*Electronics and Computer Science Faculty of Engineering and Physical Sciences University of Southampton*

Josh Pattman

12 December 2022

# Investigating Optimisations Of Swarm Learning With Respect To Real World Challenges

Project Supervisor:

Dr Mohammad Soorati - *m.soorati@soton.ac.uk*

Second Examiner:

Dr Jo Grundy - *j.grundy@soton.ac.uk*

A project progress report submitted for the award of
**Computer Science with Artificial Intelligence**

# Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.
- I have acknowledged all sources, and identified any content taken from elsewhere.
- I have not used any resources produced by anyone else.
- I did all the work myself, or with my allocated group, and have not helped anyone else.
- The material in the report is genuine, and I have included all my data/code/designs.
- I have not submitted any part of this work for another assessment.
- My work did not involve human participants, their cells or data, or animals.

# Abstract

Federated learning is a technique that allows a machine learning model to be trained on data distributed across multiple data islands without ever sharing the data. This approach protects privacy by keeping the data decentralized. Swarm learning is a similar technique that eliminates the need for a central server, ensuring that not only the data, but also the communication, is completely decentralized. The goal of this project is to design and implement a swarm learning algorithm, then test the algorithm in multiple scenarios inspired by the real world. The algorithm will then be evaluated against other machine learning techniques to investigate if swarm learning is a viable technology for the modern world.

# Contents

# Background

## 1.1 Background Research

### 1.1.1 Federated Learning

Many modern machine learning algorithms require large volumes of diverse data to achieve optimal performance. Real-world data is often distributed among multiple organizations that are unable to share it with each other due to privacy regulations such as GDPR [1], reducing the amount of data available to train models on and negatively impacting trained performance [2]. Federated Learning (FL) [3] is a technique in machine learning that aims to train a single model using all available data across organizations without requiring any data to be shared among them.

There are a multitude of published FL frameworks [4], each with different merits and drawbacks for certain use cases. Federated Averaging (FedAvg) [5] is a commonly used yet simple framework, which splits training into iterations where three steps take place:

1. A copy of the current model is sent to each agent
2. Each agent performs some training with their copy of the model and their own private data
3. The trained models from each agent are sent back to the server to aggregate into a global model

The global model improves over time, and after a certain number of steps, the training is complete and the global model is the final output.

As it does not require data to be shared between agents, FL is naturally beneficial for privacy sensitive tasks compared to conventional machine

learning where the data is aggregated in a central location ***CITEME***. Additionally, as FL performs training on multiple agents in parallel, it can make better use of available training resources in situations where processing power is distributed among multiple nodes ***CITEME***.

One branch of FL is distributed federated learning (DFL). This algorithm works in a very similar way to FL, but replaces the central server with an elected leader in a network of agents [6]. DFL has been shown to increase fault tolerance and security over FL.

### 1.1.2 Swarm Learning

Swarm learning (SL) [7] is a subcategory of FL which operates in a completely distributed and decentralised manner. SL still allows a network of agents to collaborate to learn a shared model, but at no point is a central server used or a leader chosen. In contrast to FL, SL methods usually use a blockchain based system to coordinate a global model.

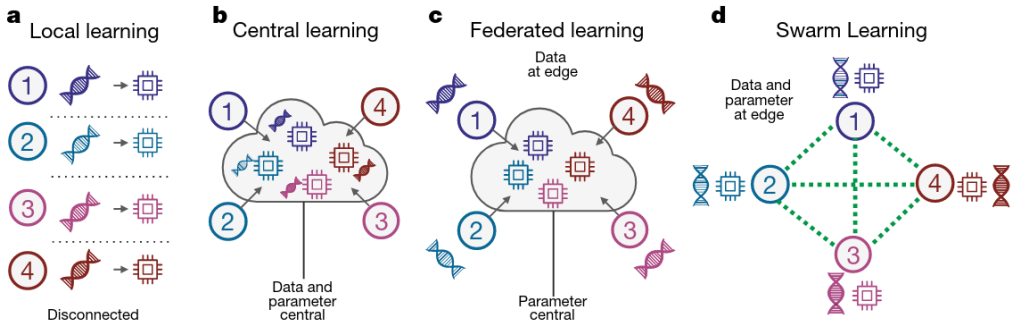



Figure 1.1: Diagram of swarm learning compared to other type of learning taken from [7]

In SL, agents learn in a similar fashion to FL in training steps. However, the agents do not need to have their training steps synchronised. Following are the typical training steps of a swarm learning agent:

1. An agent obtains a copy of the current model from the blockchain
2. The agent performs some training with their copy of the model and their own private data
3. The updated model is merged with the current model on the blockchain and sent back to the blockchain for other agents to use

The model on the blockchain will improve over time, and after a certain number of steps the blockchain model is the final output.

SL exhibits many of the same benefits of FL over conventional learning, but it also improves upon FL in some aspects. As is often the case when comparing decentralised algorithms to their centralised counterparts [8], the absence of a central server makes SL more resilient to failures than centralized FL approaches such as FedAvg. The lack of a leader election protocol also means that SL may be better suited to tasks where networks of agents are sparsely connected, as leader election in dynamic networks is a complex problem and can add large amounts of overhead [9]. Additionally, the removal of the need for a server in SL reduces the likelihood of performance bottlenecks for very large swarms.

### 1.1.3 Challenges for Federated and Swarm Learning

There are numerous challenges associated with implementing FL or SL compared to conventional learning systems. Some of the common challenges are described below:

#### Training Overhead

Since FL and SL require communication between multiple agents, there is an inherent overhead cost associated with data transfer time and synchronization of agents. Additionally, the centralization of FL can result in server bottlenecking. This is an area where SL offers an improvement over conventional FL. One potential solution to reduce bottlenecks is to divide the agents into groups, each with its own central server, and enable those servers to communicate with a core server [10]. This has the potential to reduce the burden on the central server and improve overall performance.

#### Unevenly Distributed Labels and Features - Non-IID Data

When a dataset is collected from various sources and locations, it will have a specific distribution of features and labels. However, the distribution of these features and labels may differ among the different sources from which it was collected, meaning the data is referred to as non-iid. For example, a dataset containing security camera footage of car and bus accidents may have more of one type of accident at some locations than others (labels). The cameras at different locations may also have varying brightness or contrast settings (features). This can negatively impact the training of SL and FL

algorithms [11] because each agent is effectively proposing a solution to a different problem than the rest. There is a significant amount of research being conducted on this topic, with various approaches being proposed [12].

One approach is to partition agents into different clusters, where each member of a cluster communicates with the same server. The servers at the centre of each cluster will perform FL with their respective agent, but still synchronise the cluster model to all other cluster servers [10]. This allows each sub-cluster to develop its own modified version of the global model to suit its needs.

A different approach is to use batch normalisation layers on each agent [13], which adds a degree of standardisation to activations in the network. This method is focussed on non-iid features as opposed to non-iid labels, and has shown to have excellent results as opposed to a multitude of other FL techniques.

### Sparsely Connected Networks

For FL, ideally every agent would be able to have direct communication with the central server. Similarly for SL in the best case, every agent would have direct communication with every other agent. However, these cases are rarely ever realised. The effect of having a sparsely connected network is more detrimental to FL than SL mainly because FL needs server communication, but SL and blockchain have been proven to work in sparse settings. This can however still slow the training of SL.

### Data Transfer Limits

In the real world, it is sometimes not possible to transfer entire models between agents on a regular basis, due to data transfer limits. It has been proposed that a possible solution could be to use FedAvg, but only transfer segments of the network at each training step [14]. This approach outperformed FedAvg by a factor of between 2.25 and 3.01 with respect to training speed, due to the removal of the central server bottleneck and reduced data transferred [14][section. 5.2].

### Low Processing Power Agents

FL and SL are promising techniques for robotic systems [15] and Internet-of-Things (IoT) devices [16]. However, these systems often do not have the same high level of processing power as a central machine learning server. For this reason, it is imperative that FL and SL systems are as efficient as

possible with the results of their training as to minimise the loss caused by reduction in processing power.

# Project Goals

## 2.1 Problem

Currently, there is more data stored around the world than ever before. However, there are also many recent regulations that prevent the data from being used by machine learning algorithms. In many cases, this is because the data owners are not legally able to send the data to a central server for processing and training. The result of this is that either each data owner has to train their own inferior models, or no models are trained at all due to the lack of data.

In short, the world contains vast amounts of data divided into data islands, with few efficient methods to processing the data. If used correctly, this data could improve lives.

## 2.2 Proposed Solution

One of the key features of SL is that data never needs to be shared among the agents in the learning network, which is beneficial for the protection of private data. Given this, SL seems well-suited to the problem at hand. In addition, SL may be able to more effectively utilize the processing power of a network of connected devices. However, SL is still a relatively new algorithm and faces many real-world challenges that have only been addressed individually in existing research.

## 2.3 Goals

To investigate possible optimisations of the SL algorithm on simple problems, whilst adding real world constraints incrementally, and eventually arriving at a robust SL algorithm that addresses many real world issues in one.

1. Create an SL model on a simple dataset such as MNIST without any real world constraints
2. Incrementally add real world problems (listed below) to the SL algorithm
3. Mitigate the effects of each problem by implementing either novel ideas or methods from existing literature
4. Ensure that the end product is as reproducible as possible such that it can be re-implemented for specific use cases

**Real-Word Problems**

Below are some common real-world problems that effect SL:

- **Unevenly distributed features/local bias** - Features in the dataset are not distributed evenly between agents. For instance, when classifying MNIST (a dataset that contains images of digits 0-9), one agent may see more 7s and one may see more 2s
- **Sparsely connected networks** - Each agent does not have direct communication with every other agent, but instead can only communicate with a few neighbours
- **Data transfer limits** - An agent may not be able to send an entire model over the internet, due to connection speed
- **Low processing power agents** - Agents have much lower processing power than a standard machine that one may train a model on. For example, agents may not have access to a GPU

## 2.4 Focussing On Project Goals

At the outset, the project goal covered a very wide topic and was unachievable: It was to *control a swarm of drones to detect objects such as natural disasters, whilst also improving accuracy of the model over time.* However, following multiple stages of research and meetings, the idea was focussed into a smaller, achievable goal. The shift in focus from object detection to SL was primarily motivated by the author's interest in the potential usefulness of a general framework for SL algorithms.

# Progress of Implementations

## 3.1 Basic MNIST classifier

A basic MNIST classifier was constructed using Keras in Python. The classifier consisted of a single agent that trained on the dataset. The primary motivation for this approach was to establish a baseline for comparison with the performance of the SL algorithm. This solution was trained on a single GTX 1660. MNIST was picked as it is simple, well documented, and has a built in function in Keras to load it with minimal code.

## 3.2 Simple swarm learning

In this experiment, a swarm of agents was created using the same model as that used in the Basic MNIST classifier. Each agent had access to the entire dataset for training, and all agents were able to communicate with each other. The agents operated in a loop, where they would each train for one epoch on their own copy of the training set, and then average their model weights with those of their neighbours. This implementation involved a total of five agents, who shared a single GTX 1660 among them. The graphs below depict the accuracy of an agent from the swarm, and also the accuracy of the lone agent.

According to figure 3.1, at the outset, the swarm's accuracy took longer to converge to its highest value than that of the single agent algorithm, which may be attributed to the additional computational overhead associated with SL. After a certain period, the swarm's accuracy improved to the same level as that of the single agent. However, while the single agent began to exhibit signs of overfitting, the swarm's accuracy continued to increase. Ultimately, the swarm achieved a higher level of accuracy than the single agent.

It was observed that the agents achieved their final accuracy in a reduced number of training steps when their training loops were offset by even inter-

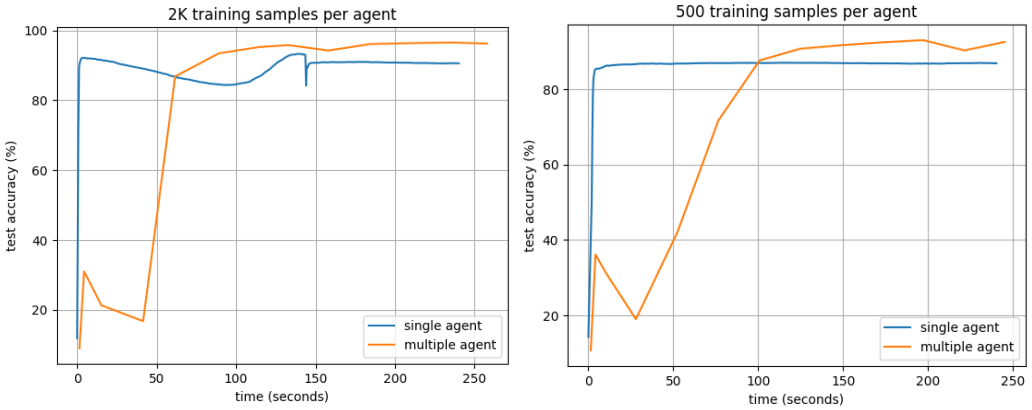

Figure 3.1: Training accuracy over time for an agent from a swarm and a solo agent training with conventional methods, each agent has access to whole dataset.

vals of time. The author hypothesises that this may be due to the potential loss of training epochs when the agents' training is synced, as some of the agents may skip the most recent training when requesting updates.

## 3.3 Reduced dataset swarm learning

The focus of this experiment was to investigate the impact of reducing the amount of data available for training each agent. For this purpose, a fixed number of samples were randomly selected from the training set for each agent, and these samples were not altered throughout the training process. The experiment was conducted with 2000 and 500 samples per agent.

As seem in figure 3.2, in both cases it was observed that the swarm required a longer time to reach its maximum accuracy. However, when it did, the swarm's peak accuracy exceeded that of the single agent by approximately 5%-10%. Both cases show a spike in accuracy near the start of swarm training, followed by a decrease in accuracy, and then a stable increase. This is likely due to the joining of new agents with untrained networks, which effectively add a random model to the pool of weights. A simple solution to this problem would be to have newly joined agents immediately request a model from their neighbouring agents and use that as their base network.




(a) 2000 samples per agent

(b) 500 samples per agent

Figure 3.2: Training accuracy over time for an agent from a swarm and a solo agent training with conventional methods, each agent has access to 2000 or 500 random samples from the dataset.

# Project Planning

## 4.1 Planned Phases of the Project

The project is divided into several phases, which must be completed in sequence. Throughout these phases, the interim and final reports will be developed concurrently as a skeleton.

1. Research - Finding and reading existing literature and developing fully the project idea
2. Simple Implementation - Development of a proof-of-concept SL algorithm trained on a basic dataset, without any complex problems
3. Interim Report - Filling out and finalising the interim report
4. Main development - Iterative development of the SL algorithm will occur, repeating four sub-phases:
   (a) Further research on real-world problem
   (b) Implement real-world problem and test current solution
   (c) Implement mitigations and test / evaluate
   (d) Document any discoveries and evaluations of mitigations
5. Final report - Filling out and finalising the final report
6. Housekeeping - Extra jobs that were not able to be completed in other phases

## 4.2 Completed Work

A Gantt chart was created to plan the completion of different sections of the project prior to the interim stage. This chart was closely followed and proved

to be a valuable tool for time management. The Gantt chart was developed after the initial project brief was submitted.

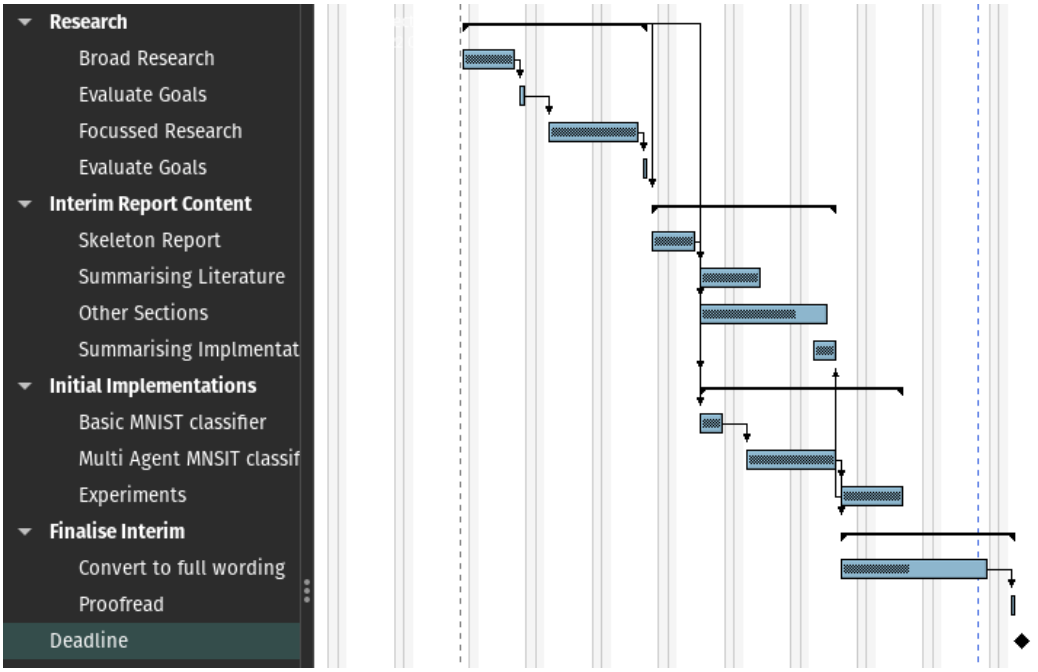



Figure 4.1: Gantt chart for pre-interim planning

## 4.3 Remaining Work

A second Gantt chart was created to plan the project after the interim stage. This chart includes plans for two problems, with a buffer in the form of the housekeeping phase in case of unforeseen delays. Only two problems were planned for however a third may be implemented if the project is ahead of schedule.

## 4.4 Risk Assessment

### 4.4.1 Personal Issues

#### Description

Personal issues which cause the author to be unable to do work, such as illness.

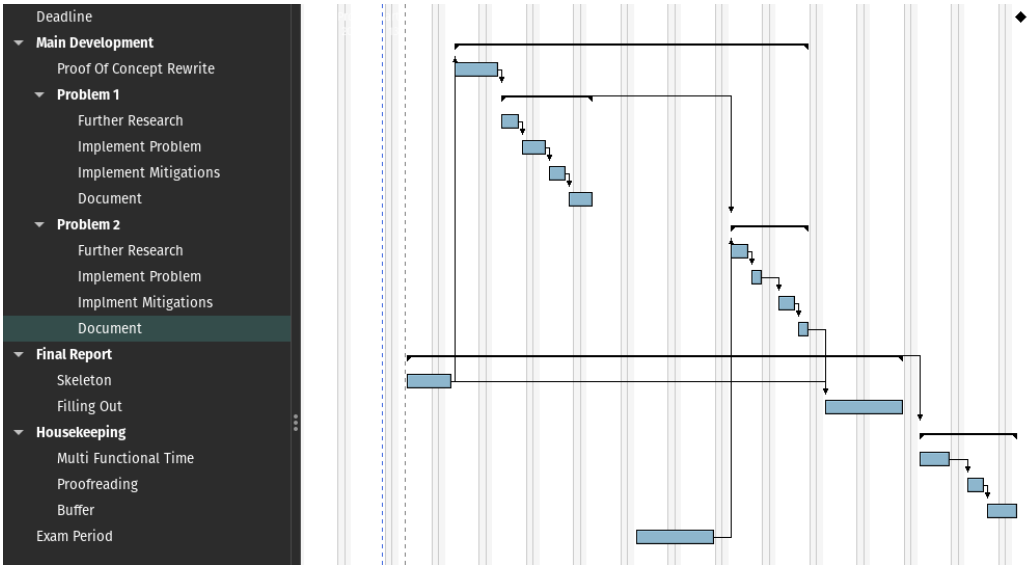



Figure 4.2: Gantt chart for post-interim planning

#### Risk Calculations

*Severity (1-5):* 3
*Likelihood (1-5):* 3
*Overall Risk (1-25):* **9**

#### Mitigation

The codebase will be designed such that individual sections and modules have minimal dependencies on other sections. This means that, even if the author is unable to work for a period of time, some less critical sections can be omitted without significantly impacting the rest of the project.

### 4.4.2 Hardware Failure

#### Description

Failure on the authors local computer of any kind, such as a graphics card or storage breakage.

#### Risk Calculations

*Severity (1-5):* 4
*Likelihood (1-5):* 2

*Overall Risk (1-25):* **8**

#### Mitigation

The project will be regularly backed up to GitHub. If a core component of the work computer breaks, the author has access to a personal laptop and the Zepler Labs. The deep learning environment along with dependencies is backed up to the authors Google Drive in the form of a docker image, so that switching to a new computer would be a smooth process.

### 4.4.3 Algorithm Does Not Work Work

#### Description

The SL algorithm does not function as well as expected. However, this is very unlikely as SL has been proven to work in numerous papers.

#### Risk Calculations

*Severity (1-5):* 5
*Likelihood (1-5):* 1
*Overall Risk (1-25):* **5**

#### Mitigation

It may be possible to shift the project away from SL and onto distributed FL with leader election. FL is more commonly used and therefore has more literature, meaning that it is more likely to be an achievable goal to implement it.

# Conclusion

In conclusion, this paper has shown that SL has the potential to outperform conventional machine learning methods in some cases. While SL presents its own challenges, a plan has been proposed to address these issues and prepare SL for real-world applications. Preliminary testing has been encouraging, and future experiments may provide further evidence of the superiority of SL over conventional machine learning.

# Bibliography


[1] M. Kop, "Machine learning & eu data sharing practices," Stanford-Vienna Transatlantic Technology Law Forum, Transatlantic Antitrust ..., 2020.

[2] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305–311, 2020.

[3] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[4] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[5] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016.

[6] J.-H. Chen, M.-R. Chen, G.-Q. Zeng, and J.-S. Weng, "Bdfl: a byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 8639–8652, 2021.

[7] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, S. Ktena, F. Tran, M. Bitzer, S. Ossowski, N. Casadei, C. Herr, D. Petersheim, U. Behrends, F. Kern, T. Fehlmann, P. Schommers, C. Lehmann, M. Augustin, J. Rybniker, J. Altmüller, N. Mishra, J. P. Bernardes, B. Krämer, L. Bonaguro, J. Schulte-Schrepping, E. De Domenico, C. Siever, M. Kraut, M. Desai, B. Monnet, M. Saridaki, C. M. Siegel, A. Drews, M. Nuesch-Germano, H. Theis, J. Heyckendorf, S. Schreiber, S. Kim-Hellmuth, P. Balfanz, T. Eggermann,

P. Boor, R. Hausmann, H. Kuhn, S. Isfort, J. C. Stingl, G. Schmalzing, C. K. Kuhl, R. Röhrig, G. Marx, S. Uhlig, E. Dahl, D. Müller-Wieland, M. Dreher, N. Marx, J. Nattermann, D. Skowasch, I. Kurth, A. Keller, R. Bals, P. Nürnberg, O. Rieß, P. Rosenstiel, M. G. Netea, F. Theis, S. Mukherjee, M. Backes, A. C. Aschenbrenner, T. Ulas, A. Angelov, A. Bartholomäus, A. Becker, D. Bezdan, C. Blumert, E. Bonifacio, P. Bork, B. Boyke, H. Blum, T. Clavel, M. Colome-Tatche, M. Cornberg, I. A. De La Rosa Velázquez, A. Diefenbach, A. Dilthey, N. Fischer, K. Förstner, S. Franzenburg, J.-S. Frick, G. Gabernet, J. Gagneur, T. Ganzenmueller, M. Gauder, J. Geißert, A. Goesmann, S. Göpel, A. Grundhoff, H. Grundmann, T. Hain, F. Hanses, U. Hehr, A. Heimbach, M. Hoeper, F. Horn, D. Hübschmann, M. Hummel, T. Iftner, A. Iftner, T. Illig, S. Janssen, J. Kalinowski, R. Kallies, B. Kehr, O. T. Keppler, C. Klein, M. Knop, O. Kohlbacher, K. Köhrer, J. Korbel, P. G. Kremsner, D. Kühnert, M. Landthaler, Y. Li, K. U. Ludwig, O. Makarewicz, M. Marz, A. C. McHardy, C. Mertes, M. Münchhoff, S. Nahnsen, M. Nöthen, F. Ntoumi, J. Overmann, S. Peter, K. Pfeffer, I. Pink, A. R. Poetsch, U. Protzer, A. Pühler, N. Rajewsky, M. Ralser, K. Reiche, S. Ripke, U. N. da Rocha, A.-E. Saliba, L. E. Sander, B. Sawitzki, S. Scheithauer, P. Schiffer, J. Schmid-Burgk, W. Schneider, E.-C. Schulte, A. Sczyrba, M. L. Sharaf, Y. Singh, M. Sonnabend, O. Stegle, J. Stoye, J. Vehreschild, T. P. Velavan, J. Vogel, S. Volland, M. von Kleist, A. Walker, J. Walter, D. Wieczorek, S. Winkler, J. Ziebuhr, M. M. B. Breteler, E. J. Giamarellos-Bourboulis, M. Kox, M. Becker, S. Cheran, M. S. Woodacre, E. L. Goh, J. L. Schultze, C.-. A. S. (COVAS), and D. C.-. O. I. (DeCOI), "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, pp. 265–270, Jun 2021.

[8] J. C. Varughese, R. Thenius, T. Schmickl, and F. Wotawa, "Quantification and analysis of the resilience of two swarm intelligent algorithms.," in *GCAI*, pp. 148–161, 2017.

[9] J. Augustine, T. Kulkarni, and S. Sivasubramaniam, "Leader election in sparse dynamic networks with churn," in *2015 IEEE International Parallel and Distributed Processing Symposium*, pp. 347–356, IEEE, 2015.

[10] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning," *CoRR*, vol. abs/2005.01026, 2020.

[11] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[12] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pp. 794–797, IEEE, 2020.

[13] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," 2021.

[14] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," 2019.

[15] Y. Xianjia, J. P. Queralta, J. Heikkonen, and T. Westerlund, "Federated learning in robotic and autonomous systems," *Procedia Computer Science*, vol. 191, pp. 135–142, 2021. The 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), The 16th International Conference on Future Networks and Communications (FNC), The 11th International Conference on Sustainable Energy Information Technology.

[16] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.