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: Mohammad Soorati Second Examiner: ?

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

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 present paper conducts experiments with swarm learning and discusses the results. The experiments focus on adapting swarm learning for real-life situations and addressing issues that arise when applying swarm learning to certain use-cases. Furthermore, this paper examines methods for improving the performance of swarm learning when compared to standard centralized learning, demonstrating that it is a promising technique not only for protecting privacy, but also for making better use of available computing power.

Contents

Chapter 1

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, which can hinder the ability of machine learning models to reach their full potential. Federated Learning (FL) [1] 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 with different merits and drawbacks [1][p. 12]. Federated Averaging (FedAvg) [2] is a commonly used yet simple framework, which breaks 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 beneficially for privacy sensitive tasks compared to conventional machine learning where the data is aggregated in a central location ***CITEME***. However, 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***.

1.1.2 Swarm Learning

Swarm learning (SL) [3] is a subcategory of FL which operates in a completely distributed and decentralised manner. In contrast to FL, SL methods usually use a blockchain based system to coordinate an agreed upon global model ***CITEME***. Similarly to FL though, SL uses repeated iterations during training, but each agent does not have to perform iterations in a synchronised manner:

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. The absence of a central server makes SL more resilient to failures than centralized FL approaches such as FedAvg ***CITEME***. 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 data would not need to be forwarded as frequently throughout the network, as is the case with distributed FL techniques ***CITEME***. Additionally, the removal of the need for a server in SL reduces the likelihood of performance bottlenecks for very large networks.

1.1.3 Challenges for Federated and Swarm Learning

There are numerous challenges associated with implementing FL or SL compared to conventional learning systems. Much research has been done. However, many of the proposed algorithms for FL have not been adapted to work with or tested on SL.

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 [4]. 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 typically 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. 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 different brightness or contrast settings (features). This can negatively impact the training of SL and FL algorithms ***CITEME*** because different agents are effectively proposing solutions to different problems. There is a significant amount of research being conducted on this topic, with various approaches being proposed.

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 [4].

A very different approach is to use batch normalisation layers on each agent [5]. This method is more focussed on non-iid features than labels, but is 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. These cases are rarely ever realised however. The affect 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 ***CITEME***.

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 [2], but only transfer segments of the network at each training step [6]. 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 [6][section. 5.2].

Low Processing Power Agents

FL and SL are promising techniques for robotic systems [7]. However, edge robotics 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.

Chapter 2

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 comfortable or 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 addition to this, the world also has more electronic devices than ever before. As these devices become more and more powerful, using each device to train a machine learning model on a small amount of data becomes a possibility. However, these devices will still not outperform a fast computer alone.

2.2 Proposed Solution

One of the key features of swarm learning 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, swarm learning seems well-suited to the problem at hand. In addition, swarm learning may be able to more effectively utilize the processing power of a network of connected devices. However, swarm learning 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 swarm learning algorithm on simple problems, whilst adding real world constraints incrementally, and eventually arriving at a robust swarm learning algorithm that addresses many real world issues in one.

1. Create a working implementation on a simple dataset of swarm learning
2. Incrementally add real world problems to the swarm learning algorithm
3. Mitigate the effects of the problem on the accuracy of the algorithm 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.

Problems

Below are some real world problems with swarm learning. This is not a comprehensive list and more problems could be researched if necessary.

- **Unevenly distributed features/local bias** - This is where feature in the dataset are not distributed evenly between agents. For instance, when classifying mnist, one agent may see more 7s and one may see more 2s.
- **Sparsely connected networks** - This refers to a situation where each agent does not have direct communication with every other agent, but instead can only communicate with a few neighbours.
- **Data transfer limits** - This is a situation where an agent may not be able to send an entire neural network over the internet, due to connection speed.
- **Low processing power agents** - This situation is where 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

The plan for this project went through multiple iterations before the final plan was formulated:

1. The initial plan was to *control a swarm of drones to detect objects such as natural disasters or people needing help, whilst also improving accuracy of the model over time*

2. After discussion, the plan was changed to be *perform edge processing and distributed object detection on many camera perspectives of an environment to decide where disasters are happening, whilst learning to improve the model over time*

3. Following the initial research phase, the plan was narrowed to *explore ways to optimise swarm learning for distributed detection of a simple abstract object*

4. Finally, after the second phase of research, the settled upon plan became *Investigate possible optimisations of the swarm learning algorithm.* This is the current plan.

The shift of focus away from object detection and towards swarm learning is mainly for two reasons:

1. The author finds the swarm learning aspect of the project to be the most interesting part, especially after researching the subject

2. A general framework of improvements and algorithms on swarm learning is much more useful to the real world than an implementation on a simple simulation.

Chapter 3

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 swarm learning algorithm. This solution was trained on a single GTX 1660.

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, which depict the accuracy of the a single agent from the swarm, and also the accuracy of the lone agent.

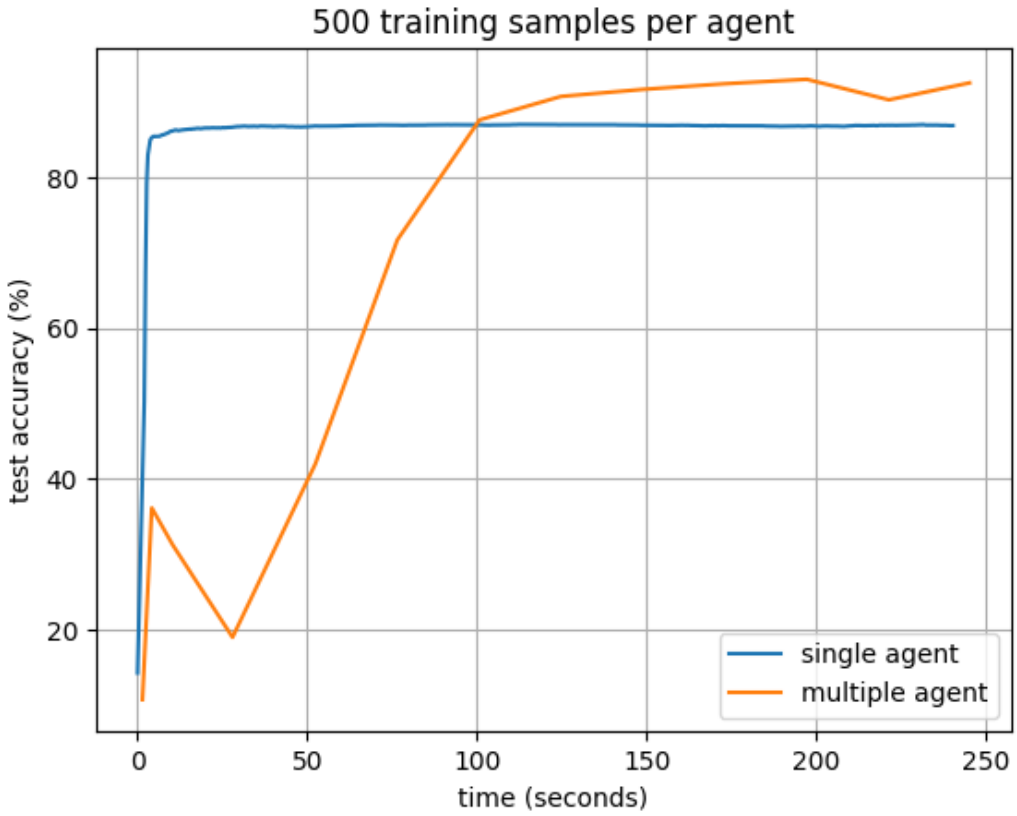
At the outset, the swarm's progress in terms of accuracy was slower than that of the single agent algorithm, which may be attributed to the additional computational overhead associated with swarm learning. After a certain period, the swarm's accuracy improved to the same level as that of the single agent. 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 intervals of time. The author posits 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. Further investigation is necessary to confirm this hypothesis.

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.



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%.

The graphs for 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 inclusion 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. This approach has not been implemented due to time constraints.

Chapter 4

Project Planning

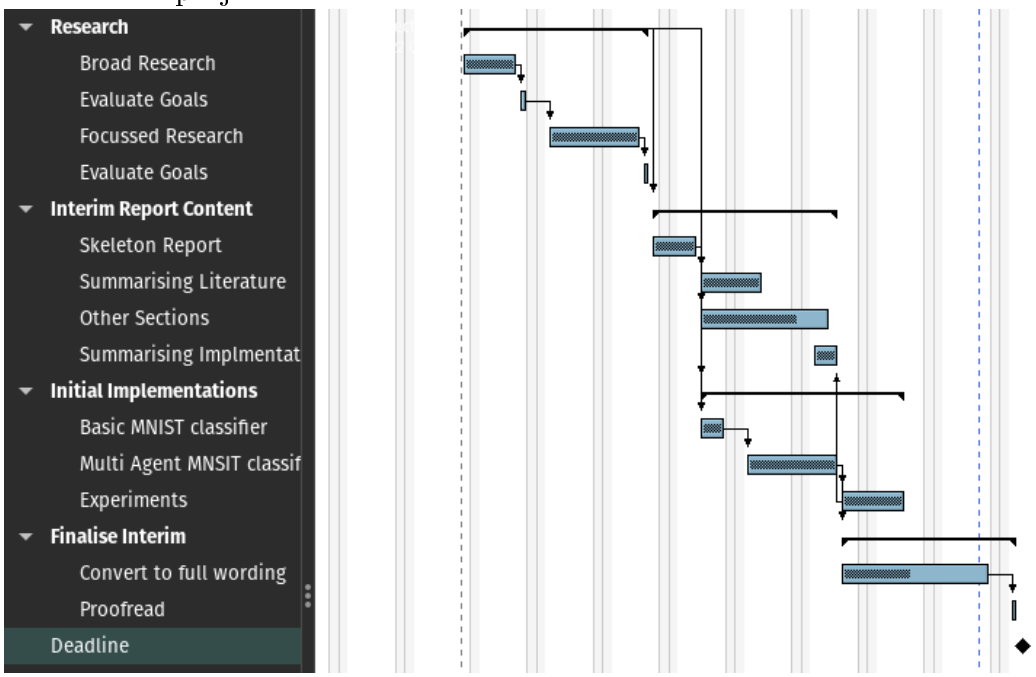
4.1 Planned Phases of the Project

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 - This phase consists of finding and reading existing literature. It is at this time that the project idea is developed fully.
2. Simple Implementation - In this phase, a proof of concept implementation will be developed. It will entail a simple swarm learning algorithm learning on a basic dataset, without any complex problems. This implementation will be build upon in further phases.
3. Interim Report - This phase is focussed on filling out and finalising the interim report.
4. Main development - In this phase, the iterative development of the swarm learning algorithm will occur. This is actually a repetition of 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 - In this phase, the final report will be written up.
6. Housekeeping - This is the final phase which is used for any extra jobs that were not able to be completed in other phases.

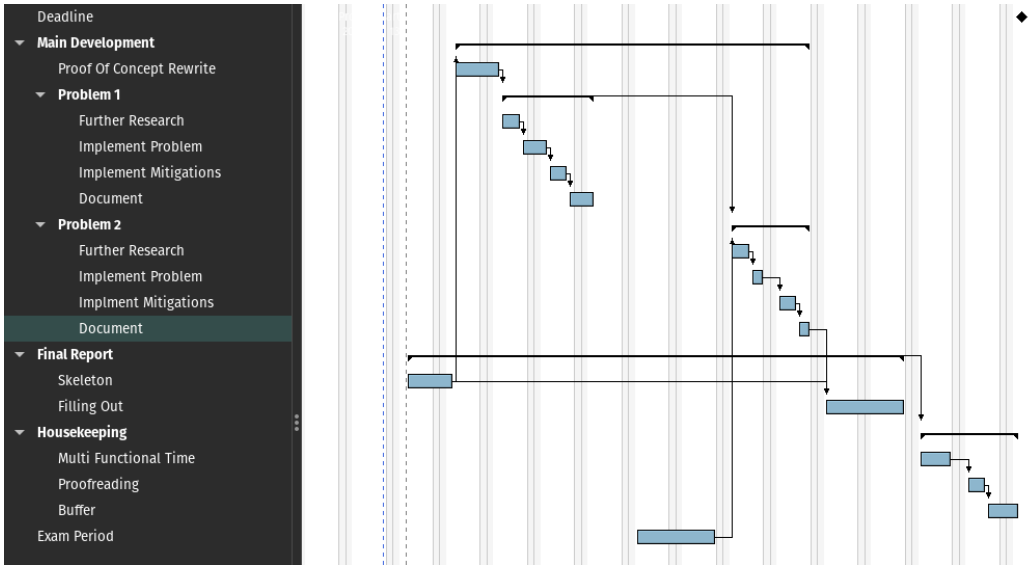
4.2 Completed Work

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.



4.3 Remaining Work

A second Gantt chart was created to plan the project after the interim stage. This chart was developed near the end of the first semester, after a more thorough understanding of the project had been obtained through research. The chart only includes plans for two problems, as it was determined that this is the minimum number required for the scope of the project. If the problems are completed more quickly than anticipated, additional problems will be incorporated into the development phase. A large block of time has been allocated in the housekeeping phase to act as a buffer in case any other aspect of the project takes longer than expected. The ideal outcome is for the project to be submitted at the beginning of this period.



4.4 Risk Assessment

4.4.1 Personal Issues

Description

This risk entails all personal issues which cause the author to be unable to do work, such as illness.

Risk Calculations

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

Mitigation

As much as possible, 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 - Local Computer

Description

This risk entails a 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

To mitigate storage based failures, 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 Fails to Work

Description

This risk describes a situation where the algorithm of swarm learning does not function as well as expected. However, this is very unlikely as swarm learning has been proven to work in numerous papers CITE ME.

Risk Calculations

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

Mitigation

This is not an ideal situation, nevertheless it may be possible to shift the project away from swarm learning and onto federated learning. Federated learning is more commonly used and therefore has more literature, meaning that it is more likely to be an achievable goal to implement it. This would be a last resort though.

Bibliography

[1] 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.

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

[3] 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.

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

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

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

[7] 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.