FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Selection Principles for Federated Machine Learning in Privacy-Sensitive Settings

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Computational Logic

eingereicht von

Anastassiya Pustozerova
Matrikelnummer 11832651

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao. univ. Prof. Dr. Dipl.-Ing. Andreas Rauber
Mitwirkung: Mag.rer.soc.oec. Dipl.-Ing. Rudolf Mayer

Wien, 1. März 2020

Anastassiya Pustozerova	Andreas Rauber

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Selection Principles for Federated Machine Learning in Privacy-Sensitive Settings

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computational Logic

by

Anastassiya Pustozerova
Registration Number 11832651

to the Faculty of Informatics

at the TU Wien

Advisor: Ao. univ. Prof. Dr. Dipl.-Ing. Andreas Rauber
Assistance: Mag.rer.soc.oec. Dipl.-Ing. Rudolf Mayer

Vienna, 1st March, 2020

Anastassiya Pustozerova | Andreas Rauber

Erklärung zur Verfassung der Arbeit

Anastassiya Pustozerova
Barichgasse 28/9, 1030 Vienna, Austria

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. März 2020

Anastassiya Pustozerova

Acknowledgements

At the end of this tremendous journey through my master's program, I feel extremely grateful to all the people who were surrounding me and helping on this path. Thank you for your support and for sharing this beautiful time of my life with me.

First of all, I thank my parents for their decision to invest in my education. Thanks to my mother, my main hero in this world, for constant support through the good times and the bad.

Further, I thank the European Union for providing various possibilities for people to receive education and develop collaborations between different countries and universities. I believe that education and cultural exchange are fundamental parts of the global prosperity.

Thanks to TU Dresden, UNIBZ and TU Wien, the universities at which I was lucky to study.

Thanks to my friends in Saint Petersburg, who were supporting me already during my bachelor studies and who still keep in touch even two years after I have moved away.

Thanks to my friends, who were accompanying me in EMCL. Thank you for your help in studying, for all these lovely moments of moving, cooking, traveling and chilling together.

Thanks to my dear colleagues and friends from SBA, for your moral and technical support while I was working on my thesis.

Thanks to my supervisor Andreas Rauber, for his guidance during the work on this thesis.

Many thanks to Rudolf Mayer for giving me an opportunity to work on this very interesting topic, surrounded by a nice team. Thank you for being just an awesome advisor.

PS. These acknowledgements were written at the shore of the Pacific Ocean, inspired by the sun and sounds of waves.

Kurzfassung

Bedingt durch die rasche Entwicklung von Technologien und die wachsende Menge an Daten, die in mobilen Computersystemen, Cloud Computing und dem Internet der Dinge gesammelt werden, spielt Maschinelles Lernen (Machine Learning) heutzutage eine immer wichtigere Rolle bei der Entscheidungsfindung in vielen Anwendungsgebieten.

Die Daten, die für das Erlernen von Machine Learning Modellen beötigt werden, sind jedoch in der Regel privater Natur. Mit zunehmendem Bewusstsein über die Wichtigkeit des Datenschutzes wird die Frage, wie Daten unter Wahrung der Privatsphäre der Benutzer genutzt werden können, immer relevanter.

Federated Learning ist ein Ansatz, der es ermöglicht, Machin Learning auf verteilten Daten anzuwenden, während der Datenbesitzer die Daten nicht mit Anderen teilen muss. Federated Learning baut hierbei auf der Idee auf, die Modelle lokal bei jedem Dateneigentümer zu trainieren, und nur diese Modelle auszutauschen und zu einem globalem, gemeinsamen Modell zusammenzufassen. Dadurch wird die Menge der auszutauschenden Informationen stark reduziert, und damit die Angriffsfläche für einen Angreifer verringert. Modelle, die während des Federated Learning Prozesses ausgetauscht werden, können jedoch immer noch Informationen über ihre Trainingsdaten preisgeben.

In dieser Diplomarbeit werden Risiken für die Privatsphäre beim Federated Learning bewertet, indem Angriffe auf die Privatsphäre, z.B. durch so-genannte Membership Inference Angriffe, in verschiedenen Federated Learning Szenarien durchgeführt werden.

Es wird weiters eine empirische Evaluierung des Erfolgs von Angriffen sowie von Verteidigungsstrategien. mit dem Ziel, einen Ausgleich zwischen Datenschutz und Nützlichkeit der Modelle zu finden, durchgeführt.

Abstract

Machine learning nowadays plays an important role in decision making in various industries, especially with the rapid development of technologies and data gathered in mobile computing systems, cloud computing, and the Internet of Things. However, the data used for training machine learning models usually has private nature. With increasing concerns of data privacy, the issue of making use of the data while preserving users' privacy is getting more critical. Federated learning is an approach which allows using machine learning on distributed data while the data owner can keep the data on his/her own side. The main idea of federated learning is to train machine learning models locally at each data owner's place, and to aggregate only the model from each participant of the training to a globally shared model. This greatly reduces the amount of information needed to be exchanged, and thus reduces the attack surface for an adversary. However, models that are exchanged during the federated learning process can still leak information about their training data. In this work, we evaluate privacy risks in federated learning by performing privacy attacks, e.g. membership inference attacks, in different federated learning settings. We design empirical evaluation of the success of attacks and mitigation strategies, aiming for a trade-off between privacy and effectiveness of the models.

Contents

CHAPTER 1

Introduction

1.1 Motivation and problem statement

With recent developments of technologies such as mobile computing systems, cloud computing, and internet of things (IoT), the opportunities for data collection and processing have been increasing rapidly. Machine learning is used to obtain insights and predictions based on data. It already plays an important role in decision-making in various industries, such as health care, media, education, manufacturing, or marketing. However, data processed in such industries often contains private or sensitive information. Leaks or thefts of data may cause a loss of customer trust, or reveal advantageous technologies to competitors. Therefore, it poses a threat to businesses. Privacy concerns are generally rising and becoming a global issue. On May 25th, 2018 the EU Parliament has enforced "The EU General Data Protection Regulation" (`https://gdpr-info.eu/`), intending to protect and empower individual data privacy. Different organizations and companies have faced issues in regards to changing their data policy, e.g. by implementing data anonymization and secure computation tools to provide users with data privacy. The main challenge is ensuring users' privacy and at the same time make maximum use of their data to improve services. The goal is therefore to design privacy preserving frameworks that accumulate and securely process confidential data.

Global usage of mobile computing devices with growing computational capabilities empowers the development of mobile edge computing. One of the approaches that helps to keep data collection and processing closer to the user's equipment is *federated learning*. Addressing the issues of data ownership and locality, federated learning allows preserving users' privacy, by applying machine learning over distributed data. In traditional machine learning settings, data is centralized on a server, which entails concerns of privacy and responsibility for the data. In federated learning, model parameters are independently and locally learned, based on each client's data, thus avoiding data centralization. These parameters are aggregated to a new, globally shared model. Since model parameters

represent only a rough abstraction of the data, privacy protection is considerably simplified and the risk of information leaks is reduced.

Federated learning can be considered with a largely varying number of processing nodes over which data is distributed: from several institutes collecting data, but not being able to share it, to billions of mobile devices with owners requiring their data to stay with them.

However, federated learning has as well several challenges. One of them is the additional effort in communication, which could slow down the learning process. Another problem is the reduction in model parameter information content, which could jeopardize learning quality. Moreover, there are still possibilities to attack the data through possession of access to the models, therefore additional privacy and security mechanisms may be necessary.

In this thesis, we develop guidelines for federated learning approaches, and define parameter selection principles for effective, efficient and privacy-preserving usage of federated learning. The principles are formulated based on the analysis of the results of an empirical evaluation of the federated learning method. The evaluation includes the analysis of federated learning efficiency and effectiveness and privacy preserving properties. The privacy level is evaluated by simulation attacks with the goal of inferring information from machine learning models about their training set. We also propose and evaluate mitigation strategies for improving privacy properties of federated learning.

1.2 Research questions

Within the scope of this thesis, we investigate the following research questions:

1. What is the effect of federated learning on models efficiency and effectiveness compared to models trained on centralized data?
2. Which privacy risks are in federated learning and how they can be mitigated?
3. What are the most suitable settings for federated learning to ensure privacy under membership inference attacks while preserving a level of model effectiveness and efficiency close to machine learning on centralized data?

1.3 Main results

We consider federated learning organized in a sequential and parallel manner. We can show that models trained in federated learning settings reach, and in some cases even top, the performance of the models trained on centralized data. By performing a membership inference attack in federated learning settings we provide evidence that the method does not exclude privacy risks, as the models that are shared in the federated learning process may still leak information about their training set. Additionally, we analysed

different mitigation strategies to improve the privacy properties of federated learning. In particular, we evaluated the effect of adding noise to the data, with variants of decreasing the amount of added noise at each iteration towards the end of the training process.

1.4 Structure of the work

In Chapter 2, we discuss the federated learning state-of-art, its categorisation and applications. Further, we also review privacy attacks on machine learning and discuss mitigation strategies for improving machine learning models privacy properties.

Chapter 3 is dedicated to describing the experimental setup, including the choice of datasets and machine learning models. We also discuss different approaches to federated learning architecture including sequential and parallel federated learning settings. Subsequently, we describe the membership inference attack, and develop scenarios for the attack in federated learning settings. Moreover, we discuss the structure of possible mitigation strategies.

Chapter 4 respond to the core finding of the study. We compare the performance of federated learning related to centralized learning, including models' accuracy and their training time. We perform a membership inference attack in centralized settings, as well as in sequential and parallel federated learning. Analyzing the attack results, we evaluate several mitigation strategies and their influence on attack performance. Based on the experimental evaluation, we give recommendations for a federated learning setup in order to find a trade-off between utility and privacy of the model.

In Chapter 5, we summarize the work, give conclusions from the obtained results and discuss future work.

CHAPTER 2

Federated learning applications and privacy issues

In this chapter, we give a brief introduction to supervised machine learning, discuss approaches to privacy-preserving machine learning. Then we thoroughly describe federated learning approach, its categorisation and applications. Last sections is dedicated to the review of existing attacks on privacy in machine learning and also mitigation strategies.

2.1 Supervised machine learning

Arthur Samuel popularized the term *Machine Learning* in 1959 presenting it as the method for optimizing strategies for checker playing program [1]. The main idea behind machine learning is to give computers the ability to solve problems without explicitly specifying (programming) the solution algorithm.

Machine learning is actively used in various applications nowadays: smart personal assistants, recommendation systems, fraud detection, spam detection and other prediction tasks. Essentially, *Machine Learning* is the process of learning a function that takes an input and returns a prediction. The function tries to learn the inherent structures in the data. There is *supervised* and *unsupervised* machine learning. Unsupervised machine learning tries to e.g. find structures in the data when there are no defined output. It includes methods like principal component, cluster analysis and others. In supervised machine learning, the function is learnt based on the examples of known inputs and outputs. Supervised machine learning considers *classification* and *regression* tasks. Regression is the task of predicting a numeric score for an input. In a classification task, the output is categorised into a limited and fixed number of classes.

In this work, we consider classification tasks on structured data. As an input we have records (n rows) from a table denoted as X_j^i, where $j = 1, ..., n$ is the number of instances

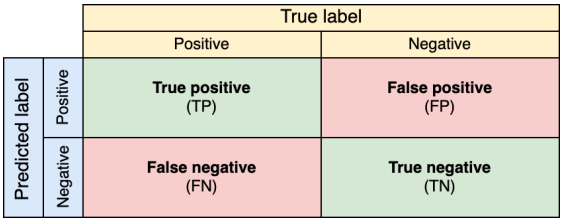


Figure 2.1: Confusion matrix for binary classification task

and i is the number of attributes. Output is the correct label (or class) of the record - y_j. Machine learning algorithms allow learning the function f that will take the record and give a correct label as an output: $f : X_j^i \rightarrow y_j$. Some of these algorithms are k-Nearest Neighbours, Support Vector Machine [2], Decision Tree, Random Forest [3] and Multi-layer Perceptron (or more generally – Neural Networks) e.g. [4].

In supervised machine learning, the desired function (model) is learnt on so-called training data. Effectiveness of a model is evaluated on testing (validation) data, which was not used for training. The *confusion matrix* (sometimes also referred to as error matrix), is a special form of a contingency table representing the number of true and false predictions. It is used as a basis to evaluate model effectiveness. For a binary classification task, the confusion matrix is represented in Figure 2.1. For our classification tasks we evaluated the models' effectiveness using several scores:

- *Accuracy* – the ratio of correctly labelled records to the number of all records.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{\#samples} \tag{2.1}$$

- *Precision* (*True positive rate*) – the ratio of the records which were correctly classified as one class, to all the records that were classified as this class.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

- *Recall* (*Sensitivity*) – the ratio of records correctly classified as one class to the other records of this class that were misclassified.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

- *F1-score* combines precision and recall to one score, showing their harmonic mean.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.4}$$

We also evaluate the efficiency of models by measuring the amount of training time which was necessary to reach a sufficient level of model effectiveness.

2.2 Privacy-preserving machine learning

Machine learning creates a generalized function by processing evidence and helps to make predictions based on this evidence. Therefore, it demands more data for better predictions. Despite that, in 2020, vast amounts of data are generated every second, privacy concerns and regulations limiting access to this data are arising. Finding data for training is one of the biggest challenges in machine learning nowadays. Machine learning is not anymore only concerned with the quality of predictions, but also about the privacy of the data, based on which predictions are made. The issue is how to make use of data without violation the privacy of data owners. In this work, we consider federated learning as a tool allowing solving this issue. We evaluate to which extend federated learning preserves the privacy of data owners and maintains the quality of predictions.

In principle, there are two main approaches for preserving privacy in machine learning: *privacy preserving data publishing*, which aims at removing sensitive information from the data prior to learning, and *privacy preserving computation*, which aims at not revealing the data used during the machine learning process, but only the final outcome.

Privacy preserving data publishing includes techniques developed to prevent revealing sensitive information about individuals from a dataset while allowing to process and infer knowledge from the data. These techniques are usually based on modifications of the original dataset. For a detailed and comprehensive overview on techniques, consult e.g. [5].

As one of the most well-known techniques, k-anonymity allows preserving the anonymity of individuals by modifying the quasi-identifiers of the dataset (*quasi-identifiers* are attributes that become personal identifiers when combined) in a way that for each instance, there are at least $k-1$ other entities from which the original instance cannot be distinguished [6]. These modifications includes attributes suppression (deleting quasi-identifier attributes) and generalization, e.g. ZIP code can be modified by removing the least significant digits, or can be suppressed by replacing with a $^{|*|}$. The method is sensitive to outliers, e.g. because it is difficult to generalise quasi-identifiers. As k-anonymity does not provide any randomization, an intruder can infer some knowledge from the dataset, having some information about the records. The method also was shown to be vulnerable to other inference attacks [7].

l-diversity addresses some problems of k-anonymity and gives stronger privacy guarantees by enforcing that the values of a sensitive attributes must be well-represented in each *equivalence class* (a set of identical quasi-identifying attributes) [8]. For example, not ever sample in the same equivalence class should have "lung cancer" as the sensitive value.

t-closeness was developed to prevent attribute disclosure, which is a vulnerability of l-diversity. By separation of the information gain of an observer into two parts, the whole population and specific individuals, Li et al. [9] introduce a threshold t on the distance between the distribution of a sensitive attribute in an equivalence class and the distribution of this attribute in the whole dataset.

All these approaches are based on the idea of data generalization and suppression, which causes a loss of information. The trade-off between privacy and utility is rather difficult to optimize. Moreover, these methods are not resistant to membership disclosure [10].

Differential privacy allows to hide individuals' information and, at the same time, preserves statistical properties of a dataset [11]. An algorithm analysing a dataset is considered differentially private if by looking at its output, one cannot infer if some record was in the dataset or not. This approach has limitations in settings where the same or similar analysis is needed to be repeated several times, as the privacy budget may be used up quickly. A differentially private form of stochastic gradient was proposed for training neural network by Abadi et al. [12]. However, Bagdasaryan et al. [13] showed that this approach reduces the accuracy of the model on underrepresented classes significantly more that on the other classes.

Another shortcoming of the methods listed above is that neither of them provides full privacy protection of data if the interest of the attacker is in the *global* properties of a dataset.

Privacy preserving computation allows processing the data while keeping it inaccessible to other parties that participate in a computation, or adversaries. This is a widely used methodology for privacy protection and secure data aggregation [14].

One consider different types of adversaries in privacy preserving computations:

- **Semi-honest** (or honest-but-curious) adversaries comply with the protocol, but try to gather more information than allowed from the protocol.
- **Malicious** adversaries arbitrarily deviate from the protocol to breach security. Protocols that achieve security in this model provide a high-security guarantee.

Homomorphic encryption can be used to outsource computation and storage while preserving the privacy of data as it allows to process encrypted data without the need for decryption by the third party. One problem of this approach is that it has limitations on supporting a wide range of operations [15]. Another big issue is the speed of computations that can be several orders of magnitude slower, which renders the method unattractive for industrial application.

Secure multi-party computation (SMPC) allows several parties to jointly and securely compute a function over their inputs, while keeping the data of each party private. The approach has weaker privacy guaranties in specific settings - e.g. when there are two malicious parties out of in total three participants, joining their knowledge to infer information about the third participant's data [14], [16].

Federated learning, like SMPC, can be used in cases when training data is distributed among several parties. This approach allows processing training data locally at each party, and exchanges model parameters only. Unlike homomorphic encryption and

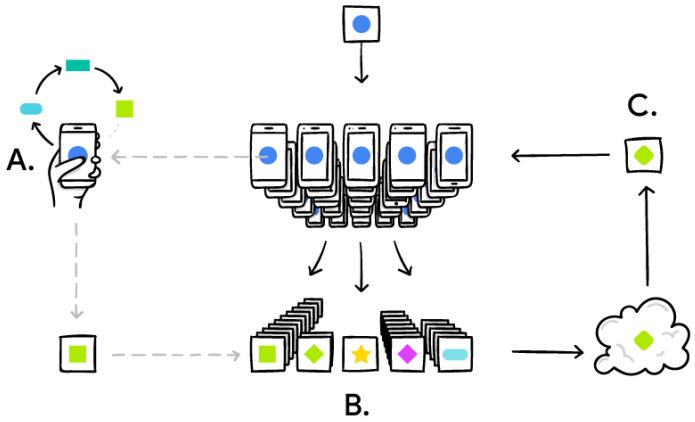


Figure 2.2: Parallel federate learning on mobile devices
(Source: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html)

secure multi-party computation, federated learning by itself, does not guarantee security and privacy, and is vulnerable to different types of attacks. However, the approach is relatively easily implementable and privacy and security risks may be mitigated by several techniques. In the following sections, we thoroughly describe federated learning.

2.3 Federated learning and its applications

The term *federated learning* was popularized in 2017 by McMahan et al. [17] as a method which enables to leave data distributed on mobile devices and learn a global model based on aggregated parameters from the local training. Its main goal was to reduce privacy and security risks by excluding the possibility of obtaining data from a server. The authors proposed federated learning for the tasks of image recognition and language modelling, and specifically for deep learning models applied in these tasks.

Figure 2.2 shows an exemplary federated learning process:

- Initialization of a global model
- A. Client trains model locally on her data,
- B. Models from all users are collected at a central coordinator,
- C. A new global model is aggregated from the collected individuals' models and sent back to clients for the following local training.

The authors proposed the *FedAvg* algorithm, which performs models averaging of the locally trained models based on a multi-layer perceptron and stochastic gradient descent.

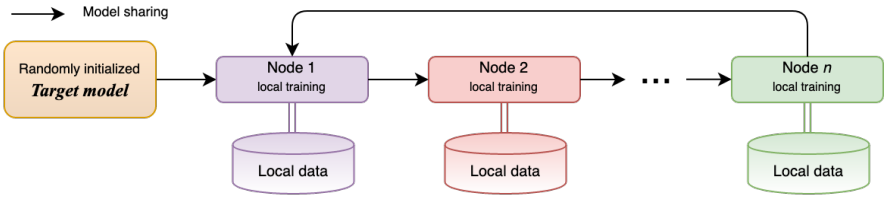


Figure 2.3: Sequential federated learning. A randomly initialized model is locally trained at the first client and then passed to the next node in the sequence. After completing a full round of n nodes, the model is passed again to the first node for repeating the training process.

FedAvg combines local gradients on a server, then computes model weights updates for each collected local gradient, and then takes weighted averaging of these updates. The local training of the same, randomly initialized model at different clients followed by aggregation and averaging was shown to achieve substantially lower loss relative to the independent training of small subsets of the full dataset.

2.3.1 Federated learning categorisation

There are several ways to categorize federated learning, based on the properties of the datasets the participants in the federated learning hold:

- **Horizontal federated learning** stands for the case when different datasets are matching in feature space, but don't share the same records.
- **Vertical federated learning** is used in the scenario when several datasets share the same records, but consist of different feature space.

Federated learning can be distributed over a rather small or large amount of processing nodes. Depending on the task, different types of learning the global model can be chosen:

- **Sequential** federated learning, or cyclic incremental learning [18], can be applied in scenarios with a small number of processing nodes. In this case, a randomly initialized model is sent to one of the learning participants for local training. After training, the node transfers the updated model to the next party in the sequence. This process (cf. Figure 2.3) may take several cycles, similar to batch training in neural networks. The core benefit of this approach is that it does not require the presence of an aggregator. However, it is not a feasible approach with a large number of clients. In the case when the participants of federated learning are malicious users, they can corrupt the model or be a threat to the privacy of the preceding node.

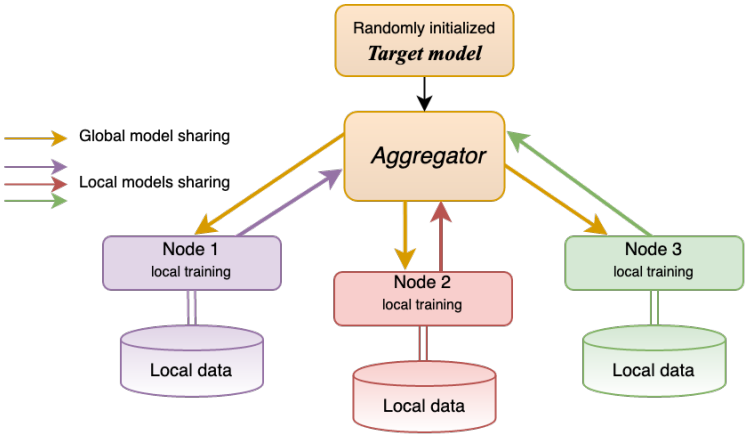


Figure 2.4: Parallel federated learning. The *aggregator* initializes a global model with random weights and shares it to every node in the setting. Each node trains the model in parallel on its local data and then returns it to the secure aggregator. From the locally trained models, a new global model is aggregated and shared to the clients for the following training cycle.

- **Parallel federated learning**, which is often used as an equivalent to federated learning, is based on the idea of aggregating parameters from different nodes to a global model at a (secure) aggregator, and then sharing this global model to each of the participants of the federated learning (see Figure 2.4). McMahan et al. [17] proposed the *FedAvg* algorithm, which allows to average the weights of several models via an aggregator to obtain a global model. Having a large number of clients, one may use all, or choose the subset of participants and send them the same, randomly initialized model. After local training at each party, the model parameters are aggregated by taking the average and updated at each node. At the next round, another subset of nodes is taken for the models aggregation, and so on. This approach enables using data from a massive number of nodes (clients). However, it demands a trusted aggregator.

In this work we consider horizontal federated learning in parallel and sequential settings.

2.3.2 Applications

Nowadays, smartphones and tablets are the most commonly used mobile computing devices. They posses decent computational capacity, as well as access to a large amount of their users' data. Analysing such data and building models trained on it makes devices "smarter" and more efficient in use. Traditionally, when one wants to analyse data from a group of different sources, she needs to collect the data on a server for processing, which may threaten the privacy of the users data. Federated learning was proposed as a method by McMahan et al. [17] allows to process data locally on mobile devices, sharing only

model updates. Federated learning allows using computational power at the edge (closer to the location where data originates) while keeping sensitive data with its owner.

Hard et al. applied parallel federated learning to the task of next-word prediction in a virtual keyboard for smartphones with a recurrent neural network [19]. Similar language modelling tasks for mobile keyboard suggestions were considered in [20]. The authors proposed an *attentive aggregation* method for federated learning, which takes into account the distances between local models and a global one. Popov et al. [21] showed on the task of language modelling that local training on a user's device, combined with a random model averaging from other users, improves language model performance in the next word prediction task.

Parallel federated learning has further applications for problems in IoT. It was used for training deep reinforcement learning agents and allowed to reduce the amount of communicating data [22]. Nishio et al. [23] improved the federated learning protocol for making the process more efficient in a heterogeneous mobile devices network. In the scenario when resources are limited, their protocol enables to select the optimal clients, whose updates are then collected and aggregated for improving the global model. Mills et al. [24] developed a communication-efficient version of *FedAvg* by adapting federated learning with Adam optimizer and a tool for compressing uploaded models, which helped to significantly reduce the number of federated learning cycles before convergence and communication coasts.

In spite of federated learning having been inspired by an idea of processing data distributed over mobile devices, this approach has been used also in other settings, e.g. by collaborations of institutes for different tasks based in the medical imaging domain. Gathering medical data is a challenging task due to various legal, technical and privacy issues, especially among international institutions, which makes federated learning a suitable solution for this case.

Intel, in collaboration with The Center for Biomedical Image Computing and Analytics (CBICA), showed that federated learning could train a deep learning model reaching up to 99% of the accuracy of the same model trained over centralized data [18]. They compared the performance of parallel and sequential (cyclic incremental) federated learning and showed that parallel federated learning achieves better performance.

An approach similar to federated learning was used by Deist et al. as a distributed learning methodology for analyzing lung cancer patient data from five different institutions[25]. The authors used the Alternating Direction Method of Multipliers (ADMM), an algorithm that solves convex optimization problems splitting them into smaller parts. It was used for learning support vector machine models. The performance of distributed learning algorithm was compared to a centralized one and was shown to be less efficient due to the communication coasts.

Jochems et al. [26] used federated learning with a Bayesian network model for horizontally partitioned data within several medical institutions. They had data from 287 patients, who had lung cancer, and were predicting dyspnea, a common side effect after radiotherapy

treatment of lung cancer. The data was stored at five different institutions in Europe. The goal was to train a well-predicting model on the data from all five institutions, while avoiding data sharing. First, they defined the structure of the network based on previous studies and adapted the network probabilities by aggregating feedback from each hospital. The contribution of each partner was proportional to the number of patients in the institution. In the second round, they send the averaged network to each party, where they locally trained probabilities. The local probabilities were collected and averaged. They showed that the federated learning approach allows to achieve performance close to centralized learning, while avoiding sharing patients' data.

Federated learning was also used for analyzing brain diseases with brain images data from different institutions, which could not be shared due to privacy and security reasons [27]. The authors proposed a framework for data standardization, confounding factors correction, and high-dimensional features variability analysis in federated learning.

2.3.3 Frameworks

There are several implementations of federated learning, which are still developing their functionality and efficiency, but can be used for federated learning applications.

- **Pysyft**[1] is a Python library for secure and private Deep Learning, containing an implementation of federated learning, secure multiparty computation, and differential privacy. The library is based on Pytorch[2], an open-source machine learning framework.
- **Tensorflow Federated**[3] is an open-source framework for machine learning on decentralized data developed by Google, containing an implementation of the *Federated Averaging* algorithm proposed in [17]. They also made a funny comic describing in an easy way how federated learning works [4].
- **FedAI**[5] is an open-source project initiated by Webank's AI Department to provide secure computing. It implements secure computation protocols based on homomorphic encryption and secure multi-party computation. It supports federated learning architectures and secure computation of various machine learning algorithms, including logistic regression, tree-based algorithms, deep learning [28].

In our experiments, we use Pysyft and our own implementation of federated learning, based on the Pytorch framework. The choice of the framework was made based on the availability of tutorials. Moreover, Pysyft has an implementation of Federated Learning with SMPC, which one can use for mitigating privacy leaks in federated learning.

[1] https://github.com/OpenMined/PySyft
[2] https://pytorch.org/
[3] https://www.tensorflow.org/federated
[4] https://federated.withgoogle.com/
[5] https://www.fedai.org/

2.4 Federated learning challenges

In the following section, we discuss particular challenges that appear in federated learning application.

- **Communication costs.** A federated learning setting may consist of a large number of processing nodes, which makes communication of models' parameters one of the core problems. Konečný et al. proposed methods which allow to reduce the communication cost in federated learning [29]. They considered the case of sparse not independent and identically distributed data (non-IID data) among a large number of devices and showed that the global model can be trained with a small number of rounds communicating the parameters. Sattler et al. [30] developed a new communication protocol which was shown to be more robust compared to *FedAvg*.

- **Unbalanced data.** Different parties of the federated learning process may generate different amounts of data. Moreover, in the case when the number of processing nodes is large, each node may contain data from a significantly different distribution and will not be able to represent the general training distribution. This may increase the complexity of the task, as well as the communication costs. The Non-IIDness problem in federated learning is frequently discussed in the literature ([30], [22], [29]). Li et al. showed that there is a trade-off between *FedAvg* convergence rate and communication efficiency [31].

- **Systems Heterogeneity.** Devices collaborating in federated learning may have different communication and computational capacities, e.g. memory, network connectivity (when devices are offline or using a slow or an expensive connection) or battery capacity. This may lead to unstable model transferring, therefore the risks of unavailability must be taken into account. Nishio et al. [23] addressed this problem by developing a new federated learning protocol *FedCS* (Federated learning with client selection). The protocol enables to reduce training time relatively to the original federated learning protocol while maintaining high effectiveness of the models.

- **Security risks.** There are possibilities to intrude the federated learning process with malicious purposes. For instance, poisoning attacks, when an adversary manipulates the training data of the model to change models prediction results in the desired way, either to reduce accuracy or misclassify some particular input [32]. This attack was already considered in federated learning setting by Xie et al. [33]. Another type of attack on models is when an intruder throws an adversarial input to the model, modified in the way to evade detection and cause misclassification [34], [35]. There is also a threat of model stealing when machine learning model has high commercial value [36], an attack which has not been studied for federated settings yet.

- **Privacy risks.** While federated learning enables to leave the private data with its owner, there are still risks of data leakage from the models. Intruders may attack the machine learning models that are shared in the federated learning process, to steal sensitive information. Examples of such attacks are membership inference [37], model inversion [38] or attribute inference[39] (described in more details in Section 2.5). In this thesis, we consider membership inference attack on federated learning in different scenarios, and also test several mitigation strategies.

2.5 Privacy attacks on machine learning models

The main goal of federated learning is to preserve data privacy. However, the distributed analysis of the data may not be enough guarantee for this. The risks may appear on the model level, if some sensitive information can be extracted from the model.

One distinguishes several scenarios based on model accessibility while performing privacy attacks on machine learning models:

- *White-box access* is the full access to the model, including information about its type, architecture and parameters.
- *Black-box access* is the access only to the model as a service, enabling to use it for predictions without any access to information about the model type, architecture and parameters.
- *Grey-box access* is access to some information about the model, e.g. model's weights or models hyperparameters (optimizer, learning rate).

A *model inversion* attack tries to recreate data samples that represent the underlying original objects. This has been shown to work in very specific settings, such as in the case of recreating pictures of the persons to be identified by a face recognition system [38]. While the attack is powerful, it is more difficult to perform in general settings, especially if there is no correspondence between a specific object of interest, and a class. A system that e.g. distinguishes between classes that represent large groups of individuals is unlikely to reveal information about a single member of that group. This attack also has also been used as a measure of achieved privacy, e.g. Baumhauer et al. [40] evaluated the quality of *linear filtration* (an algorithm for sanitization of classification models) by comparing performance of model inversion attacks.

Song et al. [41] show how machine learning models can leak information about the training data set with black-box access to the model. They apply specific modifications to the training algorithm, e.g. encoding correlated values, and synthesizing maliciously augmented data. They demonstrated that it is possible to reveal a subset of the training data while preserving a high quality of the machine learning models. This scenario considers the case when a client (data holder) uses a model trained by some provider,

who wants to get sensitive data from the client. It was shown that the provider can maliciously train this modified model which will reveal data from the client's training set.

Another type of attack is *attribute inference* (or disclosure). The attacker has partial information about a record, based on which, in combination with other information, she infers unknown attributes about the record. For instance, recommendation systems may leak users' private information when the adversary has access to their public data. Weinsberg et al. [39] showed how to infer the gender of a user from a recommendation system based on the ratings which the user gives.

In [42], the authors showed how to infer the statistical properties of a training set from a machine learning model by training a meta-classifier. This meta-classifier is aimed to detect the changes of the main machine learning classifier after training on the target training data, and infer from it information about the training data. The attack is also resistant against differential privacy, as it is targeted to general statistical information of a dataset.

Another vulnerability of machine learning models was revealed by the membership inference attack in [37]. The main idea of the attack is, having a record or data instance, to infer if it was in the model's training set or not. For predicting the membership, the authors train an *attack model* on the output received from so-called *shadow models*, which are build to imitate the behavior of a target model. The authors showed that even with only black-box access to the target model the precision of the attack may reach 93,5 %. The main reasons for leakage, in that case, were model overfitting and a small diversity of the training data. Also, the attack showed better performance on classification tasks with larger number of target classes.

Truex et al. [43] compared the performance of membership inference with different machine learning algorithms and also considered the case of a membership attack in federated learning settings. They showed that a malicious member of a federated leaning process can perform a membership attack, and that the bigger the difference between distributed datasets, the more vulnerable they are to insider membership inference.

Salem et al. [44] showed that membership inference is even more powerful and an intruder can perform an effective attack not only without having information about the model type, architecture and data distribution. They further used a more computationally efficient way for the attack, without training several shadow models, but only one, and by training this shadow model on a dataset which is not related to the target one.

Melis et al. [45] performed several attacks in federated learning settings. They showed that malicious participants of the training can perform membership inference attack and also reveal properties characterizing a subset of the data. For instance, for a dataset of individuals' photos with a gender classification task, they could infer from the model if a person was wearing glasses, though it was out of the scope of the target model. They designed inference attacks based on leakage from the gradients.

Nasr et al. [46] considered membership inference attack in white-box settings. They

developed a new attack algorithm which learns on gradient vectors of the models trained by using stochastic gradient descent (SGD). In [37], models which overfit the training data were shown to be more vulnerable to membership inference with black-box access. In [46], the authors showed that even models which generalize well are vulnerable to white-box access membership inference. They considered membership inference attack in federated learning settings performed by one of the participants of federated learning. They showed that an insider-intruder can perform membership inference attack against other participants even if a global model has high effectiveness.

In this work, we perform membership inference attack (based on [37]) in a federated learning setting with different scenarios, and compare the performance with an attack on models trained on centralized data. Unlike in previous, related work [43], we perform experiments with broader variation in federated learning settings, e.g. we consider sequential learning, a different number of federated learning participants, and consider datasets with different number of classes in the classification task. Moreover, we evaluate mitigation strategy as noise addition in federated learning settings and its influence on overall model performance.

2.6 Improving privacy properties in federated learning

A possible solution to prevent a machine learning model from leaking information of its training set, is to apply some other privacy-preserving or secure computation techniques. For instance, in [47], federated learning was complemented with secure multiparty computation and differential privacy, which provided end-to-end privacy guarantees with respect to the participants.

Bonawitz et al. [48] proposed a secure aggregation protocol for computing the weighted averages based on SMPC. The authors considered the task of next-word prediction using deep neural networks. They showed that a secure aggregation protocol, which allows to compute sums of individual's models updates with SMPC, guarantees that the adversary will be able to infer if one or more users would use a particular word, but will not be able to distinguish which one.

Shokri et al. [49] implemented a federated learning system with *Selective Stochastic Gradient Descent* (SSGD), where participants of the learning share selected gradients to other parties. To limit the availability of knowledge inference from shared gradients, they also applied differential privacy on the shared parameters. They showed that the accuracy of a differentially private selective stochastic gradient descent with a larger number of participants is higher than in the case of standalone training.

Based on the system presented in [49], Phong et al. [50] utilized Selective Stochastic Gradient Descent with additive homomorphic encryption. They showed that shared gradients may leak information about training data by recovering pictures of the original dataset from the gradients. Therefore, the authors developed a system with homomorphic encryption, which allows for preservation of accuracy of the training while guaranteeing

security against an honest-but-curious aggregator. In this system, the gradients are first encrypted and then stored on the server. Computations over the gradients are enabled by additive homomorphic property. The system was shown to be secure against honest-but-curious server, which is not guaranteed in [49].

Several defence mechanisms were tested in [37], but were ineffective against membership inference, e.g. increasing entropy of the prediction vector, limiting the prediction vector to some number of classes, regularization, etc. Truex et al. [43] discussed the potential application of differential privacy as mitigation for the attack. Salem et al. [44] showed that adding a drop out layer to the network deteriorates the performance of membership inference. However with a dropout of more than 0.5, the target model was loosing effectiveness and performed badly on the main classification task.

CHAPTER 3

Membership inference attacks and mitigation strategies in federated learning

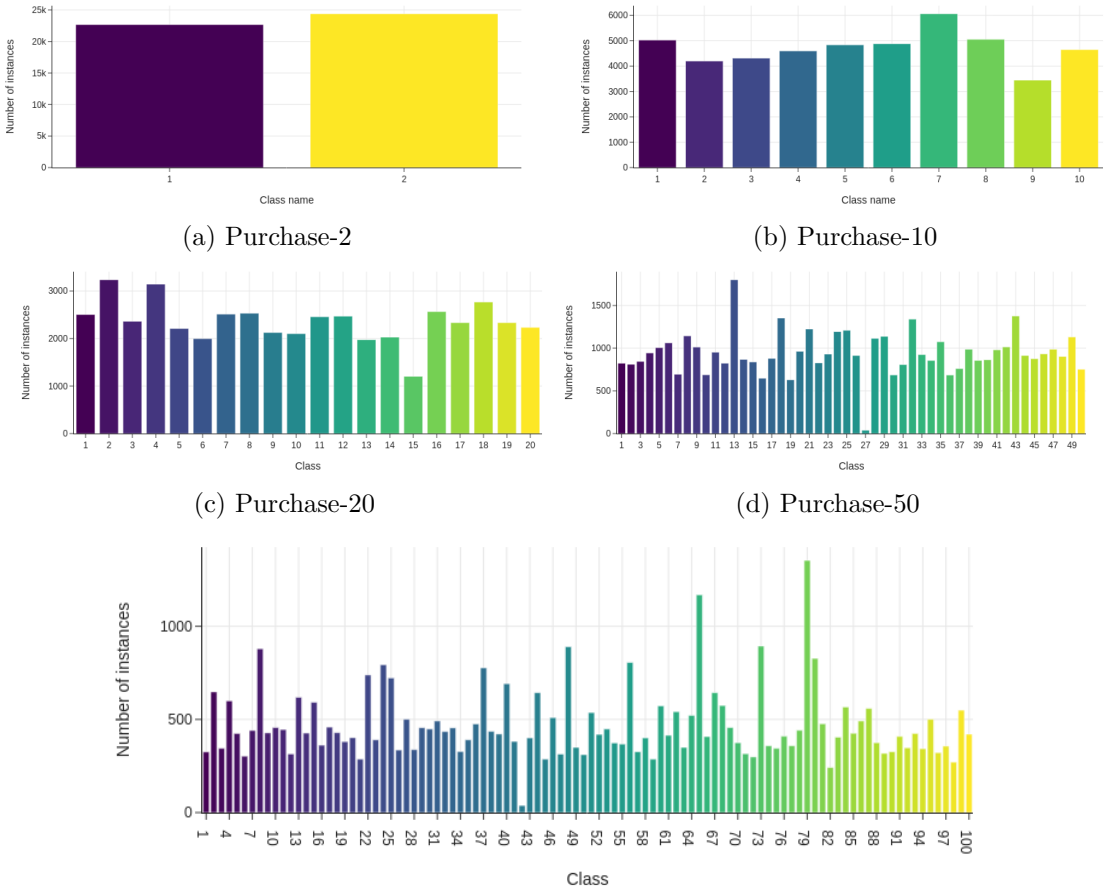
3.1 Dataset and model description

For evaluating membership inference performance in federated learning, we use two datasets, which were considered in other works on membership inference attacks, but in centralized settings [37].

Purchases dataset The *Purchases* dataset is a smaller version of the "Acquire Valued Shoppers" dataset from Kaggle[1] As none of the previous works directly published their dataset and described all the data preprocessing steps, we are re-creating the dataset as closely as possible from the given description and making it available at Zenodo[2].

The original dataset contains around 350 million rows of transactions data. We take a subset of the first 60 million rows due to limited computational capacity. From these transactions data, we choose 600 products (as it was in [37]) and rearrange the dataset to contain only two features: the user id and the product name. Then we delete duplicates and represent every product as an attribute. Thereby, we get a new dataset with around 47 thousand individuals and 600 binary features, representing different products and showing if an individual purchased the product. To create different classification tasks, we use *k-means clustering* to cluster the data with different numbers of target classes (2, 10, 20, 50 and 100). Each class represents the purchase behaviour of a group of customers, therefore the classification task is to predict the behavioural group for a specific customer.

[1] https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data.
[2] dx.doi.org/10.5281/zenodo.3697278



(a) Purchase-2

(b) Purchase-10

(c) Purchase-20

(d) Purchase-50

(e) Purchase-100

Figure 3.1: Purchase datasets' classes distribution

In this work, we denote the Purchase dataset with 2 classes as Purchase-2, correspondingly Purchase-10 for 10 classes, and so on.

The resulting data distribution among classes is generally rather similar in size, especially with smaller number of classes (see Figures 3.1a, 3.1b). A number of very small and very large classes appear in the datasets clustered with 20, 50 and 100 classes (see Figures 3.1c – 3.1e).

We used 30,000 instances for training the model, 3,000 for test set, and the rest is used for training shadow models and building the attack model training set.

Location dataset The *Location* dataset includes data from April 2012 to September 2013 about global check-ins in the Foursquare application [3]. It consists of 33,278,683

[3] https://sites.google.com/site/yangdingqi/home/foursquare-dataset

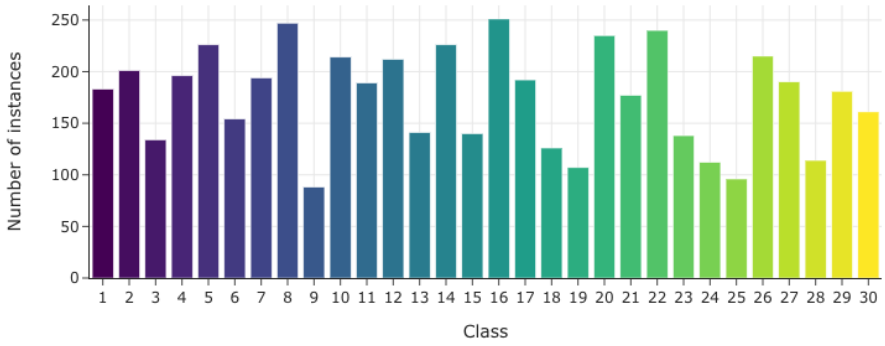


Figure 3.2: Location dataset: Data distribution among the classes.

check-ins made by 266,909 users on 3,680,126 venues. We choose the subset of the data representing the check-ins in Bangkok, as it was done in [37]. The data for each location contains latitude and longitude of the place as well as its category (Restaurant, Hotel etc.). First, we select only check-ins made in the Bangkok area and obtain $1,056,940$ records. Then we split the Bangkok area to 22 equal parts along the longitudinal coordinates and 26 parts along the latitudinal, which resulted in $22X26$ sections (this ratio was chosen after the search of parameters, in order to minimize the empty areas). After deleting the sections without any records, we end up with 351 different areas of Bangkok. Subsequently, we delete users who had less than 25 records as well as filtered out locations with less than 100 check-ins. As a result, we obtained 150 areas and 143 categories of check-in spots. Using *One-hot-encoding* we turn this attribute into binary values. The final version of the dataset contained $5,280$ records representing unique users with 293 binary attributes showing characteristics of all the places that these users have visited. The classification task, as in the previous case, is created with *k-means* clustering. For this dataset, we choose a classification task with 30 classes, based on [37]. Figure 3.2 represents data distribution among the classes. We make use of $1,200$ instances for training the target model, also based on [37] and 400 instances for the test set. The rest of the data is used for training the shadow models. We also make this dataset available at Zenodo [4].

Neural network model We employ the same model architecture as in [37] to reproduce and compare the membership inference attack performance. For the Purchase dataset, we use a neural network with one hidden layer of 128 neurons, a *Softmax* layer and a *Tanh* activation function (see Figure 3.3). *Softmax* is a function, that takes a vector as an input and normalizes it into a probability distribution (a vector, the components of which sum up to 1). *Tanh* is a nonlinear activation function, bound to the range $(-1, 1)$ and defined by Formula 3.1.

[4] dx.doi.org/10.5281/zenodo.3697278

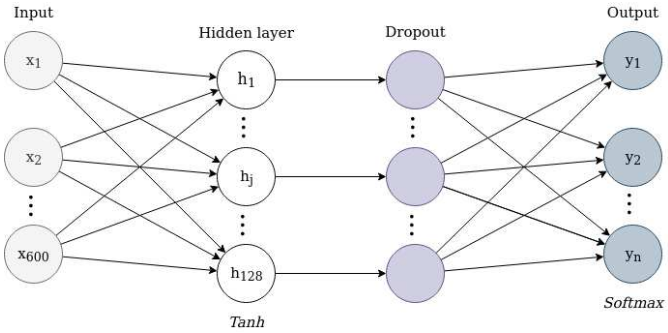


Figure 3.3: Neural network model architecture for the Purchase dataset

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{3.1}$$

We perform a hyper-parameters search over different optimizers (SGD, Adam) with various learning rates $(10^{-1}, 10^{-2}, 10^{-3}, 10^{-4})$. With all the considered learning rates, besides 10^{-2} the models would not converge to the accuracy close to the state of art result in [37]. Moreover, in [37], the authors use the same learning rate. Based on this, for the following experiments we then use the Adam optimizer with a 0.001 learning rate and *NLLLoss* (the negative log-likelihood loss) for training the network. The Adam optimizer (adaptive learning rate optimization algorithm [51]) allows to train the neural network more efficiently, yet does not generalize as well as SGD (stochastic gradient descent). However, in our settings the Adam optimizer performed significantly better. Therefore we decide to use the Adam optimizer instead of SGD, as was described in [37]. We also include a *Dropout* layer – a regularization that randomly drops a part of the units in the network to avoid overfitting. The dropout rate is set to 0.9 for the Purchase-2 dataset, and to 0.5 for the other classification tasks. We chose these values for dropout layer after a grid-search on the values for the dropout rate from 0 to 0.9 with the step of 0.1. With chosen parameters we were able to reach similar to state of art effectiveness of target models as in [37] (see results in the Chapter 4).

The choice of a neural network was also a natural fit for the sequential federated learning, as it is based on incremental optimization. In the case of parallel federated learning, we use weights averaging. By collecting the updated local models from each node, for each weight in a new global model, we take an average of all corresponding weights from local models.

For the Location dataset, we also use a neural network with a similar architecture as for the Purchase dataset, as in the previous work by Salem et al. [44]. The neural network for the Location dataset (see Figure 3.4) consists of one hidden layer with 128 neurons with a *Tanh* activation function and a *Softmax* layer. In this case we do not use a dropout layer, as adding the layer did not improve the effectiveness of the model. We also chose the Adam optimizer with 0.001 learning rate and *NLLLoss*, as they performed

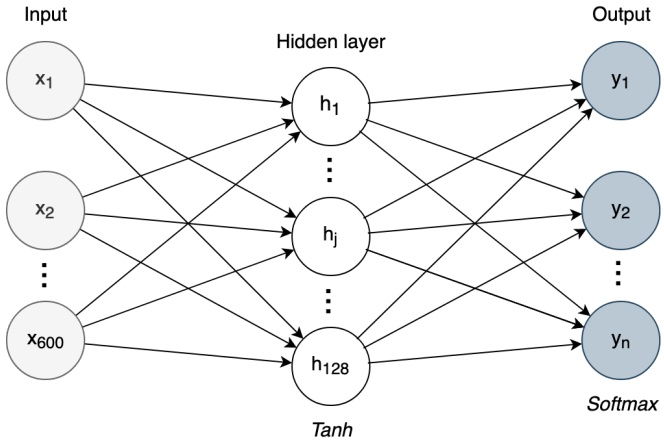


Figure 3.4: Neural network model architecture for Location dataset

the best on the parameter search. As for the Purchase dataset, we tried SGD and Adam optimizers with $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$ learning rates. We take the results from [44] as a baseline and reach it with selected parameters (see results evaluation in Chapter 4).

3.2 Federated learning setup

For each of sequential and parallel federated learning, we assume that there is a fixed set of clients in a federated learning setting, and each client has a fixed unique local dataset. We perform experiments with similar data distribution in the nodes, which is reached by random sampling the instances among the processing nodes. The original dataset is randomly split into equal parts among the corresponding number of processing nodes in each federated learning setting.

We compare the effectiveness and efficiency of sequential, parallel federated learning and centralized learning. We measure the accuracy of the model trained in these three different ways for different classification tasks. For efficiency, we compare the time of the training.

Sequential federated Learning. For the *Purchase* dataset, we consider scenarios with 3, 10, 15 and 20 processing nodes in a sequence, and for the *Location* dataset with $2, 3, 4, 5, 6$ nodes. Sequential federated learning is mostly for settings with smaller number of nodes, therefore, we consider cases with maximum of 20 nodes. Each node had the same number of epochs and was using the full individual dataset for training the model. We distribute the data equally among the different number of processing nodes, i.e. with two nodes, each party has 1/2 of all instances in the training dataset, with three nodes 1/3, and so on. We perform random sampling for this distribution.

For both datasets, as the first step, we randomly initialize weights of the model and sent it to the first node in the sequence for the following training. At each node, we

train the model with the same number of epochs and then sent an updated model to the next client in a sequence. The training is conducted with several so-called cycles C (or federated rounds), following the same order (sequence) at each round.

Parallel federated Learning. In this case, we consider the same number of processing nodes and data distribution as in the sequential scenario for both datasets. We randomly initialize a model and send it to all the clients in the federated learning setting. Having the same initial model at each node at a start significantly increases the effectiveness and efficiency of the training [17]. After each node in the setting has trained the model with the same number of epochs, we collect all the updated models, averaged their weights, and obtain a new global model. Then, the new global model is sent to each client and the training process was repeated with C federated rounds.

3.3 Membership inference attack

Membership inference attack allows to infer from a trained machine learning model whether a specific data instance was in its training set or not. As the baseline for evaluation of the attack performance in the federated learning settings, we reproduced the membership inference attack from the original paper on models trained on centralized data [37]. The following description of the attack is based on the notation given in [37].

Let D^{train}_{target} be the data on which the *target model* f_{target} was trained. $(x^i, y^i)_{target} \in D^{train}_{target}$ is an instance of the f_{target} training set, where x^i is an input vector and y^i is the output label. c_{target} is an output of the target model f_{target}: $c_{target} = f_{target}(x^i)$, represented by a probability vector (each component represents the probability of the record to be of a specific class, the sum of the components is equal to one).

Membership inference is essentially a binary classification task of predicting if a sample record was "in" or "out" of the training set of a target model. To solve this task we train an attack model f_{attack}. The problem is formulated as follows: having an instance (x, y) and a model f_{target}, predict if the instance was "in" or "out" of the training set of the model D^{train}_{target}. The attack model input consists of a probability vector of size c_{target}, plus the correct label y. The output of the attack model is 1, if the record was in the training set and 0, if it was not.

The next step is to create the training set for our attack model. In general, the actual target model is not available to an adversary for this step, as it is kept by the legitimate owner. We can however approximate it by using models with similar architecture, which are called shadow models. The idea underlying the usage of shadow models is that similar models behave similarly on analogous type of data [37]. We simulated the target model behaviour by training several *shadow models* f^j_{shadow} with the same architecture, but different initial weights. We assumed that an intruder is part of the federated learning process, therefore she has access to the model architecture. We trained 20 shadow models on $D^{train}_{shadow_j}$, which is a part of the original dataset similar to D^{train}_{target}, but disjoint to it

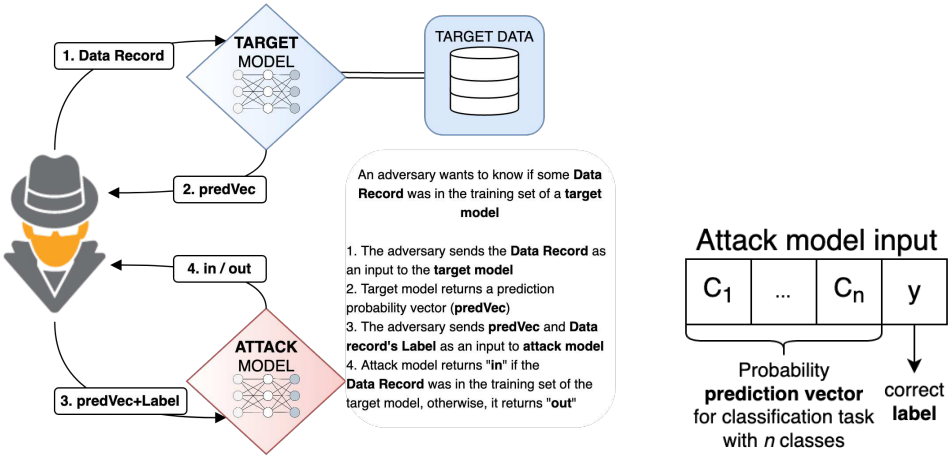


Figure 3.5: Membership inference attack scheme [37]. An intruder passes a *data record* to a *Target Model* and gets a prediction probability vector. This prediction vector and the correct class label is an input for an *Attack Model*, which will predict whether the record was in the training set of the *Target Model* or not.

(see Figure 3.7). These shadow models were used to build the training dataset for the attack model (see Figure 3.6).

For each $(x^j, y^j)_{shadow}$, a record in the $D^{train}_{shadow_j}$, we obtain the probability vector $c^j_{shadow} = f^j_{shadow}((x^j, y^j)_{shadow})$. The tuple (c^j_{shadow}, y^j) was the input for the attack model with label 1. In the same manner, we obtain probability vectors for instances which are not in the training set of the shadow model and label them 0.

After training the attack model on the data obtain from shadow models, we test the attack model on the target model. As for the test dataset for the attack model, we use the instances which were in the training set of the target model and instances which were not in the model training set, with the ratio $1 : 1$.

We select a model architecture for the attack model also based on [37]. The architecture of the attack model for both Purchase and Location dataset is a neural network with one hidden layer of 64 units, a *ReLu* activation function on the hidden layer and a *Sigmoid* function on the output layer, as shown in Figure 3.8.

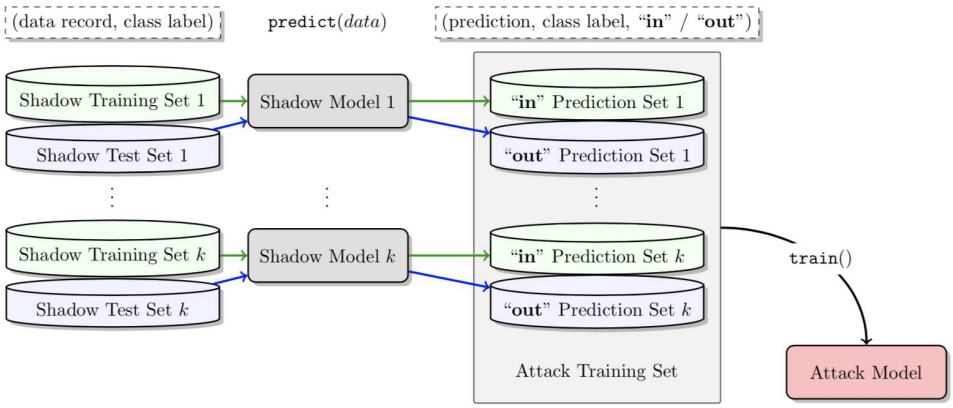


Figure 3.6: Building the attack model training set from shadow models outputs [37]. The *Shadow Models* are trained and tested on individual *Shadow Training Sets*. The *Attack Training Set* consists of the records labeled "**in**" and "**out**". "**in**" class records contain prediction vectors of *Shadow Training Set* plus authentic labels. The "**out**" class consists of prediction vectors of the *Shadow Test Set* plus authentic labels.

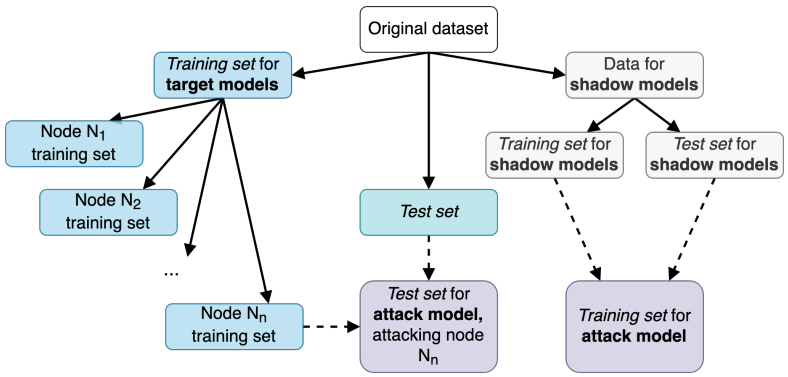


Figure 3.7: Example of splitting an original dataset to training set, test set and a set for training shadow models.

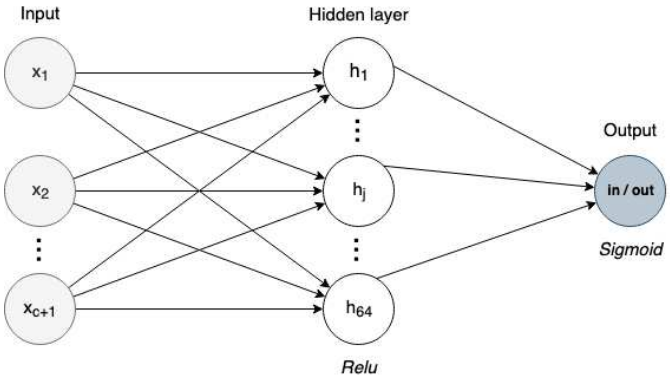


Figure 3.8: Attack model architecture

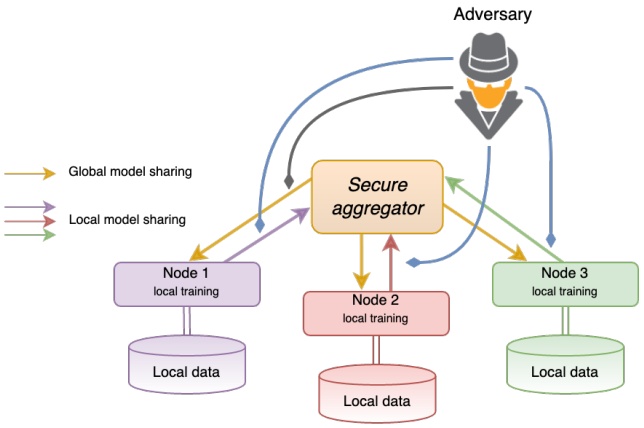


Figure 3.9: Membership inference attack in parallel federated learning settings.

3.4 Membership inference attack in federated learning settings

In both sequential and parallel learning, we consider the case that the intruder is part of the federated learning process. An outsider adversary may have access to the final model, and even if she can infer that some records were in the training set, she is not able to distinguish from which particular party they are. This is the same attack as in the centralized scenario, where we have access only to the final model.

We simulate the attack on each node's training data in federated learning settings. In parallel learning, we consider the scenario where an intruder can steal the model during communication between nodes and a secure aggregator (cf. Figure 3.9). Therefore she has access to all the models trained in every node (see Figure 3.9 blue arrows), as well as a global model which is shared after aggregation (see Figure 3.9 grey arrow). We perform

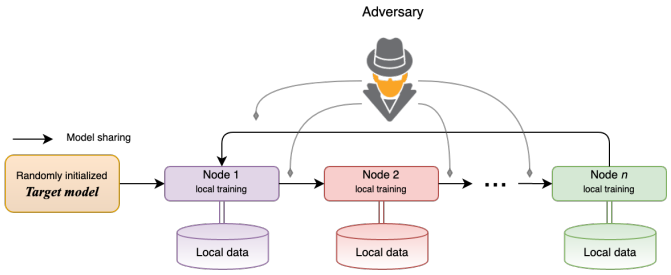


Figure 3.10: Membership inference attack in sequential federated learning settings.

membership inference attack on a node N_1 with a model f_{N_1}, which was trained on the data $D_{N_1}^{train}$. Further, we compare the attack performance on the global model, and the other local models from nodes N_k, $k \neq 1$. In all attacks, the test set of the attack model is a combination of the global test set (not used for training any other models) and data from $D_{N_1}^{train}$ (i.e. the training data of a target node N_1), with ratio 1:1. Therefore we measure how much information about the training set of node N_1 can be extracted from each of the (i) local model trained in this node, (ii) the global model, and the (iii) local models trained in the other nodes.

In sequential federated learning, we consider a scenario when an adversary can steal a shared model (see Figure 3.10). We attack the node N and compared results of the attacks (i) right after the training in the target node and (ii) after training at each following node. By performing membership inference attack at each step of federated learning, we evaluate in which possible scenarios an adversary will have

In a real world scenario, an intruder may have access to the target models only at some particular points of training. In the evaluation chapter, we more closely discuss which particular parts of federated learning are more vulnerable to membership inference, and when.

We evaluate the performance of the target model on the training and test set. The test set is the subset of the main dataset, which was not used for training target models, nor shadow ones (see Figure 3.7). As a training set, we consider the training set of each node, as well as the whole training set from all the nodes. The membership inference attack was evaluated by measuring attack accuracy, precision and recall on the attack test set.

3.5 Mitigation strategies

The empirical evaluation shows that federated learning is vulnerable to membership inference attack. Particularly, in sequential federated learning, right after training in the target node, the attack on the target node's data has higher accuracy, than attacks on the models trained in other nodes. In parallel learning, comparing the attack accuracy on the target node model to the attack accuracy on the models trained on the other nodes' data, we also saw higher performance in the first case.

Overfitting a machine learning model to the training data leads to higher membership inference performance (see Chapter 4 and [37]). There are several approaches to avoid overfitting and improve generalization properties of a machine learning model, e.g. data augmentation, adding a dropout layer, reducing model complexity, adding noise to inputs during training, or early stopping.

In this work we evaluate how adding noise to the training data influences both the membership inference attack and the target model performance. During experiments we search for a trade-off between privacy and effectiveness of the model on the main classification task.

Noise addition to the original data. Both Purchase and Location datasets were one-hot-encoded, therefore all the attributes are binary values. To introduce noise into the training set, we take a specific number of random samples from the data corresponding to a selected noise ratio and swap the values in all attributes. We perform experiments with constant noise in the data, as well as with gradually reducing or increasing the noise during the training process, depending on the setting.

3.6 Summary

In this chapter we discussed the design on the experimental setup for evaluating federated learning effectiveness and its resistance to a membership inference attack. We closely discussed the Purchase and Location datasets, and how they were prepossessed. Based on previous works and conducting hyper-parameter search, we chose the architectures and hyper-parameters for the models. For Purchase dataset We discussed attack scenarios in federated learning settings and developed mitigation strategies.

CHAPTER 4

Evaluation of attacks and mitigations

In this chapter, we present the results of the empirical evaluation of membership inference attack performance in federated learning settings. We also evaluate the effectiveness and efficiency of the considered federated learning settings and the impact of mitigations to the federated learning performance.

4.1 Efficiency and effectiveness evaluation

For evaluating the efficiency of federated learning in different settings we measure training time and model accuracy during ten cycles of federated learning. We compare the performance of the models with different numbers of processing nodes and training epochs per node. The effectiveness is evaluated by measuring the accuracy of models on the test set (which was not used for training at any node). We also train a model on centralized data and compare its effectiveness and efficiency to federated learning results.

4.1.1 Purchase dataset

Sequential federated learning. Figure 4.1 shows the results of sequential federated learning on the Purchase-2 and Purchase-10 datasets (i.e. for 2 and 10 class settings) with different numbers of nodes and training epochs per node. For Purchase-2 in all considered numbers of nodes (see Figures 4.1a to 4.1d), we see a similar trend: regardless of the number of epochs, the models reach a high accuracy already in the beginning of the training. After 100 seconds of training at all different nodes, the training accuracy levels up at a score of around 0.96 and fluctuates only slightly afterwards.

Figures 4.1e to 4.1h represent the results for Purchase-10 dataset. All the settings varying the number of nodes show a worse accuracy score with 200 epochs. The accuracy scores of

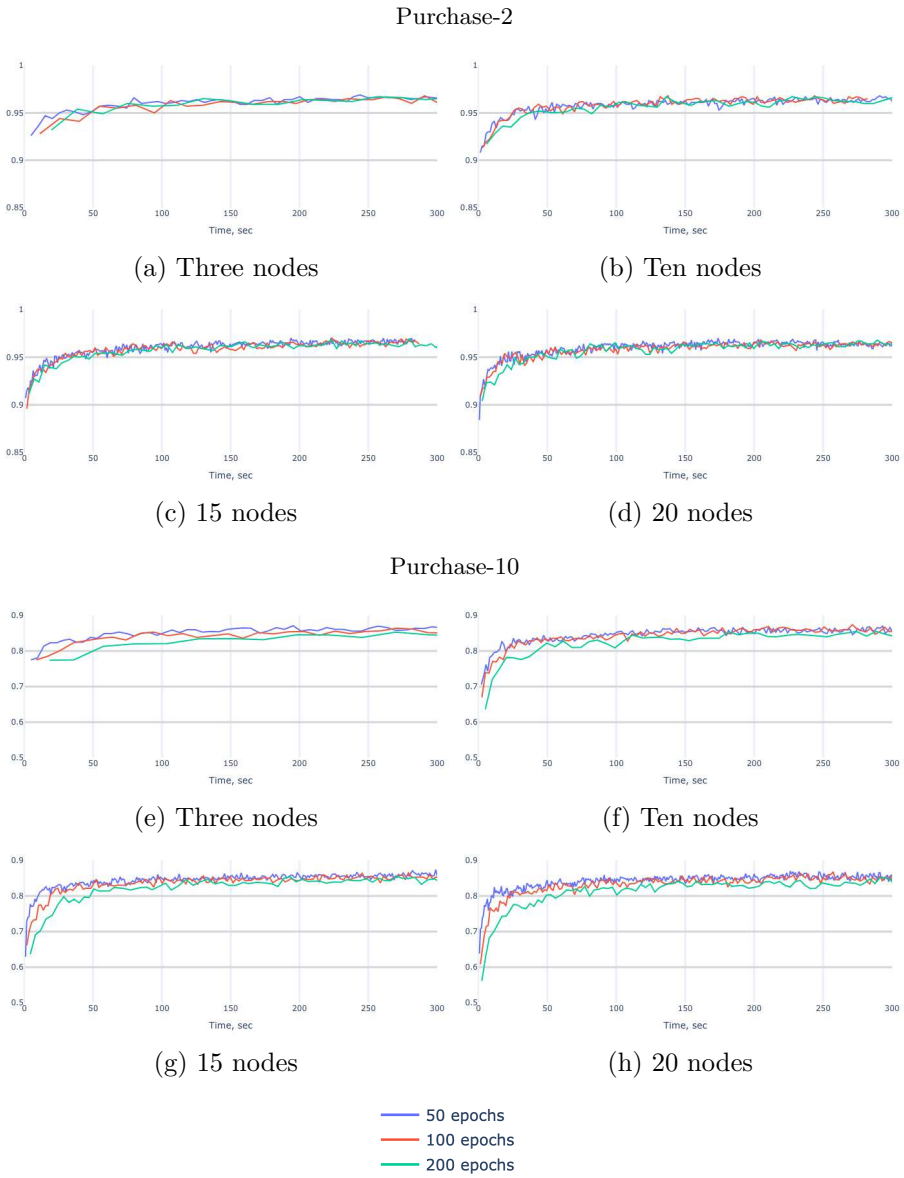


Figure 4.1: Target models accuracy on Purchase-2 and Purchase-10 datasets in sequential federated learning settings with different numbers of processing nodes and different numbers of training epochs per node.

the models trained with 50 and 100 epochs are relatively similar to each other. However, training with 50 epochs per node provides the best results in all settings and reaches the score of 0.86 at each case with a different number of nodes.

The Purchase-20 dataset shows, even more, the inefficiency of using a high number of epochs per node (see Figure 4.2d). In the cases with 3, 10 and 15 nodes (see Figures 4.2a

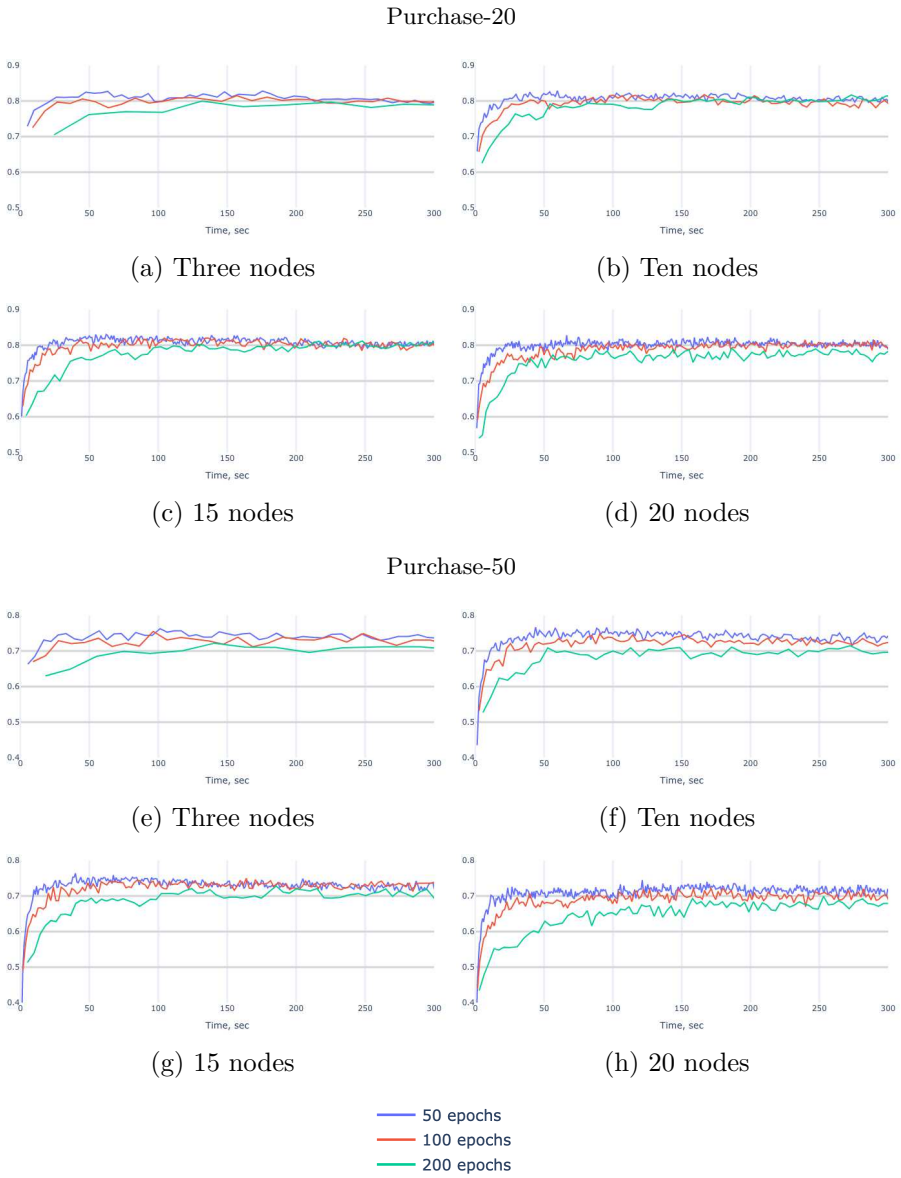
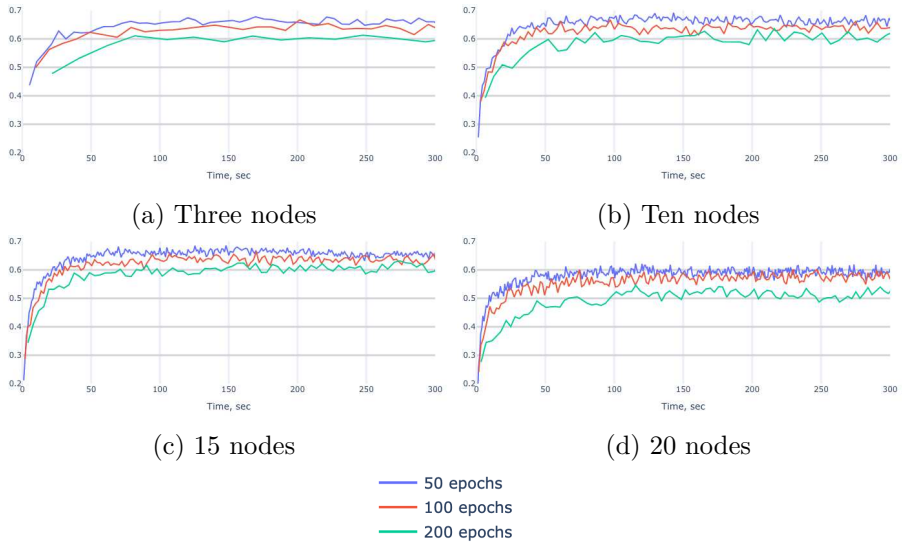


Figure 4.2: Accuracy score of target models on the Purchase-20 and Purchase-50 datasets in sequential federated learning settings with different numbers of processing nodes and different numbers of training epochs per node.

– 4.2c), it takes more time for the 200 epoch training to reach an accuracy similar to 50 and 100 epochs. In sequential federated learning with 20 nodes, the test set accuracy with 200 epochs has the worst score of all represented training times. The highest score of 0.82 is achieved with only 50 training epochs per node.

On the Purchase-50 dataset, the trend with lower performance of 200 epochs remains



(a) Three nodes (b) Ten nodes

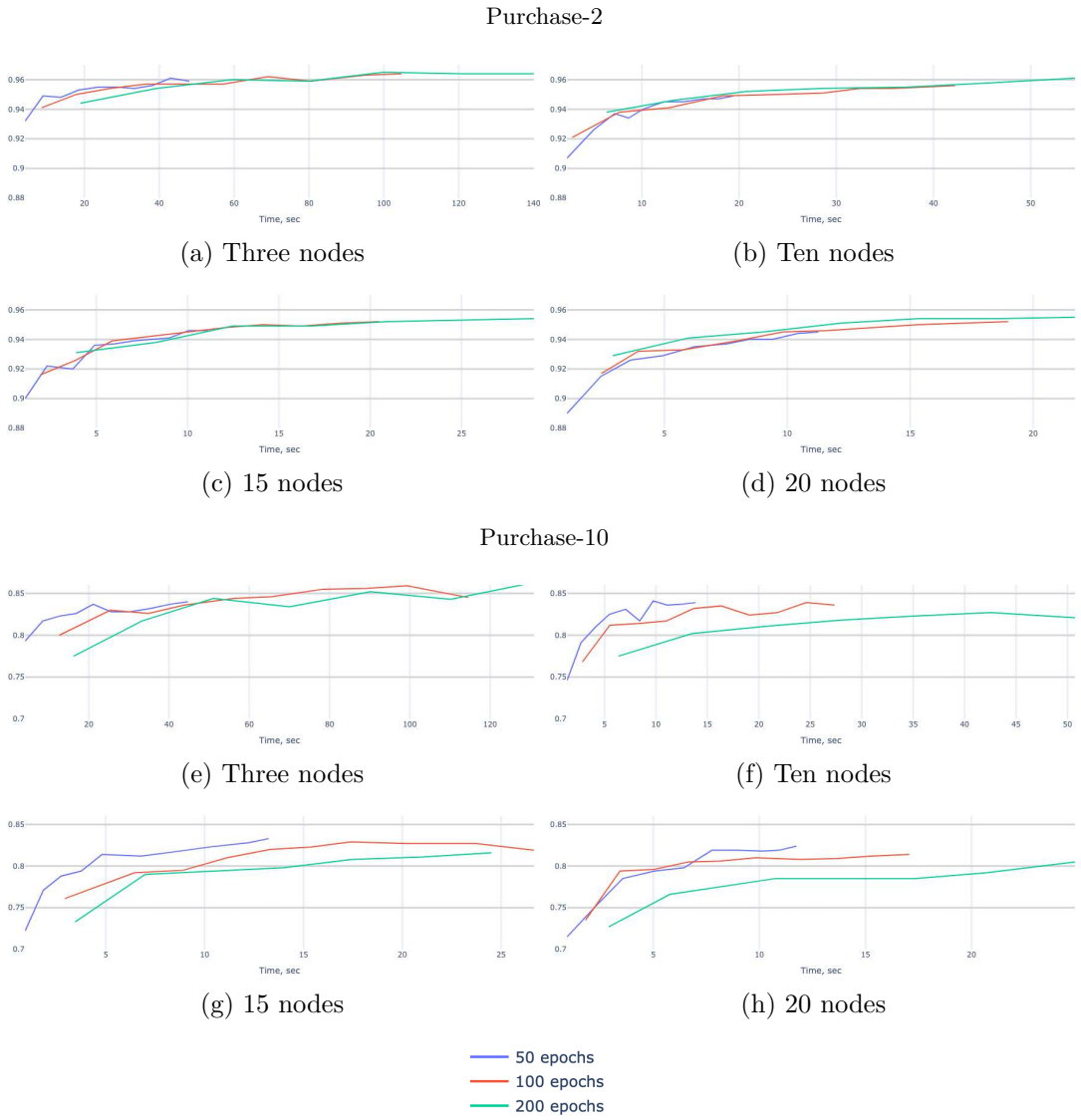
(c) 15 nodes (d) 20 nodes

Figure 4.3: Target models accuracy on the Purchase-100 datasets in sequential federated learning settings with different numbers of processing nodes and different numbers of training epochs per node.

unchanged (see Figures 4.2e to 4.2h). However, the difference in results of training model with 50 and 100 epochs is increasing, and we can see the advantage of using 50 training epochs at each node. For the Purchase-50 dataset, the highest accuracy score is around 0.76 and it is achieved only by training 50 epochs at all considered numbers of nodes in the federated learning setting. However, in the case with 20 nodes, the maximum accuracy on the test set reaches at most 0.745 (cf. Figure 4.2h).

We observe similar effect of different number of epochs on Purchase-100 dataset (refer to Figure 4.3) On the Purchase-100 with 20 nodes, the maximum accuracy on the test set is 0.662 which is the lowest score comparing to the cases with a less number of nodes in the setting. With 3 and 15 nodes in the setting, the maximum accuracy is around 0.67 (see Figures 4.3a, 4.3c). With ten nodes, we reach the accuracy of the model of 0.69 (see Figure 4.3b).

Parallel federated learning. Results for parallel federated learning on the Purchase-2 and the Purchase-10 datasets are represented in Figure 4.4. In the parallel setting, we evaluated the effectiveness of the global model which is obtained by averaging the local models trained at each node. For a different number of epochs (50, 100 and 200), we considered ten federated cycles. On the Purchase-2 dataset with three nodes in the federated learning setting (see Figure 4.4a), training with 50 epochs per node reaches the accuracy score of above 0.96 after 40 seconds of training. For ten and more nodes in the setting, the maximum accuracy is reached only with 200 epochs and closer to the end of the considered time interval.



(a) Three nodes (b) Ten nodes

(c) 15 nodes (d) 20 nodes

(e) Three nodes (f) Ten nodes

(g) 15 nodes (h) 20 nodes

Figure 4.4: Target models accuracy on the Purchase-2 and Purchase-10 datasets in parallel federated learning settings with different number of processing nodes and different number of training epochs per node.

On the Purchase-10 dataset, we observe different behaviour, compared to the Purchase 2 dataset. In this case, training with 100 epochs performs the best with three nodes (see Figure 4.4e). With 10 and more nodes in federated learning (see Figures 4.4f, 4.4g, 4.4h), 50 epochs leads to the highest accuracy score. Moreover with 50 training epochs at each node the global model converges faster. 200 epochs per node do not reach the accuracy of 50 and/or 100 in all these three cases, within the considered time frame and the number of epochs.

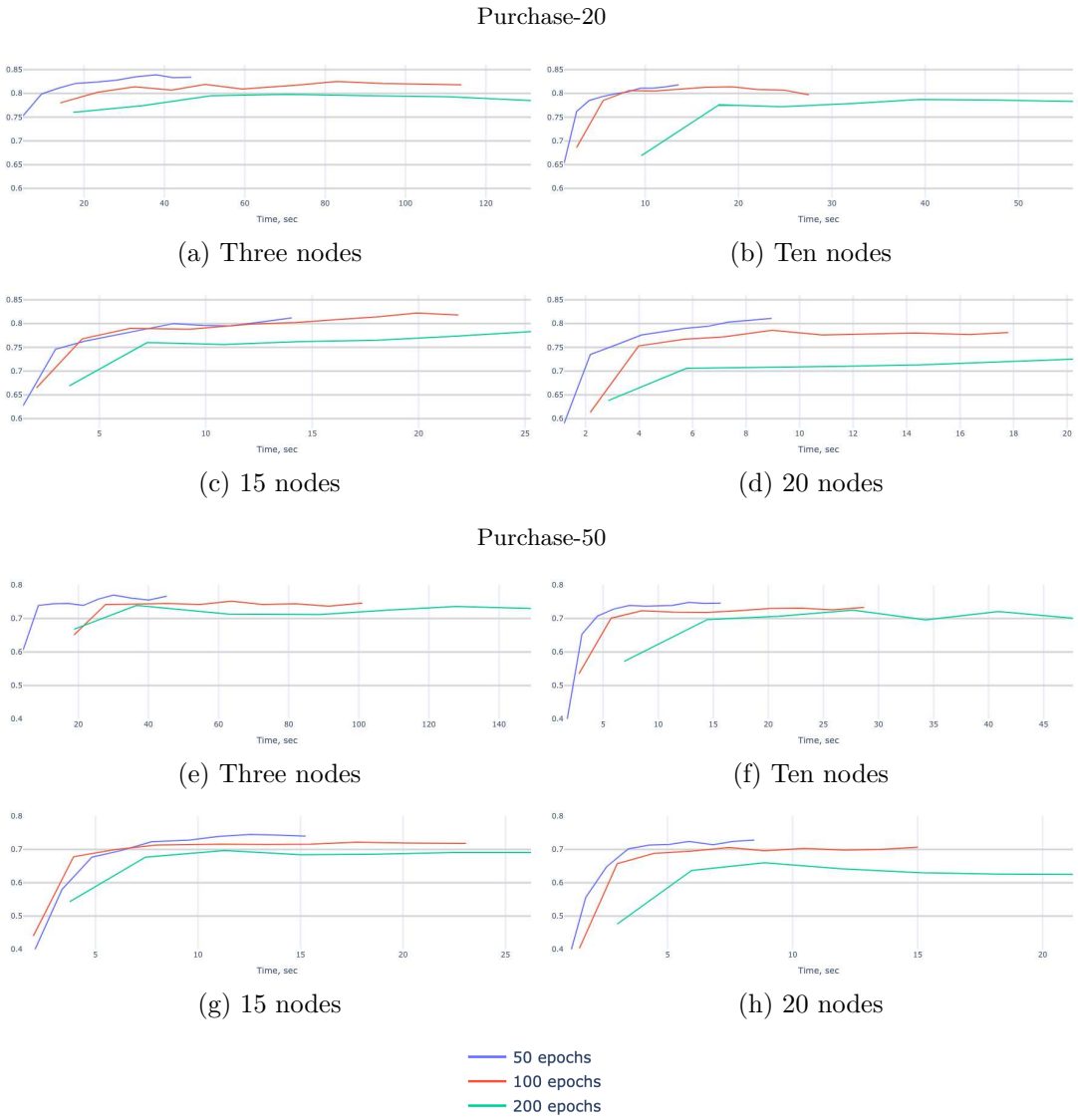
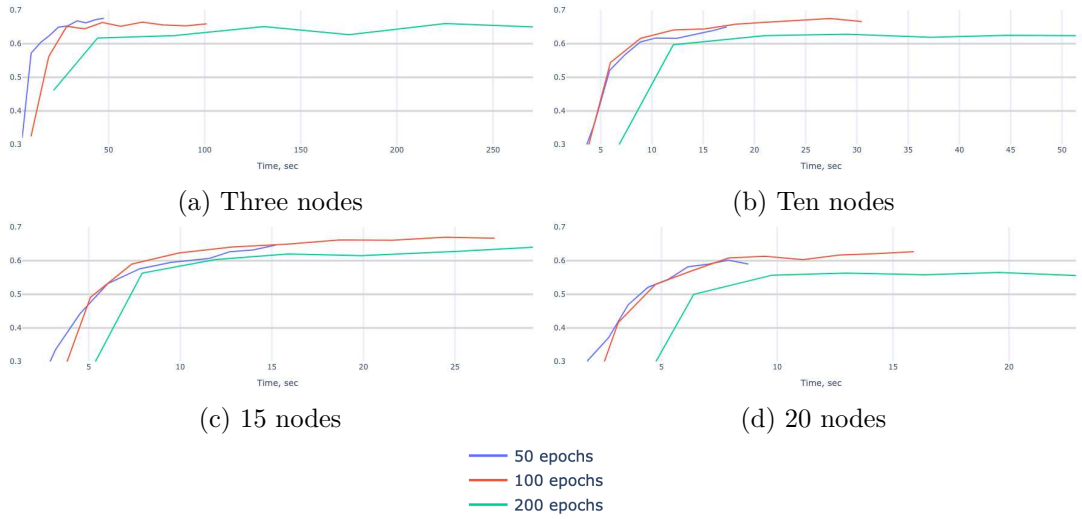


Figure 4.5: Target models accuracy on the Purchase-20 and the Purchase-50 datasets in parallel federated learning settings with different numbers of processing nodes and different number of training epochs per node.

We can see a similar trend for Purchase-20 and Purchase-50 (see Figure 4.5). With a lower number of training epochs per node, models reach a higher accuracy score faster. In general, the more nodes in the federated learning setting, the faster the convergence. However, with three nodes in the setting (see Figure 4.5a), the accuracy score is the highest and reaches the value of 0.84, in 40 seconds of training. With ten and more nodes we reached at most the accuracy score of 0.82, within ten federated learning cycles.



(a) Three nodes (b) Ten nodes

(c) 15 nodes (d) 20 nodes

Figure 4.6: Target models accuracy on the Purchase-100 dataset in parallel federated learning settings with different numbers of processing nodes and different numbers of training epochs per node.

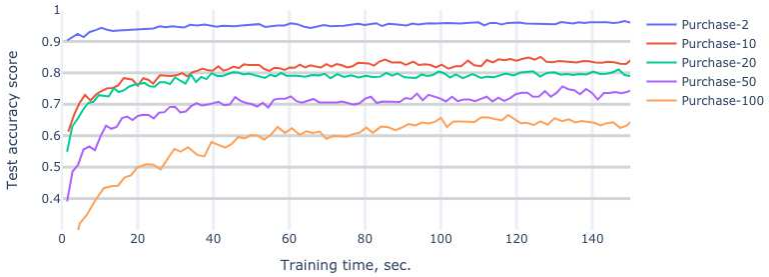


Figure 4.7: Effectiveness and efficiency of models trained on centralized data for different classification tasks from Purchase dataset.

For Purchase-50, the maximum accuracy score of the target model is around 0.73. It is also reached fastest with 50 training epochs in each node. With three and ten nodes in the setting, the maximum accuracy score is higher and reaches the value of 0.75.

For the Purchase-100 dataset (see Figure 4.6), 50 training epochs at each node result the best accuracy score only for three nodes in the setting (see Figure 4.6a). For ten and more nodes, the best accuracy of the target model is reached with 100 epochs per node. However, with 20 nodes in the setting, the best reaches accuracy is 5% worse than with 10 and/or 15 nodes (refer to Figures 4.6b, 4.6c, 4.6d).

Comparing parallel and sequential federated learning to centralized Figure 4.7 shows the performance of models trained on centralized data. We use training parameters as stated in Chapter 3 and reach similar to state of art result as in previous

works [37], [44].

Sequential federated learning on Purchase-2 dataset (see Figures 4.1a to 4.1d) reaches accuracy of 0.96 after 40 seconds of training, which is comparable to centralized case. Sequential learning on Purchase-10 and Purchase-20 datasets allows training model with the same accuracy as in centralized data case within less training time.

On Purchase-10 (see Figures 4.1e to 4.1h), the accuracy on the set reaches baseline accuracy of 0.84 before 50 seconds training in sequential federated learning, while in centralized it takes over 80 seconds. However, the more the number of nodes in the setting the longer it takes to reach the same result.

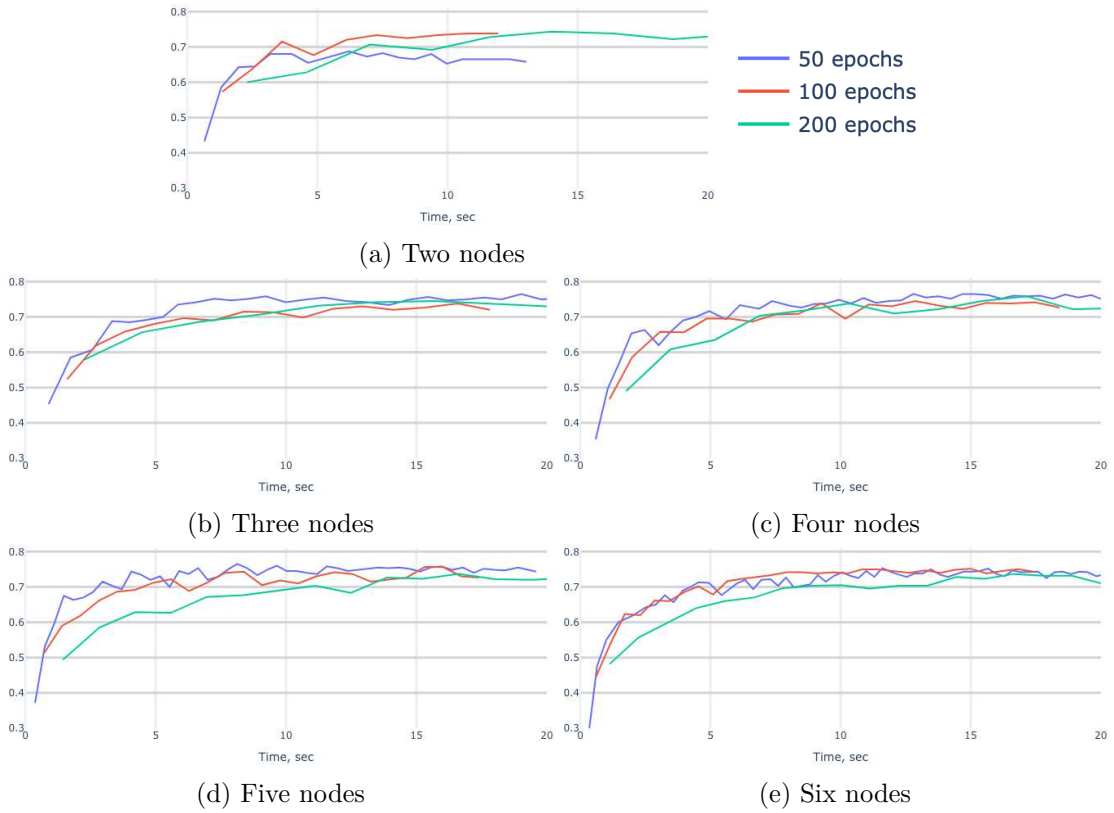
On Purchase-50 and Purchase-100 with 20 nodes in the federated learning setting, we also reach the baseline results faster than in centralized learning. However, in sequential learning with 20 nodes (see Figures 4.2h and 4.3d), model effectiveness is 4-6% lower than in the centralized case. In the settings with all other numbers of nodes, the state-of-art accuracy can be reached within 60 cases (in most of the cases even faster), while in centralized learning it takes more than 100 seconds.

Parallel federated learning allows training models simultaneously. Therefore, with more nodes in the setting, the growth of the model accuracy goes faster. With three nodes on Purchase-2 dataset (see Figure 4.4a), we reach a baseline accuracy of 0.96 within 60 seconds in a parallel setting, which is worse than the sequential case. However, with a larger number of nodes, the time needed to reach the baseline is shorter. With 15 and 20 nodes in the setting, we reach 2-3% less accuracy within considered training time.

On Purchase-10, parallel learning with three nodes (see Figure 4.4e) performs similar as in sequential case, reaching accuracy of 0.845 after 50 seconds of training versus 80 in the centralized setting. With 10 nodes the same result is reached after 10 seconds training (see Figure 4.4f). With larger number of nodes we reach slightly worse accuracy of the model, e.g. with 20 nodes the accuracy score is 0.83 (see Figure 4.4h).

On Purchase-20 classification task with parallel federated learning with three nodes (see Figure 4.5a), the accuracy score outperforms the baseline result for almost 5%, reaching accuracy score of 0.84 at 38 seconds training. However, with a larger number of nodes in the settings parallel learning allows to reach the baseline accuracy faster (e.g. with 15 and 20 nodes parallel takes around 8 seconds versus 20 in sequential and 153 in centralized).

For Purchase-50 and Purchase-100 in parallel learning, we reach baseline accuracy in all settings besides 20 nodes case. With 20 nodes, the best-reached accuracy is 2% less for Purchase-50, and 5% less for Purchase-100. For cases with a fewer number of nodes in the setting, the baseline accuracy is reached faster than in centralized and parallel cases even with three nodes in the setting. For example, parallel federated learning on Purchase-100 with ten nodes reaches an accuracy of 0.65 in 15 seconds, sequential learning in the same settings takes 50 seconds and centralized 100 seconds.



(a) Two nodes

(b) Three nodes

(c) Four nodes

(d) Five nodes

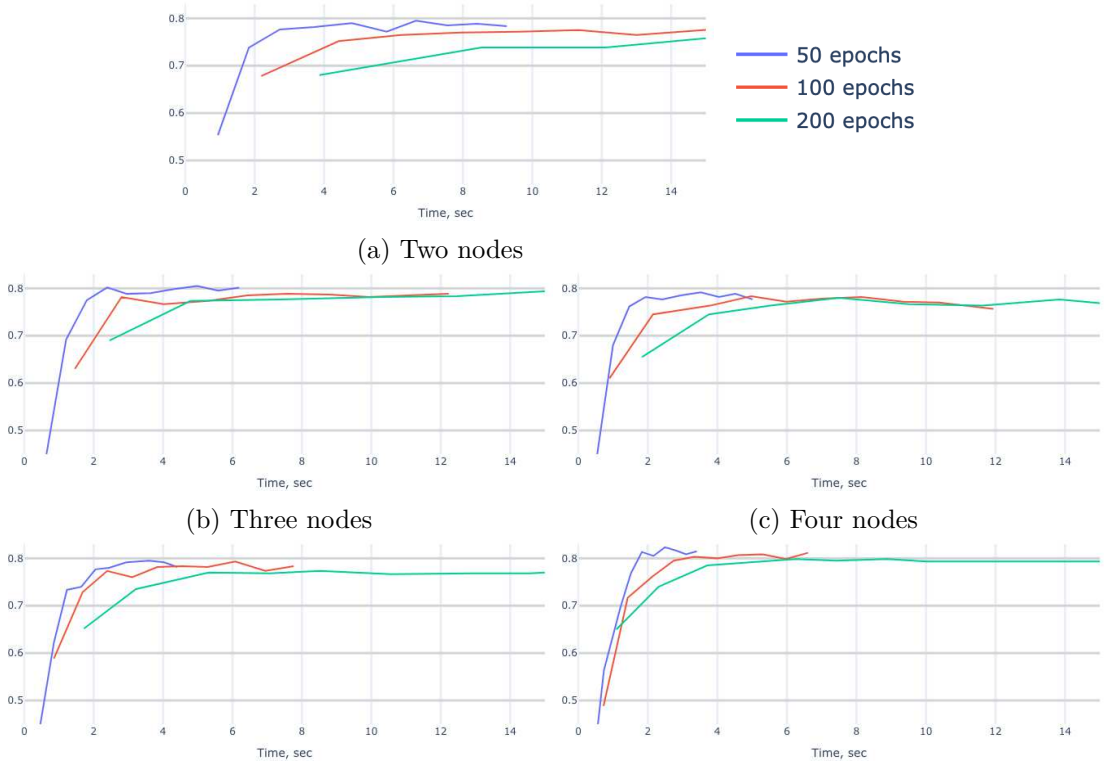
(e) Six nodes

Figure 4.8: Target models accuracy on Location dataset in sequential federated learning settings with different number of processing nodes and different number of training epochs per node.

4.1.2 Location dataset

Sequential federated learning. Figure 4.8 represents the results for models' effectiveness and efficiency on the Location dataset in sequential federated learning. In general, there are no specific trends for an optimal number of epochs per node. In cases with three, four and five nodes (cf. Figures 4.8b to 4.8d), 50 epochs perform the best among considered number of epochs, and reach the highest accuracy score faster. With two nodes in the setting (refer to Figure 4.8a), training with 100 epochs results the best accuracy score in the shortest time. With six nodes (see Figure 4.8e) 50 and 100 epochs perform similarly. However, for all cases, training model with 200 epochs per node also allows to reach good accuracy score.

Parallel federated learning. The performance parallel federated learning on the Location dataset is represented in Figure 4.8. We observe that with a lower number of epochs per node the accuracy of the global model grows faster and reaches maximum in a shorter interval. For all considered number of nodes, training local models with 50



(a) Two nodes

(b) Three nodes

(c) Four nodes

(d) Five nodes

(e) Six nodes

Figure 4.9: Target models accuracy on the Location dataset in parallel federated learning settings with different numbers of processing nodes and different numbers of training epochs per node.

epochs per node performs the best.

The maximum accuracy on the test set is reached with six nodes in the federated learning setting 4.9e. In two seconds after starting federated learning process we reach the accuracy of around 0.8. With 100 epochs we reach the same accuracy in three seconds after the start, and with 200 epochs it takes around five seconds to achieve the same effectivness.

With two, three, and four nodes in the setting (see Figures 4.9a to 4.9c), the maximum accuracy of the target model is reached only with 50 epochs per node. Training models with 100 and 200 epochs results similar performance.

Federated learning on five nodes (see Figure 4.9d) also preserves the trend of higher performance with 50 epochs. The maximum accuracy of 0.8 is reached in three seconds with 50 epochs. However with 200 training epochs we do not reach the same accuracy score at all.

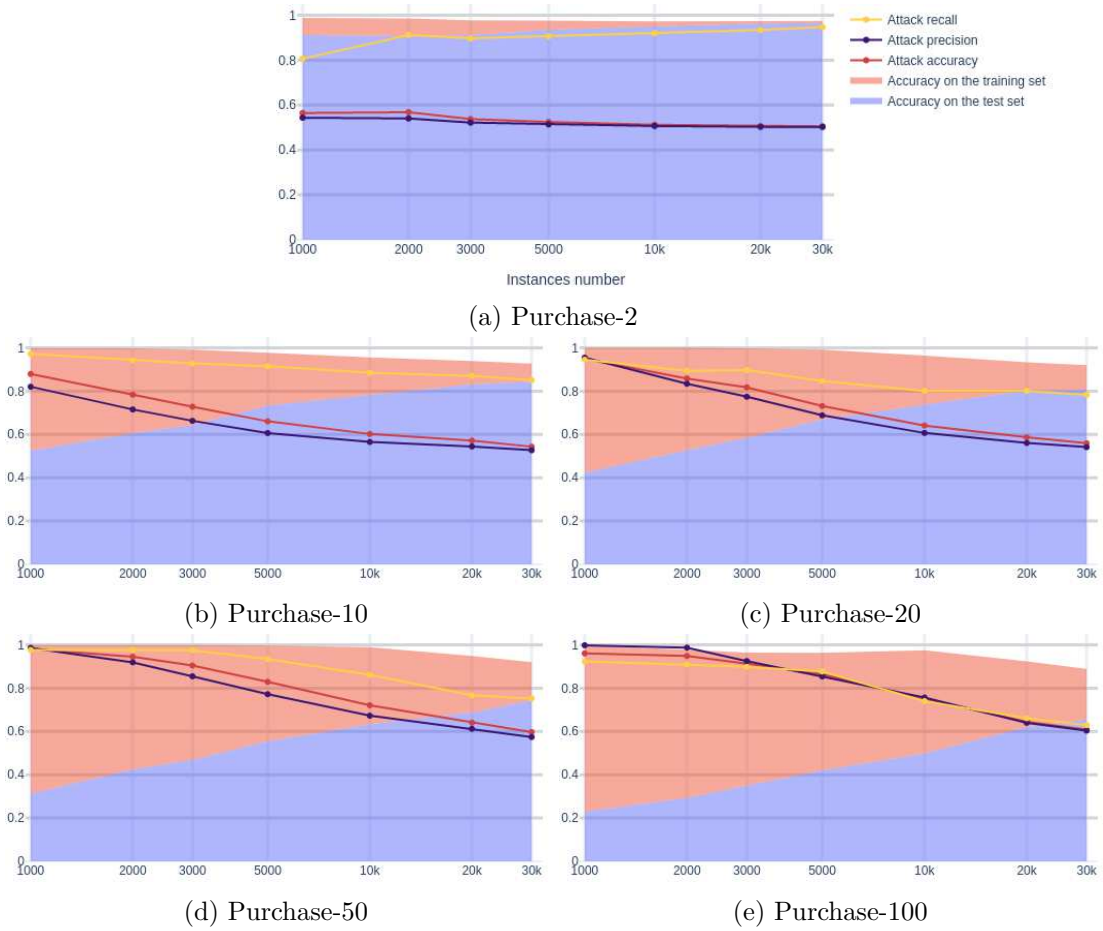
4.1.3 Summary

We considered federated learning effectiveness and efficiency in different settings. Depending on the dataset and the number of nodes in the setting using larger or smaller number of training epochs at each node can be more beneficial. For Purchase dataset in sequential and parallel federated learning training with 50 epochs performed the best. However, on Purchase-2 in sequential federated learning, we reached the similar accuracy with 50, 100 and 200 epochs. The parallel federated learning on Purchase-2 dataset resulted the best score with 200 training epochs at each node.

In general, parallel learning speeds up the learning process due to the simultaneous training of the local models. Therefore, with larger number of nodes in the setting the faster global model converges. However, on the tasks with large number of classes (Purchase-100), the accuracy score was lower, than with less nodes in the setting. This can be explained by the fact, that with 20 nodes in the setting, each node has fewer instances in our settings. Thus machine learning model does not have enough instances to generalize well.

Comparing parallel and sequential federated learning to machine learning on centralized data, we see that federated learning allows reaching the centralized learning accuracy for most of the settings. However, parallel federated learning with 20 nodes does not allow to reach comparable accuracy within considered training time. Therefore, either more federated rounds should be considered in the setting or other numbers of epochs at each node.

In the next sections we evaluate privacy properties of federated learning by performing membership inference attack. We compare the federated learning results with performance of machine learning on centralized data.



(a) Purchase-2

(b) Purchase-10

(c) Purchase-20

(d) Purchase-50

(e) Purchase-100

Figure 4.10: Membership Inference on the Purchase dataset on models trained on centralized datasets of different sizes

4.2 Membership inference attack on centralized data

As the baseline for evaluation federated learning efficiency, effectiveness and privacy properties, we first train a model on centralized data. The Purchase dataset with $10,000$ records in the training set was trained with 10 batches ($1,000$ instances per batch), 100 epochs per batch, and 5 iterations. The Location training dataset, consisting of 30 classes and $1,200$ records, was trained with 6 batches (200 instances per batch), 50 epochs and 4 iterations. All hyperparameters were chosen after performing a grid search. During grid search process we varied parameters like the number of batches (from 1 to 10), number of training epochs per node (25, 50, 100, 150), number of iterations (from 1 to 10).

The results of centralized machine learning on the Location dataset are presented in Table 4.1. We see that the accuracy on the training data is 35% higher than on the test data. Due to the small number of instances, we could not reach a better score on the

Table 4.1: Membership inference attack on the Purchase dataset (with $10K$ instances in the training set) and the Location dataset

Dataset	Training Accuracy	Test Accuracy	Attack Accuracy	Attack Precision	Attack Recall	Attack F1
Purchase-2	0.972	0.951	0.512	0.507	0.92	0.653
Purchase-10	0.956	0.783	0.602	0.565	0.885	0.69
Purchase-20	0.964	0.737	0.641	0.607	0.801	0.691
Purchase-50	0.989	0.634	0.721	0.673	0.861	0.755
Purchase-100	0.974	0.499	0.751	0.757	0.74	0.748
Location	0.983	0.648	0.806	0.734	0.960	0.832

test set. The membership inference attack accuracy is 0.806 with a precision of 0.734.

For the Purchase dataset, the results show that the more classes in the classification task, the bigger is the gap between the training and test set accuracy. This may be explained by the fact that with a large number of classes the classification task becomes more difficult, e.g. because classes are becoming more similar to each other. Further in the work, we investigate this effect. From the data distribution (see Figure 3.1), we also see that some classes are underrepresented, which makes the classification task even harder. The precision of attack models is also growing proportionally to the number of classes, and reaches a maximum of 0.757 on the Purchase-100 dataset. This can be explained by the fact that with larger number of classes in the classification task the input for the attack model is growing (as a part of the input consist of a prediction vector obtained from the target model).

We evaluated how the number of instances in the training set influences membership inference attack performance on the Purchase dataset. We trained the target model on a different number of instances in the training set: from $1,000$ to $30,000$ (see Figure 4.10). With a smaller number of records in the training set, the model remembers training data more specifically, overfits, and has a worse performance on the test set. Since the model learns too much from its training set when it is too small, the membership inference attack has a significantly better performance in these cases. This trend is similar for different classification tasks with a larger number of classes (see Figures 4.10b–4.10e). However, the Purchase-2 dataset (cf. Figure 4.10a) is more resistant to membership inference: even with 1000 instances, the accuracy of prediction is around 0.5, which is at the baseline of random guessing. With an increasing number of instances in the training set, for all classification tasks, we see a trend of increasing accuracy of the classification on the test set, while the membership inference attack performance gradually decreases.

For all of the following experiments with federated learning settings, we use the Purchase dataset with a total of $30,000$ instances in the training set. Subsequently, we will compare all the attack and classification performance results on Purchase dataset with the results of this setting.

4.3 Membership inference attack performance in sequential federated learning

In the sequential federated learning scenario, the membership inference performance was evaluated by attacking the target model after training at each node, i.e. in each cycle. As the target model, we chose one node and observed how the attack performance is changing following the training at the next parties. We simulated attacks on every node's data. Half of the attack test set consists of one selected node N_k training data (where k is the number of nodes in federated learning setting), the other half is the test data that was not used for training in any of the nodes.

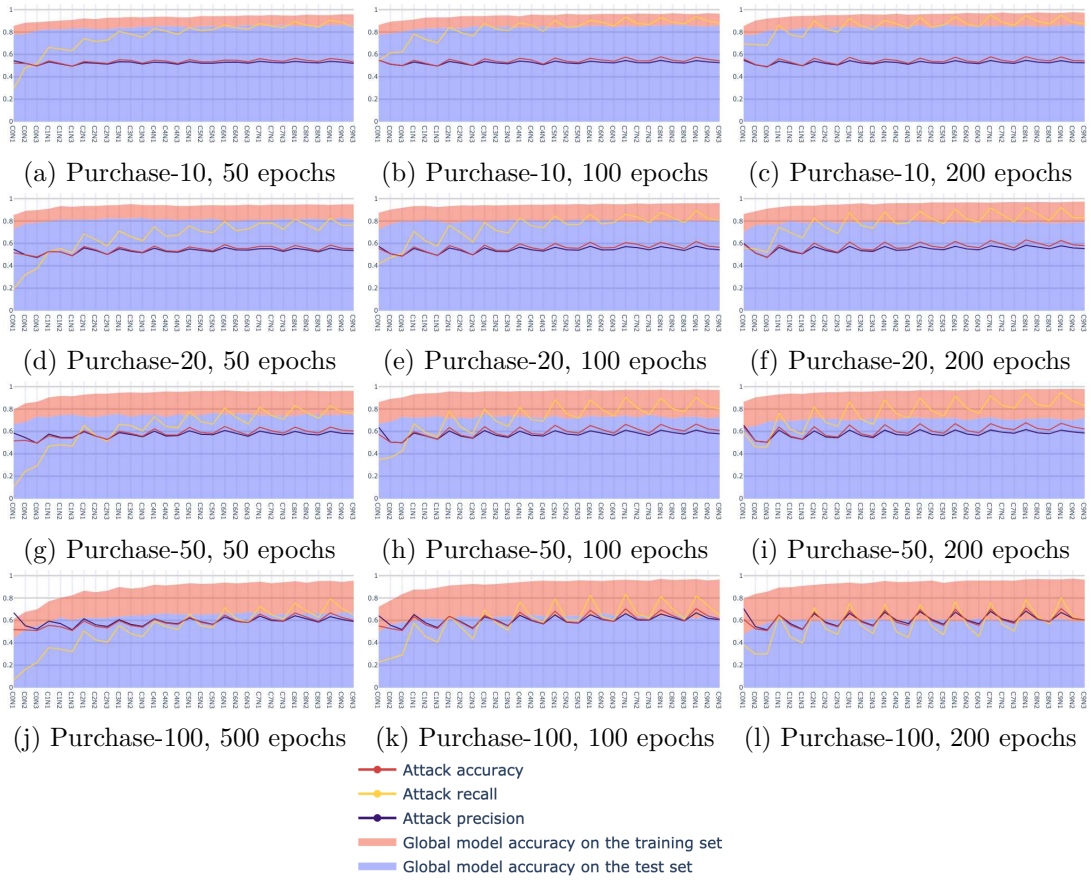
The vertical axis of the plots (see Figures 4.11 – 4.15) shows the score of the target model (i.e., the accuracy on the whole joint training set, accuracy on test set) and the attack model scores (i.e. the attack accuracy, precision and recall on the attack test set). The horizontal axis shows the state during the training process of the target model, where C stands for a federated cycle, and N stands for the node number. Thus, $CiNj$ is the specific instance of the target model after training for the i^{th} cycle at the j^{th} node.

4.3.1 Purchase dataset

Three nodes. Figure 4.11 shows the results for sequential federated learning with three nodes, including the accuracy on the training and test set of the target model (the state of the target model corresponds to the x-axis), and the membership inference performance. We evaluated the membership attack accuracy, recall and precision on the test set, which consists of the training data from the node N_1, and test data in ratio 1:1. Purchase-2 dataset showed to be resistant to membership inference attack in all considered federated learning settings. Thus, in this work, we present results only for the Purchase dataset with ten and more classes and also for the Location dataset.

For each classification task, we tried a different number of training epochs at each node (50, 100, 200). For all classification tasks on the Purchase dataset and the Location dataset, more training epochs per node cause better attack performance. Figures 4.11a to 4.11c show that membership inference has a higher accuracy from the model which was just trained at the node N_1. However, it reaches at most the maximum accuracy of 0.578, which is only 7% higher than the baseline. The accuracy of the target model on Purchase-10 classification task is similar in all three cases of the different number of epochs.

We can see the same trend in all classification tasks with a larger number of classes: Purchase-20 – Purchase-100 (see Figures 4.11d to 4.11l). In all cases, right after training the model in the target node, the attack accuracy is higher. The membership inference attack accuracy always reaches a maximum with 200 training epochs per node. The spike after training in the target node is steeper in this case, comparing to a smaller number of epochs. Regarding the Purchase-20 dataset (cf. Figures 4.11b, 4.11d and 4.11f), we observe that the target model's accuracy on the main classification task (corresponding



(a) Purchase-10, 50 epochs (b) Purchase-10, 100 epochs (c) Purchase-10, 200 epochs

(d) Purchase-20, 50 epochs (e) Purchase-20, 100 epochs (f) Purchase-20, 200 epochs

(g) Purchase-50, 50 epochs (h) Purchase-50, 100 epochs (i) Purchase-50, 200 epochs

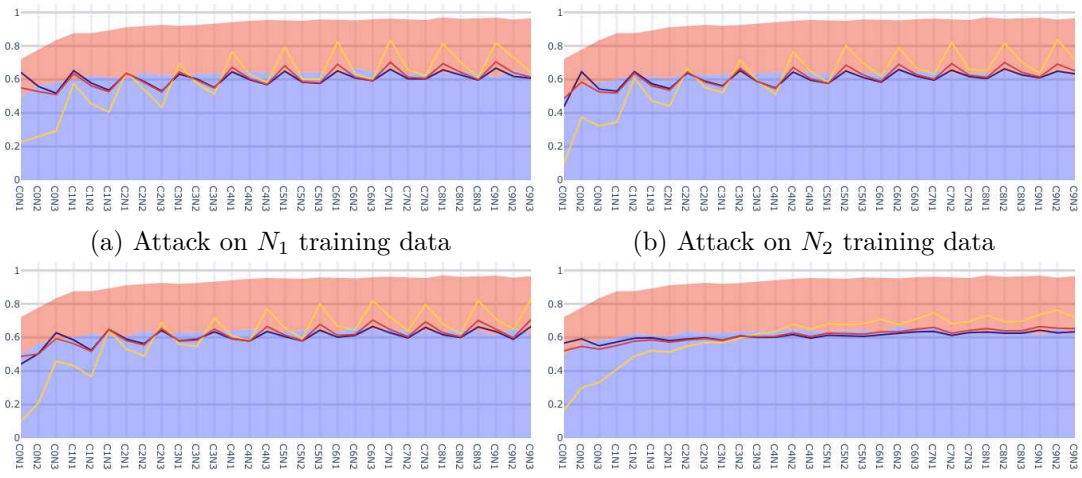
(j) Purchase-100, 500 epochs (k) Purchase-100, 100 epochs (l) Purchase-100, 200 epochs

Figure 4.11: Membership Inference in sequential federated learning with three nodes on the Purchase dataset

to the "Global model accuracy on the test set") is higher with 50 epochs than with 100 or 200 training epochs per node. The same trend we see for Purchase-50 and Purchase-100 datasets. Therefore, models trained with 50 epochs per node perform the best on the main classification task. Moreover, these models are more resistant to membership inference than models trained with 100 or 200 epochs. Interestingly, that recall in growing throughout the whole federate learning process, while accuracy and precision have a similar result at each cycle.

Evaluating the results on Purchase-50 and Purchase-100 datasets (see Figures 4.11b, 4.11g and 4.11l), we observe that membership inference attacks have better accuracy and precision on classification tasks with larger number of classes. We conclude that in federated learning, like in centralized learning, the more classes has a dataset, the more it is vulnerable to membership inference.

We observed the same behaviour of the membership inference attack while attacking the data from other nodes, as shown in Figures 4.12a to 4.12c. After the shared model



(a) Attack on N_1 training data

(b) Attack on N_2 training data

(c) Attack on N_3 training data

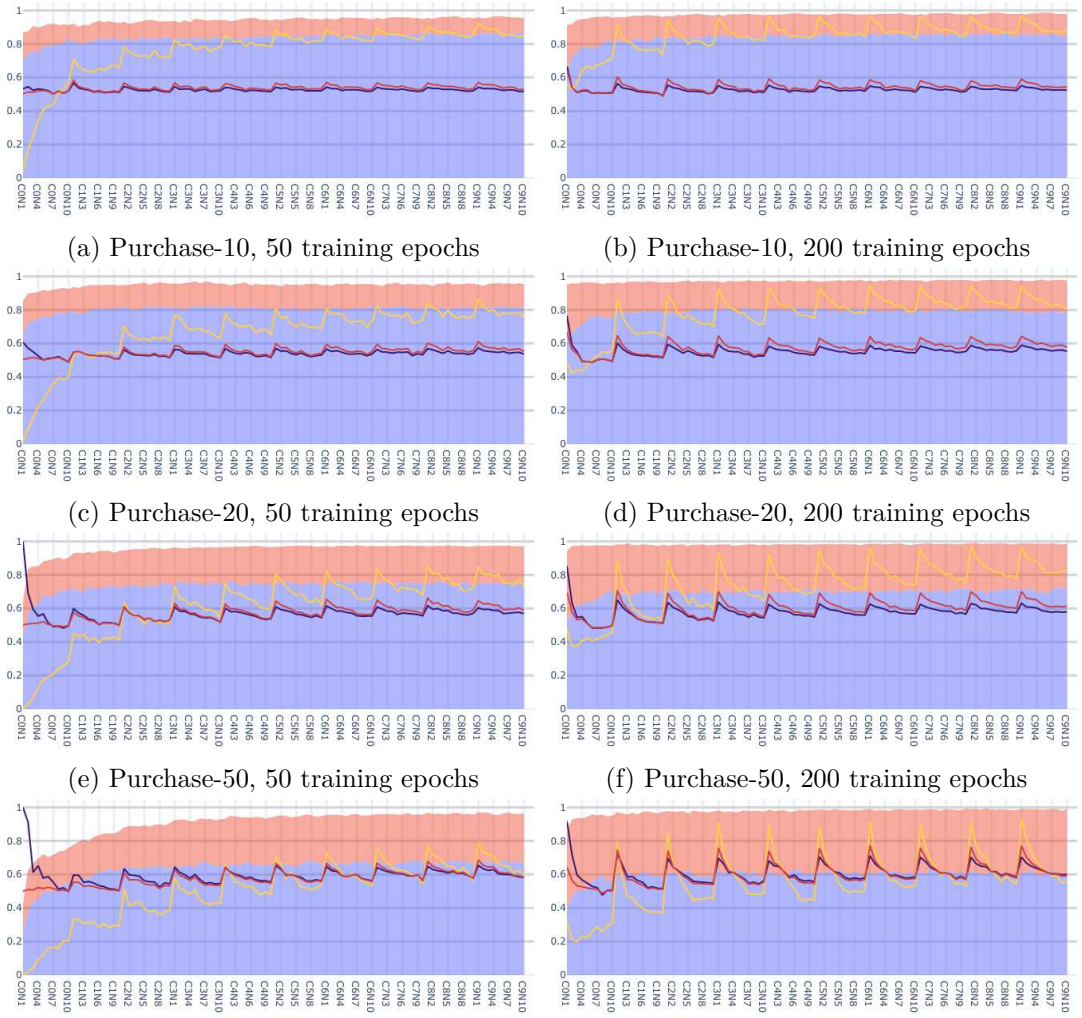
(d) Attack on global joint training data

Figure 4.12: Membership Inference in sequential federated learning with three nodes on the Purchase-100 dataset with 100 training epochs at each node. Attack on different nodes training data.

is trained in a target node (N_1, N_2 or N_3), the membership inference from this model (attacking corresponding node) has a higher accuracy, than after training in the other nodes. Attack on the global data (refer to Figure 4.12d) produces the results similar to centralized learning scenario (see Figure 4.10e with $30,000$ instances in the dataset). Membership inference attack on the global data corresponds to the attack performance on the attack test set, which consist 50% from the test data and 50% from the global training set (the joint training set from all the nodes).

Ten nodes. Sequential federated learning on ten nodes showed similar trends as training with three nodes. Membership inference attack does not reveal any information about the training set on the Purchase-2 dataset. Purchase-10 with 50 training epochs (refer to Figure 4.13a) looks similar to the corresponding setting with three nodes. Models trained with 200 epochs (see Figure 4.13b) are more vulnerable to membership inference, especially right after training in the target node. With fewer training epochs per node, attack accuracy on the model on the first cycle does not exceed the attack accuracy in the following cycles.

For the classification tasks with a larger number of classes (refer to Figures 4.13c–4.13h), we see a more prominent difference between a high and a low number of epochs of training. 50 epochs, again, do not only lead to a lower accuracy on membership inference but also give a significantly better performance on the original classification task. Training with 200 epochs on the Purchase-50 and the Purchase-100 produces attack accuracy in the target node around 10% higher than with 50 epochs of a difference in the. For Purchase-50, maximum reached membership inference accuracy is 0.7 (see Figure 4.13f).



(a) Purchase-10, 50 training epochs

(b) Purchase-10, 200 training epochs

(c) Purchase-20, 50 training epochs

(d) Purchase-20, 200 training epochs

(e) Purchase-50, 50 training epochs

(f) Purchase-50, 200 training epochs

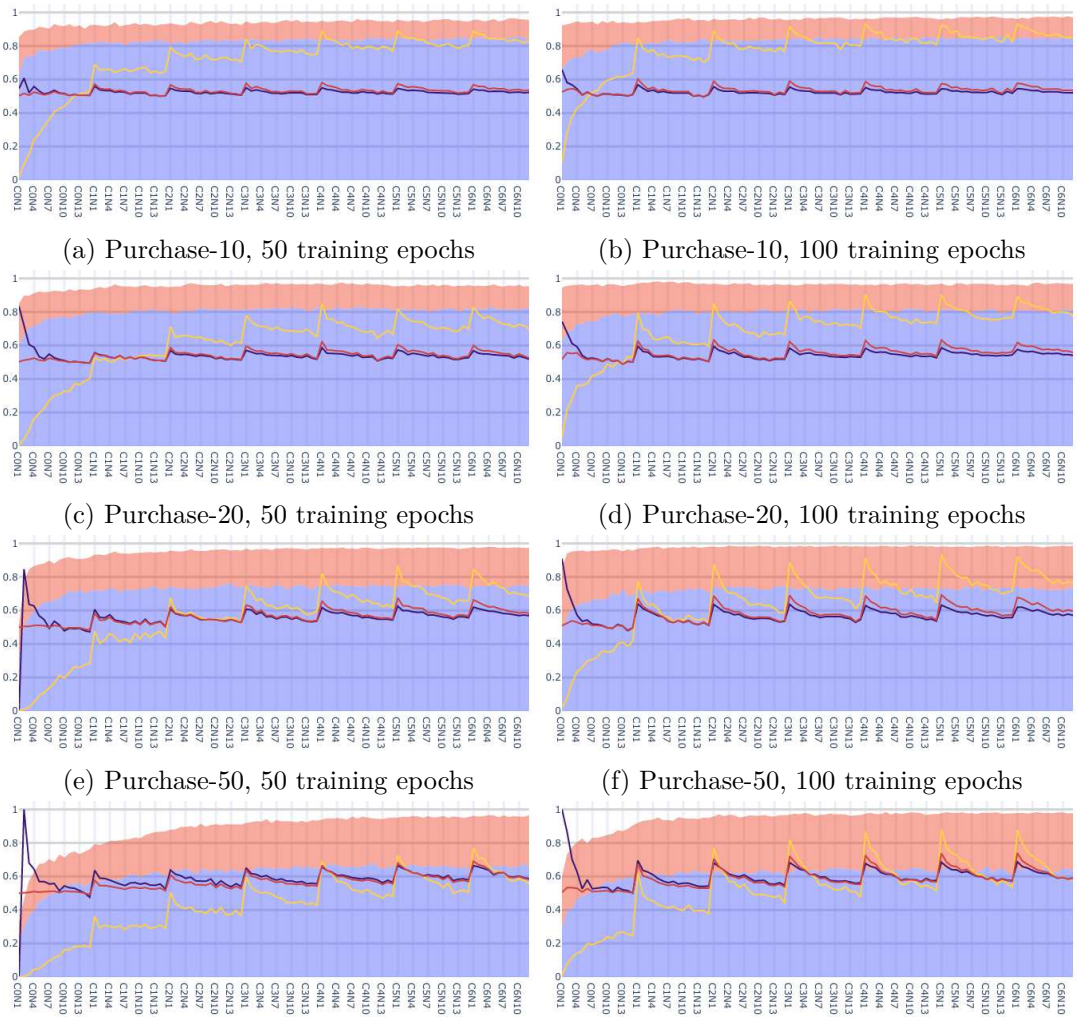
(g) Purchase-100, 50 training epochs

(h) Purchase-100, 200 training epochs

Figure 4.13: Membership Inference in sequential federated learning with ten nodes on the Purchase dataset

For Purchase-100, the membership inference from the model trained in the target node reaches at most 0.75. For the recall score, the difference is even more pronounced (up to 40% in some cases). With 200 epochs, the attack recall also has a sharper variation than precision and accuracy do. The spikes of accuracy and precision of the attack on the target model are also more significant at each considered cycle.

15 and 20 nodes. Sequential federated learning on 15 nodes behaves similarly to cases with fewer nodes. Figure 4.14 shows results for the setting with 15 nodes in the cases of 50 or 100 epochs of training per node. Again we see the high peaks in the membership inference attack accuracy, precision and recall on the target model right after training in



(a) Purchase-10, 50 training epochs

(b) Purchase-10, 100 training epochs

(c) Purchase-20, 50 training epochs

(d) Purchase-20, 100 training epochs

(e) Purchase-50, 50 training epochs

(f) Purchase-50, 100 training epochs

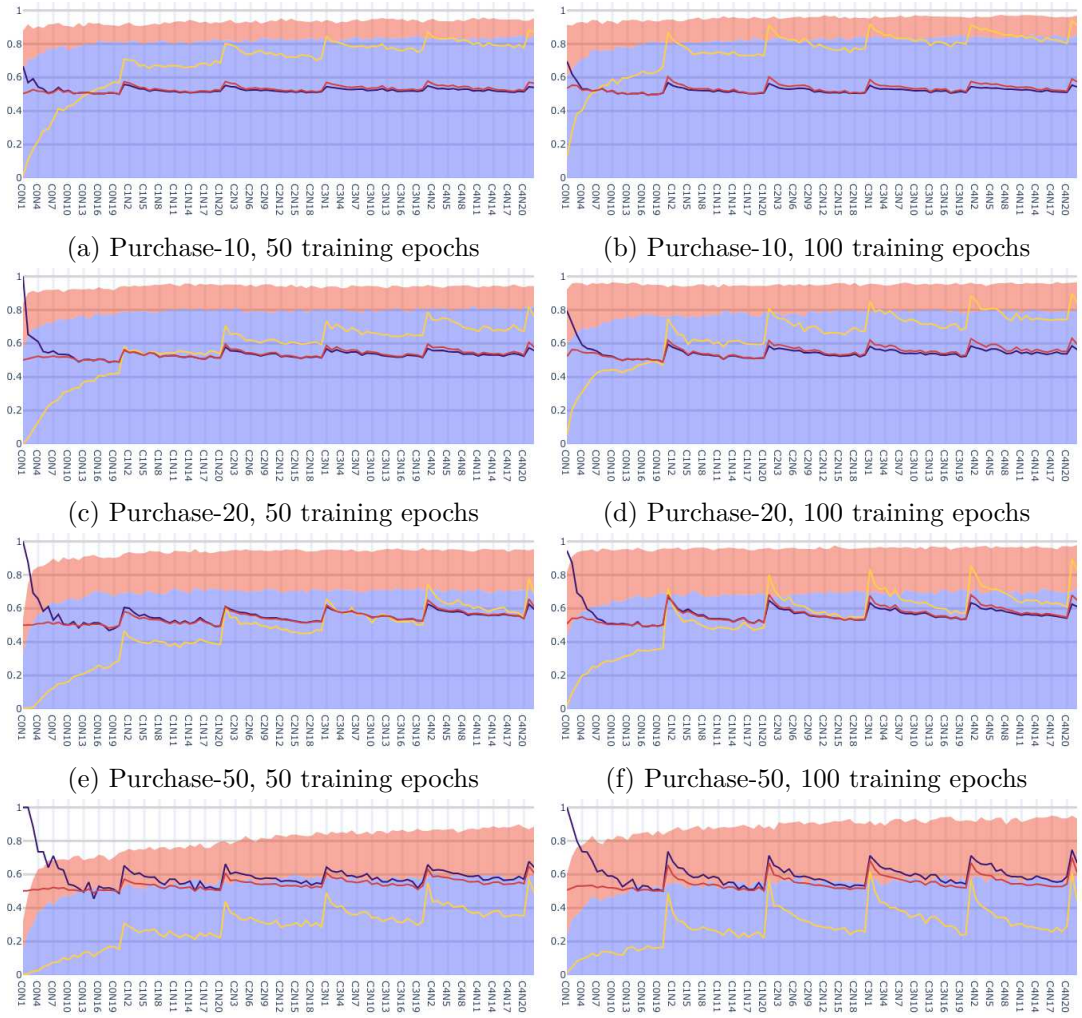
(g) Purchase-100, 50 training epochs

(h) Purchase-100, 100 training epochs

Figure 4.14: Membership Inference in sequential federated learning with 15 nodes on the Purchase dataset

the target node N_1. The attack performance is also increasing with more classes in the classification task. The same holds for the case with 20 nodes (see Figure 4.15). However, on the Purchase-100 dataset, the attack has lower performance, than with less number of nodes in the federated learning setting.

Summary Table 4.2 represents the results for sequential federated learning with the different number of epochs per node, on the Purchase dataset classification tasks. We compared the accuracy of the target model on the test set in the last federated cycle, as well as the maximum attack accuracy reached during the whole federated learning process. In general, training with 50 epochs results in the best performance, while the



(a) Purchase-10, 50 training epochs

(b) Purchase-10, 100 training epochs

(c) Purchase-20, 50 training epochs

(d) Purchase-20, 100 training epochs

(e) Purchase-50, 50 training epochs

(f) Purchase-50, 100 training epochs

(g) Purchase-100, 50 training epochs

(h) Purchase-100, 100 training epochs

Figure 4.15: Membership Inference in sequential federated learning with 20 nodes on the Purchase dataset

membership inference attack achieves the worst scores in this setting. An exception is a Purchase-2 dataset, on which 200 epochs provide the best performance on the main classification task with an accuracy score of 0.971 and more with the different number of nodes in the setting. The membership inference does not work on Purchase-2 in any settings that we tried.

All the other datasets (Purchase-10 – Purchase-100) show the same tendency that with a larger number of epochs the attack accuracy is higher. The attack accuracy also increases with a growing number of nodes in the settings. Nevertheless, the setting with 20 nodes and 200 epochs shows worse attack accuracy than for 15 nodes, but only on the

Table 4.2: Membership Inference attack on the Purchase dataset in sequential federated learning with a different number of processing nodes.

50 training epochs per node

Dataset	Target model accuracy on the test set				Attack accuracy			
	3 nodes	10 nodes	15 nodes	20 nodes	3 nodes	10 nodes	15 nodes	20 nodes
Purchase-2	0.969	0.968	0.97	0.97	0.516	0.526	0.526	0.53
Purchase-10	0.874	0.87	0.874	0.873	0.58	0.586	0.593	0.597
Purchase-20	0.828	0.829	0.83	0.828	0.617	0.63	0.633	**0.645**
Purchase-50	0.763	0.767	0.764	0.745	0.661	**0.689**	**0.686**	**0.69**
Purchase-100	0.678	0.692	0.686	0.622	0.698	**0.736**	**0.736**	**0.737**

100 training epochs per node

Dataset	Target model accuracy on the test set				Max attack accuracy			
	3 nodes	10 nodes	15 nodes	20 nodes	3 nodes	10 nodes	15 nodes	20 nodes
Purchase-2	0.968	0.968	0.97	0.969	0.518	0.524	0.523	0.53
Purchase-10	0.864	0.874	0.865	0.868	0.588	0.608	0.63	0.649
Purchase-20	0.815	0.818	0.821	0.815	0.619	0.639	**0.648**	0.643
Purchase-50	0.755	0.748	0.75	0.721	0.668	0.691	**0.704**	**0.703**
Purchase-100	0.667	0.667	0.664	0.602	0.707	0.744	**0.761**	0.744

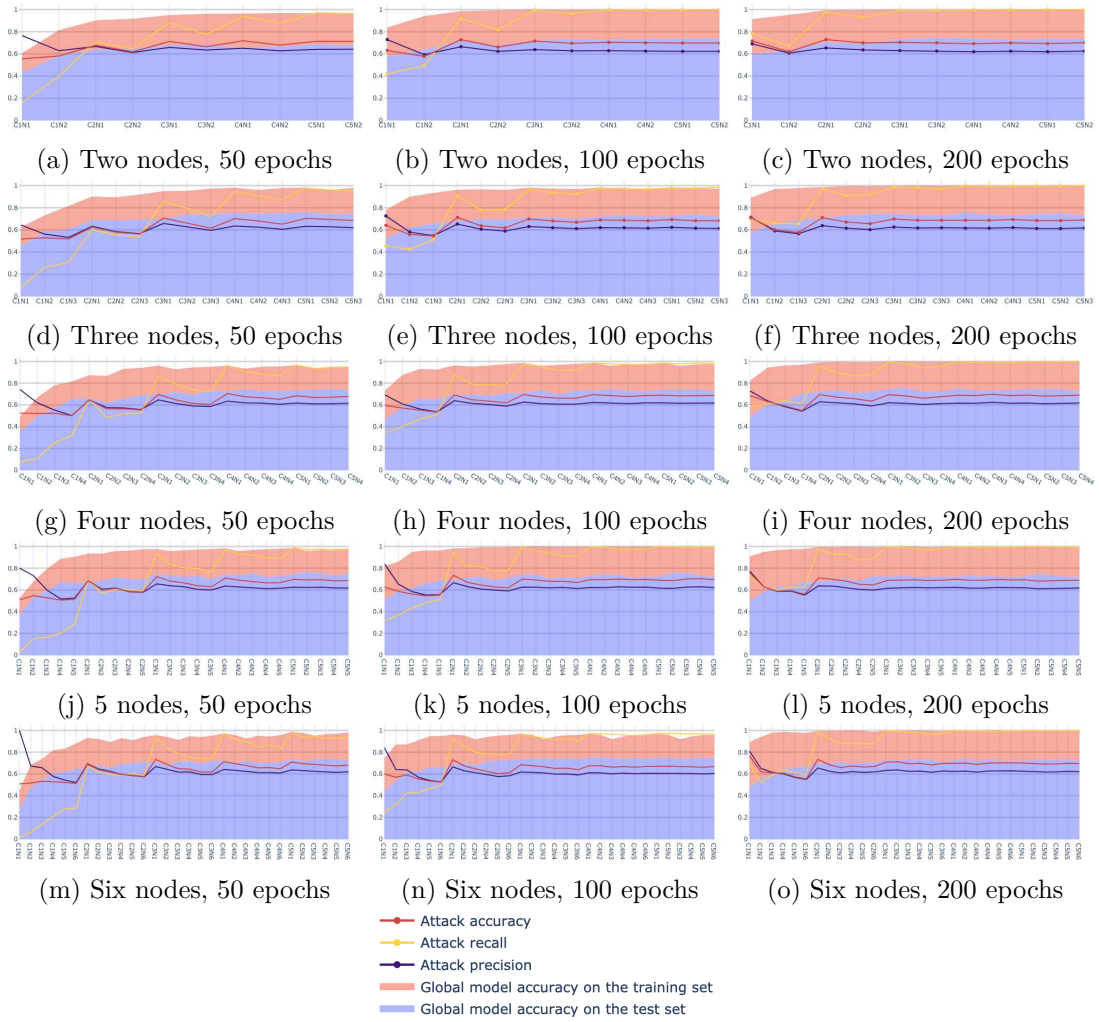
200 training epochs per node

Dataset	Target model accuracy on the test set				Max attack accuracy			
	3 nodes	10 nodes	15 nodes	20 nodes	3 nodes	10 nodes	15 nodes	20 nodes
Purchase-2	0.973	0.971	0.972	0.972	0.515	0.536	0.535	0.537
Purchase-10	0.866	0.868	0.865	0.861	0.591	0.658	0.687	**0.701**
Purchase-20	0.808	0.817	0.818	0.796	0.634	0.677	0.716	**0.723**
Purchase-50	0.726	0.729	0.728	0.715	0.688	0.745	**0.777**	**0.783**
Purchase-100	0.619	0.636	0.632	0.548	**0.724**	**0.78**	**0.808**	**0.775**

Purchase-100 dataset. With 100 epochs, the attack accuracy with 15 nodes is also higher than with 20 nodes on Purchase-100, Purchase-50 and Purchase-20.

With three nodes in the setting, we have 10, 000 instances at each node, which corresponds to the experiments in centralized learning with 10, 000 instances represented in Table 4.1. Comparing these results in centralized learning with sequential federated learning with three nodes, we observe, that membership inference accuracy is lower in federated learning. Even with 200 epochs per node maximum attack accuracy is lower in sequential federated learning that in centralized.

The specific problem in sequential federated learning setup is the drastic rise of the attack performance after training at a target node. We conclude that when there are several malicious users in sequential federated learning, they can combine their knowledge and observe the dynamics of membership inference attack performance at the different stages of the training and break the privacy of data records in a federated learning setting. We investigate mitigation techniques for this issue in Section 4.5.



(a) Two nodes, 50 epochs (b) Two nodes, 100 epochs (c) Two nodes, 200 epochs

(d) Three nodes, 50 epochs (e) Three nodes, 100 epochs (f) Three nodes, 200 epochs

(g) Four nodes, 50 epochs (h) Four nodes, 100 epochs (i) Four nodes, 200 epochs

(j) 5 nodes, 50 epochs (k) 5 nodes, 100 epochs (l) 5 nodes, 200 epochs

(m) Six nodes, 50 epochs (n) Six nodes, 100 epochs (o) Six nodes, 200 epochs

Figure 4.16: Membership Inference in sequential federated learning with a different number of nodes on Location dataset

4.3.2 Location dataset

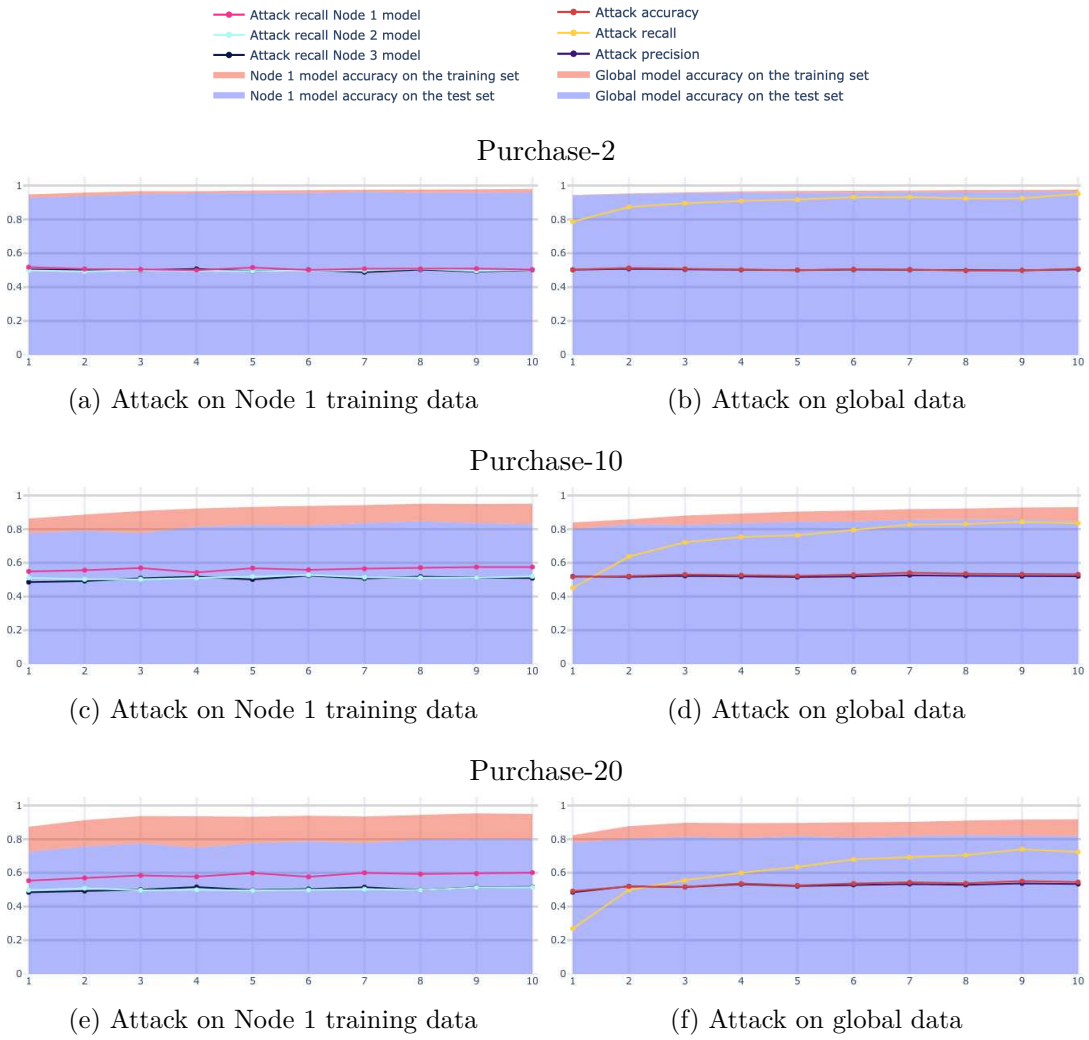
We observe different results for the Location dataset, which has 30 classes and 1,200 instances in the training set. Figure 4.16 shows membership inference in sequential federated learning on a different number of nodes (from two to six). For the Purchase dataset, membership inference from the node N_1 maintains a high level of accuracy during the whole represented federated learning process. In the Location dataset, we see high variations only in the beginning of the training. In most of the cases, in the third federated cycle ($C3$) the accuracy reaches a plateau and does not rise drastically in the attacked node. This can be explained by relatively small amount of data at each node. On the third cycle the model learns maximum from the data and does not change much

after each following training.

Federated learning on two nodes (refer to Figures 4.16a to 4.16c) reaches the highest accuracy of the target model with 100 epochs per node. Membership inference accuracy, in this case, reaches the maximum in the second cycle and then levels off. With 200 epochs, we see a similar picture. However, with 50 epochs per node, the global model performs worse, while the attack accuracy is similar to other cases.

The settings with three to six nodes (see Figures 4.16d–4.16o) give similar results with the same number of epochs. The best accuracy of the target model is reached already with 50 epochs for each case with a different number of nodes in the setting. In this case, the membership inference has the highest accuracy on the third federated cycle and then remains on the level of 0.7. With 100 epochs, the maximum attack accuracy is reached on the second federated cycle and remains at the same level until the end of learning, at around 0.7. However, the maximum attack accuracy is higher with a larger number of nodes. The maximum membership accuracy is reached already in the first cycle with 200 epochs. On the next cycles, the membership inference performs worse and does not change much after training in the target node.

We conclude that membership inference on Location dataset is more successful when the target model is trained with a larger number of epochs per node. However, the results on Location dataset show that membership inference from the model trained in the target node differs with attacks on models trained in the other nodes only at first federated cycles.



(a) Attack on Node 1 training data (b) Attack on global data

(c) Attack on Node 1 training data (d) Attack on global data

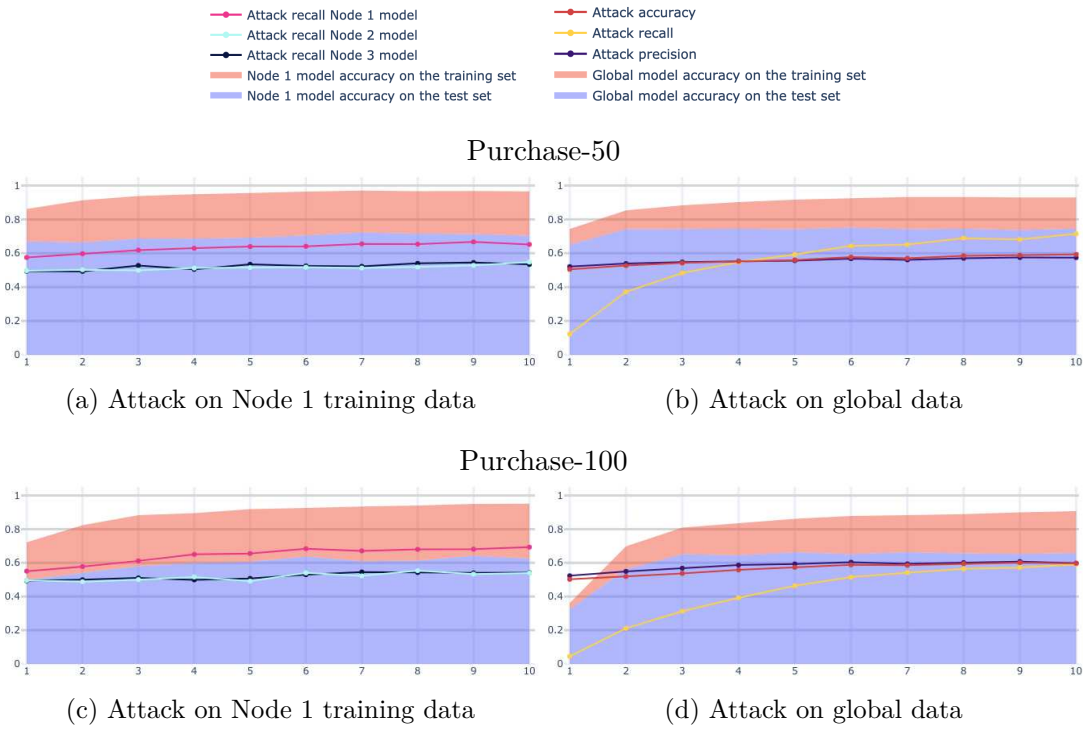
(e) Attack on Node 1 training data (f) Attack on global data

Figure 4.17: Membership Inference in parallel federated learning with three nodes on different classification tasks (Purchase-2, Purchase-10, Purchase-20) with 100 training epochs at each node in every FL cycle

4.4 Membership inference attack performance in parallel federated learning

4.4.1 Purchase dataset

Three nodes We evaluated the performance of parallel federated learning on different classification tasks and found the same pattern as in centralized and sequential federated learning scenarios: with a higher number of classes in a classification task, the membership inference performs better (see Figures 4.17, 4.18). On the Purchase-2 dataset (refer to Figures 4.17a and 4.17b), we could infer neither the members of the Node 1 training set



(a) Attack on Node 1 training data

(b) Attack on global data

(c) Attack on Node 1 training data

(d) Attack on global data

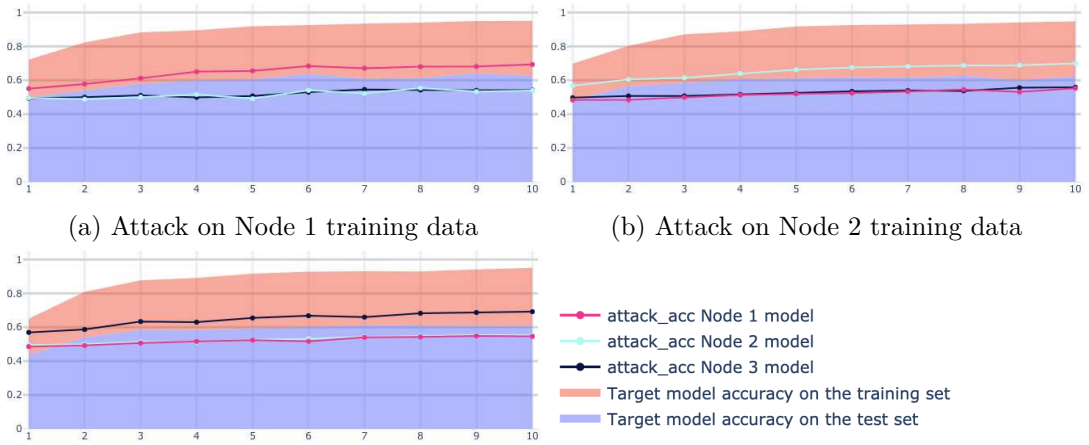
Figure 4.18: Membership Inference in parallel federated learning with three nodes on different classification tasks (Purchase-50, Purchase-100), with 100 epoch training at each node on every FL cycle

from its local model nor the members of the whole training set from the global model (when we try to infer from the global model in an instance was in the training set in some of the nodes).

Membership inference on Purchase-10 and Node 1 training data (see Figure 4.17c) performs slightly better than on Purchase-2 with an accuracy of around 0.57 at each considered federated learning cycle. In this case, one can see the difference between membership inference attack on the different models. The attacks on the models trained by Node 2 and Node 3 give a baseline performance of random guessing, while the attack on Node 1 model results in better accuracy. So an intruder can recognize in which node the data was used. For the global models and the joint global data, the attack performance stays on the baseline (see Figure 4.17d) of random guessing.

Results for Purchase-20 are similar to Purchase-10, however, the accuracy of an attack on Node 1 (refer to Figure 4.17e) is by 4% better than in the Purchase-10. The recall score of the attacks on global data on Purchase-10 and Purchase-20(see Figures 4.17d and 4.17f) increases with every new federated learning cycle.

We also compared the membership inference performance attacking data from each of the nodes in these settings. Figure 4.19 shows the accuracy of attacks on the training



(a) Attack on Node 1 training data

(b) Attack on Node 2 training data

(c) Attack on Node 3 training data

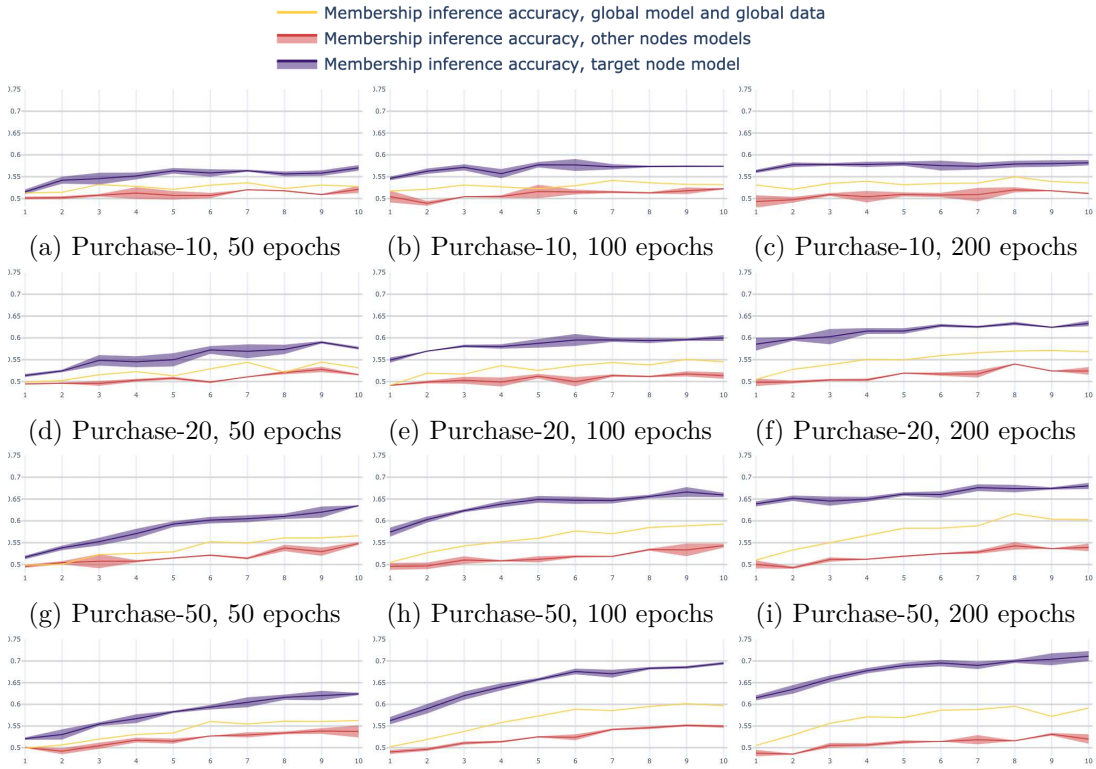
Figure 4.19: Accuracy of Membership Inference attack on model from different nodes, in parallel federated learning with three nodes, Purchase-100 classification task with 100 epoch training at each node on every FL cycle

data from different nodes. In all cases, we see that the attack on each node's training data has a higher performance if we target an attack on the model trained on this data. The accuracy score shows that a membership inference attack on a target node's model has a high performance, in each federated learning cycle. Moreover, the averaging of the models at the end of each federated cycle does not decrease the attack performance, which reaches its peak in the tenth cycle for all three nodes.

We also analyzed the effect of a different number of epochs in the training of a target model. Similar to the case of the sequential federated learning, the lower number of epochs gives a better accuracy on the test set for the main classification tasks. Also with 50 training epochs membership inference attack has the worst accuracy comparing to a larger number of epochs per node.

Figure 4.20 represents membership inference attack on the training data of node N_1. Each plot represents three different scores. We perform experiments attacking each node in the setting. We perform membership inference attack on ten different attack test set (for each node its own attack test set). The red and purple lines show the mean and standard deviation of the scores from these ten different attack test sets. In parallel federated learning, each node trains a model in parallel on its data. The purple line shows the attack accuracy on the model which was trained by the target node. The red line represents the attack accuracy on models trained in all the other nodes. The yellow line represents the attack accuracy on the global model with the global joint test set, one half of which consists of training data from all ten nodes with equal proportions and the other half is the test set.

On Purchase-10 and Purchase-20 (cf. Figures 4.20a to 4.20f), the membership inference



(a) Purchase-10, 50 epochs (b) Purchase-10, 100 epochs (c) Purchase-10, 200 epochs

(d) Purchase-20, 50 epochs (e) Purchase-20, 100 epochs (f) Purchase-20, 200 epochs

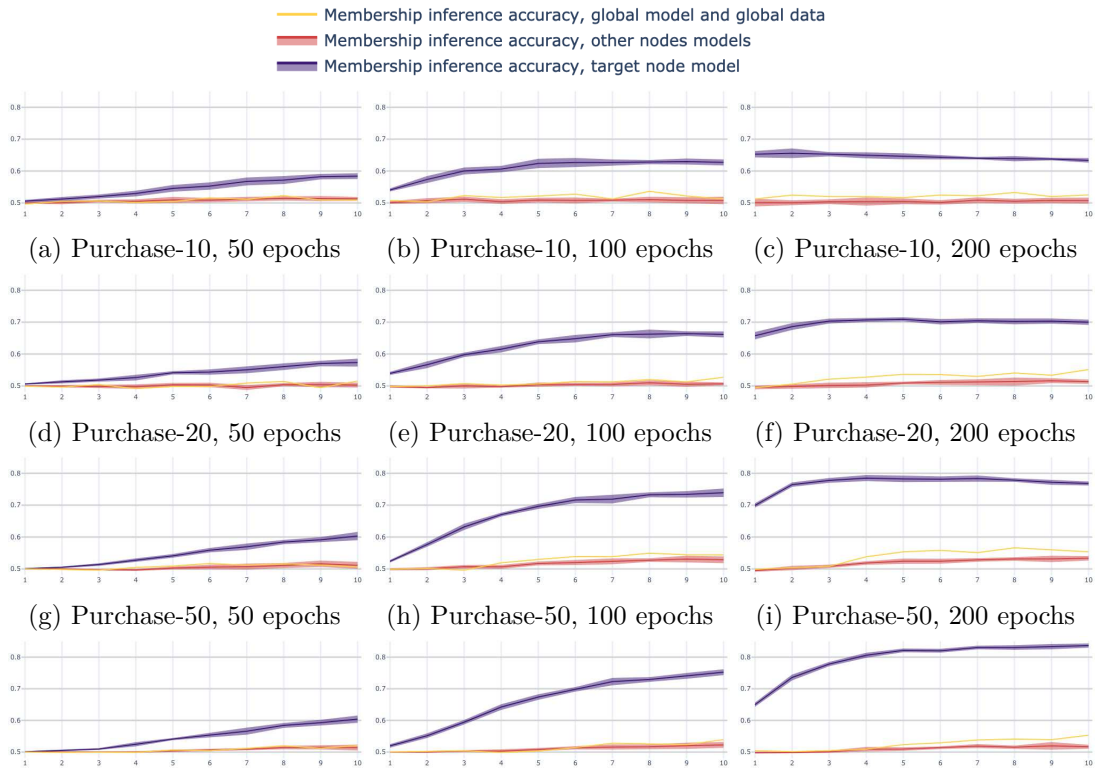
(g) Purchase-50, 50 epochs (h) Purchase-50, 100 epochs (i) Purchase-50, 200 epochs

(j) Purchase-100, 50 epochs (k) Purchase-100, 100 epochs (l) Purchase-100, 200 epochs

Figure 4.20: Membership Inference in parallel federated learning with three nodes on the Purchase dataset.

is rather small for all epochs we tested (50, 100, 200), and does not vary much, reaching at most 0.58 on Purchase-10 and 0.63 on Purchase-20. On Purchase-50 dataset (see Figures 4.20g to 4.20i), we see that the membership inference performance is better with a larger number of training epochs per node. The same trend is observed for the Purchase-100 dataset (refer to Figures 4.20j to 4.20l). During the whole considered federated learning process, the membership inference accuracy increases and we reach the maximum attack accuracy in the last federated cycle we analysed. However, the growth of accuracy slows down and flattens by the tenth cycle in most of the cases.

Ten nodes Figure 4.21 shows the results for membership inference attack in parallel federated learning with ten processing nodes. For all classification tasks on the Purchase dataset, we compared the accuracy of the membership inference attack with a different number of training epochs. In all cases, we observe the trend that the attack accuracy is increasing with an increase in the number of epochs during the training. There is a noticeable difference in the attack performance for utilizing either a 50 or a 100 epochs. However, the difference between 100 and 200 epochs is relatively small. Moreover, for all



(a) Purchase-10, 50 epochs (b) Purchase-10, 100 epochs (c) Purchase-10, 200 epochs

(d) Purchase-20, 50 epochs (e) Purchase-20, 100 epochs (f) Purchase-20, 200 epochs

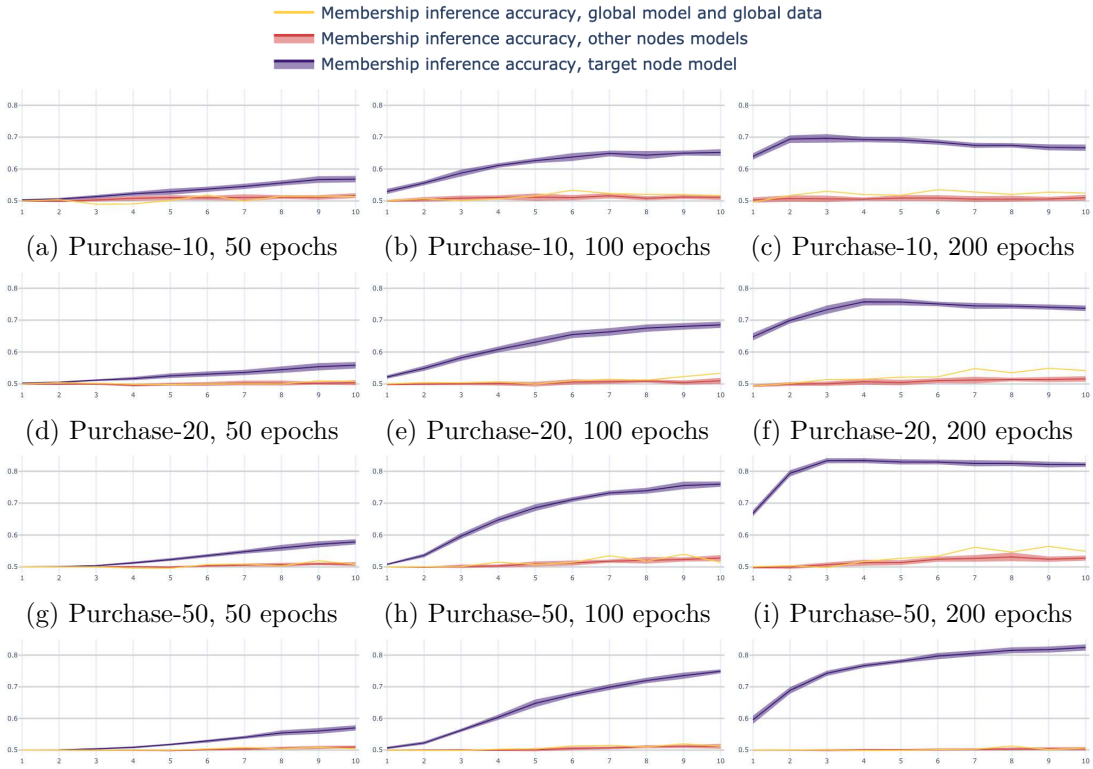
(g) Purchase-50, 50 epochs (h) Purchase-50, 100 epochs (i) Purchase-50, 200 epochs

(j) Purchase-100, 50 epochs (k) Purchase-100, 100 epochs (l) Purchase-100, 200 epochs

Figure 4.21: Membership Inference in parallel federated learning with ten nodes on Purchase dataset.

various epoch numbers, the attack accuracy increases with every new federated cycle. The maximum attack accuracy is reached always with 200 epochs; for the Purchase-10 it is 0.656 (cf. Figure 4.22c), for the Purchase-20 0.709 (cf. Figure 4.22f), for the Purchase-50 0.785 (cf. Figure 4.22i), and for the Purchase-100 the maximum achieved attack accuracy is 0.836 (cf. Figure 4.22l. As well as in the previous setting, with ten nodes in parallel federated learning, we also see that membership inference attack performs better on the dataset with a larger number of classes in a classification task.

15 nodes Membership inference attack in parallel federated learning with 15 nodes performs similarly to the setting with ten nodes. Figure 4.22 shows that with a larger number of epochs, the accuracy of membership inference increases for all represented classification tasks. For the Purchase-10, training with 100 and 200 epochs (see Figures 4.22b and 4.22c) causes an even higher attack accuracy score, relatively to the case with ten nodes. This may be explained by the fact that in our setting with 15 nodes, each participant has less of the training data than in the setting with ten nodes (2,000 instances per node versus 3,000 instances). With less training data, the model



(a) Purchase-10, 50 epochs (b) Purchase-10, 100 epochs (c) Purchase-10, 200 epochs

(d) Purchase-20, 50 epochs (e) Purchase-20, 100 epochs (f) Purchase-20, 200 epochs

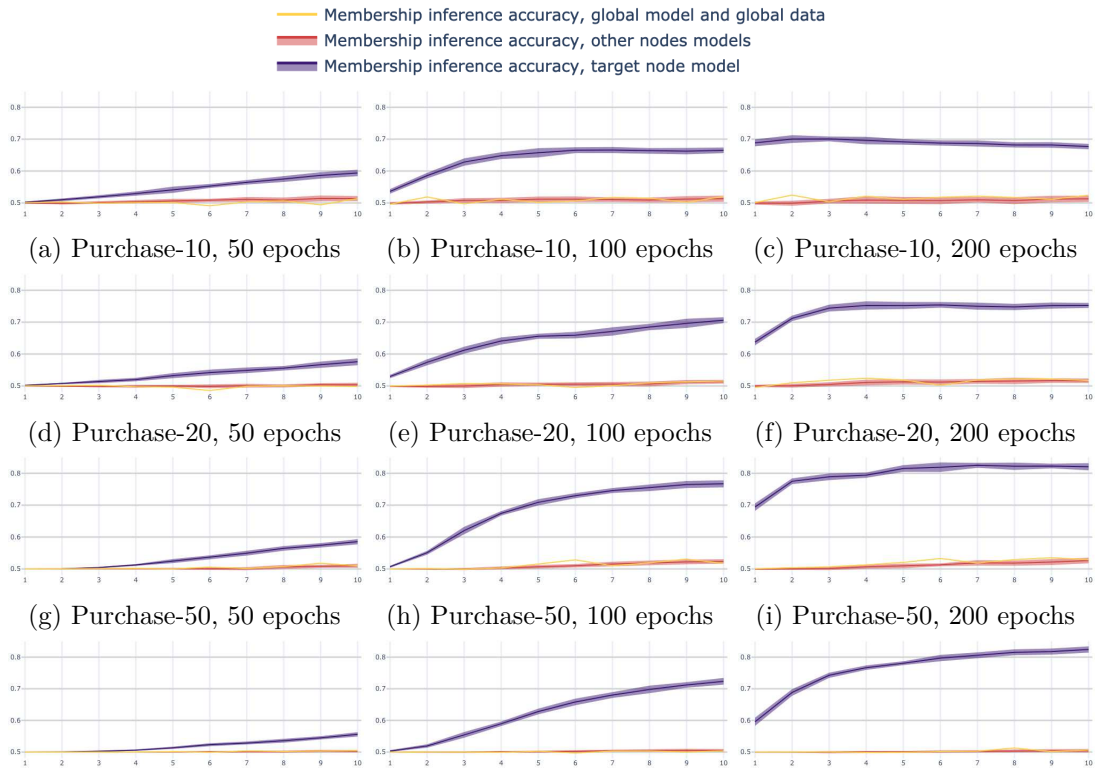
(g) Purchase-50, 50 epochs (h) Purchase-50, 100 epochs (i) Purchase-50, 200 epochs

(j) Purchase-100, 50 epochs (k) Purchase-100, 100 epochs (l) Purchase-100, 200 epochs

Figure 4.22: Membership Inference in parallel federated learning with 15 nodes on the Purchase dataset

remembers the training data more specifically and generalizes worse, which causes a relatively high accuracy of membership inference. We observe the same trend for all other classification tasks trained with 100 and 200 epochs per node (See Figures 4.22e – 4.22l). However, training with 50 epochs per node produces a different result: with 10 nodes, the attack accuracy is on average 2% higher than in the setting with 15 nodes (see Figure 4.22a,d,g,j). Moreover this result is similar while attacking all the nodes in the setting.

20 nodes Figure 4.23 represents the results for membership inference on Purchase dataset with 20 nodes in a federated learning setting. One can notice a similar trend: with the larger number of epochs per node, membership inference accuracy increases. Interestingly, the attack accuracy grows faster on the dataset with 50 classes (see Figures 4.23g to 4.23i) than with 100 classes (see Figures 4.23j to 4.23l) As well as in cases with less number of node in the setting, the membership inference accuracy grows throughout the whole considered federated learning process for 10 federated cycles.



(a) Purchase-10, 50 epochs (b) Purchase-10, 100 epochs (c) Purchase-10, 200 epochs

(d) Purchase-20, 50 epochs (e) Purchase-20, 100 epochs (f) Purchase-20, 200 epochs

(g) Purchase-50, 50 epochs (h) Purchase-50, 100 epochs (i) Purchase-50, 200 epochs

(j) Purchase-100, 50 epochs (k) Purchase-100, 100 epochs (l) Purchase-100, 200 epochs

Figure 4.23: Membership Inference in parallel federated learning with 20 nodes on Purchase dataset.

Summary Table 4.3 represents the results of the evaluation of parallel federated learning with a different number of nodes in the settings. Comparing the effectiveness of federated learning with a different number of epochs per node, we conclude that for the Purchase-10, Purchase-20 and the Purchase-50 classification tasks, 50 training epochs per node gives the highest accuracy on the test set on the main classification task. At the same time for all classification tasks, the membership accuracy is the lowest in the settings with 50 training epochs per node. The Purchase-100 dataset has the best test set accuracy with 100 training epochs on the settings with 10, 15 and 20 nodes. However, the membership inference is as well more effective in this case than with 50 epochs. The trend of more effective membership inference accuracy with the larger number of classes in classification task persists.

Comparing results of three nodes to centralized learning performance from Table 4.1, we observe that membership inference performs worse even with 200 epochs per node. That can be explained due to the generalizing the model at each federated learning iteration with models from other nodes. Therefore the target model is distorted by the models trained on other data, what makes the membership inference less successful.

Table 4.3: Membership Inference attack on the Purchase dataset in parallel federated learning with different numbers of processing nodes.

50 training epochs per node

Dataset	Target model accuracy on the test set				Attack accuracy			
	3 nodes	10 nodes	15 nodes	20 nodes	3 nodes	10 nodes	15 nodes	20 nodes
Purchase-2	0.961	0.949	0.946	0.952	0.511	0.519	0.528	0.530
Purchase-10	0.84	0.841	0.833	0.835	0.578	0.6	0.591	0.612
Purchase-20	0.839	0.818	0.812	0.822	0.581	0.593	0.572	0.591
Purchase-50	0.77	0.748	0.745	0.741	0.635	0.625	0.591	0.601
Purchase-100	0.676	0.65	0.647	0.621	0.626	0.621	0.588	0.571

100 training epochs per node

Dataset	Target model accuracy on the test set				Attack accuracy			
	3 nodes	10 nodes	15 nodes	20 nodes	3 nodes	10 nodes	15 nodes	20 nodes
Purchase-2	0.964	0.956	0.952	0.952	0.515	0.525	0.53	0.539
Purchase-10	0.859	0.839	0.829	0.814	0.575	0.642	0.665	**0.678**
Purchase-20	0.825	0.814	0.822	0.786	0.607	0.677	0.699	**0.723**
Purchase-50	0.752	0.733	0.722	0.707	0.664	0.757	0.772	**0.793**
Purchase-100	0.664	0.675	0.67	0.627	0.699	**0.768**	0.761	0.741

200 training epochs per node

Dataset	Target model accuracy on the test set				Attack accuracy			
	3 nodes	10 nodes	15 nodes	20 nodes	3 nodes	10 nodes	15 nodes	20 nodes
Purchase-2	0.967	0.963	0.958	0.965	0.516	0.527	0.541	0.540
Purchase-10	0.863	0.835	0.83	0.829	0.59	0.645	0.683	**0.692**
Purchase-20	0.814	0.801	0.789	0.786	0.642	0.713	0.75	**0.768**
Purchase-50	0.739	0.725	0.697	0.671	0.69	0.775	0.836	**0.84**
Purchase-100	0.669	0.636	0.647	0.581	0.722	0.85	**0.888**	0.84

With 200 epochs in the setting on Purchase-100 dataset, the membership inference reaches an accuracy of 0.89. At the same time, the performance on the main classification task with 200 epochs is the worst. Therefore, we conclude that a large number of training epochs on Purchase-100 dataset is not recommended to use.

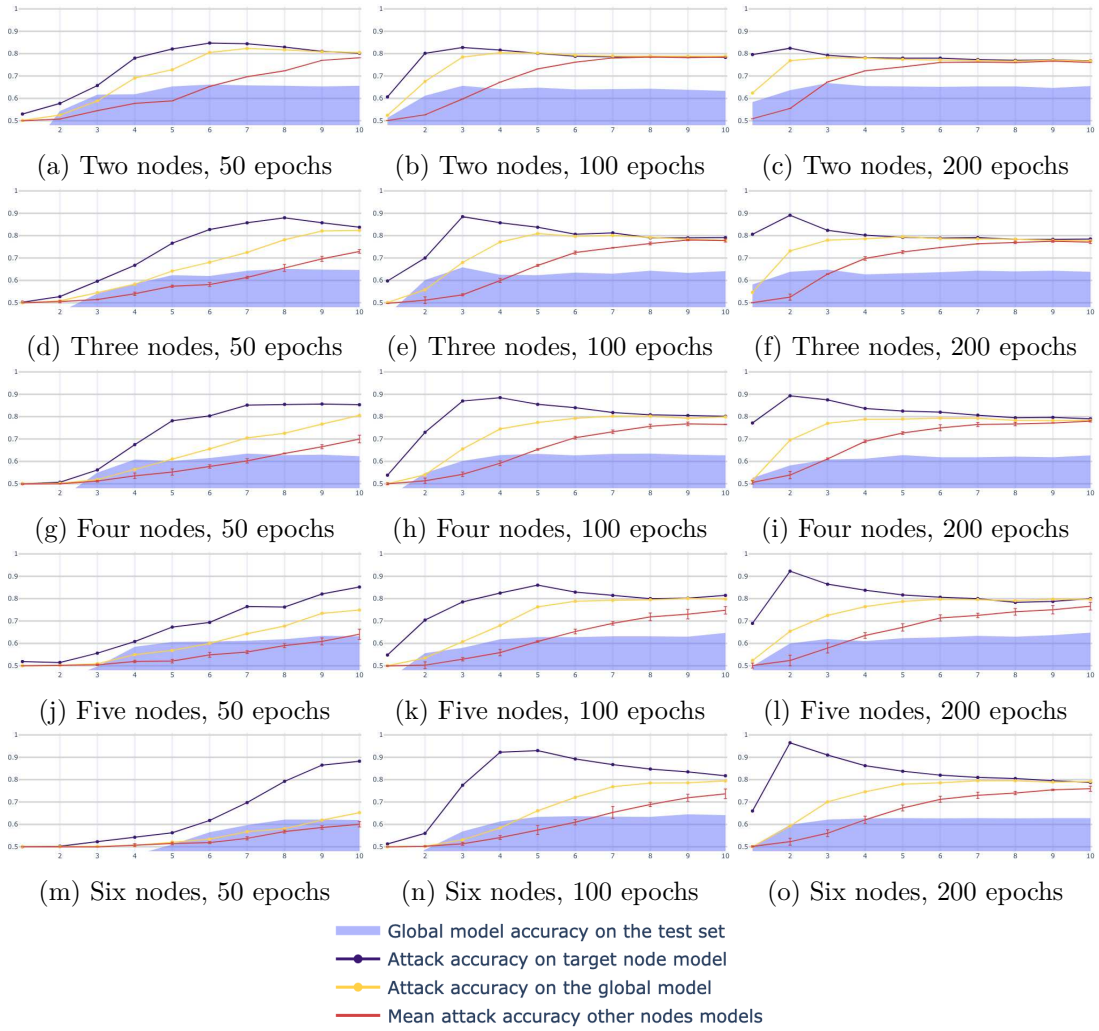


Figure 4.24: Membership Inference in parallel federated learning on the Location dataset with different numbers of nodes and epochs per node.

4.4.2 Location dataset

Figure 4.24 shows the results of parallel federated learning performance on the Location dataset. We reach high membership inference accuracy on Location dataset for all settings. With 50 epochs per node, the attack accuracy on the target node model is rather low in the beginning of the training. However, it increases drastically during the next federated learning cycles. With 100 and 200 epochs, the maximum membership inference is reached in the beginning of the training, and then it is declining. This can be explained by the fact that with 50 training epochs at each node, it takes more rounds for the model to fit the training data well, therefore membership inference score is growing slower at each cycle, than with 100 epochs. With 200 epochs, the overtraining happens

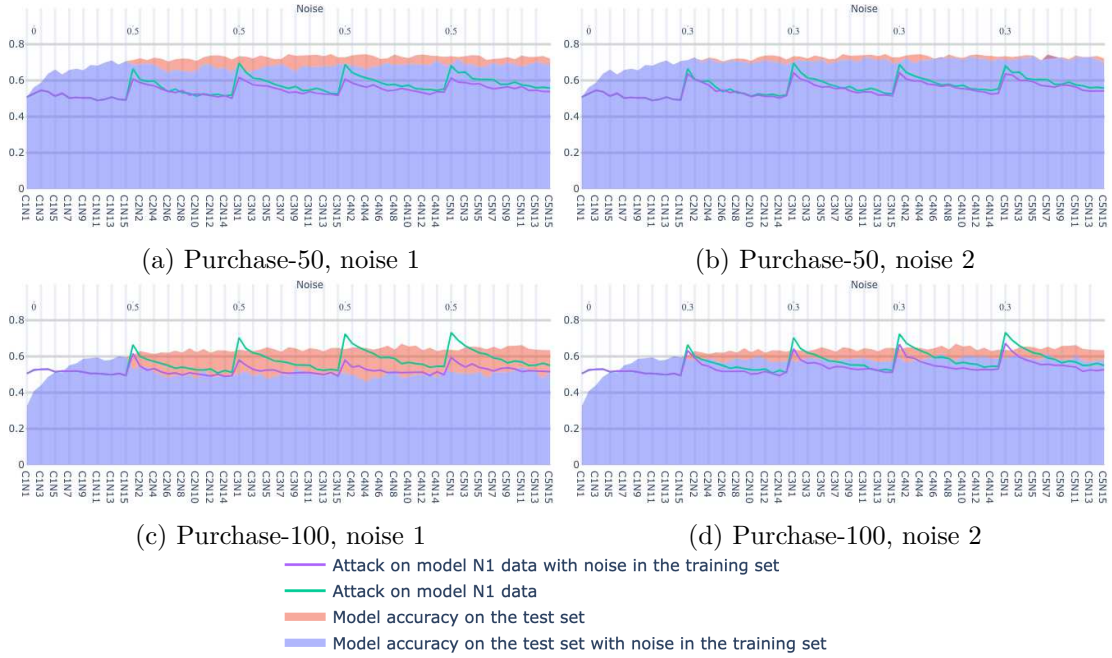
even faster. Already in the second federated cycle, the membership inference attack on the target node's model reaches a maximum score for all considered settings with different number of nodes.

On the other models (the global model and the models trained in the other nodes than the target one) the membership inference accuracy is increasing during the whole federated learning process. However, with 50 epochs per node these attacks do not reach accuracy score of the attack on a target model (except the case with two nodes). With 100 and 200 epochs per node the scores from all considered attacks converge by the end of the training. The more nodes in the setting the accuracy scores converge. When we have the same accuracy score from a global model and all local models, we cannot predict anymore from which node the data came.

With two nodes in the learning setup, we reach as highest attack accuracy a rate of 0.85 with 50 epochs per node (see Figure 4.24a). We observe that by the tenth cycle the attack on a target node's model as well as the attacks on other nodes' models, and the attack on the global model perform with the same accuracy level. With 100 and 200 epochs they give the same score starting from sixth cycle (cf. Figures 4.24b and 4.24c). However, the maximum of the attack accuracy on a model trained in the target node is reached on the third federated cycle with 100 epochs, and on the second cycle with 200 epochs.

With three nodes in the setting, the maximum of the membership inference accuracy of around 0.9 is achieved for all considered numbers of epochs (see Figures Figures 4.24d to 4.24f).

Federated learning on four nodes behaves similarly to the three-nodes case. The setting with five nodes shows an even higher result in attack accuracy, which equals to 0.92 (4.24l). With six nodes and 200 epochs we obtain an attack accuracy of 0.965 (see Figure 4.24c).



(a) Purchase-50, noise 1

(b) Purchase-50, noise 2

(c) Purchase-100, noise 1

(d) Purchase-100, noise 2

Figure 4.25: Membership inference accuracy and accuracy of the target model evaluation after adding noise to the training data at each node. The Purchase-50 and Purchase-100 datasets in sequential federated learning with 15 nodes.

4.5 Mitigation strategies evaluation

4.5.1 Sequential federated learning

As a mitigation strategy for the membership inference attack, we used noise addition to the data. We applied mitigations in federated learning for the settings which allowed us to reach good accuracy on the main classification task. We developed a defence strategy of increasing and decreasing the amount of noise based on the results of the evaluation membership inference attack performance in sequential and parallel federated learning. In both sequential and parallel learning, the noise was added to the training data of every participant in the training. We evaluate the effect of the noise on the membership inference accuracy and the accuracy of the target model on the main classification task.

Purchase dataset The sequential federated learning performs the best with 50 training epochs at each node on classification tasks with a smaller number of classes (Purchase-2, Purchase-10 and Purchase-20). Membership inference attack does not reach the accuracy score of more than 0.65 in these configurations. Thus, we choose Purchase-50 and Purchase-100 datasets for testing mitigations, as these datasets showed to be more vulnerable to membership inference.

In sequential federated learning, the highest risk of membership inference was observed in

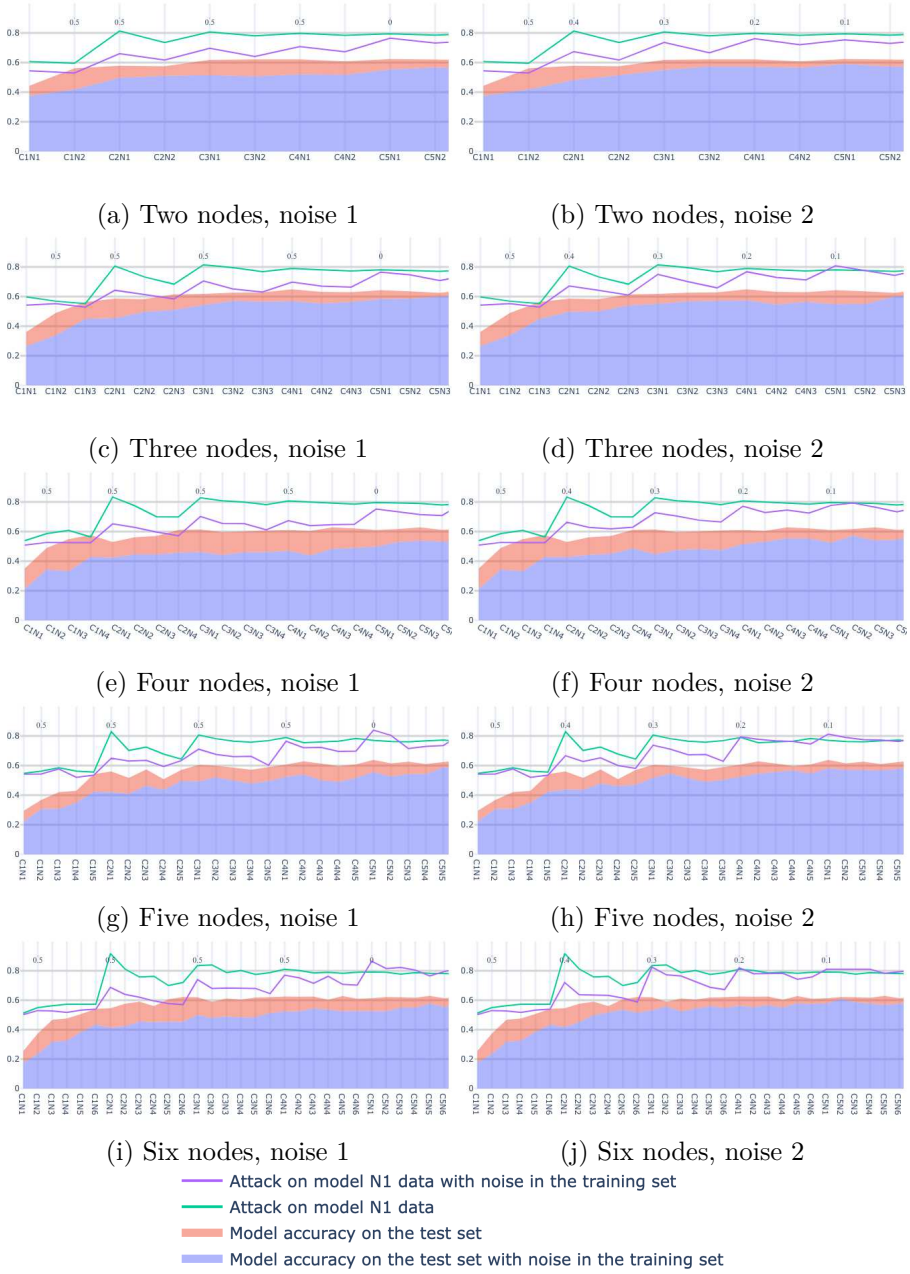


Figure 4.26: Membership inference accuracy and accuracy of the target model evaluation after adding noise to the training data at each node. Location dataset in sequential federated learning.

the settings with 15 and 20 processing nodes. Results for both cases have similar trends in membership inference attack performance. However with 15 nodes, we reached the highest attack accuracy, therefore we concentrate on the setting with 15 nodes and 100

training epochs at each node. We compare two configurations of noise. The "noise 1" corresponds to the setting with constant noise added to 50% of the training data at each node. In the case of "noise 2", we start training with "clean data", and in each following cycle, we add noise to 30% of the training data. The strategy of higher noise at the end of the federated learning process is based on the results from performance in sequential federated learning. In the first cycle, the membership inference attack has low accuracy close to random guessing, however, on the next cycles, the accuracy of attack increases to 0.7. Therefore we decide to use clean data in the first cycle. In the next cycles, we add noise to the training data to avoid overfitting, thus avoiding higher membership inference accuracy.

Figure 4.25 shows the effect of adding noise in the settings with 15 nodes on the Purchase-50 and the Purchase-100 datasets. On the Purchase-50 dataset, we decrease the attack accuracy by almost 10% by adding the noise to the 50% of training data at each federated cycle (refer to Figure 4.25a). However, we loose 3% in the target model accuracy. With noise 2 we reach a target model accuracy close to the original result without the noise. However, the membership inference accuracy decreased at most only by 5%, as compared to the training with no noise at all (refer to Figure 4.25b).

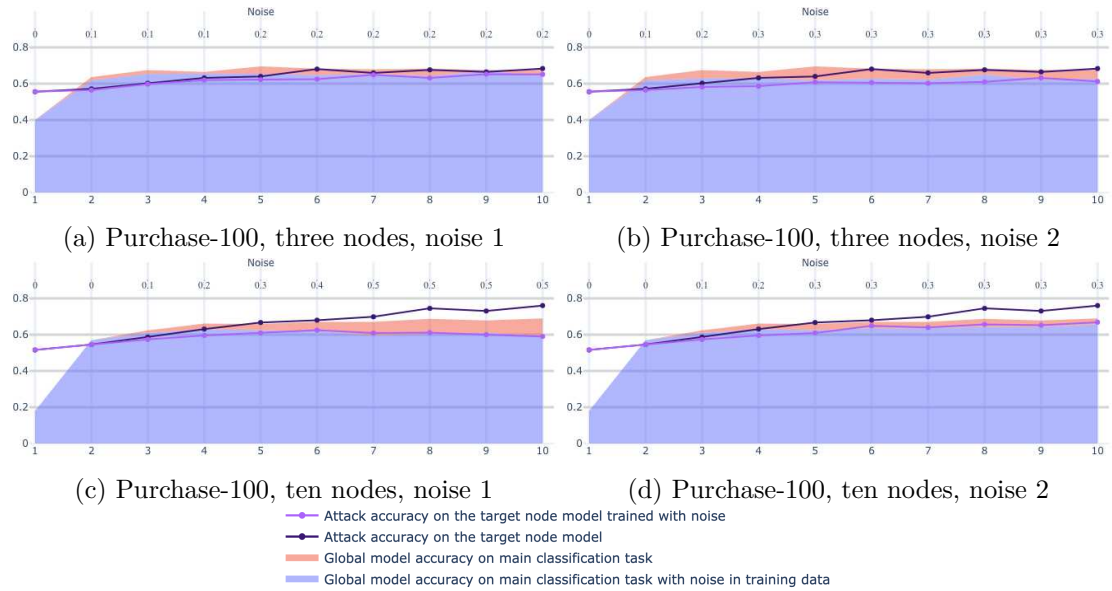
For Purchase-100, we tried the same amount of noise and observed a similar trend. With higher noise, membership inference attack drops in performance, but here we lose a much higher amount in the effectiveness of the model around 10% (refer to Figure 4.25c). With noise 2 we lose only 5% in effectiveness and also reduce the attack accuracy only by 5% (cf. Figure 4.25d).

Therefore, we conclude that the noise 2 performed better on Purchase-50 and Purchase-100 dataset allowing to reduce attack accuracy while loosing less in the effectiveness of the target model.

Location dataset For Location dataset, we present results with other noise configurations. In the first configurations, we have 50% of noise records at each node in each cycle until the last one. "Noise 2" corresponds to the setting with a decreasing amount of noise in each cycle from 0.5 to 0.1 with the step of 0.1.

The noise addition worked quite well as a mitigation strategy on the Location dataset. With two and three nodes in the setting, we reduced the attack accuracy by up to 17% and managed to maintain a high utility of the models, with difference of around 3% (see Figures 4.26a, 4.26d).

In sequential federated learning with four nodes, we lost around 7% in utility of the final model with larger amount of noise (see Figures 4.26e and 4.26f). For five and six nodes in the setting, the noise works the best at the beginning of the training, and then the membership inference returns to the original outcome when we fully eliminate the noise (refer to Figures 4.26g–4.26j). By tenth federated cycle, the effectiveness of the target model with both considered noises converges to the accuracy reached by federated learning without any noise. This is relevant for cases with all considered number of nodes.



(a) Purchase-100, three nodes, noise 1

(b) Purchase-100, three nodes, noise 2

(c) Purchase-100, ten nodes, noise 1

(d) Purchase-100, ten nodes, noise 2

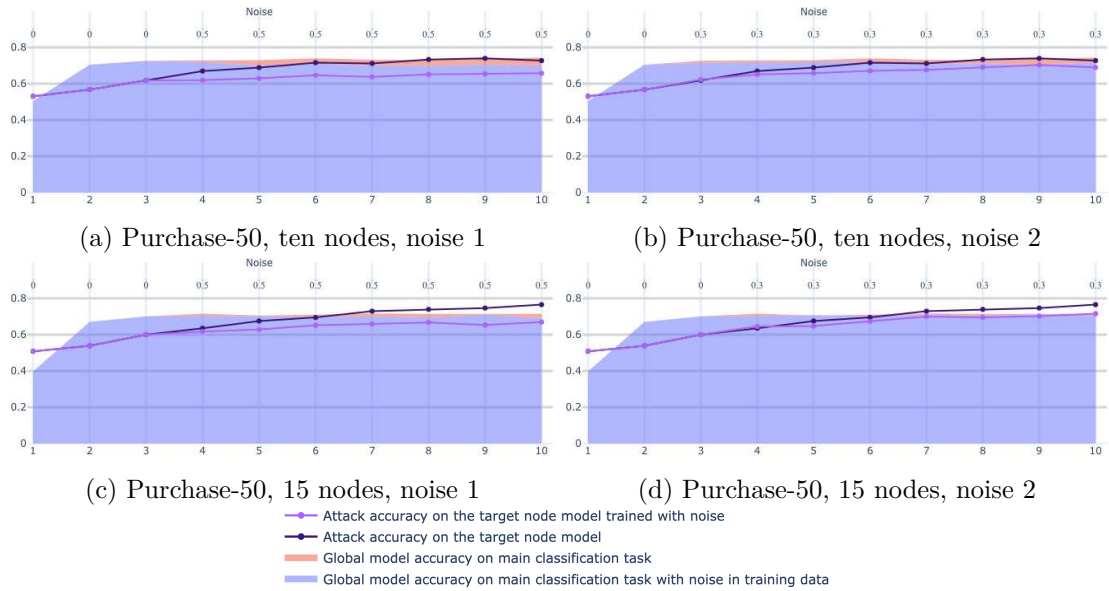
Figure 4.27: Membership inference accuracy and accuracy of the target model evaluation after adding noise to the training data at each node. Purchase-100 in parallel federated learning.

4.5.2 Parallel federated learning

Purchase dataset In the parallel federated learning, the performance of mitigation on the Purchase dataset was better than in sequential settings. We managed to reduce membership inference attack accuracy while preserving the utility of the model on the close to the original results level. Figure 4.28 shows the results for the Purchase-50 dataset with ten and 15 nodes in the setting. The setting with only 30% noise is ineffective, but adding 50% of noise to the training data for both ten and 15 nodes allows to reduce the membership inference accuracy by 8-10%, while it preserves high accuracy of the target model (refer to Figures 4.28a, 4.28c).

For the Purchase-100 dataset, we considered settings with three and ten nodes. On three nodes, the larger noise performed better (refer to Figure 4.27b). With ten nodes, noise 1 caused a 10% loss in the target model accuracy. The lower noise reduced the attack accuracy from 0.77 to 0.68 and preserved the target model effectiveness.

Location dataset The influence of the noise in parallel federated learning on the Location dataset is shown in Figure 4.29. The original membership inference had a higher performance at the beginning of the training. In the setting with two nodes (see Figures 4.29a, 4.29b) we decreased the accuracy of membership inference in the beginning of the training by 20% on the second federated cycle with noise 1 and noise 2. However, the effectiveness of the global model decreased less with noise 2, only by 4%.



(a) Purchase-50, ten nodes, noise 1

(b) Purchase-50, ten nodes, noise 2

(c) Purchase-50, 15 nodes, noise 1

(d) Purchase-50, 15 nodes, noise 2

Figure 4.28: Membership inference accuracy and accuracy of the target model evaluation after adding noise to the training data at each node. Purchase-50 in parallel federated learning.

Membership inference from a larger number of nodes in the setting (from three to six), reaches the accuracy score of around 0.9 (with six nodes in the setting the attack accuracy reaches 0.94). By adding the noise to the training data at each node, the accuracy of the attack dropped by 25% in second federated cycle with three nodes (see Figures 4.29c, 4.29d). With six nodes the membership inference accuracy was 40% lower (see Figures 4.29i, 4.29j) in the second cycle. Moreover, for all settings with the number of nodes higher than 6 we managed to preserve the accuracy of the model on the same level as in the models trained on the data without any noise. In general, both noise strategies were quite successful in the first cycles of federated learning. However, after the fifth federated cycle, we came back to the 80% membership inference accuracy. So the strategy of constantly adding the noise at each federated cycle should be considered in the future work, as well as its effect on the global model utility.

4.6 Summary

Federated learning allows to train effective machine learning models on distributed data. However models, which are shared in federated learning may leak information about their training data. By performing membership inference attacks in various federated learning settings, we showed that it is possible to infer the membership of records with higher accuracy than in centralized learning. Moreover, an adversary can infer from which particular node the record was.

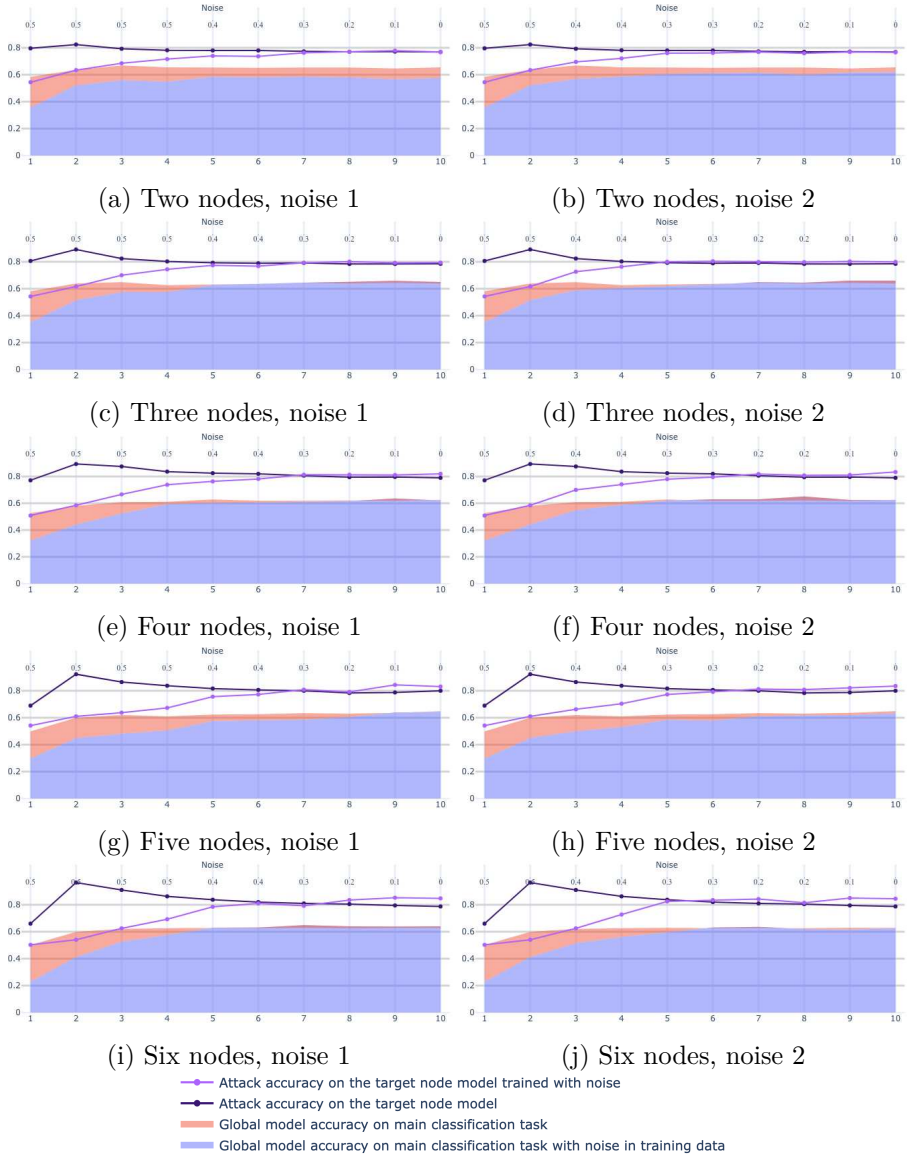


Figure 4.29: Membership inference accuracy and accuracy of the target model after adding noise to the training data at each node. Location dataset in parallel federated learning.

Evaluating the attack performance on the Purchase and Location datasets, we found that training a lower number of epochs per node generally reduces the risk of membership inference. However, when a node has a few instances, even with low number of epochs, membership inference accuracy is high.

Parallel federated learning was more efficient than sequential federated learning, especially with larger number of nodes in the setting. However, in parallel federated learning the

membership inference attack had higher accuracy in comparable settings. However, in parallel learning the lower number of epochs at each node the membership inference was not effective, while the accuracy of the global model was the highest.

Adding noise to the training data as a mitigation strategy against membership inference does not always perform well. In some cases, the target model loses too much in effectiveness with the level of noise required to noticeably reduce the membership inference attack success. However, for parallel federated learning, we found the noise strategy which allows to reduce attack accuracy while preserving the target model accuracy at the same level as models trained on the data without any noise.

Key lessons learned:

- The larger the size of the training set of the model, the worse the membership inference attack accuracy.
- The higher number of epochs at each node leads to worse effectiveness of a target model and better accuracy of the membership inference attack. Therefore a lower number of epochs is preferably to use. However, this may cause higher communication costs.
- Sequential federated learning is the most vulnerable to membership inference from a particular node right after training the shared model at a target node.
- In parallel federated learning, membership inference from the locally trained models has higher accuracy on the model trained in a target node, than in the other nodes in the setting
- The efficiency of models trained in parallel federated learning setting is higher than in sequential learning. The larger number of nodes in the setting leads to a larger difference in training time between sequential and parallel federated learning.
- Noise addition can be a mitigation strategy to decrease the membership inference attack accuracy while preserving the target model effectiveness on the level of models trained without noise.

CHAPTER 5

Conclusion

5.1 Results

In this work, we considered federated learning approaches in the tasks of privacy-sensitive machine learning. Despite the approach being relatively new, several applications in different industries have emerged. The development of mobile edge computing provokes even more growth and improvement of federated learning.

However, communication costs, data Non-IIDness (non independent and identically distributed), systems heterogeneity, privacy and security risks are core problems of the method. New algorithms and methods to improve federated learning performance are developing.

In this work, we analysed the privacy aspects of federated learning. Federated learning allows to avoid direct data sharing and allows to just transfer models to other participants of the process. However, there is still a risk of information leakage through model parameters. With several types of attacks, e.g. membership inference, an intruder can infer sensitive information from the machine learning model. We analyzed membership inference attack performance in federated learning in different scenarios, e.g. with sequential and parallel learning, with various numbers of nodes in the setting and different numbers of epochs training per node. Moreover, we tested a mitigation strategy of adding noise to the input before training to decrease privacy risks in federated learning setting. We analysed the trade-off between federated learning utility and privacy preserving properties.

Addressing the research questions from Chapter 1, our main contributions are as follows:

- We showed that federated learning allows for achieving the same effectiveness as the models trained on centralized data. In parallel learning, the efficiency of the model is higher than in centralized learning and sequential federated leaning. The parallel

setting allows to train models simultaneously in parallel, therefore the training time of the final model is shorter. However, we did not consider costs for communicating models.

- We showed that federated learning is vulnerable to membership inference attacks, as machine learning models, which are shared in the process of federated learning, leak information about their training data.
- We performed membership inference attacks in parallel and sequential federated learning settings and showed that it reaches a higher accuracy than in centralized learning. Moreover, in some scenarios, an intruder can infer to which particular node the instance belongs.
- We showed that overtraining the model leads to higher membership inference performance. Particularly, with more epochs training at each node, we reach higher membership inference attack accuracy.
- We found that a membership inference attack performs better on the models which were trained on a smaller dataset. Therefore the nodes in a federated learning setting, which have a small amount of training data, are more vulnerable to membership inference.
- We tested the mitigation strategy of adding noise to the training data. We showed that in parallel and sequential federated learning the noise can mitigate the risk of membership inference without distorting the overall performance of the global model on the main classification task.

5.2 Conclusion

- Federated learning allows training machine learning models on distribute data without high losses in the quality of the model, compared to the models trained on centralized data. Parallel learning has shown to be more effective and efficient than sequential learning. At the same time, federated learning with centralized model aggregation is easier scalable than sequential learning.
- By performing membership inference attacks we showed that federated learning does not guarantee full privacy for the clients participating in the learning process. Machine learning models, which are shared in federated learning settings can leak information about their training data.
- In the sequential federated learning scenario, we showed that there is a pronounced rise of membership inference attack performance right after training in the target node. Thereby, if an intruder has access to the model right after training in the target node, he/she can perform membership inference attack with higher accuracy. In cases when clients have small amount of training data, the models which they

train, remember the data more specifically. Therefore the attack accuracy is also relatively high in the end.

- There is also risk of data leakage in parallel federated learning. The membership inference accuracy is significantly higher at the target node, compared to other nodes' models in federated learning settings. The lower number of epochs training per node may be the solution to weaken the membership inference performance. However, fewer epochs per node in the setting may lead to an increase in the number of federated cycles and, thus higher communication costs.

- Adding noise to the training data does not guarantee resistance to membership inference, however, it works for some cases. The amount of noise, which should be added to the training data depends on the number of nodes in the setting and the size of training data. Adding noise to the data can causes a drop of utility of a global model. However in some cases one can preserve models effectiveness by gradually reducing the amount of noise in the training data.

5.3 Future work

Future work in evaluation of federated learning privacy risks includes considering other privacy attacks on machine learning models in federated learning settings, e.g. *Attribute inference* or *Model inversion*. Other machine learning models besides neural networks can be considered. Another task is to analyze federated learning performance and privacy properties on the other data types like images, audio and text data. Federated learning performance with Non-IID data was out of scope in this work, but remains one of the most actual problems in federated learning applications. Moreover, other federated learning algorithms different from federated averaging (which were proved to be more efficient) should be considered. Another field of interest are mitigation strategies, as secure multi-party computation or model parameters encryption, or other forms of noise addition to the data or exchanging models.

List of Figures

List of Tables

Bibliography

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," IBM Journal of Research and Development, vol. 3, pp. 210–229, 1959.

[2] V. Vapnik and A. Lerner, "Pattern recognition using generalized portrait method," Automation and Remote Control, vol. 24, no. 6, pp. 774–780, 1963.

[3] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, p. 5–32, 2001.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations (D. E. Rumelhart and J. L. Mcclelland, eds.), pp. 318–362, Cambridge, MA: MIT Press, 1986.

[5] R. Mendes and J. P. Vilela, "Privacy-preserving data mining: Methods, metrics, and applications," IEEE Access, vol. 5, pp. 10562–10582, 2017.

[6] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," tech. rep., Computer Science Laboratory, SRI International, 1998.

[7] L. Sweeney, "K-anonymity: A Model for Protecting Privacy," Int. J. Uncertain. Fuzziness Knowl.-Based Syst., vol. 10, no. 5, 2002.

[8] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," in ACM Transactions on Knowledge Discovery From Data - TKDD, vol. 1, p. 24, 2006.

[9] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in 2007 IEEE 23rd International Conference on Data Engineering, pp. 106–115, 2007.

[10] J. Brickell and V. Shmatikov, "The cost of privacy: Destruction of data-mining utility in anonymized data publishing," in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, (New York, NY, USA), p. 70–78, Association for Computing Machinery, 2008.

[11] C. Dwork, "Differential privacy," in 33rd International Colloquium on Automata, Languages and Programming (ICALP), (Venice, Italy), Springer, 2006.

[12] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, (CCS '16), (New York, NY, USA), p. 308–318, Association for Computing Machinery, 2016.

[13] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," in Advances in Neural Information Processing Systems 32, pp. 15479–15488, Curran Associates, Inc., 2019.

[14] S. Yu, "Big privacy: Challenges and opportunities of privacy study in the age of big data," IEEE Access, vol. 4, pp. 2751–2763, 2016.

[15] M. A. Will and R. K. Ko, "Chapter 5 - a guide to homomorphic encryption," in The Cloud Security Ecosystem (R. Ko and K.-K. R. Choo, eds.), pp. 101 – 127, Boston: Syngress, 2015.

[16] A. Ben-Efraim, Y. Lindell, and E. Omri, "Optimizing semi-honest secure multiparty computation for the internet," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16, (New York, NY, USA), p. 578–590, Association for Computing Machinery, 2016.

[17] 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, (AISTATS 2017), 20-22 April 2017, Fort Lauderdale, FL, USA, pp. 1273–1282, 2017.

[18] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," Brainlesion : glioma, multiple sclerosis, stroke and traumatic brain injuries : The second International Workshop BrainLes 2016, with the challenges on BRATS, ISLES and mTOP 2016, held in conjunction with MICCAI 2016, Athens, Greece, vol. 11383, pp. 92–104, 2018.

[19] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated Learning for Mobile Keyboard Prediction," arXiv e-prints, p. arXiv:1811.03604, 2018.

[20] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," 2019 International Joint Conference on Neural Networks (IJCNN 2019), pp. 1–8, 2018.

[21] V. Popov, M. Kudinov, I. Piontkovskaya, P. Vytovtov, and A. Nevidomsky, "Distributed fine-tuning of language models on private data," in International Conference on Learning Representations, 2018.

[22] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," IEEE Network, vol. 33, pp. 156–165, Sep. 2019.

[23] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," 2019 IEEE International Conference on Communications (ICC 2019), pp. 1–7, 2018.

[24] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," IEEE Internet of Things Journal, pp. 1–1, 2019.

[25] T. M. Deist, A. Jochems, J. van Soest, G. Nalbantov, C. Oberije, S. Walsh, M. Eble, P. Bulens, P. Coucke, W. Dries, A. Dekker, and P. Lambin, "Infrastructure and distributed learning methodology for privacy-preserving multi-centric rapid learning health care: eurocat," Clinical and Translational Radiation Oncology, vol. 4, pp. 24 – 31, 2017.

[26] A. Jochems, T. M. Deist, J. van Soest, M. Eble, P. Bulens, P. Coucke, W. Dries, P. Lambin, and A. Dekker, "Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital – a real life proof of concept," Radiotherapy and Oncology, vol. 121, no. 3, pp. 459 – 467, 2016.

[27] S. Silva, B. Gutman, E. Romero, P. M. Thompson, A. Altmann, M. Lorenzi, and U. K. Adni, "Federated learning in Distributed Medical Databases: Meta-Analysis of Large-Scale Subcortical Brain Data (Supplementary Material)," research report, Inria & Université Cote d'Azur, CNRS, I3S, Sophia Antipolis, France, 2018.

[28] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," arXiv e-prints, p. arXiv:1902.04885, 2019.

[29] J. Konečný, H. Brendan McMahan, F. X. Yu, P. Richtárik, A. Theertha Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," arXiv e-prints, p. arXiv:1610.05492, 2016.

[30] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," IEEE Transactions on Neural Networks and Learning Systems, pp. 1–14, 2019.

[31] X. Li, K. xuan Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," ArXiv, vol. abs/1907.02189, 2019.

[32] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," IEEE Access, vol. 7, pp. 47230–47244, 2019.

[33] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "{DBA}: Distributed backdoor attacks against federated learning," in International Conference on Learning Representations, 2020.

[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna Estrach, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in 2nd International Conference on Learning Representations, (ICLR 2014), 2014.

[35] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, (ASIA CCS '17), (New York, NY, USA), p. 506–519, Association for Computing Machinery, 2017.

[36] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in Proceedings of the 25th USENIX Conference on Security Symposium, (SEC'16), (USA), p. 601–618, USENIX Association, 2016.

[37] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in IEEE Symposium on Security and Privacy (SP 2017), pp. 3–18, May 2017.

[38] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15), (Denver, Colorado, USA), ACM, 2015.

[39] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft, "Blurme: Inferring and obfuscating user gender based on ratings," in Proceedings of the Sixth ACM Conference on Recommender Systems, (RecSys '12), (New York, NY, USA), pp. 195–202, ACM, 2012.

[40] T. Baumhauer, P. Schöttle, and M. Zeppelzauer, "Machine unlearning: Linear filtration for logit-based classifiers," ArXiv, vol. abs/2002.02730, 2020.

[41] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, (CCS '17), (New York, NY, USA), pp. 587–601, ACM, 2017.

[42] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," Int. J. Secur. Netw., vol. 10, no. 3, pp. 137–150, 2015.

[43] S. Truex, L. Liu, M. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," IEEE Transactions on Services Computing, vol. PP, pp. 1–1, 2019.

[44] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in 26th Annual Network and Distributed System Security Symposium (NDSS), (San Diego, CA, USA), The Internet Society, 2019.

[45] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in IEEE Symposium on Security and Privacy, San Francisco, CA, USA, pp. 691–706, 2019.

[46] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in IEEE Symposium on Security and Privacy (SP), pp. 739–753, 2019.

[47] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, and R. Zhang, "A Hybrid Approach to Privacy-Preserving Federated Learning," arXiv e-prints, p. arXiv:1812.03224, 2018.

[48] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, (CCS '17), (New York, NY, USA), pp. 1175–1191, ACM, 2017.

[49] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, (CCS '15), (New York, NY, USA), p. 1310–1321, Association for Computing Machinery, 2015.

[50] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," Trans. Info. For. Sec., vol. 13, p. 1333–1345, May 2018.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in 3rd International Conference for Learning Representations, San Diego, 2015.