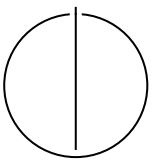
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Adaptation Techniques for using NAS Methods in the FL Setting

Max Coetzee



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

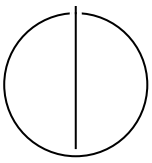
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Adaptation Techniques for using NAS Methods in the FL Setting

Titel

Author:	Max Coetzee
Examiner:	Supervisor
Supervisor:	Advisor
Submission Date:	Submission date



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, Submission date Max Coetzee

Acknowledgments

Abstract

Both *Neural Architecture Search* (NAS) and *Federated Learning* (FL) have made significant progress independently in the past decade. To benefit from the advantages of NAS methods in FL, researchers have started combining them by using NAS in FL [8] [5] [14].

[TODO: overhaul]

Applying techniques from Neural Architecture Search (NAS) to Federated Learning (FL) has been fruitful (remove) in recent years. The combination was identified as a promising research (remove) direction by [12]. It has yielded methods for finding architectures that deal with the challenges imposed by the FL setting.

Research into NAS has grown rapidly [20] since it was popularized by [23]; consequently, literature on its application to FL has grown. The last survey on NAS applied to FL compared approaches of four papers [22]. Since then, we have identified approximately 50 new papers. This motivates a new systematic survey of the landscape to identify progress and gaps in the literature.

In this thesis, we propose a map of the literature landscape based on the FL challenges they address. We achieve this by systematically evaluating the literature and identifying which challenge it solves.

We refer to the FL challenges described in [23], i.e., non-IID data, limited communication, client heterogeneity, privacy of client data, and break them down into smaller subchallenges — each subchallenge being associated with a pattern in the literature. We include personalized FL [18] as an additional subchallenge that was not originally posited, but has since drawn the community's attention.

We then analyze how the subchallenges are addressed and focus on the contribution of the used NAS method towards overcoming the subchallenge. For each subchallenge, we keep track of the NAS types used (following [20], [2]) and assess whether the underexplored methods are candidates for future research.

Neural Architecture Search

Contents

Acknowledgments **iv**

Abstract **v**

1 Introduction **1**

2 Background and Related Work **4**
2.1 Neural Architecture Search 4
2.2 Federated Learning 4
2.3 Federated Neural Architecture Search 4

3 Method **5**
3.1 Reviewed Literature 6

4 FedNAS Challenges **7**

5 Adaptation Techniques **11**

6 Discussion **12**

7 Conclusion **13**

Abbreviations **14**

List of Figures **15**

List of Tables **16**

Bibliography **17**

1 Introduction

Engineering the architecture of a neural network for a Deep Learning application is traditionally done by a team of domain experts and Deep Learning experts based on their expert knowledge and a process of trial and error. To reduce the amount of manual labour involved in this process, researchers invented *Neural Architecture Search* (NAS) [6] methods and improved them over the past decade. NAS methods employ diverse strategies to automatically search for a neural network architecture for a given Deep Learning application.

Independantly, but in parallel to NAS, researchers developed a distributed machine learning approach called *Federated Learning* (FL) [15] in response to growing concerns about data privacy. In FL, *clients* collaboratively train a model without sharing their local data. This enhances the privacy of clients' data, because model trainers can not view clients' data and client data is not collected at a central location where a single breach could expose the data of all clients.

Engineering neural network architectures in FL is as time-consuming (if not more so) as in centralised Deep Learning, therefore researchers have started investigating the use of NAS methods in FL [8] [5] [14]. Additionally, NAS methods provide an alternative to selecting a fixed architecture upfront — a so-called *prededfined architecture*. Predefined architectures can lead to slow training convergence and poorly performing models in FL, because model developers can view neither the clients' data nor, typically, client's hardware capabilities. Model developers may therefore select a predefined architecture that contains components irrelevant for generalising well from client data sets or select an architecture that trains slowly on some clients. Work has already been done that shows the use of NAS methods in FL can mitigate these issues [9] [16] [21].

Despite the potential advantages of using NAS in FL, it is still non-trivial to do so. Most NAS methods are infeasible for direct application in FL, because NAS research has focused on the traditional centralised setting as opposed to FL. Centralised NAS can make several assumptions about the search process that do not hold in the FL setting. Each assumption discrepancy creates a *challenge* for using NAS in FL, and developers have created *adaptation techniques* for overcoming them. Applying adaptation techniques to NAS methods results in *Federated Neural Architecture Search methods* (FedNAS methods) [8].

For example, one challenge arises from the fact that centralised NAS can assume

worker nodes are computationally powerful, whereas clients in most FL settings are not. Since most centralised NAS methods place large computational burdens on worker nodes, practitioners using these NAS methods in FL without modification would experience detrimental search completion times. To combat this, FedNAS developers have created adaptation techniques to reduce the computational burden on individual clients in FedNAS methods [5] [19] [10]. This typically involves reducing the overall computational work and splitting it up into smaller units, which presents a challenge, as the implementation of the resulting FedNAS method tends to be complex.

The subset of challenges faced by FedNAS methods depends on the specific FL setting, which can differ in many parameters [12]. The literature identifies two major classes of FL settings: the *cross-device* class, wherein clients are edge devices, and the *cross-silo* class, wherein clients are entire organisations, but even within these classes, there is significant variation in the setting parameters. Each FL setting violates centralised NAS assumptions to a different extent, making some challenges more relevant to them than others. For example, for FL settings in the cross-silo class, clients can be expected to be equipped with GPUs, making the challenge described above less relevant.

Developing new FedNAS methods is a length procedure that is typically a stab in the dark

Practitioners aiming to develop new FedNAS methods for a specific FL setting face uncertainty about which centralised NAS assumptions are violated and which adaptation techniques are necessary or sufficient to address them. Due to the lack of a structured overview linking FL setting characteristics to challenges and corresponding adaptation techniques, developers must rely on ad-hoc design choices, increasing development time and the risk of inefficient or incompatible method designs.

The literature on adaptation techniques is fragmented, and FedNAS methods often lack clarity regarding the targeted FL setting and the challenges they address. As a result, extending and re-using existing techniques remains difficult. This poses a problem for FedNAS developers, since they need to trade off which challenges to address for their targeted FL setting without a clear overview of adaptation techniques that would be useful for that setting. Prior literature surveys [22] [13] [7] summarise FedNAS methods on the whole, but do not dissect them in a manner that allows FedNAS developers to decide on the parts they wish to re-use. To aid the development of new FedNAS methods, we set out to answer our research question:

What challenges arise from different FL settings for FedNAS methods, and which adaptation techniques address them in the literature?

To tackle our research question, we conduct a systematic literature review of papers that present FedNAS methods. We employ grounded theory and the methodology

from [11]. For our review, we consider papers that modify NAS methods in response to the FL setting.

We define a set of fine-grained parameters to characterise the targeted FL setting of each FedNAS method based on observations of varying setting parameters in the literature. With the help of this characterisation, we identify the violated centralised NAS assumptions and catalogue the challenges that arise from them. Next, we extract unrefined adaptation techniques from the FedNAS methods and iteratively refine and merge them to obtain a set of collectively exhaustive adaptation techniques. We analyse how each adaptation technique works towards, against, or does not affect each challenge, and present our findings in the form of a discussion for each adaptation technique, as well as an overview table.

Our review aims to support the creation of new FedNAS methods by developers. By identifying the source of challenges and elaborating on them, we provide clarity on the expected challenges for a targeted FL setting. Based on the expected challenges, FedNAS developers can use our overview of adaptation techniques to guide the design of new FedNAS methods and determine whether to re-use existing techniques, extend them, or develop new ones.

In Chapter 2, we cover the background required for this thesis and related work. In Chapter 3, we describe the method with which we conduct our literature review and give an overview of the included literature. In Chapter 4, we highlight the challenges with FedNAS and how they arise from different FL settings. In Chapter 5, we describe the adaptation techniques we found and how they overcome challenges. In Chapter 6, we conduct a discussion about our work. Chapter 7 contains our conclusion.

2 Background and Related Work

NAS is contained entirely in the 5th step of the FL pipeline as described in [12].

2.1 Neural Architecture Search

2.2 Federated Learning

2.3 Federated Neural Architecture Search

3 Method

[TODO: overhaul to include method for identifying challenges and FL settings] - targeted FL setting can be stated directly, indirectly or must be inferred from experimental setup

We use various techniques from [3] and lean on parts of the methodology of [11] to perform a systematic literature review.

1. **Literature Selection:** We follow the guidelines and flow diagrams provided by PRISMA 2020 [17] for inclusion and exclusion of papers and perform forward and backwards citation searching. We identify 58 papers that present FedNAS methods. Each paper contains one or more FedNAS methods.

2. **Unrefined Adaptation Technique Extraction:** Once the set of included papers is fixed, we perform open coding on each FedNAS method to extract unrefined adaptation techniques. Any modification to a NAS method that is explicitly motivated by the federated setting is treated coded as one unrefined adaptation technique.

3. **Adaptation Techniques Coneceptualization:** We iteratively refine and merge unrefined adapatation techniques in an axial coding step to obtain a coherent set of adaptation techniques that are mutually exclusive and collectively exhaustive. Unrefined adaptation techniques with conceptually highly-similar mechanisms are merged into a single representative adaptation technique.

4. **Categorise Adaptation Techniques:** After merging, in a second axial coding step, we cluster adaptation techniques based on a) the FedNAS challenges they address and b) the conceptual similarity of their mechanisms. As a result, we obtain a taxonomy of adaptation techniques.

5. **Discuss FedNAS Challenges for Adaptation Techniques:** We discuss how each adaptation technique works towards, against, or does not affect overcoming each of FedNAS challenges.

6. **Produce Table Overviews:** Finally, we create two tables that practicioners can use to make decisions about FedNAS methods for their use case.

The first table contains a coded vector of effects over the FedNAS challenges for each *FedNAS method*. The effects per FedNAS method are the result of the effects of its adaptation techniques in aggregate. Practicioners can use this table to decide which FedNAS method suits their use case or if their use case would benefit from a new FedNAS method.

The second table table contains a coded vector of effects over the FedNAS challenges for each *adaptation technique* based on the prior discussion. If practicioners need to create new FedNAS methods, they can use this table to choose existing adaptation techniques relevant to the set of FedNAS challenges they need to address for their use case.

3.1 Reviewed Literature

add some overview and statistics about the literature

4 FedNAS Challenges

Assumption in Centralised Setting	Reality in FL Setting	Resulting FedNAS Challenge Description	FedNAS Challenge Name
Worker nodes' hardware is homogeneous.	Clients' hardware varies significantly. More pronounced in the cross-device setting than the cross-silo setting.	FedNAS methods need to be heterogeneity-aware; otherwise, the search will be delayed by low-end devices or leave high-end devices waiting in idle most of the time.	Hardware Heterogeneity
Worker nodes communicate over high-bandwidth, low-latency links.	Clients typically communicate with the central server over high-latency and low-bandwidth connections.	FedNAS must be communication-efficient; otherwise, exchanging model weights and architecture parameters becomes a bottleneck during the search process.	Limited Networking Capabilities
Worker nodes are high-end machines with powerful CPUs, GPUs, and large amounts of RAM.	Clients are edge devices with few computational resources.	The computational burden placed on clients by a NAS method needs to be adjusted such that the architecture search takes place within an acceptable time frame. This can involve splitting up computational work, usually done on single worker nodes, amongst multiple clients, making the implementation complex.	Limited Computational Resources
Training data is gathered at a central location.	Training data is distributed unevenly across clients	Since some clients have more data than others, FedNAS methods must ensure that these clients are not overrepresented in the searched architecture.	Unbalanced Client Data

The training data is drawn from the same distribution.	Each client draws training data from their own distribution.	FedNAS methods need to be robust with respect to non-i.i.d. data, which makes evaluating and ranking candidate architectures noisier than in the centralised setting.	Non-I.I.D. Training Data
Worker nodes are equally available.	Some clients are more frequently available than others.	FedNAS methods must prevent the search from being dominated by the most frequently available clients; otherwise, the resulting architecture will be biased towards them.	Variable Client Availability
All worker nodes participate in each iteration.	Only a subset of clients participates in each iteration of the search.	FedNAS methods must ensure that a realistic sample of the client population is represented in the architecture search and address high-variance performance estimates, as candidate architectures are evaluated on changing subsets of clients.	Client Participation
Worker nodes are inside the same trust domain.	All participating parties (i.e. the central server and clients) can consider another potentially malicious.	FedNAS methods must address attacks from participating parties, such as architecture parameter poisoning during the search process.	Security
Worker nodes consistently participate in the search process.	Clients can drop out of a communication round at any time, and completion times of epochs vary.	FedNAS methods need to handle client dropouts and stragglers, since interrupted or delayed evaluations of candidate architectures can slow down or destabilise the search process.	Client Reliability

Model accuracy is more important than model resource consumption in selecting architectures.	Model accuracy needs to be traded for model resource consumption in selecting architectures.	Models trained in FL tend to be deployed on the same resource-constrained clients they are trained on. Therefore FedNAS methods need to optimise the architecture of the trained model w.r.t. resource constraints.	Model Resource Constraints

Table 4.1: Discrepancies between NAS in the centralised setting and the FL setting and the resulting FedNAS challenges. *Worker nodes* perform the architecture search in the centralised setting, whereas a *server* and *clients* perform the search in the FL setting. The challenges are based to some extent on previously identified challenges in the FL setting in general [15] [12] [4] [1].

5 Adaptation Techniques

5.1

6 Discussion

7 Conclusion

Abbreviations

List of Figures

List of Tables

4.1 Discrepancies between NAS in the centralised setting and the FL setting and the resulting FedNAS challenges. *Worker nodes* perform the architecture search in the centralised setting, whereas a *server* and *clients* perform the search in the FL setting. The challenges are based to some extent on previously identified challenges in the FL setting in general [15] [12] [4] [1]. 10

Bibliography

[1] M. Arbaoui, M.-A. Brahmia, A. Rahmoun, and M. Zghal. "Federated learning survey: A multi-level taxonomy of aggregation techniques, experimental insights, and future frontiers." In: *ACM Trans. Intell. Syst. Technol.* 15.6 (Nov. 2024). ISSN: 2157-6904. DOI: 10.1145/3678182.

[2] S. S. P. Avval, N. D. Eskue, R. M. Groves, and V. Yaghoubi. "Systematic review on neural architecture search." In: *Artificial Intelligence Review* 58.3 (Jan. 6, 2025), p. 73. ISSN: 1573-7462. DOI: 10.1007/s10462-024-11058-w.

[3] J. Corbin and A. Strauss. *Basics of qualitative research*. Core textbook v. 14. SAGE Publications, 2015. ISBN: 9781412997461.

[4] K. Daly, H. Eichner, P. Kairouz, H. B. McMahan, D. Ramage, and Z. Xu. "Federated learning in practice: Reflections and projections." In: *2024 IEEE 6th international conference on trust, privacy and security in intelligent systems, and applications (TPS-ISA)*. 2024, pp. 148–156. DOI: 10.1109/TPS-ISA62245.2024.00026.

[5] L. Dudziak, S. Laskaridis, and J. Fernandez-Marques. *FedorAS: Federated architecture search under system heterogeneity*. 2022. arXiv: 2206.11239 [cs.LG].

[6] T. Elsken, J. H. Metzen, and F. Hutter. "Neural architecture search: A survey." In: *Journal of Machine Learning Research* 20.55 (2019), pp. 1–21.

[7] M. Hartmann, G. Danoy, and P. Bouvry. *Multi-objective methods in Federated Learning: A survey and taxonomy*. 2025. arXiv: 2502.03108 [cs.LG].

[8] C. He, M. Annavaram, and S. Avestimehr. *Towards Non-I.I.D. and invisible data with FedNAS: Federated deep learning via neural architecture search*. 2021. arXiv: 2004.08546 [cs.LG].

[9] M. Hoang and C. Kingsford. "Personalized Neural Architecture Search for Federated Learning." In: *1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021)* ().

[10] C. Huo, J. Jia, T. Deng, M. Dong, Z. Yu, and D. Yuan. "NASFLY: On-device split federated learning with neural architecture search." In: *2024 IEEE international symposium on parallel and distributed processing with applications (ISPA)*. 2024, pp. 2184–2190. DOI: 10.1109/ISPA63168.2024.00298.

[11] D. Jin, N. Kannengießer, S. Rank, and A. Sunyaev. "Collaborative distributed machine learning." In: 57.4 (Dec. 2024). ISSN: 0360-0300. DOI: 10.1145/3704807.

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

[13] S. Khan, A. Rizwan, A. N. Khan, M. Ali, R. Ahmed, and D. H. Kim. "A multi-perspective revisit to the optimization methods of Neural Architecture Search and Hyper-parameter optimization for non-federated and federated learning environments." In: *Computers and Electrical Engineering* 110 (2023), p. 108867. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2023.108867.

[14] J. Liu, J. Yan, H. Xu, Z. Wang, J. Huang, and Y. Xu. "Finch: Enhancing federated learning with hierarchical neural architecture search." In: *IEEE Transactions on Mobile Computing* 23.5 (2024), pp. 6012–6026. DOI: 10.1109/TMC.2023.3315451.

[15] 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*. Ed. by S. Aarti and Z. Jerry. Vol. 54. Proceedings of machine learning research. PMLR, Apr. 2017, pp. 1273–1282.

[16] E. Mushtaq, C. He, J. Ding, and S. Avestimehr. *SPIDER: Searching personalized neural architecture for federated learning*. 2021. arXiv: 2112.13939 [cs.LG].

[17] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, and D. Moher. "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews." In: *BMJ* 372 (2021). DOI: 10.1136/bmj.n71. eprint: https://www.bmj.com/content/372/bmj.n71.full.pdf.

[18] A. Z. Tan, H. Yu, L. Cui, and Q. Yang. "Towards personalized federated learning." In: *IEEE Transactions on Neural Networks and Learning Systems* 34.12 (2023), pp. 9587–9603. DOI: 10.1109/TNNLS.2022.3160699.

[19] X. Wei, G. Chen, C. Yang, H. Zhao, C. Wang, and H. Yue. "EFNAS: Efficient federated neural architecture search across AIoT devices." In: *2024 international joint conference on neural networks (IJCNN)*. 2024, pp. 1–8. DOI: 10.1109/IJCNN60899.2024.10650653.

[20] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter. *Neural architecture search: Insights from 1000 papers*. 2023. arXiv: 2301.08727 [cs.LG].

[21] J. Yan, J. Liu, H. Xu, Z. Wang, and C. Qiao. "Peaches: Personalized federated learning with neural architecture search in edge computing." In: *IEEE Transactions on Mobile Computing* 23.11 (2024), pp. 10296–10312. DOI: 10.1109/TMC.2024.3373506.

[22] H. Zhu, H. Zhang, and Y. Jin. "From federated learning to federated neural architecture search: a survey." In: *Complex and Intelligent Systems* 7.2 (Apr. 2021), pp. 639–657. ISSN: 2198-6053. DOI: 10.1007/s40747-020-00247-z.

[23] B. Zoph and Q. V. Le. *Neural architecture search with reinforcement learning*. 2017. arXiv: 1611.01578 [cs.LG].