*electronics*

MDPI

*Article*

# Federated Learning with Dynamic Model Exchange

**Hannes Hilberger *, Sten Hanke and Markus Bödenler**

eHealth Institute, FH JOANNEUM University of Applied Sciences, 8020 Graz, Austria; sten.hanke@fh-joanneum.at (S.H.); markus.boedenler@fh-joanneum.at (M.B.)
* Correspondence: hannes.hilberger@fh-joanneum.at

**Abstract:** Large amounts of data are needed to train accurate robust machine learning models, but the acquisition of these data is complicated due to strict regulations. While many business sectors often have unused data silos, researchers face the problem of not being able to obtain a large amount of real-world data. This is especially true in the healthcare sector, since transferring these data is often associated with bureaucratic overhead because of, for example, increased security requirements and privacy laws. Federated Learning should circumvent this problem and allow training to take place directly on the data owner's side without sending them to a central location such as a server. Currently, there exist several frameworks for this purpose such as TensorFlow Federated, Flower, or PySyft/PyGrid. These frameworks define models for both the server and client since the coordination of the training is performed by a server. Here, we present a practical method that contains a dynamic exchange of the model, so that the model is not statically stored in source code. During this process, the model architecture and training configuration are defined by the researchers and sent to the server, which passes the settings to the clients. In addition, the model is transformed by the data owner to incorporate Differential Privacy. To trace a comparison between central learning and the impact of Differential Privacy, performance and security evaluation experiments were conducted. It was found that Federated Learning can achieve results on par with centralised learning and that the use of Differential Privacy can improve the robustness of the model against Membership Inference Attacks in an honest-but-curious setting.

**Keywords:** Federated Learning; Differential Privacy; privacy preserving; membership inference attack

check for updates

**Citation:** Hilberger, H.; Hanke, S.; Bödenler, M. Federated Learning with Dynamic Model Exchange. *Electronics* **2022**, *11*, 1530. https://doi.org/10.3390/electronics11101530

Academic Editor: Gemma Piella

Received: 31 March 2022
Accepted: 9 May 2022
Published: 11 May 2022

## 1. Introduction

Machine learning models need large and diverse datasets to train algorithms. Ideally, data are collected collaboratively to generate robust and bias-free models. However, there are several issues that need to be addressed when training models with health data: (1) different locations of data such as in healthcare facilities, Internet of Medical Things (IoMT) devices, or mobile phones [1]; (2) data anonymisation such as removing metadata such as the patient name or date of birth, which is often not sufficient to guarantee privacy [2]; (3) imbalance of features in training, as well as inadequately large datasets [3]. Furthermore, due to strict data protection guidelines, especially in the health sector [4], it is often difficult to transport data, and consequently, they remain in unused data silos. If a hospital uses only its own data to train a machine learning model, the resulting lack of generalisation would render the model useless for other institutions [5].

Federated Learning should address these points and make it possible for healthcare organisations to collaboratively train models in different locations. Federated Learning (FL) is a method in which several clients train a model together, and the results are summarised by a central instance, such as a server. Since this technology has mainly been described for mobile devices [6], the term FL has been further subdivided into Cross-Device-FL (CDFL) and Cross-Silo-FL (CSFL). While CDFL deals with mobile or IoT devices, CSFL uses data from different organisations stored in data silos [7]. Our proposed method is mainly concerned with CSFL. Federated Averaging (FedAVG) is introduced to combine the clients' local models by calculating the average of the trained models. After the clients have trained the model on their local data, they send their parameters back to the central instance [6,8], which then carries out the FedAVG procedure. There are different methodologies for the averaging procedure, as listed in [8–10].

The use of FL in healthcare has already been evaluated in several experiments. For example, a model was trained by using Federated Learning to predict the number of hospitalisations for cardiac events in the following calendar year based on Electronic Health Record (EHR) data [1] over a cohort of 45,579 patients. Another use case potentially arises in brain tumour segmentation, as training a model is made difficult due to the limited data [11] of one clinical centre alone.

Not only institutions have an interest in training models, but also researchers who, if they are not cooperating with other companies, have to work with publicly available datasets. Here, we propose an architecture, Federated Learning-Dynamic Model Exchange (FL-DMX), which addresses the aforementioned points. FL-DMX should allow data scientists to send their model structure with the training configuration to a server, which forwards the requests to clients with data silos. The clients can define the so-called Differential Privacy themselves through configurable parameters. This method provides the advantage that several people can independently train their models on the data without having direct access to it while preserving privacy. In the end, the best model could then be taken and further evaluated. To our knowledge, such an infrastructure of a privacy-preserving dynamic model exchange has not yet been established, and therefore, it fills the gap of bringing the entire model to the clients. With the proposed approach, we contribute to ensuring that people who cannot sign such agreements, but who have to be consulted during clinical trials, also have the opportunity to train models for this purpose. This could also be a valuable contribution to university teaching, as students could train directly on real datasets without relying on toy datasets.

## 2. Related Work

In FL, there are also several challenges that need to be overcome. For example, a high communication bandwidth is required between the clients and the server, which could be reduced by restricting the participating clients or by using various compression methods. Another limitation is that the data are mostly Not Independently and Identically Distributed (Non-IID), e.g., each client does not have all classes for a classification task [4,7,12]. In this section, the security and privacy aspects are examined in more detail before the frameworks are discussed.

### *2.1. Security and Privacy*

Research areas are also concerned with attack vectors in Deep Learning and FL. Model Inversion Attacks, for example, can reconstruct features or entire record entries [13–15]. It was demonstrated by [15] that Optical Coherence Tomography (OCT) and Chest X-Ray images can be reproduced. Advanced Persistent Threat (APT) attacks are another method used in an attempt to gain access to confidential information by trying to find the codes of target systems at the onset of tasks to gain access to the system. As illustrated in [16], APT attacks can be identified and classified. Since we are incorporating TF Privacy [17] in our method and it integrates Membership Inference Attacks, as well as Differential Privacy, we will introduce them in more detail.

Membership Inference Attacks (MIAs) are used to analyse data points to determine whether they are included in the training dataset or not. The assumption in MIA is that models behave differently on data they see for the first time or have already used in training [18]. The goal is to create an attack model that can recognise these differences and thereby decide whether a dataset was part of the training or not. For this purpose, so-called shadow models are trained, whereby the attacker is aware whether certain datasets were used in these models. The target model is the model the MIA is carried out on. Supervised learning can then be used to train the shadow models so that the output is either “in” or "out", depending on whether they have the property or not. For TensorFlow Privacy [17], a modified form of this attack is used, which states that the predictions of the original model are sufficient to make statements about the membership of data points [19].

FL aims to ensure that data remain in their data silos while not being disclosed [20]. Differential Privacy (DP) attempts to maintain the global statistical distribution while reducing information about an entity [21]. DP reduces the vulnerability of Model Inversion Attacks [15], as noise is placed on data. Therefore, a noise of a Laplace or Gaussian distribution is added to increase the difference of the two datasets and to minimise the risk of making a statement about the property of an entity. Moreover, there are several reasons why the Gaussian mechanism is preferable to Laplace's, one being that the noise in the Gaussian mechanism comes from the same distribution as the error that is already present in the dataset [22].

### 2.2. *Federated Learning Frameworks*

Recent research in the context of Federated Learning infrastructure is also looking at the integration of Blockchain technology [23,24], which provides the advantage of handling trust issues between a central node and participants, as well as operations performed by the central node, e.g., the averaging mechanism in FL [25]. For healthcare, the authors of [23] described an infrastructure based on FL and Blockchain to address the challenges of data security and prediction of COVID-19 for IoMT scenarios. Yet, to keep our method simple and to clearly evaluate the potential of FL, the evaluation and integration of a Blockchain and FL will not be further addressed in this paper. Further work also deals with the use of edge servers that receive the local parameters and aggregate them before they are forwarded to the central server [26]. A data owner can freely select an edge server and receives a reward depending on the amount of data. This would have the advantage of reducing global communication between the central server and the data owners, but also lower the dropout rate, as well as efficient resource assignment. These can be self-organised in terms of the autonomy of the data owners to choose which edge servers to be connected to, the allocation of resources, such as bandwidth, by the edge servers, as well as the individual preferences by the central server regarding payment [27]. Their method was tested in an evolutionary game approach where the aim was to maximise the output for each component.

Nevertheless, other open-source FL frameworks exist, and our proposed method will be compared with a selected few. TensorFlow Federated (TFF) [28] was chosen because it uses TensorFlow Privacy and only works with TensorFlow models like our framework does as well. PySyft/PyGrid introduced a data-centric approach [29], which is similar to FL-DMX and, therefore, necessary for comparison. However, unlike our framework, PySyft/PyGrid uses PyTorch models and an iterative rather than parallel approach to training. Flower offers a framework-independent approach and has no built-in security mechanisms [30]. It was used to compare the accuracy for training with our proposed method.

#### 2.2.1. TensorFlow Federated

TensorFlow Federated (TFF) [28] is a library that is only compatible with TensorFlow [31]. TFF is described as follows [28]:

> *"TFF enables developers to simulate the included Federated Learning algorithms on their models and data, as well as to experiment with novel algorithms."*

Currently, according to this description, TFF can only be used to simulate FL. Here, we want to represent a real machine-to-machine communication, so a further description will not be provided.

#### 2.2.2. PySyft and PyGrid

PySyft and PyGrid are part of the Syft environment [32] and use PyTorch [33] to train models. PySyft uses the previously mentioned technologies FL, as well as DP, Multi-Party-Computation (MPC), and Homomorphic Encryption (HE) [34] to decouple the data from the training of the model [32]. PySyft is a library that defines the objects, abstractions, and algorithms. PyGrid offers the possibility for the implementation of FL systems with real terminals. The architecture variant Model-Centric-FL (MCFL) is a method in which a model is hosted in PyGrid. In this type, no network is needed as a component. An application such as a mobile device or a web application wants to participate in a training cycle. After training, the differences between the local and global model are returned and an averaging process is performed [32]. In Data-Centric-FL (DCFL), the initial model is sent by the model owner to the data owners, who train the model with their data. This is done via so-called workers. After the training has been carried out, the weights are sent back to the model owner. This represents an iterative process across all data owners over several rounds [32].

An extension has also been developed for PySyft/PyGrid, which is called PriMIA [15]. With this extension, different medical data formats such as DICOM and datasets from various modalities (e.g., computed tomography) can be integrated. In order to evaluate the framework, a dataset of paediatric Chest X-Rays [35] was selected and classified. In FL without DP and Secure Multi-Party Computation (SMPC) [34], the results deteriorated by about 0.03% on the validation dataset in contrast with centralised learning [15]. When SMPC is used, another small loss takes place. When DP is used, the accuracy on the validation dataset decreases by about 0.07%. This demonstrated that a similar level of accuracy can be achieved despite the use of protection mechanisms.

#### 2.2.3. Flower

Flower has an advantage over the other frameworks in that it is framework-independent and can, therefore, act with PyTorch, TensorFlow, or MXNet. The strategy represents an abstraction of an averaging process. The communication between the client and the server takes place via Remote Procedure Calls (RPCs) [30].

For DP, there is only one approach so far where the clients can decide whether they want to use this methodology or not, but this has to be implemented for each client. It must be ensured that each client has the same model stored. DP is only an implementation of the TensorFlow Privacy tutorial [17] on the server and client side.

## 3. Federated Learning in the LETHE Project

FL-DMX will be used in the Horizon 2020 project LETHE, where a multi-centre parallel-group randomised control trial is conducted with the aim to project the development of dementia and related symptoms with AI and ML methods. The models are based on MRI and clinical observational data provided by four clinical centres.

Multi-centred studies including different clinical centres need to decide on complex data agreements and processes to be able to join data together, transfer data out of the clinical centre, as well as to perform a joint data analysis. Moreover, agreements sometimes need to be signed repeatedly when new employees participate in the research. It would be more convenient if the data could stay in the clinical centre and no data need to be transferred. This would prevent researchers from developing models, as they do not have access to the data. With FL-DMX, the data stay in the clinical centres, while the researchers of the project can find the best suitable model for predicting the onset of cognitive decline, as intended in the project. Another advantage is that through FL, the cohort and the amount of data can easily be extended. This is especially relevant for rare data pools in individual centres or rare events in the data. At the same time, the cohort is expanded without the risks of transporting health data out of the clinical centre itself. Although FL has shown that it is possible to collaboratively train models and the data reside with the individual client, FL still holds the limitation that the model needs to be transferred and data can be possibly reconstructed, as happened in the aforementioned Model Inversion Attacks. To overcome this problem, we applied a privacy mechanism to enhance the security, as can be seen in Figure 1. By introducing a user management system, only authorised persons can send models to the server.

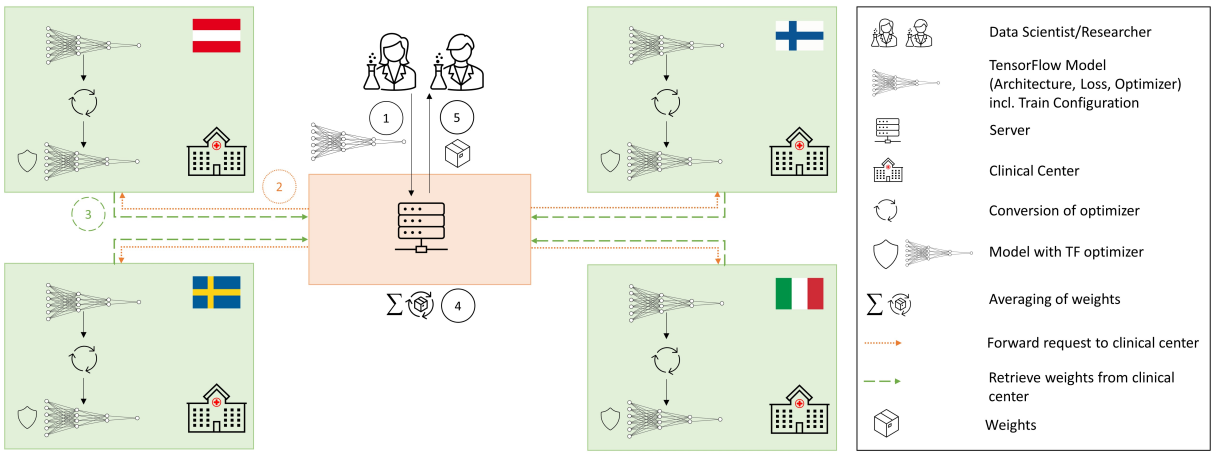

1
2
3
4
5
Data Scientist/Researcher
TensorFlow Model (Architecture, Loss, Optimizer) incl. Train Configuration
Server
Clinical Center
Conversion of optimizer
Model with TF optimizer
Averaging of weights
Forward request to clinical center
Retrieve weights from clinical center
Weights

**Figure 1.** Federated Learning in the context of LETHE: (1) the data scientist sends his/her model to a central server; (2) the server forwards request (model architecture, optimiser, loss, training configuration) to the appropriate clinical centres; (3) clinical centres send their weights after training back to the server; (4) the server averages the weights with FedAVG; (5) after the client rounds, defined in the header of the data scientist's request, have been reached, the final weights are returned to the data scientist.

## 4. Federated Learning—Dynamic Model Exchange

While Flower defines the model at each client and PyGrid hosts either the data or a model, FL-DMX deals with a dynamic transfer of the model to the clients. In the current implementation of FL-DMX, only TensorFlow is possible; PyTorch is not supported yet. The transmission of the model itself takes place via pickle or JSON, but JSON is recommended due to security reasons [36]. The communication between model owners (data scientist), the server, and the clients (data owners) happens primarily via Flask [37]. Currently, both the clients and the server have endpoints that are addressed via HTTP requests. Later on, an integration of Flask socket connections between the clients should take place in order to aggregate the data before they reach the server. DP is fully implemented in FL-DMX. Here, the optimiser and loss components of the model, once received by the client, are wrapped with their parameters into the components of TensorFlow Privacy [17] and then used for training. This holds the advantage that it can also protect against model owners who are honest-but-curious, as models always incorporate DP. After the training, the weights are sent back to the server. These are aggregated with FedAVG, and the process is repeated as many times as client rounds have been defined. The settings for DP are defined individually by the clients and are defined as follows:

- l2_norm_clip: maximum euclidean norm of a single gradient for a single training example.
- noise_multiplier: how much noise is added to the gradients.
- num_microbatches: for how many training examples the gradients are clipped at the same time; for example, with num_microbatches = 32 and minibatch = 256, 32 average gradients of 8 training examples would be clipped.

It should be noted that a more generalised form of DP, Rényi Differential Privacy (RDP), is used for TensorFlow Privacy [17]. For the theory, as well as calculation of Rényi Divergence and Rényi Differential Privacy, [22] and [38] can be consulted.

The workflow of FL-DMX training is visualised in Figure 2. The data scientist first defines the TF model. Afterwards, the components of the model (architecture, optimiser, loss) will be extracted as JSON. The information can be retrieved via the TensorFlow/Keras API. To initiate the training of his/her model, the data scientist creates an HTTP request, which consists of the training configuration in the header and the components of the model in the body. The following information must be included in the header:

- Category: dataset on which the model should be trained.
- Epochs client rounds: how many epochs a client should train before averaging.
- Client rounds: how often the weights will be averaged before the data scientist receives it.
- Batch size.

The request is then sent to the server, which extracts the header information and uses the category information to find out on which dataset the data scientist wants to train the model. However, the mapping of clients and corresponding datasets is currently static, as the required database is yet to be implemented. After forwarding the request to the clients, each client clones the optimiser and the loss and then converts the optimiser to a TF Privacy optimiser. Afterwards, the model is recompiled and the training takes place based on the configuration information of the header. The training itself uses the algorithm of [39]. This process is repeated until the number of client rounds, as defined in the header, has been reached. In each step, the weights are averaged. Finally, the averaged weights are sent back to the data scientist as an HTTP response.

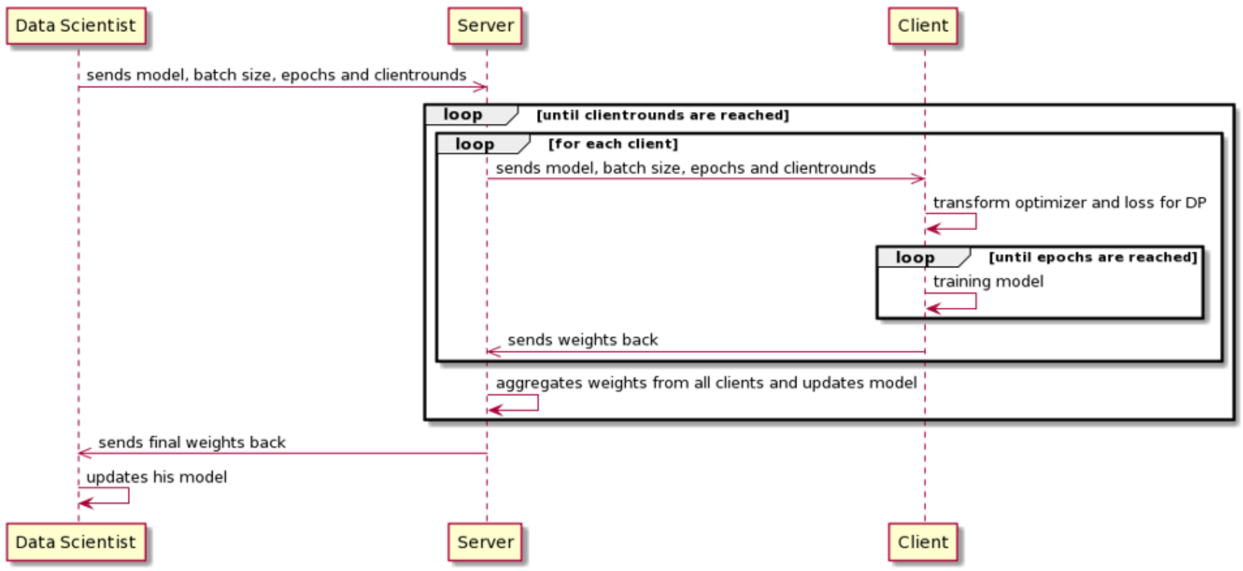

Data Scientist
Server
Client
sends model, batch size, epochs and clientrounds
loop [until clientrounds are reached]
loop [for each client]
sends model, batch size, epochs and clientrounds
transform optimizer and loss for DP
loop [until epochs are reached]
training model
sends weights back
aggregates weights from all clients and updates model
sends final weights back
updates his model
Data Scientist
Server
Client

**Figure 2.** Workflow of FL-DMX training: The data scientist defines his/her model architecture and training configuration and sends it along with the data request to the server, which coordinates the training. The server forwards the architecture and configuration to the clients, which train the model in parallel. First, they transform the optimiser and loss of the model with their DP parameters before the training is carried out. For each round, the weights are sent back and averaged by the server. This is an iterative process until the client rounds are reached. Finally, the server sends the weights back to the data scientist, who updates his/her model.

In general, the FedAVG procedure was oriented towards Flower [40], whereby no abstraction for various averaging methods has yet been integrated. The advantage of FL-DMX is the dynamic model transfer, as the model does not have to be defined for each client compared to Flower. With Flower [40], DP must be integrated by the model, and the clients have no influence on whether the security mechanisms have been implemented. With DCFL from PyGrid and FL-DMX, the training can always start because the number of clients is known by the server.

A similar procedure was adopted by PyGrid [29], but while PyGrid only works with PyTorch models, FL-DMX uses TensorFlow models. DCFL by PyGrid and FL-DMX introduce a data scientist, who initially starts the training. With FL-DMX, no iterative learning takes place, as the weights are aggregated and the individual clients are provided with the training configurations via threads. In addition, precise flags must be stored so that the server knows which client has specific data. In contrast, the IP addresses and the flags for FL-DMX are stored with the server, so that a client cannot make false entries.

However, precautions must be taken because FL-DMX trains on unknown models. One possibility would be to run the applications in a sandbox environment [41]. Another feature could be to transmit only a certain percentage of weights to the server [42]. In addition, an evaluation would have to be made after training to see how well the model is protected against attacks. If the trained model does not fulfil a condition, the weights would be withheld. Possible evaluations of subsequent attacks could be carried out by the clients with the final weights:

- Structural similarity of images when performing a Model Inversion Attack [13]; depending on the similarity [43], the weights are not sent to the data scientist.
- Evaluation of the score of an MIA [17].
- Security curves evaluating the robustness of models against attacks [44].

These attacks would be carried out by the individual clients with the final weights on their own datasets. In the process, they would send their assessment to the server. The server would then evaluate them and, depending on the result, release the weights to the model owner or not.

## 5. Methods

### *5.1. Datasets*

For a comparison of the proposed FL setup with centralised learning, we used MedMNIST. MedMNIST [45] consists of a total of ten different medical datasets. For this work, PathMNIST was used, which consists of a total of nine classes with 89,996 entries for training, 10,004 for validation, and 7180 for testing. The dataset originally examined colorectal carcinoma [46] and was subsequently processed and reduced from $3 \times 224 \times 224$ to $3 \times 28 \times 28$ pixels [45].

### *5.2. Model Generation*

For model creation, AutoKeras [47] was used. This involved a training run with AutoKeras on the full dataset using the first generated model. The training and validation dataset were combined before splitting them for each client. The model consists of one input layer and two subsequent hidden layers, as well as two dropout layers (one before and one after the flatten layer). The first dropout layer has a keep probability of 0.25, and the second dropout layer has 0.5. Afterwards, we used the model without weights to train it with our FL architecture.

### *5.3. Devices*

In general, four devices were involved in the experiments. The devices' exact specifications can be found in Table 1. In the case of central learning, a separate performance measurement was carried out for the two devices that were also responsible for learning in FL (clients). For FL-DMX, the other two devices acted as the server and data scientist, which send the model to the server.

### *5.4. Workflow and Test Setup*

We applied the PathMNIST dataset in 10 runs on different IID dataset splits (50/50 and 80/20). DP was subsequently enabled by model transformation on the client side to evaluate the proposed architecture and draw comparisons. The datasets themselves were created beforehand and then saved and distributed to the clients. The DP parameters for FL-DMX were determined empirically.

Besides the accuracy, the time was also compared, as the whole model and not only the weights are exchanged, as well as the model, more specifically the optimiser, is transformed into a TF Privacy optimiser by cloning and recompiling the model components. A

Membership Inference Attack was also used to further utilise TF Privacy, as it is included in the package.

**Table 1.** Device specifications.

| | **GPU** | **CPU** | **RAM** | **Role** |
|---|---|---|---|---|
| PC #1 | NVIDIA GeForce GTX 1650 | Intel(R) Core(TM) i5-9400F CPU @ 2.90 GHz 6/6 Core/Threads | DDR4 16 GB | Client |
| PC #2 | NVIDIA GeForce GTX 660 | Intel(R) Core(TM) i5-9400F CPU @ 2.90 GHz 6/6 Core/Threads | DDR3 8 GB | Server |
| Laptop #1 | NVIDIA GeForce MX 250 | Intel(R) Core(TM) i7-8565U CPU @ 1.80 GHz 4/8 Core/Threads | DDR4 16 GB | Client |
| Laptop #2 | NVIDIA GeForce GTX 765M | Intel(R) Core(TM) i7-4710MQ CPU @ 2.50 GHz 4/8 Core/Threads | DDR3 8 GB | Data Scientist |

For our test setup, we still hard-coded the mappings (e.g., one client owns the PathMNIST and MNIST dataset, another one only the PathMNIST) of the individual categories that a client owns at a server. In the future, this should be replaced by a database at the server side where it is specified which client has which kind of data or which data category. A more detailed description of the devices used in our test setup can be found in Table 1. In the first step, the model is created by a data scientist (Laptop #2). Afterwards, the model, as well as the optimiser and loss are extracted as JSON. Then, the corresponding training configuration is set in the header, while the model is sent in the body of the request. Finally, the HTTP request is sent to the server (PC #2), which checks the training configuration and forwards the request to the clients (PC #1 and Laptop #1) accordingly. After cloning the model components, converting the optimiser, and recompiling the model, the training with the respective epochs of the configuration is executed as described in [39]. Finally, the weights are sent back to the server, which then averages them. When the client rounds are fulfilled, the weights are sent back to the data scientist.

## 6. Results

A total of ten training runs were carried out and the average calculated. As shown in Table 2, no difference was found between centralised learning and FL-DMX. However, it can be seen here that the weaker device can negatively influence the time for execution.

DP was then implemented for FL-DMX, which performs a client-side transformation of the Keras optimiser and the loss. The parameters for DP were empirically determined or calculated as follows:

- Number of microbatches = Batch Size;
- Clipping threshold = 0.85;
- Noise multiplier = 1.3;
- Calculated $\epsilon = 0.538$ (80% of data); 0.484 (20% of data); 0.59 (50% of data);
- $\delta = 1 \times 10^{-5}$;
- Optimal divergence order ($\alpha$): 21.0.

It was expected that the implementation of DP would decrease the accuracy, since according to [17], a noise multiplier of at least 0.3 is recommended. To achieve a higher level of security, we wanted to use a greater value than 0.3 and, therefore, decided to use 1.3. Therefore, an additional comparison with an increased number of epochs was carried out as compensation. In contrast to [15], experiments were also realised that only implemented DP without MPC. Furthermore, a slightly larger $\epsilon$ with 6.0 and a $\delta$ of $1.9 \times 10^{-4}$ of [15] was chosen. The results can be understood because there is a similar loss of accuracy on the test dataset due to the introduction of DP.

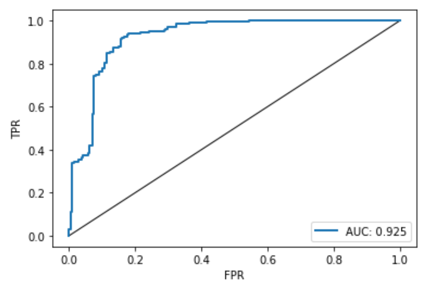
**Table 2.** Evaluation of the Federated Learning frameworks after ten runs with accuracy and duration.

| | **Parameter Set #1** [1] | | **Parameter Set #2** [2] | |
|---|---|---|---|---|
| | **Time in Seconds** | **Accuracy** | **Time in Seconds** | **Accuracy** |
| TF without FL-PC-GPU [3] | ~323.4 | 83.78 | | |
| TF without FL-Laptop-GPU [4] | ~597 | 82.93 | | |
| FL-DMX-GPU-without DP | ~294 | 82.26 | ~255 | 84 |
| FL-DMX-GPU-with DP | ~298 | 77.67 | ~254 | 79.45 |
| FL-DMX-GPU-with DP (50 Epochs) | ~593 | 80.71 | ~495 | 80.51 |

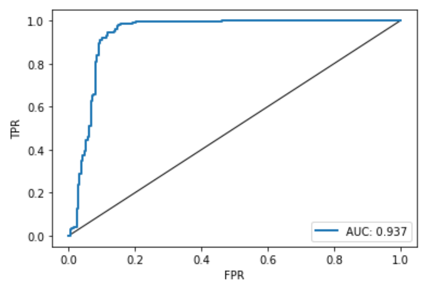
[1] For FL: Learning Rate: 0.001; Optimiser: Adam; Loss: Cross Entropy; Dropout Layer 1: 0.25; Dropout Layer 2: 0.5; Epochs: 25; Batch Size: 64; IID Data Split FL: 50/50. [2] For FL: Learning Rate: 0.001; Optimiser: Adam; Loss: Cross Entropy; Dropout Layer 1: 0.25; Dropout Layer 2: 0.5; Epochs: 25; Batch Size: 64; IID Data Split FL: 80/20. [3] Parameter sets differ in the FL distribution; values for this line are the same for both parameter sets. [4] Parameter sets differ in FL distribution; values for this line are the same for both parameter sets.

*Membership Inference Attack*

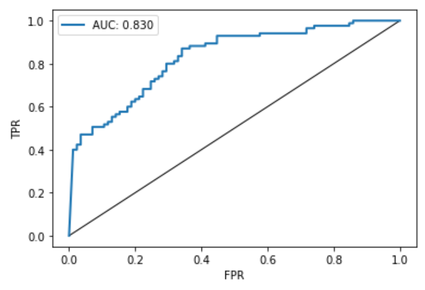
Subsequently, the MIA of [17] was applied to the models with the highest accuracy. The use of FL alone does not improve security compared to centralised learning. The use of DP reduces the likelihood of the attacker guessing whether the data point was used in the training data. Figure 3 shows the ROC curves of the respective attacks with the highest AUC. In these cases, a lower AUC implies that the attacker has a lower chance of assigning the membership probability. The AUC thus represents the success rate for the attack, meaning that lower AUC values are better in terms of security against MIA.



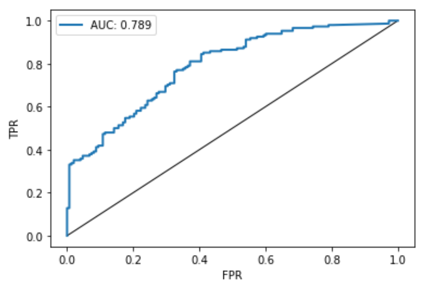
(**a**)



(**b**)



(**c**)



(**d**)

**Figure 3.** Membership Inference Attack on different models. (**a**) MIA on the model with the best accuracy without FL. (**b**) MIA on the model with the best accuracy with FL and without DP. (**c**) MIA on the model with the best accuracy with FL and DP 25 epochs. (**d**) MIA on the model with the best accuracy with FL and DP 50 epochs.

## 7. Discussion

The experiments in Table 2 showed that our proposed architecture, which incorporates a dynamic model exchange while preserving privacy, can compete with central learning in terms of accuracy in the IID setting. This is in line with previous research, where the accuracy of FL and centralised learning has been shown to be comparable [15,30].

With the introduction of DP, the results in Table 2 show a minimal loss. Specifically, the accuracy reduced from 82.26% to 77.67%. With double the number of epochs, the accuracy can be increased to 80.71%. We could not find any negative impact on the execution time by transferring the entire model, as well as transforming the optimiser into a TF Privacy optimiser. However, increased network traffic may occur. For practical applications, it must be considered whether performance or utility is more important. Flower only provides minimalistic approaches without privacy-supported implementations, in contrast to PyGrid. However, Flower could become a good alternative for mobile devices in the future, as this framework can also work with an unknown number of participants and the training accordingly only starts when a certain number is reached.

In addition to the already existing frameworks, FL-DMX was introduced, which integrates a dynamic exchange of models and, thus, introduces an additional party, the model owners or researchers (data scientists), who do not have any data, but still want to train their models with real data. This approach, in contrast to Flower, serves for a static number of participants that is always known to the server. A similar basic idea is followed in PyGrid, the difference being that FL-DMX is purely based on TensorFlow and there is no iterative learning or hosting of the data, since the training is performed directly by the clients and the server knows the classes of data. The clients with the weakest hardware would have an even greater impact, as the other clients would not be able to perform any operations and would have to wait for this one. FL-DMX also offers the possibility to set the parameters for DP independently for each client.

An example use case for DCFL and FL-DMX would be clinic networks, where the individual clinics carry out the training and are coordinated by a central body. Researchers can send their models to the central body, which then coordinates the training on the data from the clinics. However, incentives (e.g., financial) would have to be created for both the clinics and the central organisation to make their resources available. However, especially in multi-centre studies, researchers could support clinicians in training models without having direct access to the data.

Another problem is the amount of data available from the individual hospitals, as this can vary. Here, some that only have few data could provide the test datasets so that smaller hospitals can also participate and contribute to a better generalisation across sites.

A recommended enhancement for FL-DMX would be a web interface in which model owners first upload their models and these are evaluated before training. The training parameters (e.g., the parameter for the number of epochs has the maximum possible value and would, therefore, block the clients and prevent other trainings) or the model itself (e.g., provision of TensorFlow kernels that execute arbitrary code) should be checked. Basically, it is advisable to train models on a client one after the other, as the performance varies greatly when training several models at the same time. For this purpose, a kind of pipeline system could be integrated, which evaluates a model, releases it afterwards, and then trains it.

Another aspect that is still under development is the distribution of roles for providing the training and test datasets. Doctor's offices do not have the amount of data as, for example, university hospitals and would, therefore, have a smaller influence due to the averaging procedure. Therefore, the opportunity should be created for them to provide their datasets for the subsequent test set.

Further work can describe the dynamics created by FL-DMX more comprehensively. Additional questions could relate to prevention (before training the model, architecture check) and reaction (after training the model, robustness against attacks). Since each client can determine its own DP parameters, it can happen that the performance of individual ones is very poor. To circumvent this problem, poor-quality data filtering [48] could be applied. Our architecture has currently only been tested with IID distributions, and we are currently investigating on making the dynamic model exchange available for Non-IID data as well.

## 8. Conclusions

In summary, FL-DMX has results on par with centralised learning in terms of accuracy, as well as training duration, but without owning the data. Current FL frameworks offer extensive documentation, but they have to adapt their source code for changing models. "Privacy by Design" does not take place, as was demonstrated in the results on MIA. As a result, DP was introduced, which minimises the probability of a successful attack, but also has an impact on performance. It is therefore recommended to implement at least DP; however, the integration of MPC should also be considered. With FL-DMX, a dynamic approach for FL was presented in which the model is defined by a data scientist and then distributed to the clients. Security measures must be defined before and after the training to prevent data leaks. However, it could support researchers in particular in training models on real data, since the demanding regulations for data acquisition could be reduced, as these still lie with the data owners. On the other hand, clinical institutions could be supported in clinical trials without disclosing the data. This could also lead to a greater willingness to share data.

**Author Contributions:** Conceptualisation, H.H., S.H. and M.B.; methodology, H.H., S.H. and M.B.; software, H.H.; validation, H.H. and M.B.; formal analysis, H.H.; investigation, H.H. and M.B.; resources, S.H.; data curation, M.B.; writing—original draft preparation, H.H.; writing—review and editing, S.H. and M.B.; visualisation, H.H.; supervision, M.B.; project administration, H.H.; funding acquisition, S.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** The LETHE-Project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 101017405 (https://cordis.europa.eu/project/id/101017405, accessed on 10 May 2020).

**Acknowledgments:** This research is supported by a larger consortium including Alzheimer Europe, Idryma Technologias kai Erevnas, Universiteit Maastricht, Kaasa Solution GmbH, i2Grow, Stichting EGI, Extra Red SRL, Infotrend Innovations Company Limited, Combinostics Oy, and The Lisbon Council for Economic Competitiveness ASBL.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Brisimi, T.S.; Chen, R.; Mela, T.; Olshevsky, A.; Paschalidis, I.C.; Shi, W. Federated learning of predictive models from federated Electronic Health Records. *Int. J. Med. Inform.* **2018**, *112*, 59–67. [CrossRef]
2. Rieke, N.; Hancox, J.; Li, W.; Milletarì, F.; Roth, H.R.; Albarqouni, S.; Bakas, S.; Galtier, M.N.; Landman, B.A.; Maier-Hein, K.; et al. The future of digital health with Federated Learning. *NPJ Digit. Med.* **2020**, *3*, 119. [CrossRef]
3. Nguyen, D.C.; Pham, Q.V.; Pathirana, P.N.; Ding, M.; Seneviratne, A.; Lin, Z.; Dobre, O.A.; Hwang, W.J. Federated Learning for Smart Healthcare: A Survey. *ACM Comput. Surv.* **2021**, *55*, 1–37. [CrossRef]
4. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
5. Xu, Y.; Ma, L.; Yang, F.; Chen, Y.; Ma, K.; Yang, J.; Yang, X.; Chen, Y.; Shu, C.; Fan, Z.; et al. A collaborative online AI engine for CT-based COVID-19 diagnosis. *medRxiv* **2020**, *12*, 6603. [CrossRef]
6. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017.
7. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
8. Li, X.; Jiang, M.; Zhang, X.; Kamp, M.; Dou, Q. FedBN: Federated Learning on Non-IID Features via Local Batch Normalization. *arXiv* **2021**, arXiv:2102.07623.
9. Li, T.; Sanjabi, M.; Beirami, A.; Smith, V. Fair Resource Allocation in Federated Learning. *arXiv* **2020**, arXiv:1905.10497.
10. Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; Khazaeni, Y. Federated Learning with Matched Averaging. *arXiv* **2020**, arXiv:2002.06440.

11. Aledhari, M.; Razzak, R.; Parizi, R.; Saeed, F. Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Access* **2020**, *8*, 140699–140725. [CrossRef]
12. Xu, J.; Glicksberg, B.S.; Su, C.; Walker, P.; Bian, J.; Wang, F. Federated Learning for Healthcare Informatics. *J. Healthc. Inform. Res.* **2021**, *5*, 1–19. [CrossRef]
13. Fredrikson, M.; Jha, S.; Ristenpart, T. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; ACM: Denver, CO, USA, 2015; pp. 1322–1333.
14. Geiping, J.; Bauermeister, H.; Dröge, H.; Moeller, M. Inverting Gradients—How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 16937–16947.
15. Kaissis, G.; Ziller, A.; Passerat-Palmbach, J.; Ryffel, T.; Usynin, D.; Trask, A.; Lima, I.; Mancuso, J.; Jungmann, F.; Steinborn, M.; et al. End-to-End privacy preserving deep learning on multi-institutional medical imaging. *Nat. Mach. Intell.* **2021**, *3*, 473–484. [CrossRef]
16. Hassannataj Joloudari, J.; Haderbadi, M.; Mashmool, A.; Ghasemigol, M.; Band, S.S.; Mosavi, A. Early Detection of the Advanced Persistent Threat Attack Using Performance Analysis of Deep Learning. *IEEE Access* **2020**, *8*, 186125–186137. [CrossRef]
17. TensorFlow. TensorFlow Privacy. 2021. Available online: https://github.com/tensorflow/privacy (accessed on 11 October 2021).
18. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 3–18. [CrossRef]
19. Salem, A.; Zhang, Y.; Humbert, M.; Berrang, P.; Fritz, M.; Backes, M. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 24–27 February 2019.
20. Rodríguez-Barroso, N.; Stipcich, G.; Jiménez-López, D.; Ruiz-Millán, J.A.; Martínez-Cámara, E.; González-Seco, G.; Luzón, M.V.; Veganzones, M.A.; Herrera, F. Federated Learning and Differential Privacy: Software tools analysis, the Sherpa.ai FL framework and methodological guidelines for preserving data privacy. *Inf. Fusion* **2020**, *64*, 270–292. [CrossRef]
21. Kaissis, G.; Makowski, M.; Rueckert, D.; Braren, R. Secure, privacy-preserving and federated machine learning in medical imaging. *Nat. Mach. Intell.* **2020**, *2*, 305–311. [CrossRef]
22. Mironov, I.; Talwar, K.; Zhang, L. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *arXiv* **2019**, arXiv:1908.10530.
23. Samuel, O.; Omojo, A.B.; Onuja, A.M.; Sunday, Y.; Tiwari, P.; Gupta, D.; Hafeez, G.; Yahaya, A.S.; Fatoba, O.J.; Shamshirband, S. IoMT: A COVID-19 Healthcare System driven by Federated Learning and Blockchain. *IEEE J. Biomed. Health Inform.* **2022**. [CrossRef]
24. Islam, M.J.; Rahman, A.; Kabir, S.; Karim, M.R.; Acharjee, U.K.; Nasir, M.K.; Band, S.S.; Sookhak, M.; Wu, S. Blockchain-SDN-Based Energy-Aware and Distributed Secure Architecture for IoT in Smart Cities. *IEEE Internet Things J.* **2022**, *9*, 3850–3864. [CrossRef]
25. Dun, L.; Han, D.; Weng, T.H.; Zheng, Z.; Li, H.; Liu, H.; Castiglione, A.; Li, K.C. Blockchain for Federated Learning toward secure distributed machine learning systems: A systemic survey. *Soft Comput.* **2022**, *26*, 4423–4440. [CrossRef]
26. Lim, W.Y.B.; Ng, J.S.; Xiong, Z.; Jin, J.; Zhang, Y.; Niyato, D.; Leung, C.; Miao, C. Decentralized Edge Intelligence: A Dynamic Resource Allocation Framework for Hierarchical Federated Learning. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 536–550. [CrossRef]
27. Lim, W.Y.B.; Ng, J.S.; Xiong, Z.; Niyato, D.; Miao, C.; Kim, D.I. Dynamic Edge Association and Resource Allocation in Self-Organizing Hierarchical Federated Learning Networks. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3640–3653. [CrossRef]
28. TensorFlow. TensorFlow Federated: Machine Learning on Decentralized Data. 2021. Available online: https://www.tensorflow.org/federated (accessed on 18 June 2021).
29. OpenMined. PyGrid. 2020. Available online: https://github.com/OpenMined/PyGrid (accessed on 11 October 2021).
30. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Parcollet, T.; de Gusmão, P.P.B.; Lane, N.D. Flower: A Friendly Federated Learning Research Framework. *arXiv* **2021**, arXiv:2007.14390.
31. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A System for Large-Scale Machine Learning. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, Savannah, GA, USA, 2–4 November 2016; USENIX Association: Austin, TX, USA, 2016; pp. 265–283.
32. OpenMined. PySyft. 2021. Available online: https://github.com/OpenMined/PySyft (accessed on 11 October 2021).
33. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; pp. 8024–8035.
34. Ghanem, S.M.; Moursy, I.A. Secure Multiparty Computation via Homomorphic Encryption Library. In Proceedings of the 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 8–10 December 2019; pp. 227–232. [CrossRef]
35. Kermany, D.S.; Goldbaum, M.; Cai, W.; Valentim, C.C.; Liang, H.; Baxter, S.L.; McKeown, A.; Yang, G.; Wu, X.; Yan, F.; et al. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell* **2018**, *172*, 1122–1131. [CrossRef]
36. Byrne, D. *Full Stack Python Security: Cryptography, TLS, and Attack Resistance*; Manning Publications: Manning, CA, USA, 2021.

37. Ronacher, A. *Flask Documentation*. Available online: https://flask.palletsprojects.com/en/2.0.x/ (accessed on 1 July 2021).
38. van Erven, T.; Harremoes, P. Rényi Divergence and Kullback-Leibler Divergence. *IEEE Trans. Inf. Theory* **2014**, *60*, 3797–3820. [CrossRef]
39. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2016; CCS'16, pp. 308–318. [CrossRef]
40. Adap. Flower. 2021. Available online: https://github.com/adap/flower (accessed on 11 October 2021).
41. TensorFlow. TensorFlow Security. 2020. Available online: https://github.com/tensorflow/tensorflow/blob/master/SECURITY.md (accessed on 11 October 2021).
42. Dayan, I.; Roth, H.R.; Zhong, A.; Harouni, A.; Gentili, A.; Abidin, A.Z.; Liu, A.; Costa, A.B.; Wood, B.J.; Tsai, C.S.; et al. Federated learning for predicting clinical outcomes in patients with COVID-19. *Nat. Med.* **2021**, *27*, 1735–1743. [CrossRef]
43. Wang, J.; Song, Y.; Leung, T.; Rosenberg, C.; Wang, J.; Philbin, J.; Chen, B.; Wu, Y. Learning Fine-grained Image Similarity with Deep Ranking. *arXiv* **2014**, arXiv:1404.4661.
44. Trusted-AI. Adversarial Robustness Toolbox (ART). 2021. Available online: https://github.com/Trusted-AI/adversarial-robustness-toolbox (accessed on 11 October 2021).
45. Yang, J.; Shi, R.; Ni, B. MedMNIST Classification Decathlon: A Lightweight AutoML Benchmark for Medical Image Analysis. In Proceedings of the IEEE 18th International Symposium on Biomedical Imaging (ISBI), Nice, France, 13–16 April 2021; pp. 191–195. [CrossRef]
46. Kather, J.N.; Krisam, J.; Charoentong, P.; Luedde, T.; Herpel, E.; Weis, C.A.; Gaiser, T.; Marx, A.; Valous, N.A.; Ferber, D.; et al. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. *PLoS Med.* **2019**, *16*, e1002730. [CrossRef]
47. Jin, H.; Song, Q.; Hu, X. Auto-Keras: An Efficient Neural Architecture Search System. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining;* Association for Computing Machinery: New York, NY, USA, 2019; KDD'19, pp. 1946–1956. [CrossRef]
48. Edemacu, K.; Kim, J.W. Multi-Party Privacy-Preserving Logistic Regression with Poor Quality Data Filtering for IoT Contributors. *Electronics* **2021**, *10*, 2049. [CrossRef]