CE4171 : Survey paper on federated learning and IoT

Aloysius Chan Zhen Wu
School of Computer Engineering
Nanyang Technological University
Email: achan021@e.ntu.edu.sg

1. Introduction

The internet has shaped the world ever since its creation. Providing the ease of access to data as well as the ease for data creation to users all across the globe. With an estimate of about 2.5 quintillion bytes of data are being produced everyday [1], it is no doubt that we are indeed in an information age. According to Moore's law, it is stated that the number of transistors on a computer chip is doubling about every 18 to 24 months. This implies that computing capabilities are getting more developed and advanced with each passing year. As a result, digital devices are getting more and more compact with stronger capabilities such as faster processing power and larger storage capacity, thus, enabling processing capabilities on more uneconomic devices such as watches, speakers and other common electrical devices. All these coupled with the emergence of 5G technology that can further boost current capabilities such as lower latency of up to 1 millisecond, up to 10 gigabits per second (10Gbps) of speed and a larger bandwidth of 1000 times per unit area [2].Therefore, without a doubt, the concept of the Internet of Things (IoT) has become a hot topic in recent years.

IoT enables connectivity between devices, providing information specific to the targeted device's usage. This enables communication between a network of devices, thus, allowing better delegation of tasks and smarter systems. On top of that, IoT can be used to receive a collection of feedback from each individual device's environment to generate collective data from it. This data can then be studied upon to provide better allocation of resources. An example of such an application is the use of IoT for building management systems (BMS). In this example, building energy usage data such as room temperature can be obtained from temperature sensors located within the rooms in the building. These energy usage data can then be studied or trained upon using a machine learning model to help better control the way the building manages their energy internally. This will consequently, help to save cost and lower energy wastage. [3]

With the additional data generated from IoT, data scientists and analysts also stand to benefit from the additional accessibility to data for model training. This can help us develop new frameworks and model architectures that are more complex and better suited for task or domain specific use-cases for forecasting or prediction. However, these data generated can be sensitive and restricted in certain cases. For example, data revolving around personal data are governed by regulations like the General Data Protection Regulation (GDPR) or locally, in Singapore, we have the Personal Data Protection Act (PDPA). Therefore, other means of data usage while maintaining anonymity is required. One such solution is the use of Federated Learning.

2. Federated Learning

The concept of federated learning is simple, instead of the conventional centralized training for machine learning models whereby data of interest are collected in a central database, federated learning encourages the decentralization of data storage. In this case, the models used for training are instead, copied and disseminated to where the data resides for training. As a result, saving additional time required for moving large amounts of data to a centralized database and also, saving on storage space required for a centralized database. This strategy helps to uphold anonymity of the data as the data does not leave the premise of its original database and the training is done only with consent from the curator or the model receiving party (client). Subsequently, once the model is done training on its local dataset, the model's parameters are then sent back to the original federated learning server that houses the original global model, and a weight aggregation per layer of the model is performed against the original global model. This also allows the flexibility for any client to opt out of the federated learning system and stop the sharing of their private datasets without jeopardizing the overall training process.

Figure 1 shows a typical federated learning framework where a federated server is used to house a global model. Copies of the global models are then sent to the different centers for training on their private data. Once the client have finished training on their dataset, their respective trained models are then sent back to the federated server for model weights aggregation.

2.1. Federated learning use case

An example use-case of federated learning in IoT application is the collaborative machine learning developed by Google AI. This framework as seen in figure 2 makes use of client devices such as mobile phones to train on their global

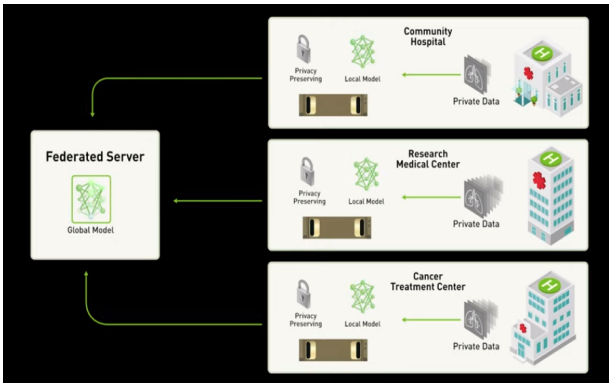

Figure 1. Typical federated learning framework [4]

model. This is tested on the Google Keyboard (GBoard) where the content and suggestions are noted in a history on-device and this is trained against a Gboard query suggestion model. Since the training process may require some time and processing power, this training is only performed when the device is in idle mode, plugged in to a constant battery source and has access to free wireless connection to avoid additional charges and deterioration of performance of the mobile phone. The model is then aggregated using a secure aggregation protocol tapping on cryptographic techniques and the model is only decrypted and updated if 100s or 1000s of users have participated in the federated learning process.

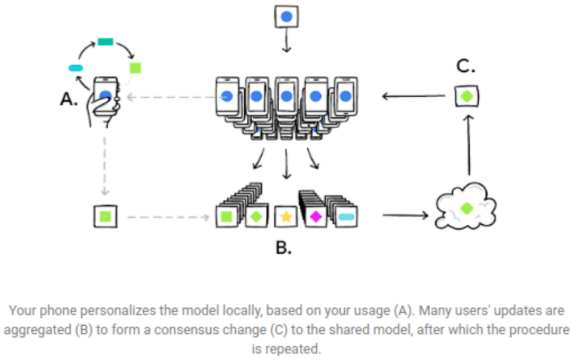

Figure 2. GoogleAI's implementation of federated learning [5]

3. Vulnerabilities of Federated learning

Although the concept of federated learning seems to uphold anonymity and data protection, there are a myriad of methodologies to gather information directly from the copy of the global model. This kind of attack on the global model can be described into 2 components, a black box inference attack or a white box inference attack. A black box inference attack represents that the attacker has no information about the layers of the model. Thus, they are only limited to the output of the model against arbitrary inputs. In this case, the attacker can only obtain the corresponding output given a data point to the global model. A white box inference attack represents that the attacker has information about the layers of the global model. In this case, the attacker is able to compute all the hidden layers, model's output, loss function and gradient of the loss with respect to the parameters of each layer. [6] This is possible by creating an attack model as seen in figure 3. The attack model is made up of Convolutional Neural Networks (CNN) and Fully Connected Network (FCN) components.

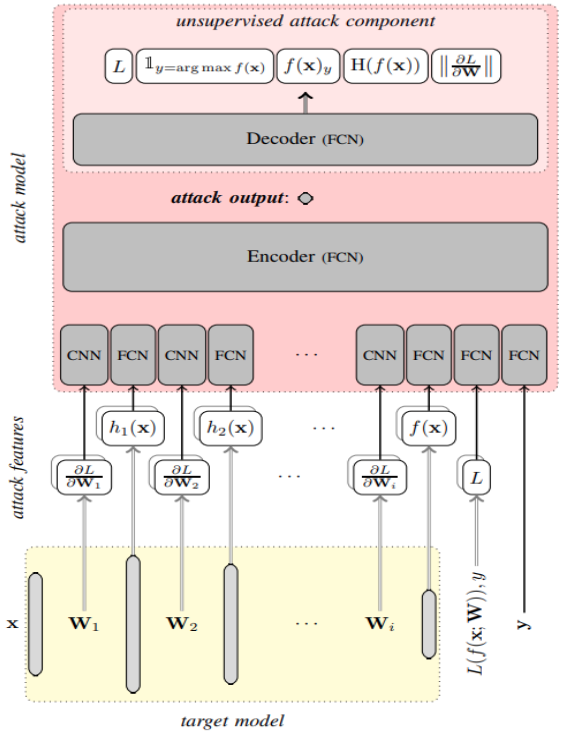

Figure 3. Attack model against a target model [6]

There are mainly 2 types of attack possible against a federated learning global model. Namely, the inference attack in the learning phase and the model inversion attack. The inference attack allows the attacker to infer and determine whether a particular individual's record is present in the training data set for the global model. This type of attack is also known as the membership inference or tracing attack. The model inversion attack, on the other hand, allows the attacker to reverse engineer input data based on the outputs of the hidden layers which are known as feature maps. By reconstructing the feature maps, we are able to vaguely make out the original input.

3.1. Inference attack

To elaborate more on the inference attack, the white box inference attack will be discussed in more details. For the white box inference attack, the attacker objective is to identify or determine the membership of a given target data

(x,y) in a training set D of target model, f. The attacker will then create an attack model that will compute all the hidden layers, model's output and loss function. Subsequently, the attacker will observe each feature T times and stack the T features before passing them to the respective attack component. The output of the CNN and FCN will then be combined and passed over to the encoder. Finally, the output of the encoder shows the embedding of the membership information. In a supervised setting, the embedding result is used to train on a probability of the target data (x,y) in the training set D. In an unsupervised setting, a decoder is trained to reconstruct important features of the attack input.

3.2. Model Inversion attack

To elaborate on the model inversion attack, we consider an example called the Fredrikson attack [6]. This attack taps on models with a small domain of feature values. The main idea of the attack is to iterate through all the possible values of an input feature x1 which in this case is a patient's genetic marker and this will allow the attacker to infer the value of x1 based on a computed weighted probability estimate that is the maximum a posteriori (MAP) estimate. Consequently, the attacker will then be able to obtain the maximum likelihood of x1 from the given model.

4. Privacy preserving techniques

Therefore, given the possible attacks that can be applicable to the use of a federated learning framework, privacy preserving techniques are required to prevent privacy leakage.

4.1. Homomorphic encryption

One type of privacy preserving technique is the use of homomorphic encryption. The idea of Homomorphic Encryption is to allow processing and manipulation of data while the data remains encrypted. This is especially useful as it allows third parties to perform operations on the encrypted data without the need to worry about privacy infringement. Since Homomorphic Encryption uses a public key to encrypt data much like all other encryption methods, only individuals with the matching private key can access the encrypted data.

Figure 4 shows the typical application of Homomorphic Encryption on a third party service provider. The data stored on the third party service provider is encrypted beforehand. The server is then able to perform operations on the user's encrypted data which will generate a corresponding encrypted result. Since the server does not have a matching private key, the result remains protected. Finally, the user can retrieve the encrypted results through a server query and decrypt the ciphertext result to obtain the plaintext contents.

From the simple example above we can see that Homomorphic Encryption is useful in securing data stored in a cloud server under the assumption that the security

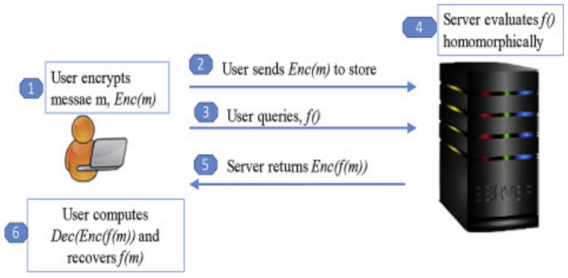

Figure 4. Typical application of Homomorphic encryption [7]

of the Software As a Service Provider is not trustworthy. Alternatively, Homomorphic Encryption can also be used when outsourcing data to commercial cloud environments or to other organizations for analysis on the data. Since the data is encrypted, there will not be any issue of privacy infringement.

There are mainly 3 types of Homomorphic Encryption, namely, Partially Homomorphic Encryption, Somewhat Homomorphic Encryption and Fully Homomorphic Encryption.

4.1.1. Partially Homomorphic Encryption. Partially Homomorphic Encryption only permits either the addition or multiplication operation on the ciphertext but the selected operation can be carried out unlimited number of times on the ciphertext. Examples of multiplicative encryption schemes are the RSA and the ElGamal encryption schemes. An example of an additive encryption scheme is the Paillier cryptosystem.

4.1.2. Somewhat Homomorphic Encryption. Somewhat Homomorphic Encryption, on the other hand, permits both addition and multiplication operations on the ciphertext but the selected operation can only be carried out a limited number of times and up to a certain complexity. Thus, due to this limit, Somewhat Homomorphic Encryption is less favourable and not really suitable for certain applications. Examples of Somewhat Homomorphic Encryption schemes are the BGN scheme and the Catalano-Fiore scheme.

4.1.3. Fully Homomorphic Encryption. Fully Homomorphic Encryption is similar to the Somewhat Homomorphic Encryption scheme but contrary to it, it permits both types of operations and these operations can be carried out unlimited number of times much like the Partially Homomorphic Encryption. However, one drawback is that Fully Homomorphic Encryption uses Somewhat Homomorphic Encryption schemes with an additional computationally expensive bootstrapping algorithm which contributes to a large computational overhead compared to plaintext operations. Thus, due to this large computational overhead, Fully Homomorphic Encryption may not be efficient in practice. Examples of Fully Homomorphic Encryption schemes are the Fan Vercauteren, Brakerski-Vaikuntanathan and YASHE schemes. In conclusion, Homomorphic Encryption schemes have a

tradeoff between functionality and efficiency. Despite the efficiency of Fully Homomorphic Encryption schemes, the additional computational and memory cost when dealing with high multiplicative depth restricts the usage of Fully Homomorphic Encryption in practice.

4.2. Differential Privacy

Homomorphic encryption is considered the early federated learning solution. This allows only the server to decrypt the aggregates. However, this does not eliminate leakages from the aggregates. In order to further mitigate leakages, privacy preserving techniques like differential privacy is used. The idea of differential privacy is to incorporate random noise such that sensitive information recovered from the adversaries are noise and imprecise. In this case, it is more difficult to breach privacy. An example of noise injection can be seen in Figure 5 [8]. Assuming the ground truth value is 3, according to each query, the response received deviates from the ground truth value. Thus, the adversaries will not be able to obtain an accurate response at each query. However, given enough number of queries, the adversaries should be able to estimate the ground truth through the average. Thus, this poses as one of the limitations of differential privacy.

Query Number	Response
1	2.915
2	1.882
3	1.292
4	4.026
5	5.346
Average	3.090
90% confidence interval	1.696 to 4.484

Figure 5. An example response to simulate noise injection [8]

In the context of federated learning, there are 2 variants for differential privacy. One of which is the Local Differential Privacy where the clients will add noise to their own dataset. This can be done either by the clients themselves before making the dataset available to the curator or by the curator. The other variant is the Central Differential Privacy where the clients will send their respective model updates to the central server without any noise added and the server will apply a Differential Privacy aggregation algorithm. Generally, the central differential privacy variant will lead to a higher accuracy as compared to the local differential privacy variant. This is because for the former, noise is added after aggregation as compared to before the aggregation for the latter. However, central differential privacy requires the clients to trust the curator to inject the necessary noise to preserve their datasets' privacy. There are a myriad of reasons for using differential privacy. One of which is the ability to quantify privacy loss. The amount of differential privacy is controllable in a way such that there is a trade-off between privacy and the accuracy of the datasets' information. Another of which is that differential privacy is immune to post-processing. Without additional knowledge of the private datasets, there is no way to compute a function of the output of a differentially private algorithm and make it less private.

$$P[M(D_1 \in A)] \leq e^{\epsilon} P[M(D_2 \in A)] + \delta$$

Figure 6. shows the definite for an (epsilon,delta)-Differential Privacy [9]

D1 and D2 represents 2 neighbouring dataset that differ by only a single entry for all possible outputs. M represents the randomized algorithm that takes in a dataset as its input. Epsilon is a metric that measures the privacy loss or also known as the privacy budget. The smaller the value of epsilon is, the better the privacy protection of the data. However, this in turn will decrease the accuracy of the dataset since the dataset will be added with more noise. In this case, when epsilon is equal to 0 and delta equal to 0, we call this as absolute privacy which provides nothing other than noise. The delta parameter accounts for the chance that the upper bound of epsilon does not hold. In other words, M is epsilon-differentially private if for every possible output x, the probability that the output x is observed does not differ from the original by more than exponential of epsilon.

4.2.1. Local Differential Privacy. In the context of local differential privacy, there are 2 ways to implement this. In the first method, the clients or participants can train on their local datasets and subsequently, before sending the updated parameters of the model to the server, they can apply noise to the updates. In the second method, clients can use the differentially private stochastic gradient descent (DP-SGD) to train the model on their datasets. This way, the clients are able to keep track of the spent privacy budget. The idea of DP-SGD is to add enough noise to hide the largest possible gradient. Since the gradient can be large for outliers and mislabeled inputs, a clipping threshold can be introduced to set a bound for the gradient.

4.2.2. Central Differential Privacy. In the context of central differential privacy, the noise is added to the aggregation function of the server. This provides participation-level differential privacy which makes it indistinguishable for the adversaries to know which client is part of the federated training process. However, this requires a degree of trust for the server since the server will be able to receive the clients' model update and to also trust that it will correctly perform the noise addition.

5. Conclusion for federated learning

In conclusion, federated learning is a strong enabling concept to uphold security and anonymity of data during the model training process. Despite the possible leakages or loopholes that pose vulnerabilities to adversaries, several encryption and privacy techniques can be set up in place as countermeasures, although at a trade off of efficiency for security. Regardless, it provides a proper solution and platform

for data scientists and analytics to access sensitive domain specific information for further research and development of more complex and task specific machine learning models.

6. Federated learning with IoT Application

The concept of federated learning can also be used in tandem with the concept of the internet of things (IoT). As mentioned previously, the utilization of IoT will enable the generation of greater volumes of data which can be used for analysis and training of machine learning models. Therefore, the concept of federated learning greatly synergises with the concept of IoT since they are both decentralized. However, there are also challenges when combining the two concepts together. There are 3 challenges in particular, namely, device heterogeneity, statistical heterogeneity and model heterogeneity.

6.1. Device heterogeneity

Device heterogeneity refers to the fact that each individual can have differing device types and models. This results in differing hardware such as CPU and memory, thus, resulting in differing computational capabilities and storage. Devices that have limited computing capabilities may become stragglers since they will take much longer time to report their model updates. Thus, slowing down on the overall federated learning process. The different environment that each individual is in also affects the network condition of the devices. Thus, a device's communication cost is also a concerning factor since the process of downloading and uploading copies of the global model can cause considerable overhead cost if their network connection does not entail free data transfer. Moreover, different battery levels can pose a problem as devices can possibly drop out from a federated learning process due to low battery or poor connection. As such, the federated learning process must be able to account for fault tolerance.

6.2. Statistical heterogeneity

Statistical heterogeneity refers to the idea that data generated from individual devices are non independent and identically distributed (non-IID) in nature. This causes individual data to be skewed. Data can have feature distribution skew if the distribution of user's activity data varies largely according to their diverse physical characteristics and behavioral habits. Data can also be label distribution skew if the number of samples in each device vary significantly. As such, non-IID can result in weight divergence which can consequently result in a poorer performing model when averaged against the global model.

6.3. Model heterogeneity

Model heterogeneity addresses the concern that different devices will want to have its own customizable local model. Contrary to traditional federated learning, traditional federated learning typically requires a consensus on a particular model's architecture such that the global model is standardized and in turn, can be effectively obtained by weight aggregation of the local model. However, in practical situations, different devices will desire their own model that are adaptive to their application environment and resource constraints. In this case, with differing local models, it is impossible to perform naive aggregation of these local models with the global model.

7. PerFit Framework

Given the 3 challenges, the research paper [10] has proposed a solution to tackle the 3 challenges in a proposed framework named PerFit. The proposed framework adopts a cloud-edge architecture that uses edge computing to alleviate the requirement for sufficient computational capability from heterogeneity devices.

7.1. Edge Computing

Edge computing is the idea of processing data at the network edge instead of processing data at a centralized data center. This in turn provides greater efficiency in data processing. With the Internet of things (IoT), the amount of volume of data is extremely large. Therefore, sending such large amounts of volume back to a centralized data center for processing can cause a huge amount of congestion on the internet and requires a large bandwidth for transmission. Therefore, edge computing aims to provide the means to process data closer to where the data originates from. This can be done by setting up storage and servers nearer to where the data is created at, providing computation and storage at the same location as the data source at the network edge. Moreover, data collected can have different importance. Those that do not require further actions are not required for storage while others may require further analysis or immediate response.

7.2. Elaboration on PerFit framework

To elaborate more on the PerFit framework, PerFit is a personalized federated learning framework for IoT applications created to address the heterogeneity challenges aforementioned. The collaborative learning process in the PerFit framework has 3 stages, namely the offloading stage, learning stage and the personalization stage. This is further illustrated in the figure 6.

7.2.1. Offloading stage. The offloading stage dictates how much of the model is offloaded to the edge for collaborative computing. If the edge is trustworthy, the whole locally trained model and local data can be offloaded to the edge for faster processing. Else, model partitioning is performed where input layers and the local data is kept on the device and only the remaining model layers are sent to the edge for device-edge collaborative computing.

7.2.2. Learning stage. In the learning stage, individual devices, along with the edge will collaboratively process the respective local models based on their respective local data. Therefore, each edge will have a collaboratively computed model that represents all the participating IoT devices. Subsequently, the information of the representative model from each edge is sent to the cloud server. The cloud server then performs aggregation on the collected models from participating edges against the global model and the process repeats until the global model converges. Subsequently, the global model is sent back to the edges for personalization.

7.2.3. Personalization stage. Lastly, in the personalization stage, the global model is then redistributed to individual devices from the edge. These devices can then make use of the global model's details to train on a personalized model using their local data.

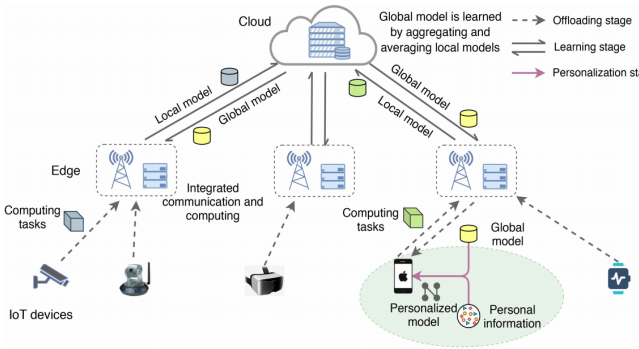

Figure 7. Personalized federated learning framework for intelligent IoT applications, which support flexible selection of personalized federated learning approaches [10]

7.3. Federated learning Mechanisms

The PerFit framework uses edge computing to provide faster processing capabilities for individual IoT devices, thus, alleviating the straggle effect from resource constrained devices. Since the local model aggregation is done at the edge server, communication overhead is reduced by avoiding direct communication from a large number of devices to the cloud server which consequently would have incurred expensive network bandwidth. Moreover, the personalization stage of the framework can accommodate the resource constrained devices by performing model pruning or transfer learning to reduce the model to lightweight models. Additionally, several personalized federated learning mechanisms can be imported to the PerFit framework. These mechanisms can be categorized into federated learning transfer learning, federated meta learning, federated multi-task learning and federated distillation.

7.3.1. Federated Transfer learning. Federated transfer learning makes use of the concept of transfer learning to help create a personalized model. Transfer learning is essentially the idea of reusing layers of a previously trained model to expand upon and train on a new set of data, thus, avoiding the need to retrain a new model from scratch. This will help to speed up the model training process as weight convergence can be achieved faster and it also saves on a large number of resources such as time to train a model from scratch and procurement of large volumes of datasets. There are 2 methodologies for transfer learning. The first methodology [11] specifies the creation of a personalized model by refining the global model according to its local data. Thus, specific layers of the global model are fixed and the remaining layers are fine tuned against the local data. Typically, lower layers are fixed and reused as they capture only common and low-level features. Higher layers are not fixed as they capture more specific distinct classification features for which will be fine-tuned with the local personalized data. Alternatively, the other methodology of transfer learning [12] views models as base layers concatenated with personalization layers. The base layers are shared layers of the global model and the personalization layers are locally trained against the local data of the IoT device. In essence, both methods have the same concept of transfer learning.

7.3.2. Federated Meta learning. Federated meta learning makes use of a secondary machine learning model to train on the personalized model. The secondary model, a meta learner, can be a long short term memory (LSTM) model trained to learn a large number of similar tasks such that it can efficiently update the personalized model's parameters using a small local dataset. This way, the personalized model can quickly adapt to the new task. Models from federated meta learning are typically more robust and useful for devices with few data samples. An example of meta learning is the model agnostic meta learning (MAML). MAML is a representative gradient based meta learning algorithm and is applicable for any model that uses gradient based training.

7.3.3. Federated Multitask learning. Federated multi task learning uses the cloud server to learn model relationships based on the uploaded model parameters from participating IoT devices without risking privacy information. In this case, models from each IoT device can learn from other devices' information. Subsequently, the device can update its own personalized model's parameters with local data and the current available model relationships information. This allows optimization of model parameters for tasks from other IoT devices and supports collaborative training of the personalized model to mitigate device statistical heterogeneity.

7.3.4. Federated Distillation. Federated distillation uses the FedMD framework [10] to allow independent designing of models using the power of knowledge distillation. Each participating IoT device translates learned knowledge to a standard format, such as class probabilities, that is understandable by other IoT devices without sharing data and model architecture. The central server then collates and computes a consensus which will then be distributed to

participating IoT devices. However, for this process, a public dataset is required for the cloud server to aggregate and average the class probabilities for each public data sample. This information is then distributed to the participating IoT devices to guide their model updates. In this case, a teacher-student framework is implemented where each IoT device is the student and the mean output of the IoT devices serve as the teacher's output. The difference in both the student and teacher output will provide learning direction for the student.

8. Conclusion for Federated learning with IoT application

In conclusion, we have observed some of the latest research and contribution to the use of federated learning with IoT application. This field is still evolving as technology gets more advanced in the years to come. Apart from innovating new methods to perform federated learning, device security is becoming a more prevalent issue in the recent years as we can see from the recent facebook and apple privacy war. Therefore, in order to appeal to the public in the uptaking of such federated learning applications, individual privacy must be guaranteed to safeguard individuals from any potential privacy leakages.

References

[1] How much data is created everyday in 2021? [Online]. Available: https://techjury.net/blog/how-much-data-is-created-every-day/gref

[2] 5g technology and networks (speed, use cases and roll out). [Online]. Available: https://www.thalesgroup.com/en/markets/digital-identity-and-security/mobile/inspired/5G

[3] Iot energy management : Beyond the bms. [Online]. Available: https://www.iotacommunications.com/blog/iot-energy-management/

[4] What is federated learning? [Online]. Available: https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning/

[5] Federated learning : collaborative machine learning without centralized training data. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[6] M. N. et al., "Comprehensive privacy analysis of deep learning : passive and active white-box inference attacks against centralized and federated learning," 2020.

[7] Fully homomorphic encryption. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/fully-homomorphic-encryption

[8] A brief introduction to differential privacy. [Online]. Available: https://medium.com/georgian-impact-blog/a-brief-introduction-to-differential-privacy-eacf8722283b

[9] M. N. et al., "Toward robustness and privacy in federated learning : experimenting with local and central differential privacy," 2021.

[10] Q. W. et al., "Personalized federated learning for intelligent iot applications : A cloud-edge based framework," 2020.

[11] Y. C. et al., "Fedhealth: A federated transfer learning framework for wearable healthcare," 2019.

[12] M. G. A. et al., "Federated learning with personalization layers," 2019.