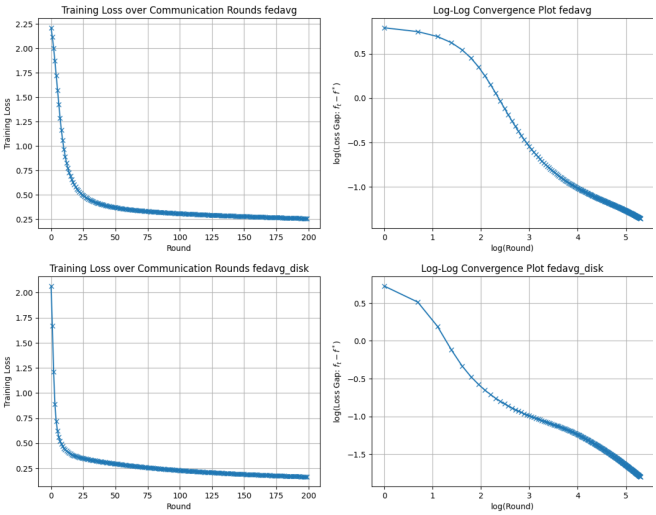

Fig. 2: training loss over rounds, and log-log convergence plot

From 2 we can clearly see an improvement on both the starting loss and on the slope of it's convergence. When we compare it's accuracy with K = 10 and T = 5 and with 30 clients we can clearly see an improvement in the accuracy.

Algorithm	Accuracy
FedAvg	0.8115
MADE	0.9068

TABLE I: Performance of MADE Method

C. Label Distribution Shift

We compare the result of FedAvg and Moon on a Dirichlet distribution with different Betas representative of different severity of label shift, respectively in 3 We can easily see that the fedAvg case smoothly increases towards the limit (84-85 %), in the other hand MOON handles data skew increasingly well as Beta rises, with peak performance around moderate to high Beta values (0.7).

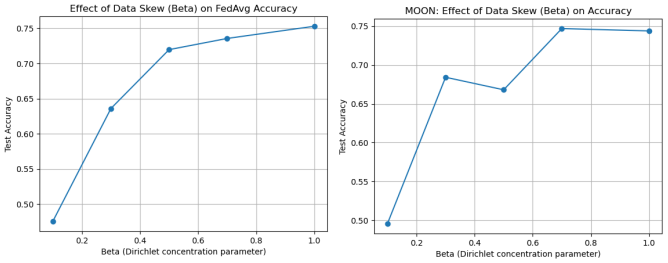

Fig. 3: FedAvg and Moon (left and right respectively) on $Dir(\beta), \beta \in \{0.1, 0.3, 0.5, 0.7, 1.0\}$

IV. Discussion

The increase of performance of FedAvg disk compared to FedAvg and even the baseline might come from the fact that we are down-weighting samples that the local clients over see and up-weighting those they rarely see. This removes some bias to the overrepresented styles, and the model learns features that generalizes across all clients. Also the fact that we clip the noise creates a bigger contrast from the digits and the background, this might make the features more clear for the model making it learn more efficiently

From Figure **??**, MOON doesn't provide any overall significant increase of performance in our experiments. This could be explained by the fact that the global contrastive loss may encourage alignment between very different local representations. Also, the use of MOON with EMNIST dataset might induces semantic inconsistencies across client embeddings especially for visually similar characters (e.g., 'O' vs. '0'), with a reduced number of classes. This leads to noisy or conflicting optimization signals that hinder convergence and reduce the overall effectiveness of MOON. Furthermore, we could explore tuning the hyperparameters μ to modify the strength of the in order to better balance between classification accuracy and representation alignment. Also, future work could explore combining MOON with adaptative weight methods.

V. Summary

In this work, we explored how to systematically generate and mitigate distribution shifts in federated learning, focusing on feature and label distribution discrepancies. Using EMNIST-Digits, we simulated realistic feature distribution shifts by grouping clients by writer identity and further accentuating this shift with Gaussian noise. MADE was used to adaptively reweight samples based on learned feature distributions, yielding noticeable performance gains over the FedAvg baseline.
For feature shift, MADE significantly improved accuracy by reweighting client data. For label shift, MOON had limited impact, likely due to representation noise in complex datasets. Our findings stress the importance of adapting FL methods to the type of data skew present.

REFERENCES

[1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," 2021. [Online]. Available: https://arxiv.org/abs/1912.04977

[2] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," 2017. [Online]. Available: https://arxiv.org/abs/1702.05373

[3] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," 2021. [Online]. Available: https://arxiv.org/abs/2102.02079

[4] H. Nguyen, P. Wu, and M. Chang, "Federated learning for distribution skewed data using sample weights," 2024. [Online]. Available: https://arxiv.org/abs/2401.02586

[5] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," *arXiv preprint arXiv:2103.16257*, 2021. [Online]. Available: https://arxiv.org/abs/2103.16257

APPENDIX A
ALGORITHMS

A. FedAvg

Algorithm 1 Federated Averaging (local SGD)

Input: Data $\{(X_i, y_i)\}_{i=1}^n$, rounds T, local steps K, learning rate γ
Initialize global model θ^0
for $t = 1$ to T **do**
  for each client $i = 1$ to n **do**
    Initialize $\theta_i^t \leftarrow \theta^{t-1}$
    for $k = 1$ to K **do**
      $\theta_i^t \leftarrow \theta_i^t - \gamma \nabla \mathcal{L}_i(\theta_i^t)$
    end for
  end for
  $\theta^t \leftarrow \frac{1}{n} \sum_{i=1}^n \theta_i^t$
end for
Return: θ^T *(final global model)* =0

B. MOON Framework

The algorithm from [5] :

Algorithm 2 The MOON framework

Input: number of communication rounds T, number of parties N, number of local epochs E, temperature τ, learning rate η, hyper-parameter μ
Output: The final model w^T

Server executes:
initialize w^0
for $t = 0$ to $T - 1$ **do**
  for all $i = 1$ to N **in parallel do**
    send global model w^t to P_i
    $w_i^t \leftarrow$ PARTYLOCALTRAINING(i, w^t)
  end for
  $w^{t+1} \leftarrow \sum_{k=1}^N \frac{|\mathcal{D}^k|}{|\mathcal{D}|} w_k^t$
end for
return w^T
function PARTYLOCALTRAINING(i, w^t)
  $w_i^t \leftarrow w^t$
  for epoch $= 1$ to E **do**
    for each batch $b = \{x, y\}$ of $\mathcal{D}^i$ **do**
      $\ell_{\text{sup}} \leftarrow \text{CrossEntropyLoss}(F_{w_i^t}(x), y)$
      $z \leftarrow R_{w_i^t}(x)$
      $z_{\text{glob}} \leftarrow R_{w^t}(x)$
      $z_{\text{prev}} \leftarrow R_{w_i^{t-1}}(x)$
      $\ell_{\text{con}} \leftarrow -\log\left(\frac{\exp(\text{sim}(z, z_{\text{glob}})/\tau)}{\exp(\text{sim}(z, z_{\text{glob}})/\tau) + \exp(\text{sim}(z, z_{\text{prev}})/\tau)}\right)$
      $\ell \leftarrow \ell_{\text{sup}} + \mu \ell_{\text{con}}$
      $w_i^t \leftarrow w_i^t - \eta \nabla \ell$
    end for
  end for
  return w_i^t to server
end function=0