COMP4801 Final Year Project

Quantitative Performance and Security Evaluation of Federated Learning on open-sourced platforms (Industry-based Project)

Group member: Fong Cheuk Ting (3035688784)

Supervisor: Prof. Yiu, Siu Ming

Abstract

The advancement of AI applications, together with the increasing concerns about data privacy, has motivated the development of a privacy-preserving AI technique, namely, Federated Learning (FL). However, an authoritative validation system that certifies qualified software is necessary to incentivize people to embrace FL technologies. This type of research is lacking in the current FL research field. This industry-based project aims to evaluate the quantitative performance and security of the open-sourced FL platform, FederatedScope. To achieve this goal, a set of evaluation metrics is proposed. The set of conducted experiments shows that FederatedScope can optimize the quantitative performance and safeguard the data security to a reasonable extent. Their optimized aggregation algorithms can narrow down the accuracy gap of 10% between the FL approach and the centralized ML approach by 1-3% for the FEMNIST dataset, while their asynchronous training mode can improve the training speed by 2-3 times for the CIFAR-10 dataset, mitigating the inherent issue of non-i.i.d. data and structural heterogeneity of FL. Furthermore, their NbAFL algorithm shows the capability to completely block DLG attacks with negligible accuracy loss. Future research may center on a deeper investigation of the features of FederatedScope, conducting a more rigorous set of experiments, or evaluating more open-sourced FL platforms to enhance the robustness, accuracy, and representativeness of the evaluation.
Acknowledgement

I would like to express my deep gratitude to Prof. Yiu and Dr. Zhang. First I would like to thank Prof. Yiu for helping me contact Dr. Zhang, the responsible person for the proposed Industry-based Project, and arrange a meeting for the project discussion. I would also like to thank Dr. Zhang for generously providing reference materials for facilitating my research process. During that period, they both gave me a lot of useful advice. Without their assistance, I would be struggling to complete the research solely by myself.
# Table of Content

List of Figures .................................................................................................................. 5  
List of Tables .................................................................................................................... 6  
Abbreviations ................................................................................................................... 7  
1 Introduction .................................................................................................................. 8     
1.1 Background and Outline ......................................................................................... 8  
1.2 Project motivation .................................................................................................... 8  
1.3 Project deliverables .................................................................................................. 10  
2 Literature Review ....................................................................................................... 13  
2.1 High-level idea of Federated Learning .................................................................... 13  
2.2 Aggregation Algorithms ......................................................................................... 14  
2.3 Challenges of Federated Learning .......................................................................... 16  
2.4 Asynchronous Training Strategy ............................................................................ 18  
2.5 Data Leakage Attack ............................................................................................... 20  
3 Methodology ............................................................................................................... 22  
3.1 Choice of Datasets ................................................................................................. 22  
3.2 Evaluation Metric for the Quantitative Performance ........................................... 23  
3.3 Evaluation Metric for Security evaluation ................................................................ 25  
4 Findings ....................................................................................................................... 26  
4.1 Experiment Environment ....................................................................................... 26  
4.2 Experimental Setup on FederatedScope .................................................................. 26  
4.3 Performance Discrepancy ....................................................................................... 28  
4.4 Benchmarking different learning tasks ................................................................. 30  
4.5 Benchmarking aggregation algorithms .................................................................... 32

3
4.6 Asynchronous Training Strategy ................................................................. 35
4.7 Data Leakage Attack and Differential Privacy ............................................. 37

5 Conclusion ........................................................................................................... 41
  5.1 Summary of the project .................................................................................. 41
  5.2 Limitations of the research ............................................................................ 41
  5.3 Future works .................................................................................................. 42

References ............................................................................................................. 44

Appendix B – Quantitative performance requirements of different AI applications [7] .... 48
Appendix C – Pseudo Code for the NbAFL DP algorithm [22] .............................. 50
Appendix D – Pseudo Code for the GAN attack algorithm [23] ............................ 51
List of Figures

Figure 1 Overview of the architecture of ASTRI’s FL validation system .........................9
Figure 2 Overview of the modules of FederatedScope [24] .............................................11
Figure 3 Event-driven architecture of FederatedScope [25] ...........................................11
Figure 4 Pseudo code for the FedAvg algorithm [14] .................................................14
Figure 5 Pseudo code for the FedOptg algorithm [15] ...................................................15
Figure 6 Data distribution of the real-world Twitter dataset ...........................................17
Figure 7 Side-by-side comparison of an i.i.d and non-i.i.d distribution sampled from one client of the MNIST dataset ..................................................................................17
Figure 8 Illustration of the difference between asynchronous and synchronous training strategy [17] ..................................................................................................................18
Figure 9 Pseudo code of CO-OP for asynchronous training strategy [14] ......................19
Figure 10 Overview of the DLG algorithm [18] ..............................................................20
Figure 11 Example of data leakage attack with different datasets [3] ............................21
Figure 12 FL model accuracy gap [7]. The narrower the gap between the blue line and orange line, the better the FL model performance ........................................................................24
Figure 13 Performance discrepancies of FEMNIST dataset on FederatedScope ..............28
Figure 14 Performance discrepancies of CIFAR10 dataset on FederatedScope ..........28
Figure 15 The architecture of asynchronous training strategy for FederatedScope [10] ....35
Figure 16 Synchronous vs Asynchronous Training Strategy on CIFAR10 dataset .........36
Figure 17 Demonstration of the attack and defence of Data Leakage attack on FederatedScope with FEMNIST dataset ..................................................................................40
List of Tables

Table 1 Validation accuracy of different aggregation algorithm against different dataset......15

Table 2 Aggregation algorithm comparison. Showing if algorithm in a row is better (+), worse (−), or comparable (=) to the algorithm in a column. [14]........................................19

Table 3 Quantitative performance of three FL learning tasks on FederatedScope..................30

Table 4 Quantitative Performance of four different aggregation algorithms on FederatedScope..........................................................32

Table 5 Benchmark of the accuracy of different FL algorithm. The best accuracy of each AI learning task is highlighted in red..........................................................34

Table 6 Experiment with CIFAR10 dataset on FederatedScope: Synchronous training vs Asynchronous training strategy ..........................................................36

Table 7 DP against DLG with FEMNIST dataset ..............................................................38

Table 8 Relationship between hyperparameters of Differential Privacy and test accuracy of FEMNIST task on FederatedScope ..........................................................39
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FL</td>
<td>Federated Learning</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>DLG</td>
<td>Data Leakage from Gradients</td>
</tr>
<tr>
<td>B2B</td>
<td>Business-to-Business</td>
</tr>
<tr>
<td>B2C</td>
<td>Business-to-Consumer</td>
</tr>
<tr>
<td>non-i.i.d.</td>
<td>Non-Identically Independent Distribution</td>
</tr>
<tr>
<td>DP</td>
<td>Differential Privacy</td>
</tr>
<tr>
<td>GAN</td>
<td>Generative Adversarial Network</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Background and Outline

With AI being the new electricity of the 21st century, AI application has become ubiquitous in our daily life. However, AI application, by nature, requires collection of users’ data in order to train the learning model on a centralized server. Some data are very sensitive, such as personal identifying information, medical record, financial transaction data, and even the words you type on the keyboard. This raises the ethical and legal concern about data privacy.

Federated Learning (FL), firstly proposed by Google [1] in 2016 for their Gboard text suggestion application, is recently becoming a promising AI technique due to tackle the problem. By definition, FL is encrypted distributed machine learning (ML) where multiple independently operating edge devices collaborate to solve a problem by sharing their locally collected data while keeping it securely stored on the device without sending it to a centralized server.

By adopting the FL technique for AI applications, the contradiction between data privacy and data sharing can be resolved. For the case of B2C (business-to-consumer), FL is useful when clients' data is very sensitive or under legal protection, such as in financial and medical applications. For the case of B2B (business-to-business), FL allows industrial entities to bridge their data silo without disclosing their own datasets such that more training data are available to improve the training performance. This is useful when valid data is scarce or mining and labelling new data are costly. It is attractive to startup companies that don't have enough historical data for their AI applications.

1.2 Project motivation

Regarding the FL technologies, industries are cautious about them because it is difficult for them to trust the “black box” as security vulnerabilities are a real concern for FL [2]. Typical security vulnerabilities include data leakage attacks. Data owners and model users may worry about such security risks. Moreover, FL suffers from high communication overhead and low convergence speed compared to the traditional ML approach. Currently, there is a lack of technical references and a standard prototype to convince industry users to embrace FL.
Therefore, a trustworthy official assessment and endorsement system is significant for FL implementation. Although there are already two governmental FL accreditation bodies (CAICT and BCTC) available, their influence is limited to Mainland China. And their process of evaluation is done in a labor-intensive manner and does not provide a reference API to solve the problem of platform silos.

To tackle this problem, ASTRI (Hong Kong Applied Science and Technology Research Institute) proposed a Reference Validation System for Federated Learning. The goal of their project is to design an authoritative, automatic, and efficient FL evaluation platform that validates commercial FL software via returning evaluation reports and issuing certificates based on the degree of computation efficiency and security the software achieved. The evaluation system (Fig. 1) consists of three different levels (Reference Technical Questionnaire, Automated Remote Evaluation System, and Inbound/Outbound Monitoring Sandbox) of validation processes. Each level consists of multiple software modules to qualitatively and quantitatively evaluate different aspects of the FL platform's performance.


Figure 1 Overview of the architecture of ASTRI's FL validation system
1.3 Project deliverables

My project is an industry-based project collaborated with ASTRI which aims to solve real-world problem. Since their project is relatively big, the scope of this project will only focus on some subtasks of their original project. My assigned job, echoing the project title, is “Quantitative Performance and Security Evaluation of open-sourced FL platforms”. The goal of the project is to write an evaluation report on one chosen open-sourced FL platform (e.g. FATE, Federated Scope, and Tensorflow Federated). Based on the same benchmark datasets, the reports should give a set of basic investigation results on the model performance and security evaluation. The investigated items will include, but not limited to the model performance, computation efficiency, and basic security evaluation showcasing one type of security attack and the related defences on that FL platform.

The evaluation report will serve as a prototype for demonstrating the sample outputs of the FL validation system as many FL software will be built using APIs of open-sourced FL platform. The evaluation will include the identification of the strength and weakness of the FL platform. This can help people to determine whether the platform is suitable for their use case and how to optimize it for better performance and security. This kind of research is significant because currently there is a lack of quantitative research on benchmarking open-sourced FL platform.

**FederatedScope, developed by the Alibaba Group, is the open-sourced FL platform this project will evaluate.** This decision was made based on several reasons which sets it apart from other open-sourced FL platforms. Firstly, the platform provides versatile features which include a vast range of built-in datasets, algorithms and ML models. It allows asynchronous training strategy and is installed with methods to protect against privacy attacks. Next, it has a event-driven architecture (Fig. 3). With such infrastructure, it is easy for FederatedScope to support different ML backends (e.g. PyTorch and TensorFlow) [10], making it a very ambitious platform. Most importantly, there is a notable lack of research about the quantitative performance and security of FederatedScope which makes it particularly valuable to evaluate FederatedScope for the purposes of this project.
This project is challenging for several reasons. Firstly, FL is a relatively new research area and inherently complex due to its distributed nature. It is also a very broad topic and many problems to investigate. It is a vast research topic encompassing a multitude of problems, such as data security, communication efficiency, optimization of aggregation algorithms, etc. In
order to provide accurate and reliable results and interpretations, it is crucial to design
rigorous experiments that account for all relevant variables and potential confounding factors.
This requires a deep understanding of Federated Learning and a thorough knowledge of the
platform FederatedScope.

For the remaining sections of this report, Section 2 will be a Literature Review Section where
existing research about FL will be discussed. Section 3 is the Methodology Section. It will
discuss the choice of dataset and the evaluation metric to evaluate the quantitative
performance and security of FederatedScope. Subsequently, Section 4 will be a list of
experiments conduct to evaluate FederatedScope. Each includes the experimental design and
hyperparameter settings. Consequently, the outcomes shall be interpreted, and their
corresponding implications highlighted. Finally, we shall conclude the report with a summary
of the project, including any limitation of the research, as well as any proposals for future
research in Section 5.
2 Literature Review

The Literature Review section of this project will provide an overview of the existing research on FL. It will introduce the high-level workflow of FL, the algorithms that aggregate clients' models, the weaknesses and challenges of FL, the asynchronous training strategy, and the two main types of attacks that FL is vulnerable to.

2.1 High-level idea of Federated Learning

![Diagram of 4 steps cycle of Federated Learning](image)

The workflow of FL can be broken down into a 4 step cycle as illustrated in Fig. 2:

1. **Broadcast model**: The central server broadcasts the global model to a subset of the clients.
2. **Local training**: Clients train the received global model with their own private data.
3. **Return updates**: Clients upload the gradient updates to the server.
4. **Federated Aggregation**: The server aggregates the collected local updates using an aggregation algorithm. The aggregated gradient is then used to update the global model. After that, we are back to step 1 to continue next round of FL. The communication round repeats to improve the global model iteratively.
2.2 Aggregation Algorithm

Aggregation algorithm controls how the server use the local updates to update the global model. We will first introduce FedAvg, one of the most fundamental algorithms (Fig. 2). In line 7, FedAvg updates the global model using the weighted sum of the local updates, where $n_k$ is dependent on the number of local training samples.

**Algorithm 1: Federated Averaging**

1. initialize $w_0$
2. for each round $t = 0, 1, \ldots$ do
3. \[ m \leftarrow \max\{\lfloor C \cdot K \rfloor, 1\} \]
4. \[ S_t = \text{random set of } m \text{ clients} \]
5. for each client $k \in S_t$ in parallel do
6. \[ w^k_{t+1} = \text{ClientUpdate}(k, w_t) \]
7. \[ w_{t+1} = \frac{\sum_{k \in S_t} \frac{n_k}{n^\sigma} w^k_{t+1}}{\sum_{k \in S_t} n_k} \]

*Figure 4 Pseudo code for the FedAvg algorithm [14]*

Instead of simply computing the weighted sum of the local updates, FedOpt uses adaptive optimizer with learning rate $\eta$ (Fig. 3, line 15). There are multiple version of federated adaptive optimizer (Fig. 3, line 12-14). Generally it can achieve faster convergence and higher accuracy than FedAvg. Experiment conducted by Reddi et al. [15] shows that FedOpt can outperform FedAvg in term of validation accuracy (Table 1).
There exist many other aggregation algorithms, such as FedProx, FedBN, Ditto, etc. Although the detailed implementation, as well as the pros and cons of every single algorithm will not be covered in this paper, it is essential to understand that the choice of aggregation algorithms will have a significant impact on the model’s quantitative performance, including the convergence rate, accuracy, and the communication efficiency.
2.3 Challenges of Federated Learning

Intuitively, FL has a worse quantitative performance compared to traditional ML as its architecture and implementation are more complex. It has a lower convergence rate, lower accuracy, and a slower training speed. There are several factors that make it challenging for FL to have achieve a comparable performance with the traditional ML approach:

Communication overhead:
FL involves the download/upload of model's weights between server and clients while ML does not require transfer of data. This will slow down the process of training. Communication overhead can be reduced by lower down the client sampling rate. Aggregation algorithms that compress the local gradient can also reduce the amount of data transfer.

Non-Identically Independent Distribution (i.i.d.) data:
In the real-world FL setting, data are distributed among perhaps thousands of clients [1]. Each edge node has different quantities of data (see example in Fig. 4), which causes discrepancy in the training speed of the local model. This adds more communication overhead to the system. Also, data on different device may have very different statistical properties (see example in Fig. 5). This leads to biased local models and pose challenge to model aggregation, making it challenging for the global model to generalize well on all clients and hence affect the accuracy. Methods to deal with this issue relies on optimization of aggregation algorithm. FedProx, for example, is a robust aggregation algorithm designed to specifically mitigate this issue [16].

Structural heterogeneity:
There are two aspects when it comes to structural heterogeneity [1]. The first one is about the computational power. Edge devices have different hardware resources. Light-weighted devices, such as smartphones or AIoT gadgets, have less hardware resources and require more time to do local training compared to personal computers. Network environment is another parameter. Edge nodes have different upload/download speed. Some nodes may even have unstable network condition, which causes clients to drop out during the training. These also add communication overhead to the system.
Figure 6: Data distribution of the real-world Twitter dataset

Figure 7: Side-by-side comparison of an i.i.d and non-i.i.d distribution sampled from one client of the MNIST dataset
2.4 Asynchronous Training Strategy

To overcome the weakness of structural heterogeneity, asynchronous training strategy was proposed and has been proved successful in Distributed ML applications [10]. For synchronous training strategy, each round, the server need to wait for all the sampled clients to return their local updates before doing the aggregation. While for asynchronous training strategy, each client can update its local gradient at its own pace without waiting for other devices (Fig. 6).

Due to the asynchronous property, clients and server typically use stochastic gradient descent (SGD) to update their models. A. Nilsson et al. [14] propose the CO-OP aggregation algorithm (Fig. 7) for the asynchronous approach. $a$ and $a_k$ represent the age of the global model and the client’s model respectively. $b_l$ and $b_u$ are the lower and upper bound of the age difference respectively. If $a - a_k > b_u$, the local model is outdated and the client should fetch the newest model from the server. If $a - a_k > b_l$, the local should train for a few more round.
before returning the update to the server. Otherwise, the local update is sent to the server and the server updates the global model using a weight w.r.t the age difference.

Although asynchronous training strategy has obvious advantage in training speed, experimental result shows that its training accuracy is inferior to synchronous training strategy with aggregation algorithms FedAvg [14]. Their paper also proofs that it is theoretically possible for CO-OP to encounter a deadlock situation.

```
Algorithm 3: CO-OP
1. $w = w_1, \ldots, w_k = w_0$
2. $a_j = 0$
3. while true do
4.   $w_k \leftarrow$ ClientUpdate($w_k$)
5.   if $a_k > y_k$ then
6.     if Client is outdated
7.       Fetch $w, a$ from the server
8.     else if $a_k < y_k$ then
9.       continue // Client is overactive
10.   else
11.     $w_k, a_k \leftarrow$ UpdateServer($w_k, a_k$) =
12.     \[
13.     \begin{align*}
14.     w &\leftarrow \frac{1}{y_k} \cdot w + \frac{a_k}{y_k} \cdot w_k, \quad a \leftarrow (a - a_k + 1)^{-1} \\
15.     \end{align*}
16.     a \leftarrow 0
17.     return $w, a$
18.   end if
19. end while
```

Figure 9 Pseudo code of CO-OP for asynchronous training strategy [14]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FedAvg</td>
<td>×</td>
<td>+</td>
<td>+</td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>CO-OP</td>
<td>×</td>
<td>×</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>FSVRG</td>
<td>×</td>
<td>×</td>
<td>=</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Aggregation algorithm comparision. Showing if algorithm in a row is better (+), worse (−), or comparable (=) to the algorithm in a column. [14]
2.5 Data Leakage Attack

Data leakage attack is the type of attack that can potentially leak client’s private data to the malicious server. Even in FL settings, gradients are still shared among all participants. Such information can reveal some properties of the training data, or more aggressively, completely recover the client’s ground truth data. Deep Leakage from Gradients (DLG), proposed by Zhu et al. [18], is the first algorithm able to achieve data leakage attack. The high-level idea of DLG can be shown in the following steps:

1. Randomly initialize a pair of dummy data and label.
2. Feed the dummy training data to the model to derive a dummy gradient.
3. Instead of updating the gradient, update the dummy data to minimize the distance between the dummy gradient and the real gradient.
4. Repeats step 2-3 to progressively reduce the distance between the dummy gradient and the real gradient. Once the dummy gradient matches the real gradient, the dummy data and label will also match the real data and label.

Note that Malicious client may also perform data leakage attack to recover other clients’ private data as the global model is broadcasted among all clients.

![Figure 10 Overview of the DLG algorithm [18]](image-url)
The experiment conducted by Wei et al. [3, Figure 9] demonstrates a successful data leakage attack targeting a different image dataset. In the initial round of FL training, the recovered images are mostly noise as the model's gradient has not yet well-fitted the training data. However, after several rounds, the recovered images quickly converge to the ground truth image (right-most image) with tolerable noise, which is negligible to the naked eye.

Since data leakage attack essentially defeats the very purpose of FL, devising strategies to defend against data leakage attack has consistently remained a major research focus within the field of FL. Conventional methods to defend against data leakage attack includes differential privacy, homomorphic encryption, Locality-sensitive hashing (LSH), and secure aggregation [1]. Differential privacy adds noise to blur client’s local gradient, but it will decrease model accuracy. Homomorphic encryption allows computation of encrypted data, but introduces new communication overhead for transmitting private key. LSH is another encryption mode that uses hashing to encrypt features. Similar inputs will generate similar outputs such that it can be used for FL. It doesn’t cause accuracy decrease and communication overhead. Secure aggregation is a protocol that makes use of a cryptographic techniques so that server can only decrypt the average update if 100s or 1000s of users have participated [8].
3 Methodology

The Methodology section will document the approach employed for evaluation of the quantitative performance and the security of FederatedScope. It includes the choice of datasets and the evaluation metric, and hyperparameter settings of FederatedScope. It will then be followed by a comprehensive list of experiments to be conducted and how they will be set up. For each experiment, we will outline the specific question we would like to investigate, the hypothesis, and the limitations of the experiment.

3.1 Choice of Datasets

Selecting the appropriate datasets is the first things to consider when conducting research to evaluating FL platform. Related researches [3, 4, 5, 6] commonly use CIFAR-10 to evaluate FL models for image classification tasks. It contains 60,000 images with 10 classes. Dirichlet distribution can be used to divide the images amongst clients to simulate non-i.i.d. data [5], which is a common situation for real-world FL applications [1]. On the other hand, LEAF is a famous benchmarking framework for learning in FL settings, which includes FEMNST, Shakespeare, Twitter, Celeba, Synthetic Dataset, and Reddit [10]. Samples in these datasets are already distributed among massive number of clients in a non-i.i.d. manner. The statistical properties of the LEAF datasets can be found in Appendix A. One advantage of choosing the above datasets is that it enables comparisons of the analysis results with other research papers.

However, note that the evaluation will only use the above-mentioned dataset, the evaluation result may not be able to accurately represent the capabilities of the FL platform handling other real-world datasets. Also, the chosen datasets may not be large enough to fully evaluate the scalability of the FL platform. The result of these limitations is that the evaluation may not generalize well on real-world practical situations.
3.2 Evaluation Metric for the Quantitative Performance

To evaluate the quantitative performance of a FL platform, it is recommended to follow the “IEEE Guide for Architectural Framework and Application of Federated Machine Learning” [7]:

**Performance discrepancy**

The most important evaluation metric for any machine learning task is the test accuracy. As mention in Section 2.3, one should expect the FL approach will have a worse test accuracy than the ML approach. Measuring such performance discrepancy is essential to quantitatively understand the trade-off between accuracy and adopting the FL approach. If the accuracy gap between the is reasonably small, industry users will be more convinced to adopt FL. Additionally, this will help identifying the limitations of FL and provide insights on how to optimize the aggregation algorithm.

To evaluate the performance discrepancy, we will benchmark the same learning task under three different cases:

1. **FL**: the default case where training is done in the FL manner.
2. **Single node**: the case that simulate the situation there is only one client training a single model with its private local data.
3. **Centralized**: data from all clients are centralized to train on one central server, simulating the traditional ML approach.

The discrepancy will be visualized via plotting a graph of accuracy against epoch as illustrated in the example below (Fig. 10).
Computation efficiency

This includes training time, testing time, intrinsic memory usage, auxiliary memory usage, and the network requirements (upload/download bytes). Different real-world FL applications require different degrees of computational efficiency [7]. As mentioned in Section 2.3, unlike traditional ML, FL is a distributed system. It has communication overhead that hinders the training and testing time. Communication costs can be reduced by decreasing the number of communication rounds between local nodes and the central node and decreasing the model update time of the local node [1]. FL also has the issue of structural heterogeneity. Hardware resources (e.g. CPU, memory, download and upload speed) of each client node can vary a lot [1]. For example, if it is a B2C application where the client nodes are edge devices like a smartphone, only a light-weighted local model is feasible. But if it is a B2B application, it would be more tolerable to computation efficiency. In conclusion, the feasibility of FL applications is very sensitive to computation efficiency. Evaluating the quantitative performance of different AI learning tasks on each FL platform helps answer the question of "Is it feasible to run X AI applications using Y FL platform on Z devices?". For reference, tables of the quantitative performance requirements of different AI applications are attached in Appendix B [7].

Computational efficiency of FL can be affected by two variables: the learning model and the aggregation algorithm. Different learning models (e.g. Logistic Regression, CNN, etc) have different amount of parameters. Thus, affecting the computation efficiency. Some aggregation algorithms have different approaches for gradient transfer between the server and
the clients, which affect the training time and the network requirements. Therefore, evaluation of the computational efficiency will be done by benchmarking the FL platform under different learning tasks and different aggregation algorithms.

Other than the aggregation algorithm, adopting asynchronous training strategy is another effective method to improve the computational efficiency. Benchmarking synchronous training strategy vs asynchronous training strategy is significant to evaluate platform’s capabilities to optimize the quantitative performance of FL.

3.3 Evaluation Metric for Security evaluation

FL applications, depends on the sensitivity of the client’s private data, require different degree of security (see Appendix B). It is an important category for evaluating a FL platform. To test whether the FL platforms are effective against data leakage attacks. If the learning task is image classification, it is convenient to measure the security both quantitatively and qualitatively. The extent of data leakage can be quantitatively measured using Mean Square Error (MSE) or Structural Similarity Index (SSIM), by comparing the similarities between the ground truth image and the leaked image [7]. Specifically, for the DLG attack method, its effectiveness can also be measured via the gradient difference between the real gradient and the dummy gradient. Qualitatively, one can visualize the recovered data and compare it to the ground truth data to see whether the attack exposes the client’s private data effectively.

For demonstrating the attack and defence, the project will investigate the effectiveness of Different Privacy (DP) against different methods of data leakage attack, the DLG attack and GAN attack respectively. Qualitatively and quantitatively compare the extent of data leakage with and without using DP.
4 Findings

In this section, experimental results related to the quantitative performance and security of FederatedScope will be presented. Results are followed by detailed interpretations.

4.1 Experiment Environment

For reference, all FL experiments are conducted in HKU CS GPU farm phase 2 with NIVIDA GeForce RTX 2080Ti GPU card. This also ensures fairness while benchmarking the execution time of each FL task on each FL platform. Other than the execution time, the total floating point operations (flops) will also be provided. It is independent of the computational power of the machine and correlated the execution time. This ensures absolute fairness for comparison of the hardware resource consumption of each FL task.

4.2 Experimental Setup on FederatedScope

In FederatedScope, simulation of FL can be done in standalone mode, where the platform can simulate multiple participants on a single device. There are many hyperparameters to control the experimental settings, but most of them are unique to specific type of experiment [20]. There exists general and fundamental hyperparameters that are used for all kinds of experiments:

- **total_round_num**: the total number of round of FL
- **client_num**: the number of client participated in FL
- **sample_client_rate**: the percentage of clients sampled to participate in FL at each round

Unless further specified, the value of the total_round_num, client_num, sample_client_rate will be fixed at 300, 200, 0.2 respectively for the sake of benchmarking the result under the same hyperparameter settings. Picking 300 rounds for FL training because for most FL learning tasks conducted in this project, their test accuracies will converge before 300th round. For the client_num, most LEAF datasets have more than a thousand clients. FederatedScope will only pick out 200 out of all the clients to participate in FL. Otherwise, with my limited computational resources, it would be too time consuming to complete all the targeted experiments. For the sample_client_rate, it is common to set it to a reasonably small value [19]. One advantage of doing so is to further reduce the communication overhead. It also help...
addressing the issue of non i.i.d. and structural heterogeneity, thereby improving the robustness and generalization of the trained model.

In addition, the aggregation algorithm for all the experiments will be fixed to FedAvg as a baseline under further specified.
4.3 Performance Discrepancy

The performance discrepancies of FL on FederatedScope is analysed with the FEMNIST and CIFAR10 datasets respectively.

![Performance Discrepancy with FEMNIST dataset](image)

*Figure 13 Performance discrepancies of FEMNIST dataset on FederatedScope*

![Performance Discrepancy with CIFAR10 dataset](image)

*Figure 14 Performance discrepancies of CIFAR10 dataset on FederatedScope*

As expected, the experiment with the FEMNIST dataset shows that the FL case performs worse than the centralized case, by an accuracy gap of 10% (Fig. 12). The centralized case quickly reaches an accuracy of 91%, while the FL case is only at 81% at the final round. Running the FL case for further number of rounds has been tried, but the accuracy has
already converged. This suggests that there will be a huge accuracy trade-off for applying the FL technique to handwritten character classification task in order to protect the user's privacy.

More unexpectedly, the single node case reaches the same accuracy of 81%. It even converges faster than the FL case, achieving a higher accuracy from 50th to 150th round. A plausible explanation for this outcome is that the FEMNIST dataset is non i.i.d.. For the single_node case, despite the limited amount of training data, the trained model only need to fit the data of a single client and still able to generalize well on test data. While for the FL case, some outlier clients may have training samples with very different statistical properties compared to the majority clients, hindering the accuracy of the global model. In fact, the test accuracy of the top 10% clients and the bottom 10% clients are 91.85% and 63.22% respectively (refers to Table 3 column 1, they are the same experiment). The former accuracy is in close proximity to the centralized case, indicating that if the dataset is i.i.d., the test accuracy could be very close to the centralized case. The later accuracy is a strong evidence supporting the aforementioned explanation regarding the performance discrepancy.

Meanwhile, the experiment with the CIFAR10 dataset gives out a very different result (Fig. 6). The accuracy gap between the FL case and the single node case is incredibly huge. One possible interpretation of this phenomenon is that the amount of training data per client in the CIFAR10 dataset is too little for effective model training. Although both the FEMNIST and CIFAR10 datasets on average have hundred-ish samples per client, the nature of the image classification task of CIFAR10 is more complex than the FEMNIST one, which is just handwritten character classification. Thus, it needs way more than a hundred-ish samples to achieve effective training such that the model can generalize well on test data. This perfectly simulates the situation where companies do not have sufficient labeled data for training. In this case, it is much better to adopt FL and collaborate with other companies.
4.4 Benchmarking different learning tasks

Benchmarking the quantitative performance of FederatedScope under different learning tasks is important. The result can serve as a guideline to answer the questions of "Is it feasible to run X AI applications using Y FL platform on Z devices?" (refers to Section 3.2).

To test out the quantitative performance of FederatedScope under different learning tasks.

Three FL experiments (Table 3) of different learning tasks are conducted in FederatedScope under the same hyperparameters and aggregation algorithm. The three FL learning tasks are hand-written text classification (FEMNIST), vector classification (Synthetic Dataset), and image classification (CIFAR-10) respectively.

<table>
<thead>
<tr>
<th>Task</th>
<th>Handwritten character Classification</th>
<th>Vector Classification</th>
<th>Image Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>FEMNIST</td>
<td>Synthetic Dataset</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>Model</td>
<td>ConvNet2</td>
<td>Logistic Regression</td>
<td>ConvNet2</td>
</tr>
<tr>
<td>FL Algorithm</td>
<td>FedAvg</td>
<td>FedAvg</td>
<td>FedAvg</td>
</tr>
<tr>
<td>Number of clients</td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Number of rounds</td>
<td>300</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Sampling rate</td>
<td>20%</td>
<td>20%</td>
<td>6.5%</td>
</tr>
<tr>
<td>Test accuracy</td>
<td>81.30%</td>
<td>68.15%</td>
<td>74.86%</td>
</tr>
<tr>
<td>Test accuracy (Top 10% clients)</td>
<td>91.85%</td>
<td>89.53%</td>
<td>87.06%</td>
</tr>
<tr>
<td>Execution time</td>
<td>36.26 mins</td>
<td>8.31 mins</td>
<td>119 mins</td>
</tr>
<tr>
<td>Total FLOP</td>
<td>178.86GB</td>
<td>17.99GB</td>
<td>1.13TB</td>
</tr>
<tr>
<td>Model size</td>
<td>6.27MB</td>
<td>301B</td>
<td>2.05MB</td>
</tr>
<tr>
<td>Total upload/download bytes (per client)</td>
<td>847.83KB</td>
<td>54.73KB</td>
<td>5.63MB</td>
</tr>
</tbody>
</table>

Table 3 Quantitative performance of three FL learning tasks on FederatedScope

The quantitative performance available are the test accuracies, training time, total FLOP, model size, and total upload/download bytes respectively. As expected, the quantitative
The performance of different AI learning tasks varies a lot. The difference is purely dictated by the dataset and the training model. Order by the computational efficiencies, we have Synthetic > FEMNIST > CIFAR10. In terms of the complexity of the training model, ConvNet2 (Convolutional Neural Network) is much more complex and has more model parameters than Linear Regression model. The image size in FEMNIST dataset is 28x28x1 while that of CIFAR10 dataset is 32x32x3. Therefore, it takes more computational resource to train the CIFAR10 dataset even though they use the same model.

In the next section, we will benchmark the quantitative performance of different aggregation algorithms, and we can see that an optimized aggregation algorithm can also improve the computational efficiencies.
4.5 Benchmarking aggregation algorithms

In this section, all experiments are benchmarked under the same dataset but different aggregation algorithms. Other than the FedAvg and FedOpt mentioned previously in Section 2.2, there exists many other variations of aggregation algorithms which attempts to optimize the quantitative performance of FL. The detailed implementation of every aggregation algorithm will not be discussed. The focus of the project is to investigate in what extent the optimized aggregation algorithm can narrow down the accuracy gap between FL and ML.

Four different aggregation algorithms, FedAvg, FedBN, Ditto, and FedOpt, are chosen to benchmark their performance. The detailed configurations and result is presented below (Table 4).

Table 4 Quantitative Performance of four different aggregation algorithms on FederatedScope

<table>
<thead>
<tr>
<th>Task</th>
<th>Handwritten character Classification</th>
<th>Handwritten character Classification</th>
<th>Handwritten character Classification</th>
<th>Handwritten character Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>FEMNIST</td>
<td>FEMNIST</td>
<td>FEMNIST</td>
<td>FEMNIST</td>
</tr>
<tr>
<td>Model</td>
<td>ConvNet2</td>
<td>ConvNet2</td>
<td>ConvNet2</td>
<td>ConvNet2</td>
</tr>
<tr>
<td>FL Algorithm</td>
<td>FedAvg</td>
<td>FedBN</td>
<td>Ditto</td>
<td>FedOpt</td>
</tr>
<tr>
<td>Number of clients</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Number of rounds</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Sampling rate</td>
<td>20%</td>
<td>20%</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>Test accuracy (Top 10% clients)</td>
<td>81.30%</td>
<td>82.40%</td>
<td>83.47%</td>
<td>82.91%</td>
</tr>
<tr>
<td>Test accuracy (Top 10% clients)</td>
<td>91.44%</td>
<td>94.99%</td>
<td>91.44%</td>
<td>92.43%</td>
</tr>
<tr>
<td>Test accuracy (Bottom 10% clients)</td>
<td>59.65%</td>
<td>58.62%</td>
<td>59.65%</td>
<td>56.01%</td>
</tr>
<tr>
<td>Execution time</td>
<td>36.26 mins</td>
<td>20.49 mins</td>
<td>30.15 mins</td>
<td>30.9 mins</td>
</tr>
<tr>
<td>Total FLOP</td>
<td>442.34GB</td>
<td>146.99GB</td>
<td>442.34GB</td>
<td>113.96GB</td>
</tr>
<tr>
<td>Model size</td>
<td>6.27MB</td>
<td>6.27MB</td>
<td>6.27MB</td>
<td>6.27MB</td>
</tr>
<tr>
<td>Total upload / download bytes (per client)</td>
<td>847.83KB</td>
<td>689.7KB</td>
<td>847.82KB</td>
<td>228.24K</td>
</tr>
</tbody>
</table>
Out of the four aggregation algorithms, FedOpt has the fastest computational time of 20.49 minutes, while Ditto has the best test accuracy of 83.37%. However, it is difficult to conclude that any of the aggregation algorithm is the most superior one since we are only benchmarking their performance under the same AI learning task. Other than the experiments conducted in this project, the official website of FederatedScope provides sample results of all its available aggregation algorithms against three different AI learning tasks [Table 3, 13]. However, since the hyperparameter settings of these experiments are unknown and their result does not provide any information about the computational efficiencies, their results cannot be directly compared with those conducted in this project. Their result shows that the optimal FL algorithm for each AI learning task is different, which implies that there is no single best algorithm for every AI learning task.

A further observation indicates that, relative to the accuracy gap between the FL case and the centralized case in Section 4.3 (about 10%), the accuracy difference between different aggregation algorithms is small (about 1-3%). This shows that optimization of the aggregation algorithm has a rather limited impact on improving the accuracies. Meanwhile, the difference in the computational efficiencies between different aggregation algorithms is significant (Table 4). This suggests that it is more reasonable to place greater emphasis on the computational efficiency rather than the test accuracy when choosing an aggregation algorithm to use in FL, especially when the computational efficiencies are the bottleneck of that FL software.

<table>
<thead>
<tr>
<th></th>
<th>Task</th>
<th>Data</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedAvg Logistic regression</td>
<td>Synthetic</td>
<td>68.36%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Image classification</td>
<td>FEMNIST</td>
<td>84.93%</td>
</tr>
<tr>
<td></td>
<td>Next-character Prediction</td>
<td>Shakespeare</td>
<td>43.80%</td>
</tr>
<tr>
<td>FedOpt Logistic regression</td>
<td>Synthetic</td>
<td>68.32%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Image classification</td>
<td>FEMNIST</td>
<td>84.92%</td>
</tr>
<tr>
<td></td>
<td>Next-character Prediction</td>
<td>Shakespeare</td>
<td>47.39%</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Task</td>
<td>Dataset</td>
<td>Accuracy</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
<td>---------------</td>
<td>-----------</td>
</tr>
<tr>
<td>FedBN</td>
<td>Logistic regression</td>
<td>Synthetic</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Image classification</td>
<td>FEMNIST</td>
<td>85.48%</td>
</tr>
<tr>
<td></td>
<td>Next-character Prediction</td>
<td>Shakespeare</td>
<td>N/A</td>
</tr>
<tr>
<td>FedProx</td>
<td>Logistic regression</td>
<td>Synthetic</td>
<td>68.36%</td>
</tr>
<tr>
<td></td>
<td>Image classification</td>
<td>FEMNIST</td>
<td>84.77%</td>
</tr>
<tr>
<td></td>
<td>Next-character Prediction</td>
<td>Shakespeare</td>
<td>47.85%</td>
</tr>
<tr>
<td>pFedMe</td>
<td>Logistic regression</td>
<td>Synthetic</td>
<td>68.73%</td>
</tr>
<tr>
<td></td>
<td>Image classification</td>
<td>FEMNIST</td>
<td>87.65%</td>
</tr>
<tr>
<td></td>
<td>Next-character Prediction</td>
<td>Shakespeare</td>
<td>37.40%</td>
</tr>
<tr>
<td>Ditto</td>
<td>Logistic regression</td>
<td>Synthetic</td>
<td>69.67%</td>
</tr>
<tr>
<td></td>
<td>Image classification</td>
<td>FEMNIST</td>
<td>86.61%</td>
</tr>
<tr>
<td></td>
<td>Next-character Prediction</td>
<td>Shakespeare</td>
<td>45.14%</td>
</tr>
<tr>
<td>FedEM</td>
<td>Logistic regression</td>
<td>Synthetic</td>
<td>68.80%</td>
</tr>
<tr>
<td></td>
<td>Image classification</td>
<td>FEMNIST</td>
<td>84.79%</td>
</tr>
<tr>
<td></td>
<td>Next-character Prediction</td>
<td>Shakespeare</td>
<td>48.06%</td>
</tr>
</tbody>
</table>

Table 5: Benchmark of the accuracy of different FL algorithm. The best accuracy of each AI learning task is highlighted in red.
4.6 Asynchronous Training Strategy

FederatedScope offers its own method to perform FL in asynchronous mode. The high-level idea of its asynchronous training algorithm is similar to the CO-OP algorithm in Section 2.4. Local gradient from slow clients will be discounted and outdated gradient will be discarded [10]. A demonstration of asynchronous training strategy used by FederatedScope is illustrate in (Fig. 12).

There are a few hyperparameters specifically for the asynchronous training mode:

- **min_received_num**: The minimal number of received feedback for the server to trigger federated aggregation.

- **staleness_toleration**: The threshold of the tolerable staleness in federated aggregation.

The values of min_received_num and staleness_toleration are set to 40 and 10 respectively.

![Figure 15 The architecture of asynchronous training strategy for FederatedScope [10].](Image)

To see the pros and cons of the asynchronous training strategy, experiments are conducted with the CIFAR10 dataset (Table 6). For the first time, the number of training rounds is fixed at 300. The synchronous training strategy takes about 23 minutes, while the asynchronous strategy takes only 8.56 minutes, which is over two times faster. However, the test accuracy is not as good as in the synchronous case as the server does not make use of all the client’s local updates for aggregation. However, for the second time, when we fix the training time instead of the number of training rounds, the asynchronous training strategy has better
accuracy. This shows that asynchronous training strategy converges at a higher test accuracy given sufficient amount of training time. A plausible explanation of this result is that asynchronous training strategy helps mitigate the issue of non i.i.d. data, as each client can update the global model with its local gradient as soon as it is ready, without waiting for other clients to finish their training.

![Synchronous vs Asynchronous Training Strategy on CIFAR10 dataset](image)

Figure 16 Synchronous vs Asynchronous Training Strategy on CIFAR10 dataset

<table>
<thead>
<tr>
<th></th>
<th>Synchronous Strategy</th>
<th>Asynchronous Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training time</strong></td>
<td>22.99 minutes</td>
<td>8.56 minutes</td>
</tr>
<tr>
<td><strong>Test Accuracy</strong></td>
<td>58.47%</td>
<td>51.57%</td>
</tr>
<tr>
<td><strong>Test Accuracy after training for 30 minutes</strong></td>
<td>59.39%</td>
<td>62.30%</td>
</tr>
</tbody>
</table>

Table 6 Experiment with CIFAR10 dataset on FederatedScope: Synchronous training vs Asynchronous training strategy
As we can see, the quantitative performance of different FL tasks varies a lot in terms of test accuracy, execution time, model size, and network requirements. Showcasing this information on the evaluation report would be helpful for answering the question of “Is it feasible to use X FL platform to run Y FL task on Z device?” For example, training on Synthetic Dataset has a much faster training speed, smaller model size, and lighter network requirements. We can conclude that it is feasible to run such task on light-weighted AIoT devices like a smartwatch, which has very limited hardware resources.

However, this is just a very basic investigation of quantitative performance. In fact, there are a lot of dimensions to evaluate FL platforms. Other than different combinations of dataset, model, and FL algorithm, there are vertical/horizontal FL, synchronous/asynchronous strategies, hyperparameter optimization (HPO), etc. It would be too overwhelming to test out everything. Therefore, it is challenging to identify the most important dimensions to evaluate and display the quantitative result in the most informative way in the finalized evaluation report.

Also, a graph of performance discrepancy (Fig. 2) as promised in the Methodology Section also hasn’t been plotted yet due to the limited flexibility of the config file. To overcome this issue, I will have to study the platform’s API and edit the python code for more flexibility in customization.

4.7 Data Leakage Attack and Differential Privacy

FEMNIST dataset is used to demonstrate the attack and defence of data leakage attack.

For DP, FederatedScope offers the NbAFL method to add noise to the local gradients. It has two hyperparameter $\epsilon$ and $\delta$ that controls the degree of noisiness applied to the local gradient. The smaller their values, the noisier the local gradient. Pseudo code for the NbAFL algorithm can be found in the Appendix C [22]. The result of DP against DLG attack is presented in (Table 7). For each round of FL, attacker is given 500 iterations to recover the data with a reconstructed learning rate of 0.1.

<table>
<thead>
<tr>
<th>Number of</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
</table>

37
Without DP, the attacker can successfully match the real gradient with almost zero gradient loss. The recovered characters can often be clearly identified without observable noise. The lower the values of $\epsilon$ and $\delta$, the higher the gradient loss, and the less identifiable the recovered characters. Eventually, at ($\epsilon = 10, \delta = 0.1$), the recovered characters become completely noises. An additional observation is that the gradient loss of DLG is independent of the number of round, meaning that the gradient loss has already converged at the first round.

However, DP does have its drawback of reducing test accuracy. It is an intuitive consequence since DP is adding noise to the local gradient, which will make the global model ends up more biased. For reference, there is a table available at the FederatedScope official website.
showing the relationship between $\epsilon$, $\delta$ and the test accuracy for the FEMNIST dataset [13, Table 7]. Therefore, even though the recovered image is completely noisy for ($\epsilon = 10$, $\delta = 0.1$), the global model will only have a test accuracy of 11.73%. Meanwhile, for DP of ($\epsilon = 50$, $\delta = 0.76$), there is no drop in test accuracy. Although one can still see the rough black spot at the center of the recovered image, one can no longer recognize the exact letter.

Therefore, for practical FL, the optimal approach to defend against data leakage attack is to strike for a balance. Tune the DP hyperparameters to the point where the attacker cannot effectively recover the clients’ data while the loss of accuracy is minimal.

<table>
<thead>
<tr>
<th>Task</th>
<th>$\epsilon$</th>
<th>$\delta$</th>
<th>Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEMNIST</td>
<td>10</td>
<td>0.01</td>
<td>11.73</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.17</td>
<td>24.82</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.76</td>
<td>41.71</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.01</td>
<td>54.85</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.17</td>
<td>67.98</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.76</td>
<td>80.58</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.01</td>
<td>74.80</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.17</td>
<td>80.39</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.76</td>
<td>80.58</td>
</tr>
</tbody>
</table>

Table 8 Relationship between hyperparameters of Differential Privacy and test accuracy of FEMNIST task on FederatedScope

FederatedScope provides another method for data leakage attack, call the Generative Adversarial Network (GAN) attack method. It is a type of class representative attack where the attacker is only curious about the client’s private data of one specific class. It involves an attacker who possesses a local GAN. During each round of FL training, the attacker first updates the discriminator of the local GAN with the received parameters. The attacker then locally trains the generator of the GAN to generate data that can be classified as the target class. Subsequently, the attacker labels the generated data as a class that is different from the target class and inserts this data into the training batch, which is then used to perform regular local training [23]. The pseudo code of the GAN attack algorithm can be found in Appendix D.
Experiment demonstrating the attack and defence of GAN attack is conducted. The targeted class is the character “3”. The qualitative result can be found in (Fig. 14).

![GAN Attack](image)

**Figure 17** Demonstration of the attack and defence of Data Leakage attack on FederatedScope with FEMNIST dataset

For GAN attack, at the initial round, the recovered image is mostly noise, but the recovered image improves gradually. For the case without DP (first row of Fig. 14), at the 50th round, we can easily recognize that the recovered image is a handwritten digit “3”.

For the case with DP, at the initial round, the recovered image is much noisier than the case without DP. After 50 rounds of training, the recovered image is more difficult to recognize compared to the upper one, but not to the point that no one can recognize it, despite DP is heavily applied ($\epsilon = 10$, $\delta = 0.1$).

### 3.2 Table of Current Progress Status

<table>
<thead>
<tr>
<th>Schedule</th>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>By the end of September</td>
<td>Basic research on FL technologies</td>
<td>Completed</td>
</tr>
<tr>
<td></td>
<td>Draft the Project Plan</td>
<td>Completed</td>
</tr>
<tr>
<td>By the end of interim-report submission</td>
<td>Finish two evaluation report on two open-source FL platforms (1st deliverable)</td>
<td>In Progress</td>
</tr>
<tr>
<td></td>
<td>Learn how to use FederatedScope</td>
<td>Completed</td>
</tr>
</tbody>
</table>

40
5 Conclusion

5.1 Summary of the project

The goal of the project is to evaluate the quantitative performance and security of the open-sourced FL platform, FederatedScope. The work is significant guideline to help people understand the strength and weaknesses of FederatedScope and provides directions for other FL users on how to optimize FL to achieve better quantitative performance and security.

After conducting a number of experiments on FederatedScope, it is discovered that FL has a huge accuracy gap compared to the traditional ML approach mainly due to the issue of non i.i.d. data. The global model does not generalize well on outlier clients. The outlier clients hinder the average test accuracy. Even though it is possible to use optimized aggregation algorithm to improve the accuracy. It can only slightly improve the accuracy and they are better at improving the computational efficiencies. Asynchronous training strategy on FederatedScope has been proofed effective. It speeds up the training process by 2-3 times by tackling the issue of structural heterogeneity. It can also get the test accuracy further closer to the centralized training case. For the data leakage attack, DP is fully effective against DLG attack, but has limited effect against GAN attack. Meanwhile, DP has a drawback of reducing the test accuracy of FL. One should strike for a balance between accuracy performance and data security when applying DP to defend against data leakage attack.

5.2 Limitations of the research

The scope and robustness of the evaluation is limited due to several factors.
The first one is about the limited computational resources. For all the FL experiments conducted, the scale (e.g. number of FL rounds, number of clients) is set to a limited value. Not only that large scale FL takes too long to train, the memory requirements may also exceed the capabilities of the machine, resulting in an unconditional termination of FL during the mid-point of training. Such limitation may render the evaluation result less representative of large-scale real-world FL.

Secondly, it is important to note that the objective of the project is to give a demo evaluation report of open-sourced FL platform for the reference validation system proposed by ASTRI. However, it must be acknowledged that the evaluation report focusing solely on one FL platform necessarily provide a comprehensive representation of other open-sourced FL platforms, such as Tensorflow Federated and FATE. Other platforms may possess features that are absent in FederatedScope, and vice versa. It will also become possible to do comparison between FL platforms, which helps identify the strengths and weaknesses of each platform and provides insights into the difference between them.

Thirdly, the features of FederatedScope has not been fully explored. In regard to the datasets, most of the experiment use image classification dataset to benchmark the performance. Nonetheless, it is worth highlighting that there are datasets available for other types of learning tasks, such as graph learning, natural language processing (NLP), etc. Also, vertical FL and backdoor attack are significant subarkject within the research field of FL and FederatedScope does contain the related features, but they are not within the scope of the project. The hyperparameter optimization module within the FederatedScope allows users to exhaust combinations of selected hyperparameters [26] to find out the best combination of hyperparameters for fine-tuning the performance.

5.3.1 Future works

This subsection will list out the possible works to be done in the future to enrich the content of the project. It is divided into the vertical direction and horizontal direction respectively.

In the vertical direction, there exists potential to explore more features available on FederatedScope. It includes benchmarking the quantitative performance with graph learning and NLP datasets, its capabilities to achieve good quantitative performance on vertical FL, and capabilities to defend against backdoor attack. Furthermore, if adequate computational
resources are available, a large-scale FL experiment could be conducted in order to bolster the reliability of the evaluation outcomes.

In the horizontal direction, we may evaluate the quantitative performance and security of other open-sourced FL platforms. This will make the evaluation result generalize well on all FL platform. Thus, enhancing the robustness of the evaluation result. The results will be more applicable to a wider range of real-world scenarios.

The goal of this project is to address the issue of how to motivate industries to embrace FL technologies. This is significant for the massification of privacy-preserving AI applications, which helps erase the worry about privacy leakage and connecting data silos. ASTRI proposed an authoritative and automatic FL validation system to certify qualified FL software. Deliverables include trial reports to quantitatively open-sourced FL platforms and showcasing privacy attacks and related defense via web portal demo. Such kind of research is valuable as they are currently lacking in FL research field. Finally, research on building a web portal demo will begin in semester 2, after finishing the two evaluation reports. We are currently conducting FL experiments on the open-sourced FL platform FederatedScope. It is recommended to further study the advanced customization of FederatedScope framework to allow more tailor-made experimental settings to retrieve more informative data. On the other hand, we should decide and study the second chosen FL platform as soon as possible to allow quantitative comparison between the two platforms early on. The overlapping functionalities of the two platforms shall also hint us the most important dimensions while evaluating an FL platform. Finally, research on building a web portal demo will begin in semester 2, after finishing the two evaluation reports.
References


For more detailed statistical properties, visit https://leaf.cmu.edu/


1. FEMNIST
   - **Overview:** Image Dataset
   - **Details:** 62 different classes (10 digits, 26 lowercase, 26 uppercase), images are 28 by 28 pixels (with option to make them all 128 by 128 pixels), 3500 users
   - **Task:** Image Classification

2. Sentiment140
   - **Overview:** Text Dataset of Tweets
   - **Details:** 660120 users
   - **Task:** Sentiment Analysis

3. Shakespeare
   - **Overview:** Text Dataset of Shakespeare Dialogues
   - **Details:** 1129 users (reduced to 660 with our choice of sequence length. See [bug](https://github.com/MaoJY6/LEAF-Benchmarks/issues/4).)
   - **Task:** Next-Character Prediction

4. Celeba
   - **Overview:** Image Dataset based on the [Large-scale CelebFaces Attributes Dataset](https://www.robots.ox.ac.uk/~vgg/data/celeba/index.html)
   - **Details:** 9343 users (we exclude celebrities with less than 5 images)
   - **Task:** Image Classification (Smiling vs. Not smiling)

5. Synthetic Dataset
• Overview: We propose a process to generate synthetic, challenging federated datasets. The high-level goal is to create devices whose true models are device-dependant. To see a description of the whole generative process, please refer to the paper.

• Details: The user can customize the number of devices, the number of classes and the number of dimensions, among others.

• Task: Classification

6. Reddit

• Overview: We preprocess the Reddit data released by pushshift.io corresponding to December 2017.

• Details: 1,660,820 users with a total of 56,587,343 comments.

• Task: Next-word Prediction.
## Appendix B – Quantitative performance requirements of different AI applications [7]

### Table A.1—Requirements for applications

<table>
<thead>
<tr>
<th>Use case type</th>
<th>Use cases</th>
<th>Efficiency requirements</th>
<th>Security requirements</th>
<th>Privacy requirements</th>
<th>Model performance requirements</th>
<th>Economic viability requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training time</td>
<td>Testing time</td>
<td>Intrinsic memory usage</td>
<td>Auxiliary memory usage</td>
<td></td>
</tr>
<tr>
<td>B2B</td>
<td>Finance</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Health</td>
<td>0</td>
<td>0,1*</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Marketing</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>B2G</td>
<td>Government services</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Government</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Urban Computing</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Telecom.</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Education</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Internet of Things (IoT)</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Security requirement level</td>
<td>Requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>No corresponding plans for potential attacks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Successfully defending read-write attack for a model on a central server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Successfully defending data recovery for channel monitoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Successfully defending data recovery for read-write attacks database and channel monitoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Successfully defending model controlling based on 1-3 attacks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privacy requirement level</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No defending ability</td>
</tr>
<tr>
<td>1</td>
<td>Successfully defending leakage during transferring</td>
</tr>
<tr>
<td>2</td>
<td>Successfully defending database and aggregator leakage</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model accuracy requirement level</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Achieving an equivalent or more competitive performance to that of a single-data-node model</td>
</tr>
<tr>
<td>1</td>
<td>Achieving a performance with noticeable deterioration compared to that of a data-centralized model</td>
</tr>
<tr>
<td>2</td>
<td>Achieving a performance with insignificant deterioration compared to that of a data-centralized model</td>
</tr>
<tr>
<td>3</td>
<td>Achieving an equivalent or more competitive performance to that of a data-centralized model</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time efficiency requirement level</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Supporting completion in the order of weeks</td>
</tr>
<tr>
<td>1</td>
<td>Supporting completion in the order of days</td>
</tr>
<tr>
<td>2</td>
<td>Supporting completion in the order of hours</td>
</tr>
<tr>
<td>3</td>
<td>Supporting completion in minutes or seconds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory efficiency requirement level</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Supporting computations on super-computing clusters</td>
</tr>
<tr>
<td>1</td>
<td>Supporting computations on high-performance single-node machines</td>
</tr>
<tr>
<td>2</td>
<td>Supporting computations on ordinary computing power servers</td>
</tr>
<tr>
<td>3</td>
<td>Supporting computations on edge devices</td>
</tr>
</tbody>
</table>
Algorithm 1: Noising before Aggregation FL

Data: $T$, $\mathbf{w}^{(0)}$, $\mu$, $\epsilon$, and $\delta$

1. Initialization: $t = 1$ and $\mathbf{w}_i^{(0)} = \mathbf{w}^{(0)}$, $\forall i$

2. while $t \leq T$ do

3.   Local training process:

4.       while $C_i \in \{C_1, C_2, \ldots, C_N\}$ do

5.           Update the local parameters $\mathbf{w}_i^{(t)}$ as

6.           $\mathbf{w}_i^{(t)} = \arg\min_{\mathbf{w}_i} \left( F_i(\mathbf{w}_i) + \frac{\delta}{2} \|\mathbf{w}_i - \mathbf{w}_i^{(t-1)}\|^2 \right)$

7.           Clip the local parameters

8.           $\mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t)}/\max\left(1, \frac{\|\mathbf{w}_i^{(t)}\|}{C} \right)$

9.           Add noise and upload parameters

10.          $\tilde{\mathbf{w}}_i^{(t)} = \mathbf{w}_i^{(t)} + \mathbf{n}_i^{(t)}$

11.   Model aggregating process:

12.       Update the global parameters $\mathbf{w}^{(t)}$ as

13.       $\mathbf{w}^{(t)} = \sum_{i=1}^{N} p_i \tilde{\mathbf{w}}_i^{(t)}$

14.       The server broadcasts global noised parameters

15.       $\tilde{\mathbf{w}}^{(t)} = \mathbf{w}^{(t)} + \mathbf{n}_D^{(t)}$

16.   Local testing process:

17.       while $C_i \in \{C_1, C_2, \ldots, C_N\}$ do

18.           Test the aggregating parameters $\tilde{\mathbf{w}}^{(t)}$ using local dataset

19.           $t \leftarrow t + 1$

Result: $\tilde{\mathbf{w}}^{(T)}$
Appendix D – Pseudo Code for the GAN attack algorithm [23]

Algorithm 1 Collaborative Training under GAN attack

Pre-Training Phase: Participants agree in advance on the following, as pointed out also by [77]:
1. common learning architecture, (model, labels etc.) [For ex. V declares labels \( a, b \) and \( L \) labels \( b, c, l \)]
2. learning rate, (\( l_r \))
3. parameter upload fraction (percentage), (\( l_u \))
4. parameter download fraction, (\( d_u \))
5. threshold for gradient selection, (\( r \))
6. bound of shared gradients, (\( y \))
7. training procedure, (sequential, asynchronous)
8. parameter upload criteria [cf. [77]]

Training Phase
1. for epoch = 1 to \( n \)epochs do
2. Enable user \( x \) for training
3. User \( x \) downloads \( l_u \) parameters from PS
4. Replace respective local parameters on user \( x \) local model with newly downloaded ones
5. if \( \text{user\_type} \rightarrow \text{ADVERSARY} \) then
6. Create a replica of local \( \text{freshlyupdated} \) model as \( D \) (discriminator)
7. Run Generator \( G \) on \( D \) targeting class \( a \) (unknown to the adversary)
8. Update \( G \) based on the answer from \( D \)
9. Get \( n \)-samples of class \( a \) generated by \( G \)
10. Assign label \( c \) (fake label) to generated samples of class \( a \)
11. Merge the generated data with the local dataset of the adversary
12. end if
13. Run SGD on local dataset and update the local model
14. Compute the gradient vector \((\text{newParameters} - \text{oldParameters})\)
15. Upload \( l_u \) parameters to PS
16. end for
17. return Collaboratively Trained Model [At the end of training, the adversary will have prototypical examples of members of class \( a \) known only to the victim]