Class-Score aggregation vs. Weight-aggregation in Non-iid Federated Learning

Wanru Zhao
Department of Computer Science and Technology
University of Cambridge
wz341@cam.ac.uk

Ziyi Liu
Department of Computer Science and Technology
University of Cambridge
zl413@cam.ac.uk

Chih-Yuan Peng
Department of Computer Science and Technology
University of Cambridge
cyp25@cam.ac.uk

Abstract

In real-world Federated learning (FL) settings, data distribution among clients is rarely iid. This premise poses a strong challenge to traditional FL approaches. Among many solutions proposed to tackle such data heterogeneity, we provide a straightforward comparison of a few popular approaches, including FedDS and FedPer which use model-weight aggregation, and compare their performance to FedMD which uses class-score aggregation under different degree of non-iid data. In the end, we further expand each of the methods with their own individual experiments and provide a qualitative explanation of their effectiveness.

1 Introduction

Weight aggregation is the most popular technique in federated learning (FL) used by FedAvg [1], FedProx [2], and many other strategies [3][4], whereas class-score aggregation is mainly used in FedMD [5]. In weight aggregation, the parameter server uses the aggregated weights of clients' models to update the global model and returns the global model back to each client. In class-score aggregation, instead of sending back the weights, clients send their training class scores back to the server for aggregation, and optimize their local models toward the aggregated score (consensus).
In terms of privacy, we believe that class-score aggregation is more robust compared to weight aggregation. According to [6], membership inference attack infers whether a particular data point is in the training set and can be performed in two different ways, black-box and white-box, where the black-box approach performs the attack by observing the model prediction, while the white-box approach uses weight gradients collected from different layers. Hence, class-score aggregation are immune to the more invasive white-box attacks. Moreover, based on the results of [7] and [8], if the local client models are well generalised and their training data are iid, then the black-box attack is not effective. However, the white-box attack is very effective against methods that use weight aggregation. The results in [6] show that when running FL on DenseNet, the current best model on CIFAR100, the membership inference accuracy can achieve 72.2% by only observing the aggregated weights sent to clients from the parameter server.

The code repository can be found on GitHub, https://github.com/brady19990517/L46.

Besides privacy, performance is also an important factor to consider when selecting a FL strategy. However, we find little accuracy benchmarking between weight-aggregation and class-score aggregation techniques. Therefore, in this project, our main contribution is to directly compare the performance between these two methods, especially under the real-world non-iid settings. We select two methods, FedDS [9] and FedPer [10], that use weight-aggregation technique and compare their performances with FedMD which uses class-score aggregation. We reimplemented FedDS and FedPer using the Flower framework [11] and then evaluate performances on all 3 methods with CIFAR-10 training data under different degree of non-iid.
The following sections are organised as follows. In section 2, we present some related work on the data-sharing strategies, personalisation layers, knowledge distillation and different kind of personalisation techniques in FL. In section 3, we introduce the design motivation and our own improved implementation of each of the three methods. In section 4, we present the evaluation results on non-iid data along with individual experiments on each approach, and end this paper with future work and conclusions.

2 Related Work

Data heterogeneity Also referred to as statistical heterogeneity, data heterogeneity implies the data on user devices may have varied probability distributions (non-iid) [12]. Numerous factors might account for this difference, including the geographical distribution of devices (differing time zones, events), or user behaviour, etc.

Model heterogeneity arises from situations where different clients need models specifically customised to their environment. As a consequence, the model architectures from different local models exhibit various shapes, making it impossible to perform naive aggregation by traditional federated learning [13]. In this case, the challenge of model heterogeneity becomes one of enabling a deep network to comprehend the knowledge of others without requiring the sharing of data or model specifics.

Data sharing [9] is a subset of the data-based approaches [14] which passes the data or metadata between client and server to solve the problem of data heterogeneity. Naoya *et al.* [15] uses a similar concept of data sharing to deal with non-iid data, where it allows a subset of client data to be uploaded to the server, and the server trains the shared model using the combined clients' data and aggregates the trained shared model with the locally trained client models. Data augmentation uses a different idea, where it generates new data to mitigate the non-iid situation instead of passing data around. Duan *et al.* [16] makes each client send the number of samples of each class C to the server, and the server calculates the mean number of C as C' from all clients and sends C' back. The clients then use C' to augment their local data to make them more iid. Shin *et al.* [17] proposed sending each client's encoded data samples to the server, then the server decodes and augments all client's data samples using the server's base samples until the data are iid. Using this reconstructed iid data, the server trains a global shared model and send it to clients. Furthermore, in federated GAN. Instead of augmenting the data directly, the server trains a shared GAN using the data information from clients and sends it back to all clients for iid dataset generation.

Personalisation Layer When models shared across all clients are structurally identical as multi-layer neural networks, combining shared based layers with private personalised layers can help counter the effects of statistical heterogeneity [18]. In the seminal work [10], `FedPer` trains the full model in one client-side update but only aggregates the base layers as the subset of all the parameters in the model on the server. Similar methods can be seen in `FedRep` [19] and `Partial Fed` [20].

Knowledge Distillation (KD) [21][22] is to compress deep and wide networks into shallower ones, where the compressed model mimics the function learned by the complex model. The main idea of KD-based approaches is to shift knowledge from a large teacher model into a smaller student model by learning the class distributions output via softmax,as shown in fig. **??**. The idea of KD has been extended to FL to tackle user heterogeneity, by treating each user model as the teacher, whose information is aggregated into the student (global) model to improve its generalization performance.

3 Methods

3.1 Data Sharing with Weight Aggregation (FedDS)

Motivation This method is proposed by [9], which shows that the naive weight aggregation technique, FedAvg, has significantly reduced accuracy on highly skewed non-iid data. The paper also demonstrates that weight divergence is the reason for this accuracy drop, shown by the differences of weights between centralised trained SGD and federated trained (either iid or non-iid). The results show that FL with iid data has little weight divergence and thus little to no accuracy drop compared to centralised SGD, whereas extreme non-iid data where each client has data from only one class has the largest weight divergence. The weight divergence can be explained by the earth mover's distance (EMD), which is the distance between the client data distribution and the population distribution - the larger the EMD, the larger the weight divergence and the lower the accuracy. The paper reduces the EMD by proposing a data sharing strategy which we reimplement using the Flower framework. We also evaluate the strategy with different `iid_fraction` parameters built inside Flower.

Method The main idea of the data-sharing strategy is that we have a global dataset on the server. This global dataset is iid and divided into the number of clients partitions. Each global partition G has the same number of samples. Besides the global dataset, we also have a local dataset where the global-to-local ratio is 2 to 8. In terms of the CIFAR-10 dataset, the global and local datasets have 10000 and 40000 data points respectively. The local dataset is also divided into the number of clients partitions but with different `iid_fractions`. Each local partition is represented as D. Next, a warmup model is trained on $|D| \times \beta$ instances in G. The parameter β can be configured to evaluate the tradeoff between the test accuracy and the amount of G needed. The larger β is, the more global data we have for the warmup model and the higher the accuracy we can achieve when dealing with non-iid data. Then, the warmup model is sent to each client for local training together with a α portion of G. Each client then start to train its warmup model with data consisting of its local data and the received subset of global data. The locally trained model's weights are sent back to the server and aggregated using the FedAvg strategy. The data-sharing strategy is illustrated in fig.1.

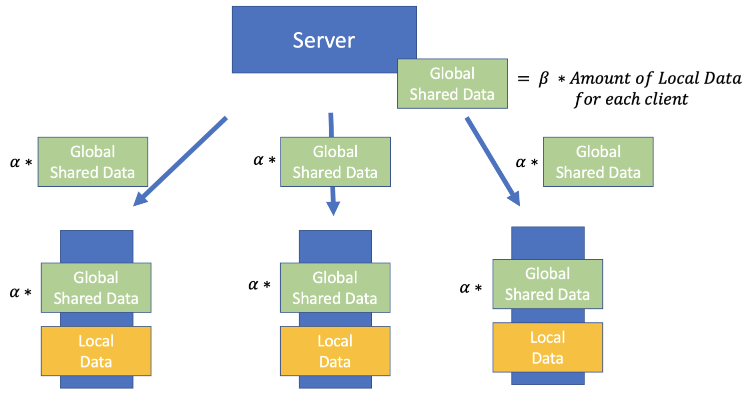


Figure 1: The internal working of the data-sharing strategy (FedDS). The green box represents the global data for each client, whereas the orange box represents the local data for each client.

3.2 Partial Weight Aggregation (FedPer)

Motivation Non-iid data, as discussed above, can induce a strong bias given a large number of clients and an essentially identical model for all clients. In many scenarios, one of the final objective is to achieve high scores in the performance metrics we employ on each client rather than on an artificially collaged central dataset. We therefore raise the question as to whether the aggregation methods that includes all model weights are capable of differentiating the varying data distributions among clients. Another consideration is security implication, where it has been shown that partial client data can be reconstructed with the client model weights alone, contradicting one of the fundamental purpose of FL. Therefore, partial weight-aggregation with personalisation layer [10] is introduced to address these two issues. The intuition behind this approach is 1) that the personalised layer is different for each client to capture different data distributions, and 2) that when only a fraction of model weights are shared, the possibility of data retrieval based on the shared parameters alone is very low.

Method Unlike the full-weight aggregation method in FedDS, we do not keep a manually constructed iid dataset or model on the server, nor is there any central training before the federated procedures. Instead, we assume all collected local data are only visible to each respective client, as we would expect from the real-word scenario out of privacy concerns. Each client has the same model structure, denoted as the `Full Model` composed of two parts - `Base Model` and `Personalised Model`. The architecture is illustrated in fig. 2. In each training iteration for a client, the `Full Model` is trained on the batched local data; however, not all weights are shared to be averaged with those from other clients - only `Base Model` weights are sent to the server and sent back after aggregation.

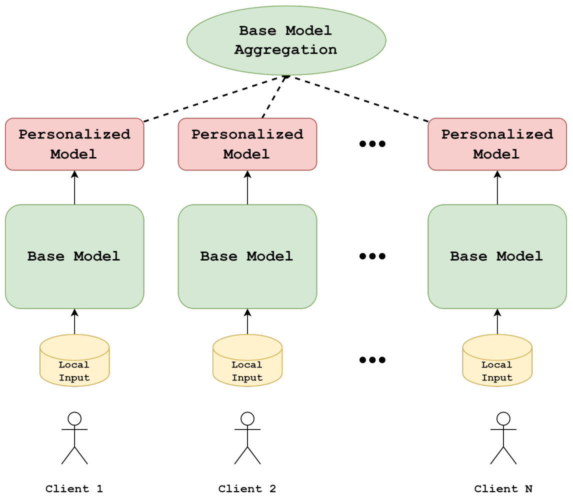


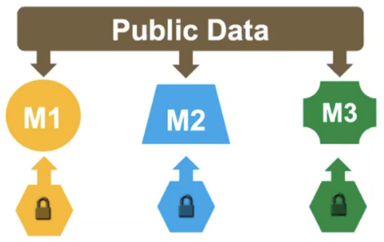
Figure 2: Diagram of Standard FedPer mechanism. `BaseModel` is shared among all clients and to be aggregated at the end of each global round, and each client has its own `Personalised Model` visible only to itself.

Typically, we can set the convolutional layers as our base model to be shared with server, while the stacked linear layers at the top can be personalised and private to the client itself, which was exactly the assumption made by this paper [10]. However, one innovation in our methods is that we interpret the `Base Model` more loosely, such that it can be any part and fraction of the model weights to be shared. Thus, we can consider the full-weight-aggregation outlined by FedDS as the extreme case in our implementation when no personalized layers are allowed. The other extreme would be fully personalised layers - that is, no FL applied at all. We will compare the performances with different degree of personalisation in section 4.

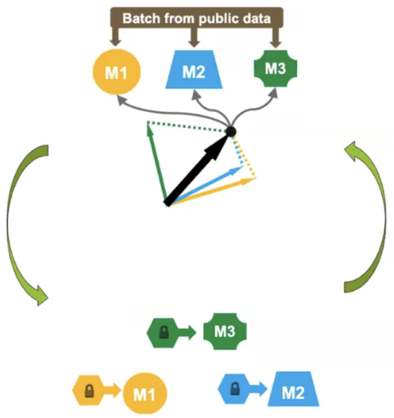
3.3 Data Sharing with Knowledge Aggregation (FedMD)

Motivation In the original federated framework, all users have to agree on the particular architecture of a centralised model. In this work, we instead explore extensions to the federated framework that is realistic in a business-facing setting, where each participant has the capacity and desire to design their own unique model. This question is intimately related to the non-iid challenge of federated learning because a natural way to tackle statistical heterogeneity is to have individualised models for each user. Existing frameworks result in sightly different models [23][24], but none, to our knowledge, delegates the model design to local clients. Moreover, transferring a small number of logits and consensus results each round significantly saves resources used in client-server communication and reduces susceptibility to attacks.

Method FedMD [5] is a federated learning framework based on knowledge distillation and transfer learning that allows clients to independently design their own networks using their local private



(a) Phase 1: Transfer learning. Each lock refers to private dataset of each client.



(b) Phase 2: Communication. The models communicate and align their logits(shown as tensors in different colors) computed from public data without applying the softmax activation layer.

Figure 3: Diagram of FedMD mechanism.

datasets and a global public dataset. First they are trained on the public dataset until convergence. Following the pretraining, each participant trains its model on its own small private dataset, as shown in fig. 3a. The clients then go through the collaborative training phase shown in fig. 3b, during which the models acquire strong and fast improvements across the board, and quickly outperform the baseline of transfer learning. Each party computes the class scores on the public dataset, and transmits the result to a central server. The server computes an updated consensus, and then each party downloads the updated consensus. Finally, each client trains its model to approach the consensus on the public dataset, as well as on its own private data for a few epochs.

4 Experiments and Results

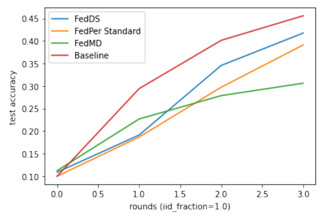
Model and Configurations

Due to very limited training resources, we conduct the experiments with a small 2-layer + 3-layer stacked linear-layer head with a total trainable parameter size of 62,006. And in order to compare the results of different aggregation approaches in a controlled manner, a collection of training configurations was set for all experiments. We use 3 global rounds and 10 local epochs on 10 clients, where we require all clients to participate. For consistency, SGD optimizer with learning rate 0.001 and momentum 0.9 is applied to all training procedures. Cutting off at 30 rounds in total instead of training till convergence is also pertinent to the real-life scenario where the local client-side training resources are limited. The `iid_fractions` are chosen from the range of $[0.1, 1.0]$, where 1.0 corresponds to iid data, whereas 0.1 is the most non-iid setting.

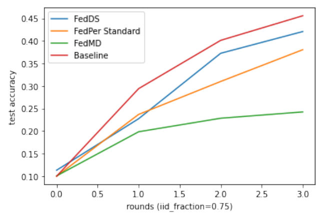
In section 4.1, we present the seminal result that compares the three main approaches described above with the baseline implementation under 6 different `iid_fractions` and provide a qualitative analysis on the performance disparity. We then expand on the individual methods with detailed and tailored experiments in sections 4.2-4.4.

4.1 Comparison of weight-aggregation and class-score aggregation

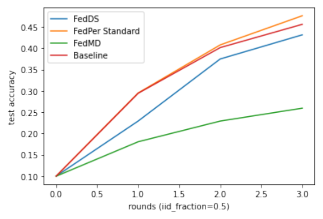
FedAvg is used as the baseline method in performance comparsion. Overall, we see from fig. 4 that methods with weight aggregation techniques (FedDS, FedPer, FedAvg) outperform class-score aggregation ones (FedMD) under every degree of non-iid data since FedMD only share trained results as logits instead of model parameters during each global aggregation round. However, when increasing the number of rounds and epochs during both the transfer learning phase and communication phase,



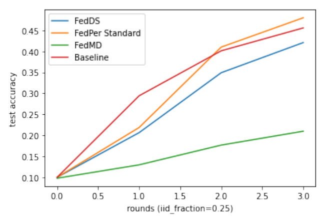
(a) iid_fraction=1.0



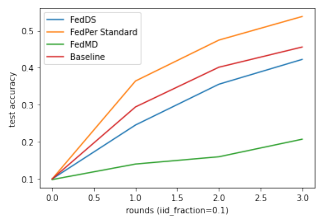
(b) iid_fraction=0.75



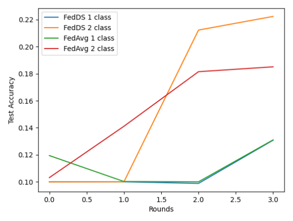
(c) iid_fraction=0.5



(d) iid_fraction=0.25



(e) iid_fraction=0.1



(f) iid_fraction=1 and 2 class

Figure 4: Benchmarking three methods (FedDS, FedPer Standard, FedMD) against the FedAvg Baseline on both iid and non-iid CIFAR-10 data.

FedMD can show its superiority in accuracy, referred to in Section 4.4. In particular, we see that FedAvg has the highest performances when the data is fairly uniformly distributed(iid=1.0, 0.75), followed closely by the other weight-aggregation methods. However, once the clients receive more skewed data, FedPer gradually exceeds the baseline in accuracy due to its capability of capturing unique client-side features despite strong class imbalance, where the impact is mediated by the personalized layers unlike FedAvg. Such advantage is most prominent in `iid_fraction=0.1` scenario (fig. 4e). Interestingly, the performance of FedDS is worse than FedAvg under all degree of non-iid data (iid=1.0 to 0.1). The reason might be that FedDS is designed for dealing with extreme cases of non-iid, where each client only contains 1 or 2 classes. The global dataset in FedDS would not offer much help if the non-iid degree isn't extreme and requires redistribution of data. On the contrary, using global dataset in less extreme non-iid data would actually mean less local data for each client to train on and thus results in lower accuracy. However, when we are in extreme cases (fig. 4f), FedDS performs better than FedAvg and the benefit of using the global dataset (reducing the non-iid degree) outweighs the drawback (less local data).

4.2 Data Sharing with Weight Aggregation (FedDS)

We conducted three experiments with the data sharing technique, and each experiment focuses on discovering the effect each parameter has on the test accuracy. There are three parameters in total, which are the `iid_fraction`, alpha (α), and beta (β).

Firstly, we compare the effect of different degrees of non-iid data have on the test accuracy. The result is shown in fig.5. We tested on 6 different non-iid settings, from `iid_fraction`=1 to `iid_fraction`=0.1. We also consider cases where each client only contains one class (`iid_fraction`=1class) and two classes (`iid_fraction`=2classes). The remaining parameters are unchanged, where β=0.25, and α=0.5. From the result, we see that the test accuracy from `iid_fraction`=1 to 0.25 are similar. However, we see that when each client has only data from two classes and one class, the accuracies drop significantly to approximately 23% and 15% respectively.

Secondly, we compare the effect of different β and α have on the test accuracy. We model both of them under the scenario where `iid_fraction`=1class, since FedDS is designed to deal with this extreme non-iid situation. In fig.6a, we see that when β decreases, the test accuracy also drop, since small β implies there are less global data used for warmup model training. In fig.6b, we see a similar result that the test accuracy decreases as α becomes smaller as smaller α means less global data to decrease the non-iid degree of clients' data.

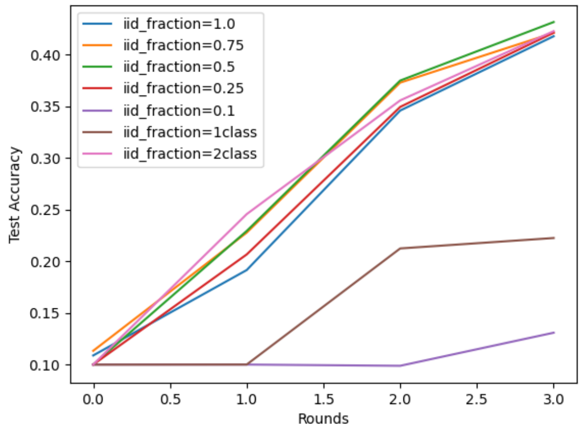
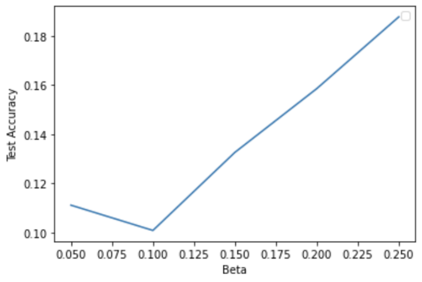
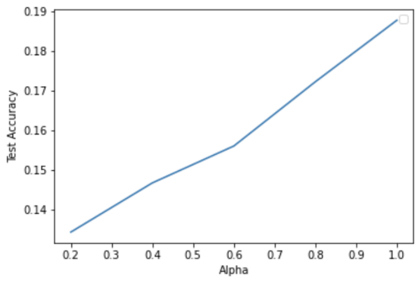


Figure 5: Comparing global model's test accuracy under different degree of non-iid with different iid_fraction using data sharing in FedDS.



(a)



(b)

Figure 6: The relationship between β (left), α (right) and the test accuracy

4.3 Partial Weight Aggregation(FedPer)

The standard `Base Model` consists of two pooled convolutional layers followed by a fully connected linear layers, while the standard `Personalized Model` takes the `Base Model` output as input and pass it to two fully connected layers, leading to prediction outputs. When we later alter the ratio of shared and personalised weight parameters, the `Full Model` is unchanged: we simply move different layers around to preserve the overall model for consistency and comparability.

Firstly, we showcase the effect of the degree of non-iid on our standard personalised model performance (fig. 7). Test accuracy shown in the plot in each round is averaged from all 10 clients to illustrate the trend more clearly (no testing was done centrally as personalisation with non-iid data is more focused on local performance improvement rather than on an arbitrary central dataset). We can observe a general trend that the more "skewed" the distributions are across all clients (smaller `iid_fraction`) , the better the averaged test results. This is a stark contrast from the results obtained from FedDS, but not unexpected. For a client with primarily data of the same class, we would expect the personalised layers unique to the client to be capable of capturing the key characteristics of the dominant class and predict that label by majority.

Secondly, we compare the effect of personalized model size on FL client performances. The `Full Model` is kept unchanged while we vary the number of parameters in personalised layers by including/excluding layers that were used in the `Base Model`. In fig. 8a, we can see the smaller base model(more personalized layers) achieved a better performance, which can explained by its higher capacity for only one client instead of being updated across all clients in the non-iid setting. The same conclusion can be found from fig. 8b, only with closer performances among the 4 experimented models. The reversed model is constructed by swapping the `Base Model` and `Personalized`

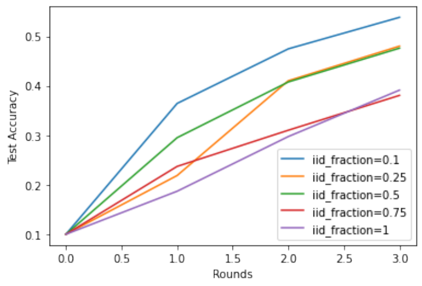
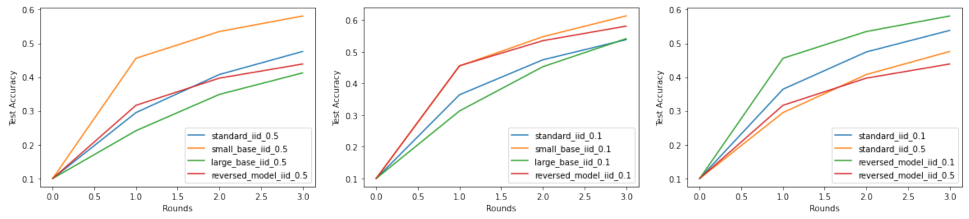


Figure 7: Comparing personalised model performance under different degree of non-iid with different iid_fraction in FedPer.



(a) iid_fraction=0.5 (b) iid_fraction=0.1 (c) standard and reversed models

Figure 8: Comparing personalised model performance under different degree of non-iid with a variety of model ratios in FedPer.

`Model` layers in the standard model, and their effects under different iid_fraction assumptions were shown in fig. 8c, where the reversed model presents a better performance in a more imbalanced setting than the standard model.

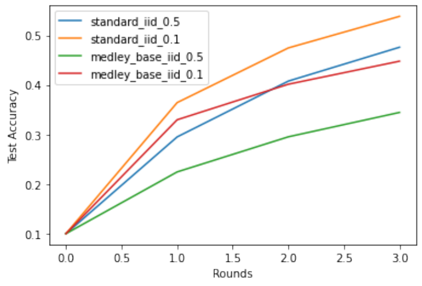


Figure 9: Comparing personalised model performance under different degree of non-iid with standard model and medley model in FedPer.

Finally, a mixed-personalisation model was constructed by alternating between Base and Personal layers.Shown in fig. 9, the `Medley Model` has a worse performance in both iid_fraction settings. With the findings above, we can conclude that the locality of personalisation has a strong influence on client-side performance.

4.4 Data Sharing with Knowledge Aggregation(FedMD)

Firstly, same as the above two methods, we compare the effect of different degrees of non-iid data have on the test accuracy. We use Adam optimizer with an initial learning rate of 0.001. The result is shown in fig.10. We can see from the figure that the test accuracy of `iid_fraction`=1 is the highest. The test accuracy of `iid_fraction`=0.75 and 0.5 are similar, and the test accuracy of `iid_fraction`=0.25 and 0.1 are similar, respectively. Overall, in this experiment, the closer the value of Non-iid is to 1, the more accurate it is, and the closer it is to 0, the less accurate it is.

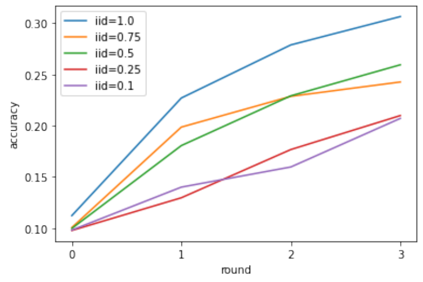


Figure 10: Comparing personalised model performance under different degree of non-iid with different iid_fraction in FedMD.

Secondly, in order to validate the method's capability of independently designing models, each of 10 participants designs a unique convolution network that differs by number of channels and number of layers. The public dataset is the CIFAR10 and the private dataset is a subset of the CIFAR100, which has 100 subclasses that falls under 20 superclasses, e.g. bear, leopard, lion, tiger and wolf belongs to large carnivores. The data are also non-iid, which means each participant has data from one subclass per superclass at training time, and during testing, participants are required to classify generic test data into the correct superclasses. For example, a participant who has only seen wolfs during training is expected to classify lions correctly as large carnivores. Therefore it has to rely on information communicated by other participants. We can see from fig. 11 that FedMD improves the test accuracy of participating models beyond their baselines (round 0), and different models shows various increasing trend in accuracy but not necessarily for the better. This can be further resolved with more carefully customised model designs in future work.

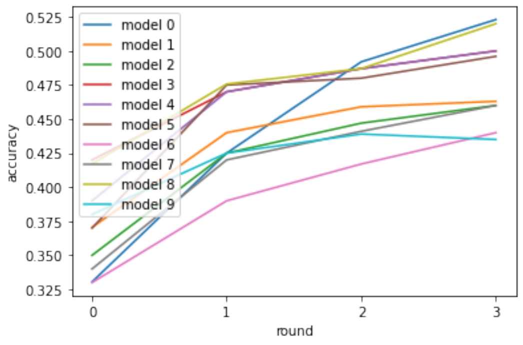


Figure 11: FedMD improves the test accuracy of participating models.

In conclusion, it is a framework that enables federated learning for independently designed models, based on knowledge distillation and is tested to work on various tasks and datasets.

5 Conclusions

Enabled by the flexible setting of server-client configurations in Flower framework, it is much easier to conduct quantitative experiments with easy evaluation on both sides in a more controllable and realistic environment on FedDS and FedPer that requires large-scale aggregation each round.

Regarding the results of our experiments, we found that methods with weight-aggregation techniques (FedDS, FedPer) outperforms those with class-score aggregation techniques (FedMD) under limited resources. Moreover, we found that FedAvg performs better than those with data-sharing (FedDS) and personalisation strategies (FedPer) when data is less non-iid, and worse when degree of non-iid increases, demonstrating the efficacy of the two approaches for data heterogeneity. Notably, FedMD shows its superiority by enabling each agent to own not only their private data, but also uniquely designed models.

References

[1] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[2] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018.

[3] Tian Li, Maziar Sanjabi, and Virginia Smith. Fair resource allocation in federated learning. *CoRR*, abs/1905.10497, 2019.

[4] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[5] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[6] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019.

[7] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2017.

[8] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting, 2018.

[9] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018.

[10] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers, 2019.

[11] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly federated learning research framework. *CoRR*, abs/2007.14390, 2020.

[12] Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*, 2018.

[13] Qiong Wu, Kaiwen He, and Xu Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society*, 1:35–44, 2020.

[14] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *CoRR*, abs/2106.06843, 2021.

[15] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani. Hybrid-fl: Cooperative learning mechanism using non-iid data in wireless networks. *CoRR*, abs/1905.07210, 2019.

[16] Moming Duan. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. *CoRR*, abs/1907.01132, 2019.

[17] Myungjae Shin, Chihoon Hwang, Joongheon Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. XOR mixup: Privacy-preserving data augmentation for one-shot federated learning. *CoRR*, abs/2006.05148, 2020.

[18] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning, 2020.

[19] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning, 2021.

[20] Benyuan Sun, Hongxing Huo, Yi Yang, and Bo Bai. Partialfed: Cross-domain personalized federated learning via partial initialization. *Advances in Neural Information Processing Systems*, 34, 2021.

[21] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[22] Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *arXiv preprint arXiv:1312.6184*, 2013.

[23] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017.

[24] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717*, 2019.