DOI: 10.15439/2023F6549
ISSN 2300-5963 ACSIS, Vol. 35

Mitigating the effects of non-IID data in federated learning with a self-adversarial balancing method

Anastasiya Danilenka
0000-0002-3080-0303
Faculty of Mathematics and Information Science
Warsaw University of Technology, Warsaw, Poland
Email: anastasiya.danilenka.dokt@pw.edu.pl

***Abstract*—Federated learning (FL) allows multiple devices to jointly train a global model without sharing local data. One of its problems is dealing with unbalanced data. Hence, a novel technique, designed to deal with label-skewed non-IID data, using adversarial inputs is proposed. Application of the proposed algorithm results in faster, and more stable, global model performance at the beginning of the training. It also delivers better final accuracy and decreases the discrepancy between the performance of individual classes. Experimental results, obtained for MNIST, EMNIST, and CIFAR-10 datasets, are reported and analyzed.**

I. Introduction

FEDERATED learning (FL) was introduced in [1]. It aims at creating a shared model, combining information from multiple sources, without sharing local data. In standard FL, training proceeds in rounds. Here: (1) global model is initialized, (2) current version of the global model is sent to selected clients, (3) they complete training, using their local data, (4) model updates are gathered on the server and used to generate a new version of the global model [1]. In this work, complete models are communicated (in (2) and (4)).

How to aggregate updates is a subject of intensive research. The basic approach is to average updates (using the FedAvg algorithm). A natural extension to FedAvg is weighting, i.e. assigning individual importance to each client, based on additional knowledge. For example, clients with more local data receive larger weights for their updates [1].

In FL, the complete dataset is never "known". Hence, the statistical properties of local and global datasets are unknown. Therefore, it cannot be established if data is identically independently distributed (IID). However, it has been established that non-IID datasets negatively affect the quality of the FL-trained model [2]. Non-IID data can be classified on the basis of the source of heterogeneity [2], i.e.: (1) data quantity skew (local datasets differ in size), (2) label distribution skew (different devices have different subsets of labels inside local dataset), (3) attribute skew (local data has unique characteristic features, noise, perturbations, etc.), (4) temporal skew (local data distributions differ over time, or data was collected in different time periods). Obviously, combinations of skews can materialize. This work concentrates on non-IID datasets with the label distribution skew. Depending on how many labels are represented in the local dataset, the skew can be extreme, with only 1 label in each dataset, to a $(C-1)$-label skew, where C is the number of labels in the set. Here, each local dataset lacks data for exactly one label. In research experiments, the data partition strategy determines, which label(s) is(are) missing.

In this context, a novel method, to overcome the problems caused by the label distribution skew, called *Adversarial Federated Learning* (AdFL), is proposed. It is inspired by adversarial attacks and is applicable, primarily, to neural networks applied to image data. To the best of our knowledge, it is the first attempt to recover local data distribution information, from the clients, using adversarial inputs and to use adversarial images to coordinate the training process.

In what follows, in Section II, related research is outlined. Section III, introduces adversarial attacks followed, in Section IV, with the description of the AdFL algorithm. Section V, presents the experimental setup and results, obtained with MNIST [3], EMNIST [4] and CIFAR-10 [5]. Conclusions and directions for future research complete this work.

II. Related work

When the problem of data heterogeneity was acknowledged, the quantity skew was mitigated by weighting the updates, based on the number of data samples in local datasets. In other approaches, to deal with the problem of non-IID data, FedProx [6] and SCAFFOLD [7], tackled the problem by restricting local model updates using proxy terms and control variates. Moreover, use of gradient correction [8], utilization of knowledge distillation [9], applied client picking [10], data sharing [11] and adapted loss functions in presence of data imbalance [12] have been explored.

While many methods reported improvements on standard datasets (MNIST, CIFAR-10, CIFAR-100), still, (i) some required additional information from clients (e.g. distribution information, averaged data), (ii) others rely on external datasets (representative of the local data), or (iii) involve training of additional, data generator, models. This may not be feasible in the real world or may introduce new privacy risks. Hence, the proposal is articulated in what follows.

III. Adversarial federated learning

In FL, in the past, adversarial techniques were used to generate data (a) to improve resistance to adversarial attacks [13], or (b) to increase the amount of locally available data [14].

Here, a different way of integrating adversarial data into FL is proposed.

A. Adversarial attack

Adversarial attacks make it possible to alter a sample from the training data, in a way that is undetectable to humans, so that the network will misclassify such (previously classified) sample [15]. In other words, when the attack is performed on a trained model then, by modifying the target sample, the adversary tries to make a valid target sample cross the decision boundary of the classifier [16], and be misclassified.

Depending on the applied changes, one can distinguish *non-targeted* and *targeted* attacks. A non-target attack aims at making the model deliver incorrect predictions. Targeted attacks set the class, to which the misclassification should be attributed. Separately, *white-box* attacks can access the model's architecture and parameters, while *black-box* can access only the model's output. Among the most famous attacks are: one-step Fast Gradient Sign Method (FGSM) [17], its iterative version I-FGSM [18], and its version enhanced with momentum MI-FGSM [19].

B. Transferability of adversarial inputs

An important property of adversarial inputs is their transferability, i.e. adversarial inputs generated for one model will mislead also other model(s) trained on similar datasets, to solve similar tasks. Note that in FL clients share the same task, model architecture, and data space. Hence, adversarial samples, generated by all clients, should be transferrable. To establish this, a series of experiments, to measure the attack success rate (ASR) – a common metric for qualifying the performance for adversarial attacks [19] – was performed. In these experiments, and in what follows, MI-FGSM was used to create targeted adversarial attacks [19]. Overall, data was IID distributed among 40 clients (all clients had samples of all classes in local datasets). During each server-side epoch, 10 clients were selected, according to the strategy described in Section IV, and performed local training. Next, clients' models (returned to the server) were used to generate one adversarial sample per class (see, Section IV-A). Hence, $10*C$ adversarial images were generated, where C is the number of classes. Next, updated clients' models classified the adversarial samples, and if the predicted class was equal to the target, the attack was qualified as successful. Table I shows the transferability metric for standard datasets (MNIST, EMNIST, and CIFAR-10) during the 5th, 25th, and 50th epochs.

TABLE I
ATTACK SUCCESS RATE (ASR) STUDY INSIDE A FEDERATED SCENARIO

	Epochs		
Dataset	5	25	50
MNIST	1	1	1
EMNIST	0.98	1	1
CIFAR-10	0.5	0.99	1

As can be seen, ASR is high, therefore, due to the high transferability of adversarial data and its connection with the decision boundaries, it is possible to derive insights about the local data, by asking clients' models to produce adversarial images, while "keeping private information private". This observation became the foundation of the AdFL algorithm.

IV. AdFL ALGORITHM

The AdFL algorithm uses adversarial images as a source of additional knowledge about FL training. To do so, the adversarial data is generated first. Here, there are two places where adversarial images can be generated. (1) Clients can produce them, as they have access to all necessary data. (2) Server can use the updated models to generate them. In AdFL, adversarial data is generated on the server, using a random noise image as a starting point. The server-side part of the AdFL algorithm is summarized in Algorithm 1.

Algorithm 1 AdFL algorithm (Server); Cl states for client; Cl_e – subset of clients picked for training on epoch e; *global distribution* tracks distribution of classes during FL training; $distr_{all}$ – estimated classes presence in clients' local datasets

```
Ensure: global model $w_0$, global distribution, clients ready
  for $e$ in $epochs$ do
    if $e == 0$ then
      $Cl_e \leftarrow$ all clients
    else
      $Cl_e$, global distribution $\leftarrow$ pick clients($distr_{all}$, global distribution)
    end if
    for $Cl$ in $Cl_e$ do
      $w_e^{Cl} \leftarrow$ run training($w_e$)
    end for
    adv data $\leftarrow$ create adversarial data([$w_e^0$, ..., $w_e^{Cl_e}$])
    if $e == 0$ then
      $distr_{all} \leftarrow$ estimate distribution(adv data)
    end if
    $CS_{[0-Cl_e]} \leftarrow$ calculate coherence(adv data, $w_e^{[0,..,Cl_e]}$)
    $w_e \leftarrow$ FedAvg([$w_e^{[0,..,Cl_e]}$], $CS_{[0-Cl_e]}$)
  end for
```

Overall, there are 6 steps that define the AdFL algorithm. Note that all AdFL-specific steps take part on the server, with no additional Client-side computations.

1) During "warm-up", round **all** clients perform local training, and return models to the server. In subsequent rounds, local training is completed by a subset of clients on the received version of the global model.
2) On the server, updated clients' models generate adversarial samples, as described in Section IV-A
3) Generated adversarial samples are used to estimate the distribution of classes across clients (see, Section IV-B).
4) Coherence scores (CS) are calculated, based on updated models and adversarial samples (see, Section IV-D).
5) CS are used as weights during aggregation, resulting in the next global model, sent to the clients.

6) From there on, the subset of clients that participate in the training is defined by the client-picking strategy (Section IV-C) and the process repeats.

A. Adversarial inputs creation

To create adversarial data, the data-free approach has been selected, to protect clients' data. Hence, the initial data source is a random noise image that, by default, is not classifiable. Starting from this image, a targeted adversarial attack is performed, using the MI-FGSM. Here, the adversarial image creation lacks malicious intent, becoming less sensitive to the amount of allowed changes. Hence, the MI-FGSM parameters have been aligned with the pursued goal. Overall, each updated client model generates C images representing classes that are present in the task. The steps of adversarial input generation are presented in Algorithm 2, with the default federated steps omitted.

Algorithm 2 Adversarial data generation

```
Ensure: $targets \leftarrow [0, ..., C-1]$
Ensure: $w_e^{[0,..,Cl_e]}$ {Clients' updated models during epoch $e$}
  for $target$ in $targets$ do
    for $w_e^i$ in $w_e^{[0,..,Cl_e]}$ do
      $adv\ img \leftarrow$ rand noise[$Ch$, $H$, $W$]{Random sample}
      for $step$ in $num\ steps$ do
        $adv\ img_{target}^i \leftarrow$ step($w_e^i$, $adv\ img_{target}^i$, $target$)
      end for
    end for
  end for
```

B. Local distribution estimation

Based on generated adversarial data, it is possible to estimate the class distribution among the clients. Here, it was established that predictions of adversarial samples, returned by the clients' updated models, are illustrative of the presence of certain classes in the local dataset (of this client). Therefore, uncovering class presence within clients' data can be used for balancing the label-skew. It is also the reason why adversarial sample generation runs for a set number of steps (30). If the target class is missing from the client's local dataset, the model will fail to generate an adversarial sample of that class. Moreover, using this knowledge, AdFL performs a warm-up epoch, by initiating training on a̲ll clients. This allows capturing a meta-level picture of class presence across clients, gathering the data needed for the client-picking routine that will occur in each training round.

After the updated models return to the server, adversarial data generation occurs, and the results of all models' predictions are used to estimate class distributions. Here, all updated models generate adversarial samples and predict the resulting samples. Next, for each model, its predictions are summarised and classes that appeared in the predictions are treated as signs of these classes being present in the client's dataset.

A set of experiments was performed to establish how precise the estimation of classes' presence in local datasets is. The experiments were performed for two extreme cases: IID setup (all classes are present), and $\leq 20\%$ of classes setup, i.e. two classes per client for MNIST and CIFAR-10 datasets, and 12 classes for the EMNIST dataset. The effectiveness was measured as the percent of classes detected from the adversarial data in total, across all clients during the warm-up FL training round, against the average number of classes detected per client. The results are listed in Table II.

TABLE II
DETECTED CLASSES (DC) METRICS FROM ADVERSARIAL DATA

	MNIST		CIFAR-10		EMNIST	
Metric	IID	2	IID	2	IID	12
DC (%)	100	100	100	100	100	97.1
Avg. DC	10	2	10	2	62	11.4

Although the experiments show that class detection can be performed quite accurately, it remains only an estimation of the actual classes' presence. Moreover, the quality of the adversarial samples depends on the number of local training epochs. Thus, tuning this parameter is important for obtaining a proper class distribution prediction. Hence, for the reported experiments, the number of local epochs is set to 10 for MNIST and EMNIST and 2 for CIFAR-10.

C. Client-picking strategy

After predicting classes that are present in the local datasets, a client-picking strategy that will mitigate the presence of local label-skew can be proposed. Based on the estimation of classes' presence in local datasets obtained during the "warm-up" round, a simple approach to balance the training process was designed. Specifically, the balance of classes during the training process is maintained by the global label frequency vector of size C, where C is the number of unique classes in the classification task. This vector is updated in each FL training epoch after a client is selected to be involved in the current training epoch. It reflects the frequency of a particular class being picked for training. To ensure equal exposure for all classes, the clients for each FL round are selected to make the values in C close to a uniform distribution. Here, a Kullback–Leibler (KL) divergence is used (for details, see [20]). This allows each FL round to include clients with rare classes in their data, by computing the KL-divergence of the global classes frequency, with respect to the uniform distribution, if a client (its data classes) is to be added to the training round. In each round, clients with the smallest KL divergence are picked. Here, note that several clients may have the same KL divergence (possibly, a minimum for the current set of clients). In this case, random client selection is applied.

D. Clients coherence measurement

Another outcome of the transferability of adversarial samples is an ability to identify "problematic clients", i.e. clients, whose models are not able to produce or identify transferable adversarial samples. In order to measure the "two-side transferability", the coherence score (CS) measure was defined.

CS calculation can be described as follows: (1) at the end of the training round, all updated models create adversarial samples for each target class, (2) each updated model predicts all adversarial samples produced by models participating in the round, (3) for each updated model the CS consists of two parts: (i) describing how good this model recognized adversarial samples from other models, and (ii) how successfully other models recognized this models samples. Therefore, models with high CS excel in both creating transferable samples and correctly classifying those created by others.

Here, the ability of the updated model to predict adversarial images produced by other models is measured according to Equation 1, where each multiplication consists of a binary flag indicating whether or not the prediction for class c generated by model k was correct, and the probability returned by the model. Predicting own inputs is omitted.

$$\text{predicted others} = \sum_{k=1}^{K} \sum_{c=0}^{C-1} \text{is correct}_{k,c} * \text{returned prob.}_{k,c} \tag{1}$$

A similar measure was applied to evaluate the ability, of the updated model, to produce adversarial images that are recognized by the other models. The individual CS is a result of a simple summation of the two scores. After the coherence scores, for all clients, are calculated, they are normalized and used as weights for the FedAvg aggregation.

V. Experimental setup, results, and their analysis

A. Data partitioning

During experiments, label-skew was simulated using three parameters: (1) number of unique classes in dataset, (2) total number of data samples in dataset, and (3) probability of class appearing in data. The number of unique classes inside the local dataset is fixed for an experiment, and all clients have the same number of unique classes. However, the set of local labels differs between clients. Total number of samples is also fixed, and all clients have the same amount of data. Moreover, local data is equally divided among classes, e.g. for 2 classes, 50% of data will be of class 1, and 50% of class 2. The probability of all classes appearing in a local dataset is defined, by drawing a sample from the normal distribution for each of the classes. Next, the resulting values are normalized to represent the probability vector. To determine the classes of a specific client, a random subset of the set size is drawn. Since the normal distribution was used, borderline cases may materialize, when few classes have a high probability of appearing. Therefore, they will be overrepresented, while a few other classes may be left out, because of their extremely low probability of occurrence. Here, random seeds were used to ensure robustness. An example of the probability of occurrence for 10 classes is presented in Figure 1.

Note that this data partitioning scheme introduces a challenging label-skew scenario, as it allows some classes to be "common" in the clients' population, while others are rare, therefore, producing a global class imbalance.

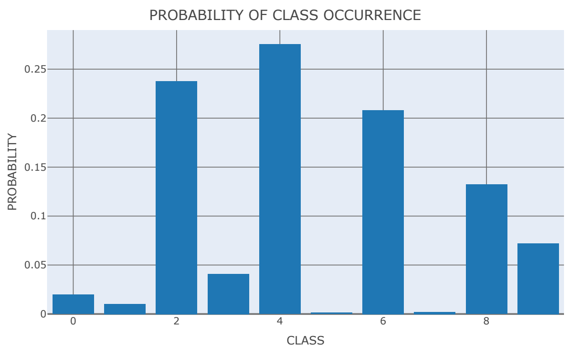

Fig. 1. Probability of occurrence for 10 classes

B. Models and hyperparameters

In the experiments, Convolutional Neural Networks (CNN) were used. For MNIST and EMNIST, a LeNet-5 architecture was used [21]. For CIFAR-10, the pre-trained version of the mobilenetv2 [22] model was used, provided by the torchvision package, with weights coming from Imagenet dataset [23]. The pre-trained version of the mobilenetv2 model was chosen to verify the applicability of the algorithm to more complex datasets and architectures that are not trained from scratch. If not stated otherwise, the cross-entropy loss function was used. The Stochastic Gradient Descent (SGD) was used as the optimizer. Model and algorithm-specific hyperparameters are presented in Table III.

TABLE III
Hyperparameters used in the experiments

Parameter	MNIST	EMNIST	CIFAR-10
Num. classes per client	2	12	2
Total classes	10	62	10
Clients per training round	10	10	10
Total clients	40	40	40
Data samples per client	400	1200	400
Learning rate	0.001	0.001	0.001
Batch size	10	10	10
Num fed epochs	10	10	2
FedProx μ	0.1	0.1	0.1
FedMix M	400	1200	400
FedMix λ	0.2	0.2	0.05
Num of adv. steps	30	30	30

C. Experimental setup

The performance of AdFL was tested on three datasets: MNIST, EMNIST, and CIFAR-10. The project, including both AdFL and other algorithms, was implemented in Python (3.7.9), using PyTorch (1.10.0). Moreover, for ready model architectures and datasets, torchvision (0.11.1) was used.

D. Experimental results and their analysis

AdFL performance is compared to FedAvg, FedProx, and FedMix algorithms (Section II). The parameters for FedProx and FedMix were as in [11]. The experiments were performed 10 times for MNIST and 6 times for EMNIST and CIFAR-10, with different random seeds, while preserving data partition. For MNIST, the resulting median test accuracy, over a set of experiments, is presented in Figure 2.

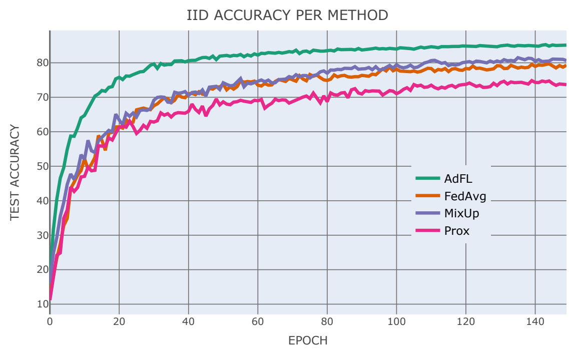

Fig. 2. Test accuracy for MNIST

AdFL improves model performance at the beginning of the training, and ensures stable performance, with slight accuracy improvement, later. This can be valuable in the case of the limitation of client-server communication rounds. The minimum accuracy improvement is about 3.3%.

For EMNIST, the median test accuracy of different algorithms is presented in Figure 3. Again, AdFL is more stable at the beginning and shows gradual improvement without significant peaks. It also results in test accuracy improvement of around 2% (compared to FedAvg).

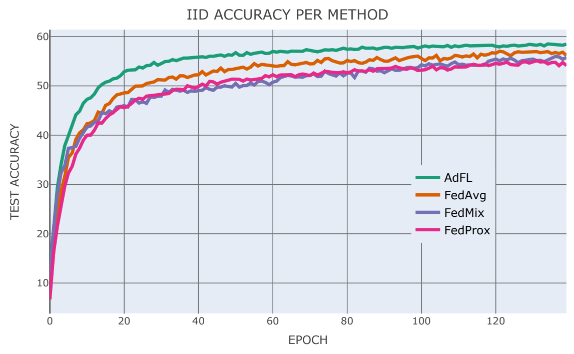

Fig. 3. Test accuracy for EMNIST

Note that, with a very challenging label-skew scenario, the performance of the models depends directly on the probabilities of class occurrence and how they are distributed among clients. Here, an additional set of 7 EMNIST experiments was performed, where the random seeds were changed each time before generating probabilities of classes occurrence and classes distribution, while preserving other parameters listed in Table III. The EMNIST dataset was chosen for testing, as it has 62 classes (as opposed to 10 classes in the remaining datasets). The average accuracy improvement and the Wilcoxon signed-rank test [24] results are presented in Table IV. As can be seen, the accuracy improvement remains for varying data partitions, with respect to FedAvg, FedProx, and FedMix algorithms.

TABLE IV
ADFL PERFORMANCE STUDY ON VARYING EMNIST DATA PARTITIONS

	FedAvg	FedProx	FedMix
Acc. impr.	1.70%	3.87%	1.88%
Std	1.11	1.61	2.91
p-value	0.0469	0.0156	0.0313

For CIFAR-10, the median accuracy is shown in Figure 4. It can be seen that, compared to previous datasets, training was less stable and the results show accuracy fluctuations even for a median of results. Still, on average, AdFL shows better accuracy from the very start of the training, resulting in a final accuracy improvement of around 2%.

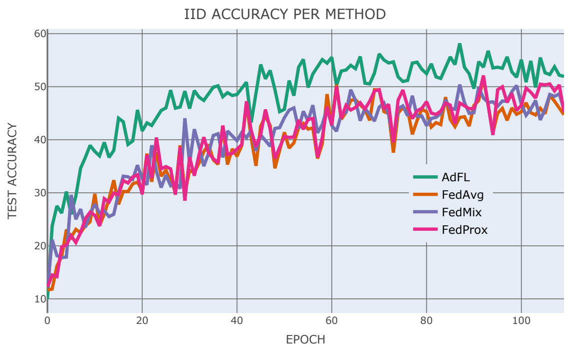

Fig. 4. Test accuracy for CIFAR-10

The summary of all experiments is presented in Table V together with the standard deviation of the final accuracy. The statistical accuracy improvement, related to AdFL, was measured with the Wilcoxon signed-rank test, on results presented in Table V and is depicted in Table VI.

TABLE V
EXPERIMENTS SUMMARY

	MNIST		CIFAR-10		EMNIST	
	ACC	STD	ACC	STD	ACC	STD
AdFL	56.44	0.51	51.77	2.36	56.67	0.41
FedAvg	53.71	1.75	47.70	2.53	53.39	1.38
FedProx	52.28	1.39	48.42	3.21	51.99	1.09
FedMix	51.03	1.53	49.13	2.30	50.81	1.47

TABLE VI
WILCOXON SIGNED-RANK TEST P-VALUE

	MNIST	EMNIST	CIFAR-10
FedAvg	0.0039	0.0313	0.0313
FedProx	0.0039	0.0313	0.0313
FedMix	0.0039	0.0313	0.0313

The results of the Wilcoxon signed-rank test show that the difference in accuracy achieved by AdFL is statistically significant. To better encapsulate the unstable performance on the CIFAR-10, the median over the last 10 epochs was taken as the final accuracy. AdFL improves the accuracy of the global model and, moreover, reduces the gap in performance between individual classes, despite their uneven distribution across local datasets. It can be measured as a standard deviation between accuracy among all classes (see, Table VII).

TABLE VII
TEST ACCURACY DEVIATION AMONG INDIVIDUAL CLASSES PER METHOD

Dataset	FedAvg	FedProx	FedMix	AdFL
MNIST	0.095	0.179	0.112	**0.025**
EMNIST	0.34	0.33	0.39	**0.26**
CIFAR-10	0.35	0.34	0.37	**0.23**

For all datasets, the standard deviation within the classes is significantly lower for the AdFL algorithm, therefore, illustrating the benefits of balanced training. Finally, the Wilcoxon signed-rank test was applied and it was found that the obtained results are statistically significant.

VI. Concluding remarks

In this work, it was shown that utilizing adversarial data on the server side, during FL training, can reveal data distribution information. Use of this information results in more balanced performance in all classes, in the case of label-skewed data. Future research can concentrate on (1) exploring properties of adversarial samples, and (2) applicability of AdFL to more complex datasets, models, and label-skew scenarios. Improvements can also be made to the client-picking strategy and the adversarial data generation process. Additional research can also explore AdFL's potential to battle other non-IID scenarios, e.g., by locating clients with corrupted data.

Acknowledgment

This work was supported by the Centre for Priority Research Area Artificial Intelligence and Robotics of Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) programme and by the Laboratory of Bioinformatics and Computational Genomics and the High-Performance Computing Center of the Faculty of Mathematics and Information Science Warsaw University of Technology.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282.

[2] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021. doi: https://doi.org/10.1016/j.neucom.2021.07.098

[3] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[4] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017. doi: 10.1109/IJCNN.2017.7966217 pp. 2921–2926.

[5] A. Krizhevsky, "Learning multiple layers of features from tiny images," https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

[6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*, I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, Eds. mlsys.org, 2020.

[7] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5132–5143.

[8] E. Ozfatura, K. Ozfatura, and D. Gündüz, "Fedadc: Accelerated federated learning with drift control," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE Press, 2021. doi: 10.1109/ISIT45174.2021.9517850 p. 467–472.

[9] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, "Fine-tuning global model via data-free knowledge distillation for non-iid federated learning," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. doi: 10.1109/CVPR52688.2022.00993 pp. 10 164–10 173.

[10] M. Tang, X. Ning, Y. Wang, Y. Wang, and Y. Chen, "Fedgp: Correlation-based active client selection for heterogeneous federated learning," 03 2021.

[11] T. Yoon, S. Shin, S. J. Hwang, and E. Yang, "Fedmix: Approximation of mixup under mean augmented federated learning," in *International Conference on Learning Representations*, 2021.

[12] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Addressing class imbalance in federated learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, pp. 10 165–10 173, May 2021. doi: 10.1609/aaai.v35i11.17219

[13] C. Chen, Y. Liu, X. Ma, and L. Lyu, "Calfat: Calibrated federated adversarial training with label skewness," 2023.

[14] Y. Lu, P. Qian, G. Huang, and H. Wang, "Personalized federated learning on long-tailed data via adversarial feature augmentation," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023. doi: 10.1109/ICASSP49357.2023.10097083 pp. 1–5.

[15] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013.

[16] O. Suciu, R. Marginean, Y. Kaya, H. D. III, and T. Dumitras, "When does machine learning FAIL? generalized transferability for evasion and poisoning attacks," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018. ISBN 978-1-939133-04-5 pp. 1299–1316.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[18] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2017.

[19] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. doi: 10.1109/CVPR.2018.00957 pp. 9185–9193.

[20] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79 – 86, 1951. doi: 10.1214/aoms/1177729694

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/CVPR.2018.00474 pp. 4510–4520.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009. doi: 10.1109/CVPR.2009.5206848 pp. 248–255.

[24] D. Rey and M. Neuhäuser, *Wilcoxon-Signed-Rank Test*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1658–1659. ISBN 978-3-642-04898-2