1 Introduction

Applying Neural Architecture Search (NAS) methods in a Federated Learning (FL) setting is an emerging field of study, which we shall abbreviate to *FedNAS*. In this chapter, we contextualise and motivate a systematic literature review of *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 Context

Relevant to this thesis, the last decade witnessed two major developments in parallel: 1. the establishment of neural architecture search (NAS) and 2. the invention of Federated Learning (FL). Together they form the cornerstone of this thesis and their origins can be explained as follows:

1. **Origin of Neural Architecture Search:** Deep learning (DL), a subfield of machine learning (ML) that makes use of deep neural networks, made key advances in various machine learning tasks such as computer vision, natural language processing (NLP), speech recognition etc. These advances led to the widespread adoption of DL with some referring to it as a deep learning revolution. However, the advances came off the back of neural network architectures that were manually designed by researchers through extensive trial and error — for example, convolutional neural networks for computer vision tasks and the transformer for NLP tasks. It did not take long for researchers to wonder whether automating the search for novel architectures could produce architectures that improved upon the best manually designed ones. In 2017, [ZL17] showed that this was indeed possible and coined the term *Neural Architecture Search* (NAS) for the automated search of neural network architectures for a given task.[1]

2. **Origin of Federated Learning:** Edge devices, such as mobile phones or Internet-of-Things devices, have become ubiquotous and increasingly store privacy-sensitive data. Traditionally, this privacy-sensitive data was either not utilised at all for

[1] [TODO: prior research in 90s]

> machine learning or it was collected in a central location for training — creating the risk of a major breach by malicious actors. Google invented *Federated Learning* 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, training happens on the edge devices in FL (typically referred to as *clients* in FL). The clients 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].

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

More than 1000 papers have been published about NAS since 2017. The seminal paper [ZL17] used reinforcement learning, a black-box optimisation technique, [TODO: summarize how it works in a black-box way]. The found architecture matched handcrafted state-of-the-art architectures on accuracy for image classification and it was faster. NAS methods proposed early on were inefficient to train[2] and a significant amount of research attention improved the situation soon after. As opposed to earlier black-box optimisation techniques, newly developed techniques tended to make use of the internals of the searched architectures to improve training efficiency — ENAS [Pha+18] and DARTS [LSY19] are prominent examples. A slow, inevitable march of progress 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 have improved upon state-of-the-art handcrafted architectures for various Machine Learning tasks (e.g. NASNet [Zop+18], AmoebaNet [Rea+19], MobileNetV3 [How+19], EfficientNetV2 [TL21] and many more).

FL has also received signficant research attention. The seminal paper on FL [McM+17] was only concerend with edge devices as clients, but since then FL has also been used for training shared models between organisations [She+20] [Day+21] [Das+22]. The former is referred to as the *cross-device* setting and the latter as the *cross-silo* setting. In fact, FL has been identified as a good fit for collabariteve distributed machine learning [Jin+24] in general. 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].

[2] The authors of [ZL17] purportedly used 800 GPUs for two weeks.

1.2 Motivation

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 also note that predefined architectures may contain components redundant for specific data sets. Additionally, predefined architectures may perform poorly on another prevelant characteristic of the FL setting: data that is not independantly and identically distributed (non-i.i.d). 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 adopting NAS in the FL setting, 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 clients' 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]. The misaligned assumptions lead to *challenges* in adopting NAS in a FL setting. 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 shall call *FedNAS* methods. We have identified 58 papers that propose FedNAS methods. Each FedNAS method makes use of various techniques to adapt a NAS method to the FL setting. We shall call these techniques *adaptation techniques* for short. In the following, we briefly illustrate three adaptation techniques.

1. For example, while 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], it 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 optimize architecture parameters towards the opposite ends of a spectrum, but averaging the architecture parameters may select for an architecture that is not favoured by any client. In [Wei+24], this is addressed by aggregating the architecture parameters into a probability distribution that can be used to sample likely architectures in the subsequent communication rounds.

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.

1.3 Research Questions

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 for researchers to choose from and even more potential adaptation techniques to address FL challenges — culminating in a large amount of possible FedNAS methods. To understand the current landscape, we ask our first research question:

(RQ1) Which FedNAS methods have been described in the literature and which adaptation techniques do they use?

As mentioned before, we expect a diverse array of adaptation techniques in the literature. Yet, we also expect some FedNAS methods to use similar adaptation techniques, if they address the same type of FL challenges. This leads us to our second research question:

(RQ2) Which adaptation techniques are similar and what types of challenges of the FL setting do they 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 third research question:

Optional (RQ3) How can adaptation techniques described in the literature be composed into new, potentially better FedNAS methods?

1.4 Methodology

To address our research questions, we divide our approach into six steps:

1. **Literature Selection:** To tackle the first part of RQ1, we follow the guidelines and flow diagrams provided by PRISMA 2020 [Pag+21] for inclusion and exclusion of papers and perform forward and backwards citation searching. Each paper presents one or more FedNAS methods.

2. **Adaptation Technique Extraction:** Once the set of included papers is fixed, we tackle the second part of RQ1 by analyzing each paper individually and extract the adaptation techniques it uses and summarize them.

3. **Categorise Adaptation Techniques:** For the first part of RQ2, we cluster adaptation techniques based on conceptual similarities and deliver a taxonomy of

adaptation techniques.

4. **Define FL Challenges:** For the second part of RQ2, we must first define a classification of FL challenges. To this end, we repurpose a prior classification [Arb+24] of recent research advances in FL. [Arb+24] suggests that newly proposed FL methods in FL research enhance one or more *evaluation metrics*. We argue that their aforementioned evaluation metrics are the result of finding a measure for how effectively a FL challenge has been addressed. Therefore the clustering of evaluation metrics into research directions is akin to clustering FL challenges into types of FL challenges.

5. **Map addressed FL Challenge Types onto Adaptation Technique Clusters:** Next, we review all clusters of adaptation techniques from step 3. and discuss how each adaptation technique cluster works towards, against, or has no effect on overcoming each of the FL challenge types defined in step 4.

6. **Compose promising FedNAS methods:** For RQ3, we propose building an "is-compatible-with" relation on the set of adaptation technique clusters. We achieve this by cross-examining each adaptation technique cluster with every other one. We then utilise the "is-compatible-with" relation together with our discussion from step 5. to identify groups of adaptation technique clusters that can be composed into FedNAS methods which could potentially shift the Pareto frontier with respect to optimality of addressing FL Challenge types.