Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 919–924
DOI: 10.15439/2023F1102
ISSN 2300-5963 ACSIS, Vol. 35

One-shot federated learning with self-adversarial data

Anastasiya Danilenka, Karolina Bogacka
0000-0002-3080-0303
0000-0002-7109-891X
Faculty of Mathematics and Information Science
Warsaw University of Technology
ul. Koszykowa 75, 00-662 Warsaw, Poland
Email: {anastasiya.danilenka.dokt, karolina.bogacka.dokt}@pw.edu.pl

Katarzyna Wasielewska-Michniewska
0000-0002-3763-2373
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland
Email: Katarzyna.Wasielewska@ibspan.waw.pl

***Abstract*—Federated learning (FL) is a decentralized approach that aims at training a global model with the help of multiple devices, without collecting or revealing individual clients' data. The training of a federated model is conducted in communication rounds. Still, in certain scenarios, numerous communication rounds are impossible to perform. In such cases, a one-shot FL is utilized, where the number of communication rounds is limited to one. In this article, the idea of one-shot FL is enhanced with the usage of adversarial data, exploring and illustrating the possibilities to improve the performance of resulting global models, including scenarios with non-IID data, for image classification datasets: MNIST and CIFAR-10.**

I. Introduction

FEDERATED learning [1] is a popular research field that attracts thousands of researchers due to its simple, yet, open for improvements idea that is inline with current trends in distributed computing infrastructures. The core of federated learning lies in its collaborative nature, which allows multiple devices (clients) to use their own private data to jointly train one global model, managed by the centralized server. In general, the federated learning workflow can be summarized as follows: (1) a global model is initialized (during the first round) or aggregated (for subsequent rounds) on a server and sent to the set of client devices, (2) client devices receive the current version of the global model and use their private data to train the model for a set number of epochs, (3) each client returns resulting updates/weights/whole model back to the server, (4) server receives updates from clients and aggregates them into the new version of the global model. The client-based training round happens multiple times and is referred to as a communication round. At the end of each communication round the updated models are aggregated into the new version of the global model utilizing the federated averaging (FedAvg [1]) algorithm, which averages the updated models' weights. The federated averaging can also be easily combined with a weighting technique, for instance, its first version [1] weighted individual client's update based on the size of the local dataset this client possessed and used during training.

Communication between the centralized server and client devices is a well-known bottleneck for the federated learning pipelines [2], therefore, techniques for improving the convergence time [3], minimizing energy consumption [4] or improving network resource management scheme [5] were studied. One of the ways to mitigate the communication burden between the server and the client is to utilize the concept of few-shot learning.

Few-shot learning [6] is usually referred to as a learning technique where during training a model only sees a small portion of data (for instance, a few examples of each class in a classification task instead of a full dataset) and then is considered ready for performing testing/inference. In the case of federated learning, the few-shot learning idea restricts the number of communication rounds that happen between clients and centralized servers, e.g. one-shot federated learning implies only one communication round [7].

Despite few-shot learning techniques being able to drastically reduce the number of communication rounds needed to train the model, new questions arise, concerning the performance of the resulting models, since machine learning models usually require numerous epochs to reach the best possible accuracy. The problem of few-shot FL is further complicated by the privacy-preserving nature of the FL, which does not allow revealing any information about the local data that clients used during local training. In some cases, the problem of non-IID data can materialize, which can further damage the performance of the resulting global model [8], [9], [10]. Non-IID data can manifest itself in many ways. One of the possible classifications can be described as follows [9]: (1) quantity skew (different sizes of the local datasets that clients possess), (2) attribute skew (local datasets have unique distinct features for the same event/object they are describing, e.g. writing style), (3) label skew (only a subset of labels is present in local data, leaving some labels with no samples), (4) temporal skew (local datasets have time-dependent nature, e.g. were collected in different moments of time), (5) preference skew (same

The work of Anastasiya Danilenka and Karolina Bogacka 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.

event/object in different local datasets has a different target value due to subjective preference, e.g. ratings). Combinations of the different skews can also be present inside one scenario.

In this article, one-shot federated learning is researched in the context of aggregation of the clients' updated models and the possibility of using adversarial images as a source of client-picking guidance and performance improvement in the presence of label skew non-IID data.

II. Related works

As was stated in section I, the decision to drastically limit the number of communication rounds increases the importance of the aggregation algorithm, which can significantly influence the best possible performance on the test dataset. One of the first approaches to one-shot FL was to utilize ensemble learning, where each updated client model was treated as a part of the ensemble [7]. It was acknowledged, that in FL scenarios, the size of the ensemble of models depends on the number of participating clients, which can reach millions of devices. Moreover, not all clients are equally "useful" in terms of the data they have. Therefore, the ensemble of the models was restricted to a subset of models depending on the selection criteria. For example, models for the ensemble could be chosen randomly, or based on some indicator. One of the possible indicators for best candidates is the local test performance of the model, which requires the model to save a portion of its local data as a test set to measure the performance of the updated global model after local training on the remaining local data. Another similar technique uses a local cross-validation performance as a performance indicator for ensemble model picking.

Distillation technique was also researched with respect to one-shot FL. Data distillation was studied as an alternative to communicating whole models/model updates from clients back to the server [11]. Instead, each client, after receiving the global model used it to distill its own local data and sent the resulting set of distilled data and targets back to the server. Although communicating clients' data even in a distilled form that cannot be directly interpreted by the human eyes may be considered a violation of the clients' privacy, the authors state, that acquiring distilled data will not let the adversary replicate the resulting global model. After receiving the resulting datasets from all clients, the server uses them to train its own global model. Some enhancements were also presented in the process of data distillation, for instance, soft labels. Another distillation technique is proposed to treat clients' models as teachers and use them for training a student model on the server side. For instance, client devices can use their data to train conditional variational autoencoder (CVAE [12]). Moreover, the ensemble of these decoders is further distilled on the server into one decoder that can further be used as a data generator for training a global model on the server side [13].

The presented approaches to one-shot FL are capable of reaching a good final accuracy while preserving the benefits of the reduced number of communication rounds. Nevertheless, they may still require training additional models (e.g. encoders) or be prone to suffering from non-IID data (cross-validation ensembles). Therefore, a new way of performing one-shot FL can be of interest. This article presents an algorithm that can acknowledge the presence of label skew non-IID data and mitigate its effect on the final model, without requesting any additional data from the clients or imposing any additional computation on the client devices by using adversarial data.

III. Adversarial attack

Neural networks are susceptible to various kinds of adversarial attacks [14]. The attacks aim at misleading the trained models into incorrect predictions, by altering the perfectly correct source data sample in a way that is unrecognizable by the human eye. This changed source sample is referred to as an adversarial sample.

There are several methods for generating adversarial samples. The method that the adversary prefers can depend on how much information about the source model the adversary has. The attack methods that require full access to the target model gradients are called 'white-box" attacks, while attacks that can operate on a limited set of information from the source model, e.g. the predictions from the target model, are called "black-box" attacks. Among the most popular attack methods, one can name: the fast gradient sign method (FGSM) [15], its iterative version I-FGSM [16], momentum-enhanced MI-FGSM [17], Carlini and Wagner (C&W) [18] attacks, and more.

Moreover, with respect to image classifiers, attacks can also be divided into two categories based on the precise intention of the attack. The attack which aims solely to mislead the trained model into misclassifying an image into any class that is not the right one is called a non-target adversarial attack, while the attack that aims at making the classifier make a mistake by predicting a certain class, set by the adversary, is called a targeted attack.

One of the fascinating properties of the adversarial samples is their transferability [19] – adversarial samples generated for one target model can also mislead models which were trained to solve similar tasks. In other words, models with similar architectures that were trained on non-intersecting subsets of the dataset will most likely be successfully attacked by the same adversarial sample. The reason behind this phenomenon is that models that have similar tasks tend to come up with similar decision boundaries. This behavior can be valuable in terms of federated learning scenarios, where clients with similar datasets are training a set of individual models with respect to the same task objective.

To sum up, during the targeted adversarial attack, the adversary creates an adversarial sample by modifying the source sample based on the selected attack method (e.g. by using gradient-based algorithms like FGSM-family methods). These changes applied to the source sample force it to cross the decision boundary estimated by the target model [20], resulting in misclassification.

IV. Proposed approach

As discussed in section II, one of the possible approaches to one-shot FL is an estimation of the most successful client models and using them for making an ensemble of models instead of combining all clients' models into one global model version. The proposed approach described in this paper uses the knowledge retrieved from the adversarial samples to find the most promising clients to be included in the ensemble of models. A more complex algorithm based on the idea of adversarial data was described in a previous article [21] and is referred to as AdFL (Adversarial FL). Although the AdFL algorithm implies training in epochs, in this article the algorithm is adapted to a one-shot FL scenario and further extended to be used in an ensemble of models. The description of the algorithm is given in Algorithm 1.

Algorithm 1 Server-side of proposed one-shot FL, where n – positive size of the model ensemble, w_0 – initialized global model

Ensure: w_0; clients are ready;
Require: n > 0;
  for client in Clients **do**
    $w_0^{client} \leftarrow$ run training on client(w_0)
  end for
  adv data $\leftarrow$ create adversarial data($w^{[0,\dots,Clients]}$)
  $CS^{[0,\dots,Clients]} \leftarrow$ calculate CS(adv data, $w_0^{[0,\dots,Clients]}$])
  $w_0^{[0,\dots,Clients]} \leftarrow$ sort desc($w_0^{[0,\dots,Clients]}$, $CS^{[0,\dots,Clients]}$)
  model ensemble $\leftarrow w_0^{[0,\dots,n-1]}$

1) The server initializes the global model and sends it to all clients, participating in training.
2) Clients perform local training with the whole data they possess for the specified number of epochs.
3) Clients send the resulting updated models back to the server.
4) After collecting all updated models, the server uses them to generate $C \times N$ adversarial samples, where C – is the number of classes in the classification task and N – is the number of updated models returned from the clients as described in section IV-A.
5) Based on the generated adversarial samples, for each updated client model, a *coherence score* (*CS*) is calculated as described in section IV-B.
6) *CS* is further used as a performance indicator to identify the top-performing models and use them as an ensemble.

A. Adversarial samples generation

In order to identify clients that can potentially be useful for the ensemble, the proposed algorithm exploits the transferability property of adversarial samples that was described in section III. Still, per definition, adversarial samples are created on the basis of existing samples drawn from the source data. This condition is generally unacceptable for strictly privacy-preserving FL. Therefore, considering the absence of the source data on the server side where the creation of the adversarial samples happens, a random noise image is considered the starting point for adversarial sample generation.

The targeted MI-FGSM method is used in this article to generate adversarial samples [17]. The algorithm can be summarised in a few steps as described in formulas listing (1), where t is the current iteration of the method, x states for the input noise image, y – target class, to which the resulting image should eventually be attributed, g_t – accumulated gradients through previous t iterations, θ – source model (in case of presented algorithm, one of the updated clients' models), and J – loss function (e.g. cross-entropy).

$$g_{t+1} = \mu * g_t + \frac{\nabla_x J(\theta, x_t^*, y))}{||\nabla_x J(\theta, x_t^*, y))||_1} \quad (1)$$

$$x_{t+1}^* = x_t^* + \alpha * sign(g_{t+1}) \quad (2)$$

The algorithm is parameterized by a number of parameters, namely, μ – decay factor, α – step size. Moreover, the resulting x_{t+1}^* image is further clipped at the end of each iteration in the clipping range e. The parameters used in this paper are different from those presented in the referenced paper, due to the different intentions for the resulting adversarial samples. Initially, adversarial samples are created to perform an attack on trained classifiers during inference time, but in the case of the one-shot FL algorithm described in this paper, the target is the transferability measure of the samples. Therefore, the constraints on the amount of the changes applied to the source image were loosened to allow more gradient information to be added to the adversarial sample. For instance, the number of MI-FGSM iterations was increased to 30, and step size α was increased to 1.

B. Coherence score

After local training, clients return the resulting updated models back to the server, where each model is used to generate one adversarial sample per class in the classification task. These samples are further used to estimate their transferability across all models. The idea behind this action is to find models that can generate transferable samples and are susceptible to samples, generated by other models. This two-side transferability might come from the similar decision boundaries that were learned during the local training step as noted in section III. Therefore, it can be assumed that such models learned somehow in a similar way.

The two-sided transferability measure is called a coherence score (CS) and is calculated out of two separate measures, each of which summarizes either the ability to create or the ability to identify adversarial samples.

The ability of a certain model to produce adversarial samples that are being recognized by other models participating in training is measured according to equation (3), where k stands for the model that was predicting adversarial samples generated by the source model, c stands for a target class that the adversarial sample was made to represent. The measure is calculated across all models from the training set with respect to the adversarial data, generated by the source model for

which the measure is computed. The predictions of this source model for its own data are omitted.

$$\text{was predicted} = \sum_{k=1}^{K} \sum_{c=0}^{C-1} \text{is correct}_{k,c} \cdot \text{returned prob.}_{k,c} \quad (3)$$

This equation uses a binary feature to identify if the prediction of the adversarial sample was correct or not. This binary flag is then multiplied by the confidence of the prediction. Therefore, no punishment is done due to the wrong prediction and less certain predictions will accumulate less significance.

The ability of the model to correctly predict classes of adversarial data generated by other models is measured according to equation (4), where k stands for the model that generated the adversarial data and c stands for the target class of the adversarial sample.

$$\text{predicted others} = \sum_{k=1}^{K} \sum_{c=0}^{C-1} \text{is correct}_{k,c} \times \times \text{returned prob.}_{k,c} \quad (4)$$

Again, the results of the model predicting its own adversarial data are omitted.

The resulting coherence score is a simple summation of the two previously described measures (equation (5)).

$$\text{coherence score} = \text{predicted others} + \text{was predicted} \quad (5)$$

Each of the models returned by the clients, participating in training, acquires its own coherence score and the list of models can then be easily sorted based on the resulting measure. The models that scored higher in CS are treated as those that provide more value to the model ensemble and are picked first. On the other hand, the coherence score can also be used for weighted federated averaging to create one global model instead of an ensemble.

V. Experimental section

To illustrate the efficiency of the presented adversarial-based method in one-shot federated scenarios, a series of experiments were performed on two datasets for image classification tasks: MNIST and CIFAR-10.

The results of the presented approach are compared to three algorithms: classic federated learning (FedAvg), and two ensembles – an ensemble of random models and a cross-validation ensemble.

In all experiments, only 10% of models were included in the ensemble, making the number of models per ensemble equal to 5. For cross-validation, during the local training part, 5 validation folds were performed, with train/validation data division being 80/20 respectively. The validation performance during cross-validation was collected. After that, the model was trained once again with the whole local data and sent back to the server together with validation metrics.

A. Data partition

Each of the experiments featured 50 federated clients that had approximately 400 images from the training dataset as their local data. The local datasets were created in a non-intersecting manner. Moreover, all classes inside the local dataset had the same number of instances (locally balanced dataset). The local training round consisted of 15 epochs, with the optimizer set to Adam, with starting learning rate of 0.001. For the test performance estimation, test sets provided by the datasets were used.

As was mentioned in section I, non-IID data can create additional challenges to federated pipelines. Therefore, during the experiments, both IID and non-IID data partition scenarios were examined. In the case of IID data, all classes were equally presented in the local datasets of clients. The label-skewed non-IID data partition was emulated by constructing each client's local dataset only from the limited number of classes. In both MNIST and CIFAR-10 experiments, the number of unique classes presented in the local dataset was limited to 4. So, before starting the FL pipeline, each client's class set was constructed individually by sampling 4 classes from the set of all classes. For each class, a probability of its occurrence in the local dataset is drawn from a normal distribution. The example of probability distribution used for CIFAR-10 experiments is showed in Figure 1.

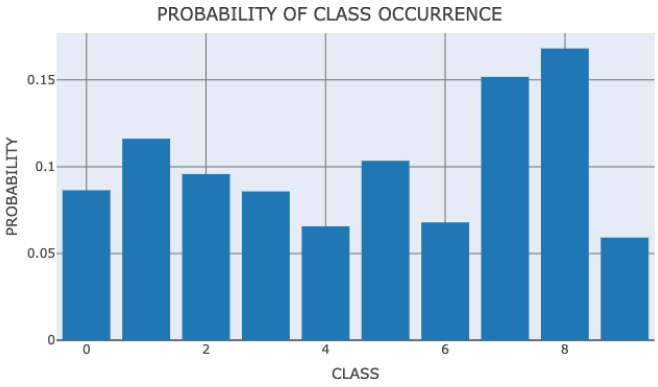

Fig. 1. Class occurrence probability for CIFAR-10 label skew experiments

Due to the custom distribution of the probabilities of class occurrence, some classes were less represented globally across client devices, while others – more.

B. Model configuration

For the MNIST classification task, a simple LeNet5 [22] configuration was used for all the experiments. As for the CIFAR-10, a custom configuration of a Convolution Neural Network was implemented, featuring 6 convolution layers, each pair followed by a maximum pooling layer, at the end followed by three dense layers.

C. Results

All experiments were performed at least 15 times to better capture the statistical significance of the results. As the training

process consists only of one epoch, the final performance of the models was summarised across multiple runs. For the MNIST dataset and IID data partition, the result is showed in Figure 2.

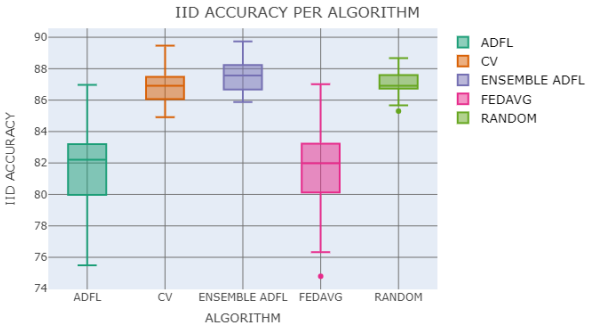

Fig. 2. Test accuracy comparison for IID MNIST experiment

It is seen, that, in general, all algorithms managed to get more than 80% of accuracy, while centralized models on the MNIST dataset reach up to 98% of accuracy. Still, some algorithms scored higher than others: ensembles of models performed better than aggregated global models – with median accuracy for AdFL, cross-validation and random ensembles being 87.6%, 86.9%, and 86.9% respectively, while aggregated FedAvg and AdFL achieved 82% and 82.2%, respectively.

In contrast, the non-IID scenario shows a different behavior, as showed in Figure 3 in addition to a way smaller resulting accuracy across all algorithms.

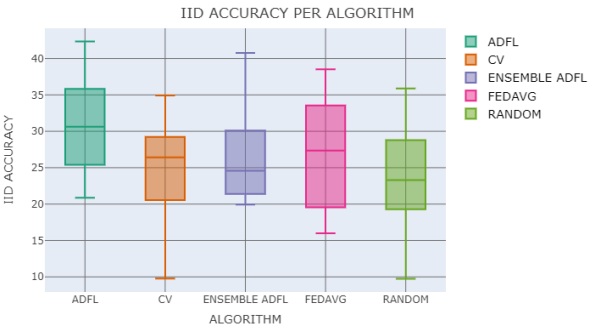

Fig. 3. Test accuracy comparison for non-IID MNIST experiment

Here, aggregated versions of the models perform better than the ensembles, with AdFL and FedAvg reaching 30.6% and 27.4%, and AdFL, cross-validation, and random ensembles reaching 24.6%, 26.4%, 23.3%, respectively.

This difference in behavior on varying datasets depending on either presence or absence of non-IIDness, may come from the fact that individual model evaluation cannot spot the non-IID clients. Therefore, it is not guaranteed that models which were exposed to heterogeneous data during training will appear in the ensemble.

To identify if this behavior can be replicated, the same experiment was performed on the CIFAR-10 dataset. The results for the IID data partition are showed in Figure 4.

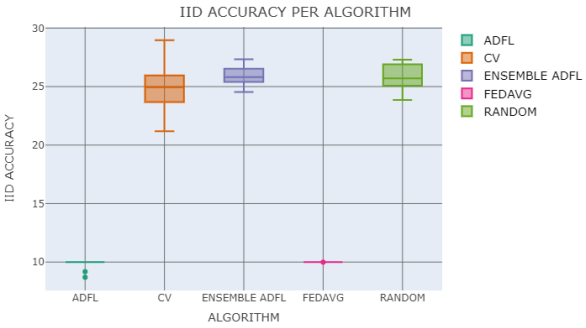

Fig. 4. Test accuracy comparison for IID CIFAR-10 experiment

In this case, again, ensemble versions perform better than aggregated models in the presence of IID data. AdFL, cross-validation, and random ensembles reached 25.8%, 24.9%, and 25.6%, respectively, while aggregated versions could not manage to achieve any meaningful results in the provided scenarios.

However, when examining the results for the CIFAR-10 non-IID scenario (Figure 5), the results differ from those observed on the MNIST dataset with non-IID data.

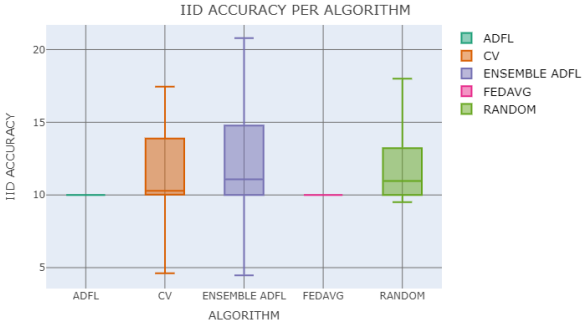

Fig. 5. Test accuracy comparison for non-IID CIFAR-10 experiment

In the CIFAR-10 non-IID scenario, contrary to MNIST, aggregated models (AdFL and FedAvg) still could not get any meaningful performance, while ensemble methods showed low, but, somehow diverse across experiments, accuracy with AdFL, cross-validation, and random ensembles achieving 11.1%, 10.3%, and 10.9% median accuracy, respectively. Although median accuracy is low, maximum accuracy for AdFL, cross-validation and random ensemble reached 20.8%, 17.5%, and 18%. This low performance may be a sign that the proposed task was overly complex and, therefore, may need more experiments with bigger local datasets or require some knowledge transfer techniques.

VI. Conclusion

In this work, a new approach to building a model ensemble for one-shot federated learning was introduced and compared with other ensembling techniques for both IID and non-IID scenarios. It was observed, that, for the MNIST dataset, in the presence of IID data, presented ensembling techniques achieve

better performance than the aggregated models, but for non-IID data the situation is opposite. For the CIFAR-10 dataset, the studied non-IID scenario presented a complicated scenario and did not replicate the results of the MNIST dataset. Still, the described technique utilizing adversarial data shows similar or better performance when compared to other algorithms with respect to test accuracy for both MNIST and CIFAR-10 image classification tasks. Further research may inspect other non-IID data scenarios, use more sophisticated model architectures and datasets, and improve the ensemble construction technique.

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. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[2] O. Shahid, S. Pouriyeh, R. M. Parizi, Q. Z. Sheng, G. Srivastava, and L. Zhao, "Communication efficiency in federated learning: Achievements and challenges," 2021.

[3] M. Chen, H. V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2457–2471, 2021. doi: 10.1109/TWC.2020.3042530

[4] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2021. doi: 10.1109/TWC.2020.3037554

[5] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication-efficient federated learning," *Proceedings of the National Academy of Sciences*, vol. 118, no. 17, p. e2024789118, 2021. doi: 10.1073/pnas.2024789118. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.2024789118

[6] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, p. 594–611, apr 2006. doi: 10.1109/TPAMI.2006.79. [Online]. Available: https://doi.org/10.1109/TPAMI.2006.79

[7] N. Guha, A. Talwalkar, and V. Smith, "One-shot federated learning," 2019.

[8] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, vol. abs/1806.00582, 2018. [Online]. Available: http://arxiv.org/abs/1806.00582

[9] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *CoRR*, vol. abs/2106.06843, 2021. [Online]. Available: https://arxiv.org/abs/2106.06843

[10] C. Xiao and S. Wang, "An experimental study of class imbalance in federated learning," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, dec 2021. doi: 10.1109/ssci50451.2021.9660072. [Online]. Available: https://doi.org/10.1109\%2Fssci50451.2021.9660072

[11] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu, "Distilled one-shot federated learning," 2021.

[12] L. Pinheiro Cinelli, M. Araújo Marins, E. A. Barros da Silva, and S. Lima Netto, *Variational Autoencoder*. Cham: Springer International Publishing, 2021, pp. 111–149. ISBN 978-3-030-70679-1. [Online]. Available: https://doi.org/10.1007/978-3-030-70679-1_5

[13] C. E. Heinbaugh, E. Luz-Ricca, and H. Shao, "Data-free one-shot federated learning under very high statistical heterogeneity," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=_hb4vM3jspB

[14] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, 2020. doi: https://doi.org/10.1016/j.eng.2019.12.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S209580991930503X

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014. [Online]. Available: https://arxiv.org/abs/1412.6572

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

[17] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," 2018.

[18] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017. doi: 10.1109/SP.2017.49 pp. 39–57.

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.

[20] 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. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/suciu

[21] A. Danilenka, "Mitigating the effects of non-iid data in federated learning with a self-adversarial balancing method," 2023, submitted to publication.

[22] 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