1 Introduction

Applying Neural Architecture Search (NAS) methods in a Federated Learning (FL) setting is an emerging field of study, which we shall call *FedNAS*. In this chapter, we motivate the study of FedNAS and *adaptation techniques* used by FedNAS methods to adapt NAS to the FL setting. Our motivation leads us to our research questions and ultimately, our proposed methodology to address them.

1.1 Motivation

NAS and FL have independently made significant progress in recent years and both are increasingly adopted in practice.

The goal of NAS is the automation of the laborious neural network architecture engineering process responsible for so many of the advances made in Deep Learning in recent years [EMH19]. The most recent wave of NAS research was initiated by [ZL17] in 2017. Reinforcement Learning, a black-box optimisation technique, was used to show that an automatically searched architecture can outperform handcrafted state-of-the-art architectures for image classification. NAS methods proposed early on were inefficient and a significant amount of research attention improved the situation soon after. Newly developed efficient techniques tend to make use of the internals of the searched architectures (ENAS [Pha+18] and DARTS [LSY19] are prominent examples). These techniques are still widely used today and have significantly accelerated NAS by orders of magnitude. Faster NAS has enabled broader adoption and consequently more experimentation for finding architectures with better accuracy, smaller size and faster training convergence. Architectures found via NAS now frequently improve upon state-of-the-art handcrafted architectures (e.g. NASNet [Zop+18], AmoebaNet [Rea+19], MobileNetV3 [How+19], EfficientNetV2 [TL21] and many more).

FL, on the other hand, has become a viable privacy-enhancing choice for Collaborative Distributed Machine Learning [Jin+24]. Ever more valuable training data is collected on edge devices like smartphones, referred to as *clients*. Traditionally, this privacy-sensitive data is either not used at all or it is collected in a central location for training — increasing the risk of exploitation by malicious actors. Google invented FL in 2016 to address this issue and allow the use of decentralised training data without the data leaving the device on which it was generated. Instead of performing training

at a central location with a cluster of high-end machines, clients in FL train a shared model on their local data and coordinate weight updates to the shared model via a central server. FL "*embodies the principles of focused collection and data minimization, and can mitigate many of the systemic privacy risks and costs resulting from traditional, centralized machine learning*" [Kai+21]. Since its inception in 2016, FL has been adopted for various ML tasks in production systems by organisations like Google [Yan+18], Apple [Ji+25] and Owkin [Old+22].

Users of FL naturally expect to leverage existing NAS methods in the FL setting to produce architectures that achieve state-of-the-art accuracy. Apart from the generic benefits, NAS has been identified as a good fit for the FL setting. A large body of work in NAS has focused on finding smaller architectures with reduced inference latency that still have reasonable accuracy [Whi+23]. Such architectures are ideal for deployment on the often resource-constrained clients in the FL setting. [Kai+21] notes that predefined architectures may not be an optimal choice for FL, where user-generated data is not visible to model developers. They note that predefined architectures may contain components redundant for specific data sets or perform poorly on non-i.i.d. data (which is prevalent in FL). When the architecture search takes place on the clients, architectures can be adjusted according to the data and distribution present on them, yielding architectures that are better-suited to the data than those a model developer would select. Dealing with the distributed, non-i.i.d. data of the FL setting has been a key challenge for FedNAS methods, and research into the matter has shown promise [HAA21] [Yao+21] [DLF22] [Liu+24] [Yan+24]. Beyond being tailored to the data and distribution of each client, architectures can also be personalised to a client's heterogeneous computation and bandwidth budgets, as shown by [Kha+24] [DLF22] [YL24] [Yua+22].

However, research in NAS methods has focused on a central NAS setting, and the methods rely on assumptions that do not hold for the FL setting. These assumptions include an abundance of computational resources, homogeneity of hardware for training and deployment, high availability of worker nodes, access to the entire dataset, access to the data distribution, etc. [Kai+21]. Consequently, most centralised NAS methods are unfeasible for direct application in the FL setting. Adopting NAS methods in the FL setting requires adapting them to the FL setting, giving rise to what we will call *adaptation techniques*. In the following, we briefly illustrate three adaptation techniques.

1. For example, naively training a one-shot supernet with a large search space, where each client trains the entire supernet and aggregates the weights of the entire supernet, works for the cross-silo FL setting [HAA21], but does not translate to the cross-device setting. Clients in the cross-device setting are generally less powerful, and such a training scheme would result in detrimental completion

times. Instead, in one FedNAS method [DLF22], researchers have opted to reduce the computational burden on clients by only sampling and training subnets within the client's training budget.

2. Another problem arises when using standard FedAvg to average the architecture weights for DARTS-based supernets [LSY19]. Clients may tend towards architectures at the opposite ends of a spectrum, but averaging the architecture weights may select for an architecture that is not favoured by any client. In [Wei+24], this is addressed by aggregating architecture weights into a probability distribution that can be used to sample likely architectures in the subsequent communication round.

3. Typically, when NAS methods are run in a central environment, they implicitly assume that the machines on which NAS is performed are homogeneous and contribute equally to the architecture search. This assumption does not hold in the FL setting, where the variance in computational resources between clients can be substantial. Consequently, higher-end clients will tend to be used more in training, introducing bias to the searched architecture. [ÖÖ25] adapts a NAS method to overcome this problem by grouping clients into clusters according to their computational speed and network bandwidth. Small models are trained on low-end client clusters, while larger models are trained on high-end client clusters.

As shown by the examples above, the conditions of the FL setting break NAS methods in a way that requires novel adaptation techniques. While NAS research has focused on finding ways to perform NAS faster and finding better and smaller architectures, a key challenge of FedNAS methods lies in achieving these goals despite the challenges imposed by the FL setting — and adaptation techniques are key. No consolidated body of knowledge exists that can inform researchers on existing adaptation techniques and aid them in designing new techniques. To mend this, we perform a systematic review of adaptation techniques in this thesis.

1.2 Research Questions

The vast number of FedNAS methods yields a diverse array of adaptation techniques. Each FedNAS method utilises a set of adaptation techniques to adapt a NAS method to the FL setting. Some FedNAS methods have overlapping sets of adaptation techniques, but most sets of adaptation techniques are disjoint. Nonetheless, most literature does not draw clear boundaries around individual adaptation techniques, leading us to our second research question:

(RQ1) How can we identify adaptation techniques? Which design patterns in adaptation techniques recur?

Each adaptation technique pushes the FedNAS method as a whole in a direction with regard to optimality towards FL challenges. Identifying this direction for each adaptation technique is important to understand potential trade-offs. This leads us to our third research question:

(RQ2) Which FL challenges do the adaptation techniques address?

Adaptation techniques are heavily dependent on the targeted FL challenge and the NAS method used. Naturally, this does not allow for most adaptation techniques to work together. Additionally, some adaptation techniques are incompatible due to other reasons. For example, the architecture weight aggregation technique proposed by [Wei+24] aims to gradually increase the likelihood of sampling the best-performing architectures during training. In contrast, [Kha+24] introduces a technique that attempts to maintain a diverse set of architectures throughout the training process. We would still like to know which existing adaptation techniques can be combined in what way, possibly paving the way for creating more powerful FedNAS methods tailored towards specific use cases. This leads us to our fourth research question:

(RQ3) How can existing adaptation techniques be composed into FedNAS methods? How can we choose a set of adaptation techniques based on a use case?

1.3 Methodology

Using either NAS or FL on its own is a complex task. Combining them introduces even more possible knobs to turn on the system as a whole. There are many NAS methods and approaches to address FL challenges — culminating in a large amount of possible FedNAS methods and literature on the subject. To identify relevant papers, we follow the guidelines and flow diagrams provided by PRISMA 2020 [Pag+21] and perform forward and backwards citation searching. Each paper represents one FedNAS method.

Once the set of FedNAS methods is fixed, we address RQ1 by identifying the set of adaptation techniques for each FedNAS method and providing a conceptual definition of the techniques. After each adaptation technique has been defined, we identify patterns in the techniques and cluster similar ones into categories. A category of adaptation techniques captures the general sense of the techniques it represents. For each category, we provide discriminant rules that clearly separate one category of

adaptation technique from the others.

For RQ2, we make use of prior work on a taxonomy of FL challenges [Arb+24]. We review all categories of adaptation techniques and discuss how each technique works towards, against, or has no effect on overcoming each of the FL challenges. The result is Table 1, which displays adaptation technique categories on the y-axis and FL challenges on the x-axis..

For RQ3, we propose building an "is-compatible-with" relation on the set of adaptation techniques. We achieve this by cross-examining each adaptation technique with every other one. To reduce the number of comparisons, we make use of the taxonomy of NAS methods provided in [Whi+23] and place each adaptation technique in one or multiple of the NAS method bins. Only adaptation techniques within the same bin need to be compared.

Finally, we utilise the "is-compatible-with" relation and Table 1 to select a few promising adaptation technique combinations and discuss their potential compared to the state of the art.