Tackling the Impact of System and Data Heterogeneity on the Global and Local Accuracy of Federated Learning

Alexandru-Andrei Iacob
Computer Laboratory
University of Cambridge
aai30@cam.ac.uk

Abstract

Federated Learning is a form of distributed Machine Learning working over large numbers of resource constrained devices. It attempts to reduce communication costs by alternating local (on-device) training with global (server-side) aggregation of model parameters without ever directly sharing client data. While Federated Learning was initially proposed as a way of training an accurate global model, recent work has shifted towards optimising the accuracy of both the local model of a given client and the global model. However, the high degree of statistical heterogeneity between the data partitions of each client, together with differences in client system specifications, make the simultaneous balancing of these two goals difficult. This research proposal reviews the recent work in the field and suggests ways of handling data and system heterogeneity by leveraging advances in Federated Learning and Neural Architecture Search. This can be achieved by applying model adaptation techniques—such as clustering—for clients with heterogeneous data, and optimizing the amount of local computation based on client resources through reinforcement learning.

1 Introduction

The canonical Federated Learning (FL) problem was introduced by McMahan et al. [2017]. The optimization objective of FL was formalised by Li et al. [2018] as follows:

$$min_w f(w) = \sum_{k=1}^{n} p_k F_k(w),$$

where f is the objective function of the global model, formed as a weighted sum of the local client objective functions F_k weighed by factors p_k. Multiple versions of FL have since been devised (see Kairouz et al. [2019]). This proposal focuses on the standard FL: the setting where a centralised server coordinates training rounds.

The primary challenges which differentiate FL from other distributed Machine Learning methodologies are data and system heterogeneity.

Data Heterogeneity: Devices may hold data partitions which are highly dissimilar from the global distribution. Data heterogeneity encompasses multiple potential divergences which depend on the features, labels, and quantity of data. For an overview, see Kairouz et al. [2019, sec. 3].

System Heterogeneity: Devices operating within the federated network are unlikely to have uniform hardware specifications or reliable internet connections. As such, clients may disconnect during the training process. Abdelmoniem et al. [2021] provide an investigation of the impact of system heterogeneity on FL performance.

Yu et al. [2020], Kulkarni et al. [2020], and, Mansour et al. [2020b] have challenged the canonical objective of training a shared global model. As indicated by Yu et al. [2020], global models perform significantly worse for clients with unusual data and/or system characteristics. Consequently, optimizing solely for the average global case means that such clients receive negligible benefit—outside of time efficiency—from participating in the Federated Learning process compared to training a local model directly.

This local–global accuracy trade-off interacts with both data and system heterogeneity as well as the architecture of the model being trained.

2 Proposal

The primary goal is to tackle the local–global trade-off while accounting for data and system heterogeneity. This can be achieved via a combination of methods, each of which handles a different part of the problem and could be applied independently.

2.1 System Heterogeneity

Li et al. [2018] introduces FedProx, an aggregation algorithm capable of handling system heterogeneity by allowing for a varying amount of local training to be completed. However, the number of local updates must be specified as no means of automatically determining the amount of local computation to be performed is provided.

I propose a system for predicting local hyper-parameters, such as the number of epochs (times a model is trained over the data), based on the model architecture and client system characteristics. The system would encode the model architecture using a learnt neural representation, or embedding, such as arch2vec [Yan et al., 2020] from the field of Neural Architecture Search. Client system characteristics would be initially summarised using a hand-crafted feature vector. In order to train the system to predict values for a given hyper-parameter, a reinforcement learning algorithm—such as the one presented by Williams [1992]—would be applied during training with each client as an agent capable of learning parameter settings from experience. This design is necessary to make the problem computationally tractable.

Federated Learning has a substantial training time and using such an embedding is crucial to allow transfer between architectures to avoid re-training on each individual model. Furthermore, using the accuracy from each training round rather than the final model accuracy alone—as other optimization methodologies like genetic algorithms do—is necessary for efficiency and is the primary motivation for applying reinforcement learning. It deserves mentioning that neural architecture representation is novel and therefore not tested extensively.

2.2 Data Heterogeneity

While the above method handles system heterogeneity, it does not directly address data heterogeneity. In fact, the system may fail to provide good local parameters if the data distribution of a client differs significantly from those used in training. Consequently, training must be done alongside a technique that mitigates the impact of data heterogeneity. The next section explores some potential solutions.

2.2.1 Clustering

Grouping, or clustering, clients with similar data and systems around a mutual server can help the convergence of aggregate models by providing a balance between local versus cluster accuracy. The version of clustering which is of interest in FL is that proposed by Mansour et al. [2020b], which groups clients by minimising a loss function. This means that clients requiring similar models end up on the same cluster. The work of Mansour et al. [2020b] can be developed through better initialization using a procedure similar to Arthur and Vassilvitskii [2006]'s, or by applying one of the methods presented by Yuan and Yang [2019]. In order for a clustered algorithm to interoperate with the local computation optimizer, a version of the optimizer would have to be trained on each cluster. Consequently, the number of clusters should be fairly low.

It is noteworthy that once clustering is applied, a single global model ceases to exist. In order to use clustering in situations where a global model is necessary, we require a way to aggregate the models trained for each cluster. A standard ensemble technique could be applied [Dong et al., 2020], however, I favour using an iterative voting method for categorical tasks [Meir, 2017]. Iterative voting is a new development in the field of Computational Social Choice [Brandt et al., 2016] where voting proceeds in rounds and voters may change their preferred output. Previous investigations conducted during my undergraduate research work indicate that this method of voting may improve compromise within a group of agents. If applied, it would represent one of the first uses of iterative voting in a practical setting.

3 Related Work

The work of McMahan et al. [2017] directly poses some of the challenges tackled by this proposal. The authors justified the need for Federated Learning on the basis of communication efficiency and privacy by following the principles of focused collection and data minimisation outlined in White House [2013]. While privacy concerns in FL are outside the scope of this proposal, the curious reader may investigate differential privacy [McMahan et al., 2018, Wei et al., 2020], secure aggregation [Segal et al., 2017], or the overview provided by Kairouz et al. [2019, sec. 4].

While their update-averaging algorithm—FedAvg (A.1)—proved resilient towards data heterogeneity when training a global model in practice, it did not offer a means of handling system heterogeneity directly, nor did it provide strong bounds on the impact of highly unbalanced data distributions, resulting in potential scenarios with large performance degradation.

Since the publication of McMahan et al. [2017], a series of methods for enhancing the usefulness of FL in highly heterogeneous scenarios have been proposed. Furthermore, solutions tackling the previously mentioned local–global accuracy trade-off are novel and warrant being addressed independently—although they do occasionally address statistical heterogeneity as well.

3.1 Tackling Heterogeneity When Training a Global Model

Zhao et al. [2018] provide a bound on the impact of a skewed data distribution on model accuracy and attempt to rectify its effects by using a small, globally shared data pool, deviating from the goal of Federated Learning. Smith et al. [2018] tackle both data and system heterogeneity through MOCHA, a federated multi-task learning framework which attempts to construct related models for each device. However, their work is incapable of handling non-convex deep learning models.

From a pure system heterogeneity perspective, there have been two dominant directions: decreasing computational, communication and memory costs, thus allowing more clients to participate; or allowing for clients to provide varying contributions to the global model and to carry out different amounts of local computation.

While reducing communication and computational costs has been a focus from the start (see the structured and sketched updates introduced by Konečný et al. [2017]), it was intended to improve overall system efficiency given slow internet connections rather than to allow clients with worse specifications to participate. Caldas et al. [2018] reduce communication costs by applying compression to both server-client and client-server communication and reduce local computation costs via Federated Dropout—a technique which allows each device to locally operate on a smaller sub-model of the overall architecture—with the goal of allowing a wider array of client systems to participate in training. These methods are generic and do not necessitate large changes to the overall aggregation algorithm, however, nor do they address the root cause of the issue: they do not distinguish between the client system specifications. As such, they are complementary to all methods proposed in this work.

Of higher interest is the aforementioned FedProx [Li et al., 2018], which operates similarly to FedAvg while also accounting for system heterogeneity by allowing different amounts of work to be completed across clients. Consequently, limiting how far each client model can diverge from the global via a "proximal term" is necessary to obtain convergence for highly heterogeneous scenarios where some clients may perform significantly less work than average. FedProx illustrates some of the challenges of FL aggregation algorithms which have inspired this proposal as it does not define a standard means of deciding how much work a client should do or how far a client model should diverge. Furthermore, it handles neither data heterogeneity nor local–global trade-off.

give equation

3.2 Tackling local–global Accuracy Trade-off

As discussed, the lower accuracy that underrepresented clients receive is inherently tied to either their statistical heterogeneity (they have unusual data), system heterogeneity (they do not get to participate in training as much because of worse hardware), or both. One of the early systems, FedPer, which attempts to address this issue was conceived by Arivazhagan et al. [2019]. FedPer allows all clients to train a set of shared base layers in the Neural Network, while individual personalisation is achieved via one or more top layers being trained for each client separately. Such layers help in situations with high statistical heterogeneity while reducing variation between client accuracies.

Multiple other methods of personalisation, or local-adaptation, have been developed and applied since. These have been covered and experimentally validated in the work of Yu et al. [2020]. They range from a similar personalisation-layer setup Arivazhagan et al. [2019] to knowledge distillation [Hinton et al., 2015,

Gou et al., 2021] and multi-task learning [Kirkpatrick et al., 2017, Zhang and Yang, 2021] approaches. Although such methods are generally applicable, they have two major shortcomings. First, they do not intend to account for system heterogeneity explicitly. Second, they have been extensively tested alongside FedAvg, makeing their interaction with more complex aggregation algorithms such as FedProx unclear.

Most influential towards the proposed system is Mansour et al. [2020a]. Their work focuses on developing efficient personalisation methods. Specifically, they refer to data interpolation, model interpolation, and client clustering. Data interpolation is applied after first training the global model by adapting a client model initialised from the global one to minimise the chosen loss function over a combination of the client data and the global data. Model interpolation refers to training both a local client model alongside the global one. Their clustering personalisation method, HYPCLUSTER (), is particularly well-suited towards a generic FL optimisation system as it does not require large changes to the aggregation algorithm and can be used to tackle both local–global trade-off and data heterogeneity. It works similarly to k-means by constructing a set of clusters, randomly sampling a number of clients and assigning them to the cluster with lowest loss, and finally running an FL algorithm for each cluster thus updating the cluster models. This design makes it ideal for interoperation with any black-box optimisation method as each cluster can be paired with its own optimization process.

add appendix

3.3 Model Aware Parameter Optimisation

While a survey of Reinforcement Learning (RL) [Arulkumaran et al., 2017, Kaelbling et al., 1996]—or hyper-parameter optimisation [Yang and Shami, 2020] in general—and Neural Architecture Search [Elsken et al., 2019, Wistuba et al., 2019, Ren et al., 2020] is beyond the scope of this work, two research directions related to these fields warrant discussion.

To tackle large search spaces and allow for transfer-learning [Zhuang et al., 2021] to occur, recent work in hyper-parameter optimisation has shifted from operating on a case-by-case basis, with complete hand-crafted encodings, towards learnt abstract representations.

This shift is evident when considering device placement optimisation for computation graphs. REGAL [Paliwal et al., 2019] uses a Graph Neural Network (GNN) [Wu et al., 2021] to construct node embeddings. Such embeddings are then inputted into a policy network to predict the starting parameters of a genetic algorithm, which does device placement optimisation, on a per-node basis. For training, it applies the REINFORCE [Williams, 1992] algorithm with the memory consumption of the device placement found by the genetic algorithm as its reward function. Concurrently with REGAL, Placeto [Addanki et al., 2019] use a similar GNN-based embedding and RL algorithm to directly optimise device placement by outputting a node placement based on both the current node's embedding and the device placement of previous nodes.

To adapt such transferable approaches towards parameter optimisation to FL, it is necessary to construct an embedding of the entire model which is being trained. A potential solution comes from the field of Neural Architecture Search (NAS), an area which has seen its own shift from explicit complete encodings to more generaliseable search spaces. Arch2vec [Yan et al., 2020] is a recent attempt at an unsupervised neural architecture representation constructed outside the NAS process, without requiring the usage of accuracies as labels. The authors find that their means of unsupervised encoding is a useful pre-processing step for NAS because it projects similar architectures to similar, compact representations.

4 First-year Outline

The first year of research will be primarily dedicated towards constructing a research prototype (pilot experiment) while investigating any new developments in the field. Notably, my main project for the MPhil in Advanced Computer Science focuses on an empirical examination of the personalisation techniques presented in Yu et al. [2020]. Any relevant findings will therefore be used to inform the first-year research work.

Given the exploratory nature of the work, the following outline will be quite broad and cover the 9-month period between the start of Michaelmas term and the first-year review on the 30th of June.

Months 1 and 2: Dedicated to detailed research on recent findings in the field, planning the experimental design alongside a preliminary estimation of the necessary computational resources to carry out the experimental design. Exploration of alternative avenues and potential improvements with supervisor discussion and feedback.

Months 3 and 4: Prototype implementation, using the Flower [Beutel et al., 2020] Federated Learning framework to orchestrate the experiments. This will result in the delivery of the clustering algorithm to Flower as well as an initial implementation of the federated Reinforcement Learning algorithm and model embeddings. Running the small-scale experiments to gather preliminary data and practically decide the viability of the proposed research direction. If the prototype does not prove viable, a series of alternative optimisation methods, embeddings or different approaches may be considered to replace it (sec.5). It may be possible to begin writing parts of the problem statement, introduction and background sections for the first-year review document.

Months 5 and 6: If the prototype proves practical, the planned experiments will be carried out. Experimental write-up and interpretation for the first-year review will begin after the experiments are finished. According to the results, the overall plan and course for the PhD will be re-evaluated—with input from the supervisor—in preparation for the first-year review.

Months 7 and 8: First-year review document full write-up in parallel with the exploration of potential avenues for developing a full system ready to be used externally and capable of being extended to other research directions. A full draft should be finished within the first weeks of month 8 and sent out to the supervisor for feedback.

Final month and first-year review: The period before the first-year review will be dedicated towards editing and finalising the document while preparing for the oral examination. Based on assessor feedback, all necessary modifications to the overall plan and first-year review document will be incorporated before being sent to the Secretary of the Degree Committee.

5 Research Considerations

Reinforcement Learning Structure: Adapting Reinforcement Learning—initially intended for developing policies for agents that interact with a clearly defined environment—towards this abstract domain is not straightforward. REGAL [Paliwal et al., 2019] receives one reward signal based on the performance of the device placement solution found by the genetic algorithm, while Placeto [Addanki et al., 2019] allows for intermediate rewards. In the context of the proposed system, two different approaches are apparent at the time of writing. One would consider performance on the validation set at the end, whereas the other would re-test performance either after every aggregation round or at set intervals while taking actions to change hyper-parameters during training—a potential necessity to make the problem computationally tractable. An argument could be made for the usefulness of having two reward signals: the first signal from the performance on the cluster-level model validation performance and the second from client-level validation performance using that client's local data. Finally, RL has difficulties with large action spaces, as such, optimising continuous hyper-parameters or trying to handle large numbers of them would require careful consideration of the action space design.

Alternative Optimisation Methods: While Reinforcement Learning was initially considered because of its promising application to device placement, it is not guaranteed to be successful given the large training times—especially if rewards are only registered at the end of the entire training process—and complexities of Federated Learning. Such difficulties would also affect other optimisation methods, e.g. genetic algorithms, relying on a high number of evaluations. One viable alternative would be Bayesian Optimisation [Frazier, 2018] because of its ability to minimise evaluations through the acquisition function for selecting promising candidates. However, it is known that Bayesian Optimisation has difficulties scaling to high-dimensional spaces [Shahriari et al., 2016], which may pose future challenges as FL algorithms grow in terms of hyper-parameter count.

6 Conclusion

As a paradigm meant to allow computation on client devices and to restrict data-sharing as much as possible, Federated Learning distinctly suffers from data and system heterogeneity, as well as a conflict between the goal of training a global model and providing good results for each client's particular characteristics. This proposal has surveyed previous work addressing these issues and proposed several methods which complement one-another—parameter optimisation for local computation, clustering, and iterative-voting based ensembles—and could be used to construct a new FL system which is compatible with previous aggregation algorithms. Such methods are intended as a starting point for PhD research and are meant to evolve over the course of the first year.

References

Ahmed M Abdelmoniem, Chen-Yu Ho, Pantelis Papageorgiou, Muhammad Bilal, and Marco Canini. On the impact of device and behavioral heterogeneity in federated learning. *arXiv preprint arXiv:2102.07500*, 2021.

Ravichandra Addanki, Shaileshh Bojja Venkatakrishnan, Shreyan Gupta, Hongzi Mao, and Mohammad Alizadeh. Placeto: Learning generalizable device placement algorithms for distributed machine learning, 2019.

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

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical Report 2006-13, Stanford InfoLab, June 2006. URL `http://ilpubs.stanford.edu:8090/778/`.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. doi: 10.1109/MSP.2017.2743240.

Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

Sebastian Caldas, Jakub Konečny, H. Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements, 2018.

Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, 2020.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Peter I. Frazier. A tutorial on bayesian optimization, 2018.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, Mar 2021. ISSN 1573-1405. doi: 10.1007/s11263-021-01453-z. URL `http://dx.doi.org/10.1007/s11263-021-01453-z`.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *CoRR*, cs.AI/9605103, 1996. URL `https://arxiv.org/abs/cs/9605103`.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks, 2017.

Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency, 2017.

Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning, 2020a.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated. *arXiv preprint arXiv:2002.10619*, 2020b.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models, 2018.

Reshef Meir. Iterative voting. *Trends in computational social choice*, pages 69–86, 2017.

Aditya Paliwal, Felix Gimeno, Vinod Nair, Yujia Li, Miles Lubin, Pushmeet Kohli, and Oriol Vinyals. Reinforced genetic algorithm learning for optimizing computation graphs. *arXiv preprint arXiv:1905.02494*, 2019.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2020.

Aaron Segal, Antonio Marcedone, Benjamin Kreuter, Daniel Ramage, H. Brendan McMahan, Karn Seth, K. A. Bonawitz, Sarvar Patel, and Vladimir Ivanov. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017. URL `https://eprint.iacr.org/2017/281.pdf`.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016. doi: 10.1109/JPROC.2015.2494218.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning, 2018.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020. doi: 10.1109/TIFS.2020.2988575.

White House. Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy. *Journal of Privacy and Confidentiality*, 4(2), Mar. 2013. doi: 10.29012/jpc.v4i2.623. URL `https://journalprivacyconfidentiality.org/index.php/jpc/article/view/623`.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. Does unsupervised architecture representation learning help neural architecture search? *arXiv preprint arXiv:2006.06936*, 2020.

Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.07.061. URL `https://www.sciencedirect.com/science/article/pii/S0925231220311693`.

Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

Chunhui Yuan and Haitao Yang. Research on k-value selection method of k-means clustering algorithm. *J*, 2(2): 226–235, 2019.

Yu Zhang and Qiang Yang. A survey on multi-task learning, 2021.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data, 2018.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021. doi: 10.1109/JPROC.2020.3004555.

A Appendix

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w)**:** *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Figure 1: FedAvg, from McMahan et al. [2017]