

Cohort-based Kernel Principal Component Analysis with Multi-path Service Routing in Federated Learning

Hira S. Sikandar^a, Saif ur Rehman Malik^b, Adeel Anjum^c, Abid Khan^d and Gwanggil Jeon^{e,*}

^aDepartment of Computer Science, COMSATS University Islamabad, Islamabad, Pakistan

^bCybernetica AS Estonia,

^cInstitute of Information Technology, Quaid i Azam University, Islamabad,

^dCollege of Science and Engineering, University of Derby, Derby, DE22 1GB, United Kingdom

^eDepartment of Embedded Systems Engineering, Incheon National University, Incheon, 22012, Korea

ARTICLE INFO

Keywords:

Federated Learning (FL)
Cohorts
Security
Machine Learning (ML)
Multipath Service Routing (MSR).

ABSTRACT


Federated Learning (FL) is a machine learning (ML) strategy that is performed in a decentralized environment. The training is performed locally by the client on the global model shared by the server. Federated learning has recently been used as a service (FLaaS) to provide a collaborative training environment to independent third-party applications. However, the widespread adoption in distributed settings of FL has opened venues for a number of security attacks. A number of studies have been performed to prevent multiple FL attacks. However, sophisticated attacks, such as label-flipping attacks, have received little or no attention. From the said perspective, this research is focused on providing a defense mechanism for the aforesaid attack. The proposed approach is based on Type-based Cohorts (TC) with Kernel Principal Component Analysis (KPCA) to detect and defend against label-flipping attacks. Moreover, to improve the performance of the network, we will deploy Multi-path Service Routing (MSR) for edge nodes to work effectively. The KPCA will be used to secure the network from attacks. The proposed mechanism will provide an effective and secure FL system. The proposed approach is evaluated with respect to the following measures: execution time, memory consumption, information loss, accuracy, service request violations, and the request's waiting time.

1. Introduction

Federated learning (FL) is a variant of distributed learning, in which a client can train the model without sharing data with others. FL is a promising ML method that aims to solve the problem of data islands while still keeping the data safe [1]. When you use FL, you can build machine learning models that are close to users' data instead of gathering and processing it all in the same place. This is a natural extension of centralized machine learning methods [2]. Additionally, it is capable of dealing with data that is skewed, non-independent (non-I.I.D.), and identically distributed (I.I.D.), as is the case in the real world [3]. A variety of applications in recent years, including prediction of the next word [3], [4], for safety visual object identification [5], resolving the entity [6], recommendations [7], [8], [9], Industrial Internet of Things [10], plus analysis based on graphs [11], [12], [13], have profited FL. FL has been used in many applications, like mobile devices, healthcare, and industrial engineering. FL has been implemented "as a Service" (FLaaS) in [2], [14]. FLaaS provides independent third parties an environment to train models collaboratively. FLaaS has high-level APIs that make it possible to share models between apps on the same device and across networks, as well as software libraries or software development kits that make it easier for app designers to use the services in their own apps and gadgets. Figure 1 shows the working of FL.

The proposed approach is based on Type-based Cohorts (TC) with Kernel Principal Component Analysis (KPCA) to detect and defend against label-flipping attacks. KPCA will be used to secure the network from attacks. Checking the types of each task submitted by the client and then clustering them into a group based on their similarity is TC. KPCA has already been used to mitigate label-flipping attacks. Our approach is to combine TC with KPCA to improve performance and reduce the attack surface in the FLaaS environment. Moreover, to improve the efficiency of the network, we will deploy Multi-path Service Routing (MSR) for edge nodes to work effectively. MSR will redirect the FL client requests to the other edge node by checking if the closest edge node is not available to respond to a specific

*Gwanggil Jeon

 gjeon@inu.ac.uk (G. Jeon)

ORCID(s):

Cohort enabled Federated Learning

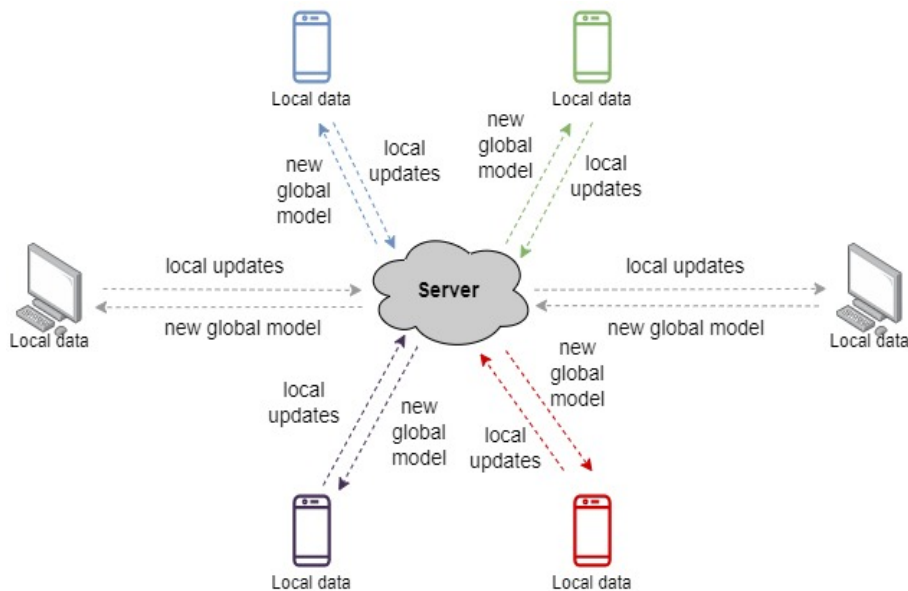


Figure 1: Federated Learning Architecture

task for training.

FL is a privacy-conscious way to train models that don't need to share data and allow players to join and leave federations without having to give up their data [15]. Nonetheless, subsequent research has proven that FL does not always give enough assurance of privacy and resilience. Current FL techniques are susceptible to exploitation. FL's attack surfaces have increased owing to the distribution's features. Malevolent local employees, for example, may try to get private information from honest local workers, or they may work together to harm the global model's performance [16]. Any malicious member is capable of inferring sensitive information about other participants, tampering with the global model aggregation, or poisoning the aggregated model. Privacy breaches might happen if gradients are sent third-party or server during the training process [17], [18], [19]. This information could be sent to a third-party or server [4], [20]. For example, as indicated in [21], an attacker can get a lot of information about the local even if a tiny percentage of gradients are leaked. Recent research also shows that someone who wants to steal training data can just look at the gradients [19], [22]. When it comes to scalability, FL systems are susceptible to data poisoning attacks [23], [24] and model poisoning attacks [25], [26], [27], [28]. Data poisoning and gradient uploads may be used by malicious players to assault the global model's convergence or install hidden triggers inside the global model., respectively (model poisoning).

Centralized machine learning techniques like homomorphic encryption (HE), secure multiparty computation (SMC), and differential privacy (DP) have been used to keep FL's privacy safe, so these techniques have been embraced [15]. Because of their high communication and processing costs, these might be inapplicable to huge FL systems. To ensure DP In aggregated activities DP demands that the collected result include random noise of a specified immensity, which is not best for FL. Furthermore, the noise enhancement necessary by DP is difficult to implement in FL. It is difficult to defend FL from a variety of robust attacks (for example, an untargeted Byzantine attack or a targeted backdoor attack). This is a result of factors. First, the defense may be done entirely on the server. This invalidates a large number of techniques of hidden defense established by centralized ML, including denoising (preprocessing) techniques [29], detection technique of backdoor samples [30], robust augmentation of data [31], a mechanism to finetune [31], and attention distillation of neural [32]. A more recent way of preventing backdoor learning is based on a complex learning process [33]. Second, the protection technique must be resilient towards both data and model poisoning attempts. The majority of extant robustness defenses are gradient aggregation techniques, like Krum/Multi-Krum [34], AGGREGATOR [35], byzantation of gradient descent [36], and gradient descent based on median [37]. Multipath service provisioning approach has been used to ensure the quality of service requests [38]. Requests are

redirected to other servers when congestion occurs at the selected server. Cohorts were initially designed for interest-based browsing. Cohorts have been used in FL to increase the performance of the FL system by dividing client's tasks into groups [39]. KPCA is a kernel principal component analysis that extracts values through eigenvectors and eigenvalues and has been used to detect and defend against label-flipping attacks in [40].

No mechanism is provided to defend against label flipping attacks in the FLaaS environment, which is why we have proposed a novel framework in this paper for FL systems to work efficiently while maintaining privacy. Firstly, Type-based Cohorts with Kernel principal component analysis (TC-KPCA) will be implemented to identify and mitigate label flipping attacks in FLaaS, and secondly, multi-path service routing (MSR) would be used for edge nodes to improve the response time of service requests and to reduce the Service Level Agreement (SLA) violations.

Our approach works by adding KPCA to the defense against label-flipping attacks. KPCA is a kernel principal component analysis that extracts values through eigenvectors and eigenvalues and has been used to detect and defend against label-flipping attacks [40]. Cohorts have been used in [39] to improve the performance of the FL system by dividing the client's tasks into cohorts, but no security mechanism has been used to provide security to the client's data. In this paper, we have combined cohorts with KPCA to achieve efficiency and security at the same time in FLaaS. TC-KPCA is a type-based cohort with kernel principal component analysis that defends specifically against label-flipping attacks. We have also used multipath service routing (MSR) [38] in our system for the efficient routing of clients' data. Our proposed scheme TC-KPCA is achieving better accuracy than the previous FLaaS [2], and that is increased by 8.33%. Execution time decreased by 82.7%. Request's average waiting time and SLA violations decreased by 22.2% and 48.9%, respectively.

The rest of the paper is structured as follows: Section II represents the Literature work of the related studies. Section III describes Background Studies, and Section IV presents System Model while Section V illustrates System Design. After that Section VI provides Experiments and Results. Lastly, Section VII presents the Conclusion of the proposed framework.

2. Related Work

There has been a lot of research work being done in federated learning and a few in federated learning as a service. Here, we will discuss some of the most current research on protecting federated learning from attacks as well as strategies for increasing the performance of machine-learning models, both in terms of security and performance. In order to allow cross-application training of ML models, federated learning "as a Service" has been implemented, and clients may submit queries to servers directly or via edge nodes in order to achieve better convergence. The privacy of data is ensured by introducing noise into the local data of each client [2]. Chuhan et al. [7] proposed a scheme named FedKD (federated knowledge distillation) in which for learning a teacher model and a student model for every client, with the intensity of the distillation being determined by their prediction accuracy. The massive teacher model is updated locally, while the small student model is shared and learned collaboratively across several customers, reducing communication costs. Real-time and accurate pocket diagnosis has been achieved as well as DI-level poisoning assaults can be defended by implementing the RFFL [41]. A Dopamine system is presented, a method for collective training of DNNs over numerous universities using confidential medical images [42]. Tests on a DR picture benchmark dataset show that Dopamine is better than previous baselines when it comes to the utility-privacy trade-off. In [43] authors presented a unique method to FL using local differential privacy. This technique enables effective training of complicated models by individuals as well as offers privacy preservation. Byzantine flexibility, security, and efficiency have been achieved by using BREa for secured federated learning, which depends on evident anomaly detection, quantized stochastic, and reliable model collection [44]. Guowen et al. [45] proposed VerifyNet, the principal protection-saving methodology supporting confirmation during the time spent preparing neural organizations. Firstly, by planning a certain methodology in view of the homomorphic hash capacity and pseudorandom advances that help the unquestionable status for every client. Then, at that point, utilization of a variety of mystery-sharing innovations has been done alongside key arrangement conventions to safeguard the protection of clients' local data and manage the clients exiting the issue in the training phase. In [46], To counter malicious clients, ensembled federated learning has been implemented and demonstrated how it can provide a strong, proven security guarantee. Also, there is a technique for calculating the certified security levels, which we have developed. A commercial center to prepare FL models in a robotized and unknown way for every member (i.e., cell phone) through 5G organizations [47]. The LDP strategy gives

powerful safeguarding efforts against participation deduction assaults by incorporating the model with noisy updates during the model preparation phase. Using a single server to coordinate a model learning process across many clients is how RoFL was built for regular FL deployments [48]. Customer parameters are aggregated inside a machine-learning model by the server on a regular basis under this implementation. Intermediary client and server modules introduced by RoFL work in a manner identical to that of current secured aggregation elements. For the enhancement of training on non-identifiable and independent data, an approach has been proposed that involves a limited sample of data that is shared globally across all the clients [49]. maximized arrangement of representations grasped by the existing local and global model improves the local updates in the MOON method [50]. Jakub et al. [51] proposed that communication costs have been reduced by two methods organized and skewed updates. In organized updates, updates can be learned from a small number of variables in a narrow area. Before sending model updates to the server, they are compressed by applying subsampling, quantization, and random rotations in skewed updates. To battle, the evil impacts of measurable heterogeneity a base plus personalization layer has been proposed for deep feed-forward neural network's federated training named as FedPer [52]. In [53] An algorithm has been proposed for the FL of current neural designs (e.g. CNNs and LSTMs) named as FedMA. By matching and averaging the secret components, a common global model is built in a layer-by-layer approach in FedMA. The model is partially aggregated and is performed by several edge hosts in HierFavg systems [54]. Thusly, the model is prepared quickly, and better correspondence calculation trade has been accomplished. In the FL system's local update step, MGD has been employed in the presented MFL [55]. MFL's global convergence features have been established and the upper limit of the convergence rate has been estimated. MFL described such parameters that can increase the convergence rate as compared to FL. For IAI, Meng et al. [56] suggested a federated learning (PEFL) approach that is both effective and privacy protected. If numerous parties conspire data cannot be disclosed to others because PEFL is not interactive. In [57] to mutually enhance the convergence rate and training loss, a communication-efficient (CE) FL system is presented. the convergence rate is enhanced by the probabilistic technique of selecting a device. A reduced number of model parameters transferred between nodes can minimize the convergence rate because of the proposed quantization approach. Runhua et al. [58] proposed a technique for the privacy-preservation of FL named as HybridAlpha which employs secure multiparty computation (SMC). By using SMC with HybridAlpha caused a reduction in training time and volume of data transmission.

3. Background Preliminaries

3.1. Federated Learning

Federated Learning (FL) is a machine learning (ML) approach where training is performed in a decentralized environment. FL is a technique of training central models using decentralized data. They are for the user, have low latencies, and protect the user's privacy. In federated learning, local data from clients train the global model. without sharing the data of clients. In its most basic version, the FL architecture comprises a director or server who resides at the heart and organizes instructive events [18]. The majority of clients are edge devices, which may count in the millions. These devices connect to the server no less than two times in every training phase. The server initially transmits weights of the existing global model to clients individually, then the local data of clients trains the global model and submits the updated parameters to the aggregator (server).

3.2. FLaaS

FL "as a Service", or FLaaS, is a system that enables various scenarios of collaborative model building by third-party applications. It also addresses the challenges that result from these scenarios, including management of permissions and privacy, usability, and hierarchical model training. It is possible to implement FLaaS in a variety of different operating contexts. The purpose of FLaaS is to provide individual apps with a straightforward method of utilizing FL, even without the time-consuming process of building and fine-tuning the algorithms, and to enable a huge number of applications to collaborate that produce models with the least amount of work that is possible. Specifically, the goal of FLaaS is to reduce the amount of work that is required to produce a model to the minimum amount possible.

3.3. Edge Computing

A form of distributed computing that incorporates intelligence into edge devices, known as edge nodes, permits data to be analyzed and processed in real time near to the origin of the data collection is called edge computing. Edge nodes are positioned right before the network's last mile, often referred to as "downstream." These nodes often have a high processing capacity and are able to route data throughout the network. Small data centers, base stations, gateways,

and converters are all examples of what might fall under this category. In edge computing data is not instantly uploaded to the centralized processing system or cloud.

3.4. Cohorts

There is a term being used Federated Learning of Cohorts (FLoC). Researchers say that FLoC is an interest-based selection proposing new privacy-preserving mechanism and its main purpose is to gather large groups of people having similar interests in contents and ads for achieving current interests in society. A sort of online monitoring is Federated Learning of Cohorts (FLoC). It divides users into "cohorts" depending on their surfing history in order to target them with attention advertising.

3.5. Kernel Principal Component Analysis (KPCA)

A nonlinear method for reducing dimensionality is Kernel Principal Component Analysis (KPCA). This is an enhancement to Principal Component Analysis (PCA), which is really a linear dimensional reduction methodology. The KPCA method operates on the presumption that a number of datasets, which cannot be differentiated from one another in the existing area, may be made differentiable by moving them to a more expansive space. The sole thing that led to the formation of the extra dimensions was the application of fundamental arithmetic operations to the dimensions of the raw data.

4. System Model

Our approach works by adding KPCA to the defense against label-flipping attacks. KPCA is a kernel principal component analysis that extracts values through eigenvectors and eigenvalues and has been used to detect and defend against label-flipping attacks [40]. Cohorts have been used in [39] to improve the performance of the FL system by dividing the client's tasks into cohorts, but no security mechanism has been used to provide security to the client's data. In this paper, we have combined cohorts with KPCA to achieve efficiency and security at the same time in FLaaS. TC-KPCA is a type-based cohort with kernel principal component analysis that defends specifically against label-flipping attacks. We have also used multipath service routing (MSR) [38] in our system for the efficient routing of clients' data. Our proposed scheme TC-KPCA is achieving better accuracy than the previous FLaaS [2], and that is increased by 8.33%. Execution time decreased by 82.7%. Request's average waiting time and SLA violations decreased by 22.2% and 48.9%, respectively.

The model in Figure 2 describes the working framework for the proposed model. Firstly, clients register themselves and try to connect to the edge nodes by sending a request, and KPCA is performed in order to detect malicious clients. Cohorts are made and sent to the server for the training, and this process repeats until the training is completed. The aim is to provide a secure and efficient FLaaS. The integrity of data or the model itself can be breached if an attacker performs the label flipping attack [2], and too many requests can increase the congestion at the edge computing nodes. Providing a TC-KPCA could be helpful for detection and defending against the aforesaid attack and MSR can deal with the congestion and can increase the efficiency at the edge nodes. KPCA has been used to extract values of all the rounds for comparison in order to detect malicious clients. The performance of the proposed system will be evaluated through information loss, accuracy, execution time, memory consumption, service request violations, and the request's waiting time. Firstly, the clients who want to use their own data to train their model training are performed in a collaborative manner. Multiple applications of the mobile client can train the model, every client has a task that is of a specific type and sends a request to the edge server for a specific server. Clients send their requests to the aggregator over the edge server for the efficient routing of requests, there is more than one server available for services. Requests are sent to the edge server based on the pre-computed values. We have used two parameters to evaluate which server is best for routing first is the response time and the second is the availability of the link. The request is sent to the server having a low response time and the best available link. Our goal is to provide the best routing for the requests which are to be routed. The server responds to the client's requests and when every participant sends a request to the same server having a low response time and best link congestion occurs at that server that can cause service request violations and increased latency delay. To deal with that we have proposed a Multi-path Service Routing (MSR) which can solve this problem of congestion. When congestion reaches a certain level at the edge server requests are redirected to the other edge servers can that provide the requested service.

Secondly, when requests reach the edge server TC-KPCA is performed based on user data for the identification of the aforesaid attack by comparing the global model updates of the previous round with the updated local model

of the current round, and then applying standardization and dimensionality reduction on the data. A malicious client can be detected by using the proposed TC-KPCA. If any of the clients have changed the labels of data, the updates are ignored or that client is blocked for the next round. After the identification of malicious clients, the tasks are then assigned to cohorts based on the similarity of tasks and then tasks are transmitted to the aggregator in preparation for the final aggregation of updates. KPCA takes values in form of samples and features of those samples in the matrix and then calculates eigenvalues and eigenvectors. Those values are then compared in each round in order to identify from aforesaid attack. On the server side, when the requests are received in the form of cohorts from all the edge server's tasks, all the cohorts are extracted and again cohorts are formed on all tasks. The same type of tasks are added to one cohort and other types of tasks are added to other cohorts and then the cohort is loaded for the execution and federated aggregation is performed on all the cohorts in a sequence and then the updated model is returned to the edge servers and servers send the updated model to the respective clients so that they can continue for the next iteration. This process is performed in certain iterations until the whole model is updated.

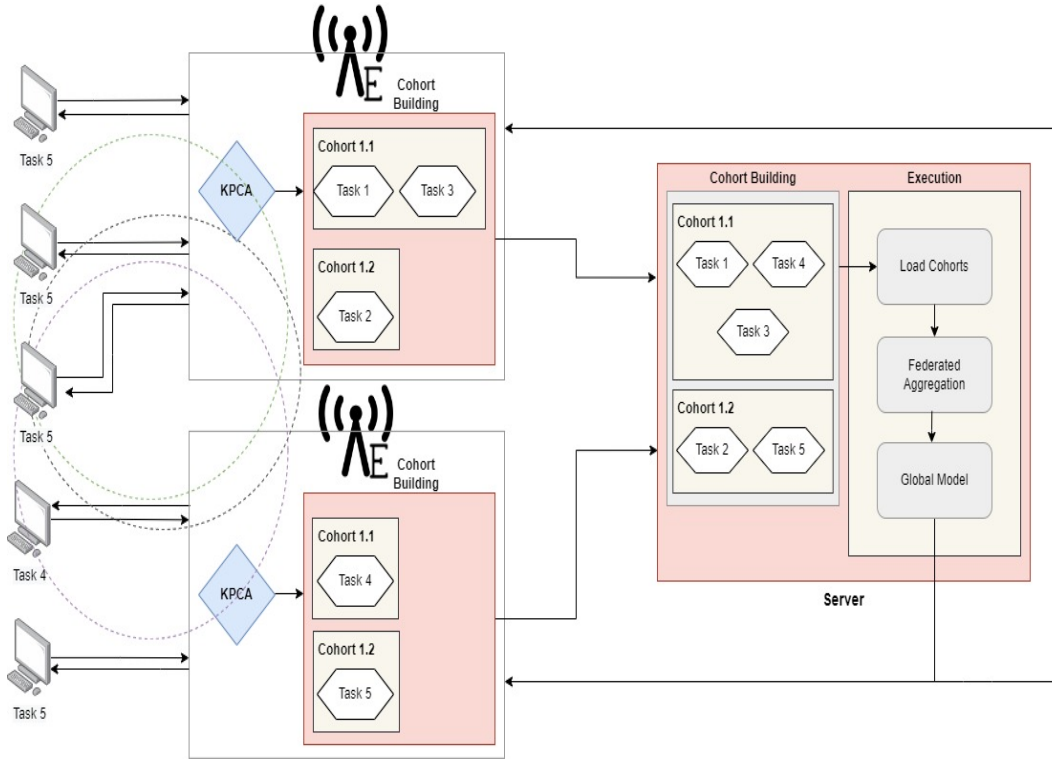


Figure 2: Working of the Proposed Framework

A request $REQ_{\epsilon_i}^{\rho_i}$ is generated by the users where ρ_i is the participant requesting for the specific service ϵ_i , and a request is sent to server based on two parameters (δ, α) which is response time and link availability. Equation 1 shows response time could be evaluated as follows:

$$\partial_{\epsilon_i}^{\rho_i, \delta_n} = \frac{e_{mi}}{\rho_{\delta_n}} + \frac{d^{\rho_i, \delta_n}}{r^{\rho_i, \delta_n}} + w_{\delta_n} \quad (1)$$

Where e_{mi} is requirement for execution, ρ_{δ_n} is processing speed, d^{ρ_i, δ_n} represents data amount, r^{ρ_i, δ_n} is transmission rate of data, and waiting time is w_{δ_n} .

Equation 2 shows link availability $\alpha_{\epsilon_i}^{\rho_i, \delta_n}$ could be obtained by getting $A\sigma(\rho_i, \delta_n)$ that is the up-time and downtime of the service link, if $c > 0$.

$$\alpha_{\epsilon_i}^{\rho_i, \delta_n} = A\sigma(\rho_i, \delta_n) = \frac{1}{c} \int_{q=0}^c A(q) dq \quad (2)$$

Table 1
List of Notations

Definition	Notations
participants	p_i
User Request	$REQ_{\epsilon_i}^{\rho_i}$
Available Servers	δ_n
Selected Server	$\pi_{\rho_i}^{\delta_n}$
Link Availability	$\alpha_{\epsilon_i}^{\rho_i, \delta_n}$
Response Time	$\partial_{\epsilon_i}^{\rho_i, \delta_n}$
Set of Vulnerable Rounds	R
Global parameters of model after training of previous round (r-1)	PJ_{r-1}
Updated parameters after training PJ_{r-1}	$PJ_{r,i}$
Creating Cohorts of model updates	CoH^{PJ}

Equation 3 shows that request $REQ_{\epsilon_i}^{\rho_i}$ can be aggregated as follows:

$$\pi_{\rho_i}^{\delta_n} = \left\{ \begin{array}{l} 0 \leq \text{Max} \partial_{\epsilon_i}^{\rho_i, \delta_n} \leq \partial \\ 0 \leq a \leq \text{Min}(\alpha_{\epsilon_i}^{\rho_i, \delta_n}) \end{array} \right\}_{\forall \rho_i \in X \wedge \forall \delta_n \in M} \quad (3)$$

where $\pi_{\rho_i}^{\delta_n}$ is selected on the basis of minimum or maximum $\alpha_{\epsilon_i}^{\rho_i, \delta_n}$ and $\partial_{\epsilon_i}^{\rho_i, \delta_n}$ from all the available servers δ_n .

5. System Design

This section will explain each component of the proposed model. Table 1 contains a list of notations that are employed in this research paper.

5.1. Client Registration

There are clients p_i that wish to train the global model using their own data that is available on their devices, and the training is carried out in a cooperative fashion. The model may be trained by several apps of the mobile client. Each client has a task t that is of a certain kind and submits a request $REQ_{\epsilon_i}^{\rho_i}$ to the edge server for a specific service ϵ_i . Because there are more than one server δ_n that may provide services, clients must submit their requests to the aggregator via the edge server in order to ensure that their requests are routed effectively. They send their requests to the edge nodes, and for that, the pre-calculated values are used to determine which request should be submitted to which edge server in order to reduce delays and congestion. The amount of time $\partial_{\epsilon_i}^{\rho_i, \delta_n}$ it takes for a server to respond is the first criterion we employed, and the availability $\alpha_{\epsilon_i}^{\rho_i, \delta_n}$ of the connection was the second. response time is computed through Equation 1, while link availability is computed via Equation 2. Both of these factors were taken into consideration. The request is then sent to the server Where $\pi_{\rho_i}^{\delta_n}$ that has the quickest response time and the most reliable connection available and calculated through Equation 3. Our objective is to offer the most efficient routing possible for the requests that need to be routed. When every participant submits a request to the same server with a short response time and best link congestion arises at that server, which may create service request violations and increased latency delay, the server responds to the requests of the clients. To address this issue, we have presented a Multi-path Service Routing, also known as MSR, which is capable of resolving the congestion issue. When the edge server hits a specific threshold level of congestion, requests are diverted to other edge servers in the network that are able to offer the service that is being requested. MSR will redirect the FL client requests to the other edge node by checking if the closest edge node is not available to respond to a specific task for training.

5.2. TC-KPCA

when requests reach the server at the edge of the network In order to identify the aforementioned attack using TC-KPCA on the user's data, first the data of the client is extracted through KPCA by extracting samples of data and then the dimensions of those samples and then they are stored in an $m \times n$ matrix and used to calculate kernel matrix via kernel formula Then, compute the kernel matrix's eigenvalue and eigenvector.

Arrange eigenvector in decreasing order, then do filtration on eigenvalues as well as on eigenvector by crossing every column in eigenvalue, eigenvectors are generated. Compute the project injector parameters of the centralized kernel to conclude the matrix formed through eigenvectors. Global updates of the model from the preceding round are compared with the updated local model from the current round, and then standardization and dimensionality reduction are applied to the data. This process is repeated until the attack has been identified. By using the TC-KPCA that has been presented, it is possible to identify a malicious client. If any of the clients have altered the data labels, the modifications are either disregarded or that client is prevented from participating in the subsequent round as shown in Figure 3. After the malicious clients have been identified, the tasks are next allocated to cohorts depending on the degree to which they are similar, and after that, the tasks are transmitted to the central server for the final aggregation of model updates to take place. KPCA begins by calculating eigenvalues and eigenvectors based on the values that are provided in the form of samples and the characteristics of those samples in the matrix. After that, in each round, the values are compared in order to determine whether or not the assault in question was successful. The steps that are involved at the edge nodes are shown in Figure 4.

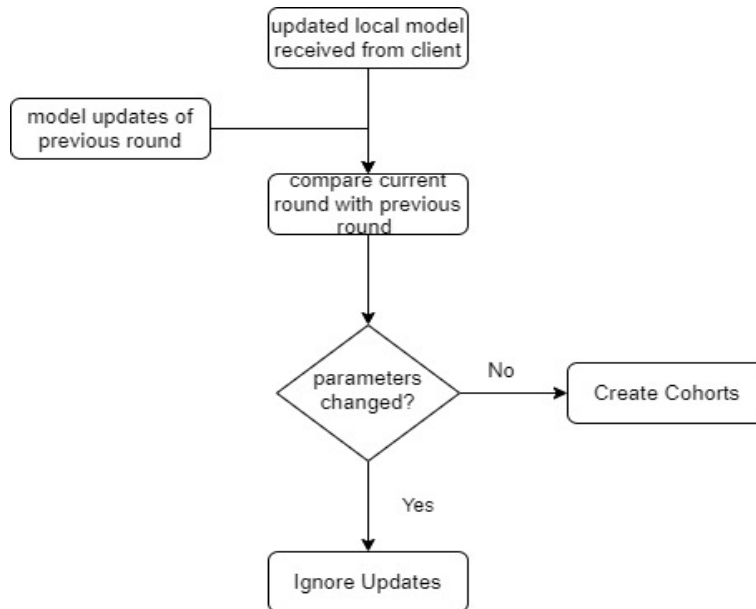


Figure 3: KPCA working

5.3. Cohorts Creation at the Server

On the server side, Figure 5 represents the process. When the requests come in the form of cohorts, the tasks on all of the edge server's tasks are analyzed to determine which cohorts should be extracted, and then new cohorts are produced on all of the tasks. The same kind of tasks is added to one cohort, while other kinds of tasks are added to other cohorts. After that, the cohort is loaded for execution, and then federated aggregation is performed on all of the cohorts in a sequence. Finally, the model is updated and sent back to the edge servers, which then transmit the revised model to the respective clients so that they can continue for the next iteration. This procedure is carried out in a series of iterations until the whole model has been brought up to date.

6. Experiments

The proposed approach is evaluated with respect to the following measures: execution time, memory consumption, information loss, accuracy, service request violations, and the request's waiting time. The information loss, accuracy and memory consumption metrics are used to evaluate the performance of our proposed approach. The information loss metric is used to verify the loss incurred by our proposed technique, while accuracy and memory consumption metrics are the same as in the base paper. Additional metrics (execution time, average waiting time, and SLA violations) have

1. $REQ_{\varepsilon_i}^{p_i}$
2. $\pi_{\delta_n}^{p_i}$
3. for $p \leftarrow 1$ to $|P|$ do
 - Populate $\pi_{\delta_n}^{p_i}$
4. end for
5. for all $\delta_n \in \pi_{\delta_n}^{p_i}$ do
6. calculate $\partial_{\varepsilon_i}^{p_i, \delta_n}$ and $\alpha_{\varepsilon_i}^{p_i, \delta_n}$
7. If $\left(\partial_{\varepsilon_i}^{p_i, \delta_n} \leq \partial \text{ where } \varepsilon_i = \varepsilon_i \right) \wedge \left(\alpha_{\varepsilon_i}^{p_i, \delta_n} \leq \alpha \text{ where } \varepsilon_i = \varepsilon_i \right)$ then
8. Update $\left(\pi_{\delta_n}^{p_i} \leftarrow \partial_{\varepsilon_i}^{p_i, \delta_n}, \alpha_{\varepsilon_i}^{p_i, \delta_n} \right)_{\delta_n = \delta_n}$
9. else continue
10. end if
11. end for
12. $REQ_{\varepsilon_i}^{p_i} \rightarrow Selc\pi_{\delta_n}^{p_i}$
13. If $\left(Selc\pi_{\delta_n}^{p_i} \right) = A$ then (A stands for selected server's availability)
14. for all $p \leftarrow 1$ to $|P|$
15. $PJ \leftarrow$ current GM (global model) parameter
16. for every $r \in R$ (R: unsafe rounds)
17. P_r (client queried for service)
18. PJ_{r-1} (GM of previous round)
19. for every $p_i \in p_r$ do
20. $PJ_{r,i}$ (updated parameters)
21. $PJ_{\nabla,i} = PJ_{r,i} - PJ_r$ (computation of delta between model update and global model by aggregator)
22. $PJ_{\nabla,i}^{src}$ (source class output connected to $PJ_{\nabla,i}$ parameters)
23. Add $PJ_{\nabla,i}^{src}$ to PJ
24. $PJ' =$ standardize (PJ)
25. $PJ'' =$ dimensionality_reduction (PJ')
- Create Cohorts (PJ'')**
26. Initialize U as $m \times n$ matrix for m tasks and n features
27. for $PJ_i \in P$
28. $r \leftarrow m^{n_i}$
29. add row r to U
30. end for
31. for feature $u \in U$ do
32. if $std(u) < \epsilon$ then
33. remove u from U
34. end if
35. end for
36. $K \leftarrow$ elbow (U)
37. $COH^{PJ} \leftarrow$ KMeans (U,k)
38. $COH^{PJ} \rightarrow$ FL server
39. end for
40. end for
41. end for
42. else
- $REQ_{\varepsilon_i}^{p_i} \rightarrow \delta_n$ where $\left(\partial_{\varepsilon_i}^{p_i, \delta_n} \leq \partial \text{ where } \varepsilon_i = \varepsilon_i \right) \wedge \left(\alpha_{\varepsilon_i}^{p_i, \delta_n} \leq \alpha \text{ where } \varepsilon_i = \varepsilon_i \right)$
43. repeat steps 14-42

Figure 4: Steps for Execution at Edge nodes

been added because we have combined Multi-path Service Routing with TC-kPCA. To evaluate MSR, we have added these three additional metrics.

6.1. Dataset Details

The CIFAR-10 dataset has been used in this paper for evaluating the proposed framework. It includes 60,000 color images at a 32 by 32-pixel resolution, arranged among 10 categories with 6,000 pictures in each category. The training uses 50,000 photos and the testing uses 10,000 images. The data of the dataset is separated into six training batches and one test batch, with ten thousand pictures in each training batch and one thousand in the test batch. The sample-set includes precisely 1000 photos taken at random from each category. The remaining photos are placed in a random sequence inside the training batches. However, some training batches may have a greater number of examples from one category than from another. The training batches have precisely 5000 examples from each category. Together, they make up the training set.

```

1. Input:  $COH^{Pj}$  received from edge nodes
2.  $W \leftarrow T_n$  ( $T_n$ : tasks extracted from cohorts)
3. Initialize  $U$  as  $m \times n$  matrix for  $m$  tasks and  $n$  features
4.   for  $W^{pj} \in P$ 
5.      $r \leftarrow n^{m_i}$ 
6.     add row  $r$  to  $U$ 
7.   end for
8.   for feature  $u \in U$  do
9.     if  $\text{std}(u) < \epsilon$  then
10.      remove  $u$  from  $U$ 
11.    end if
12.  end for
13.  $K \leftarrow \text{elbow}(U)$ 
14.  $COH^{w^{pj}} \leftarrow \text{KMeans}(U, k)$ 
15. for  $coh_i \in COH^{w^{pj}}$  do
16. initialize  $k_0^{t_j}$  for all tasks  $t_j \in coh_i$ 
17.   for  $r = 1$  do (where  $r \in R$ )
18.     for task  $t_j \in coh_i$  do
19.        $k_{r+1}^{coh_i} \leftarrow \text{UpdateClient}(t_j, k_r^{t_j})$ 
20.     end for
21.      $k_{r+1}^{coh_i} \leftarrow \frac{1}{|coh_i|} \sum_{i=1}^{|coh_i|} k_{r+1}^{t_j}$ 
22.   end for
23. sending  $k_R^{coh_i}$  and validating to all task  $T_i$  of participants  $p_i \in coh_i$  (sending it back to the edge servers)
24. end for
25. UpdateClient ( $t, k$ )
26.  $B \leftarrow (Y^r$  split tasks into  $B$  size Batches)
27. for epoch  $e=1$  do (where  $e \in E$ )
28.   for  $b \in B$  do
29.      $k \leftarrow k - \alpha \nabla_l(k, b)$ 
30.   end for
31. end for
32. return  $k$  to server
    
```

Figure 5: Steps for Execution at Server-side

6.2. Results

In this Section, we evaluate our model using various metrics. We have also compared our work the state of the art models.

6.2.1. Information Loss

Figure 6 depicts the information loss of the proposed scheme. It is shown that information loss is in direct opposition to FL rounds with an increase in FL rounds the information loss decreases. loss is calculated via Equation 4. Let A be the raw data and B be the data that has been compressed or altered. According to this definition, A and B 's mutual information is:

$$I(A;B) = H(A) - H(A|B) \quad (4)$$

where $H(A)$ is the entropy of A and $H(A|B)$ is the entropy of A under the condition that B exists. The mutual information calculates how much less uncertain A is as a result of knowing B .

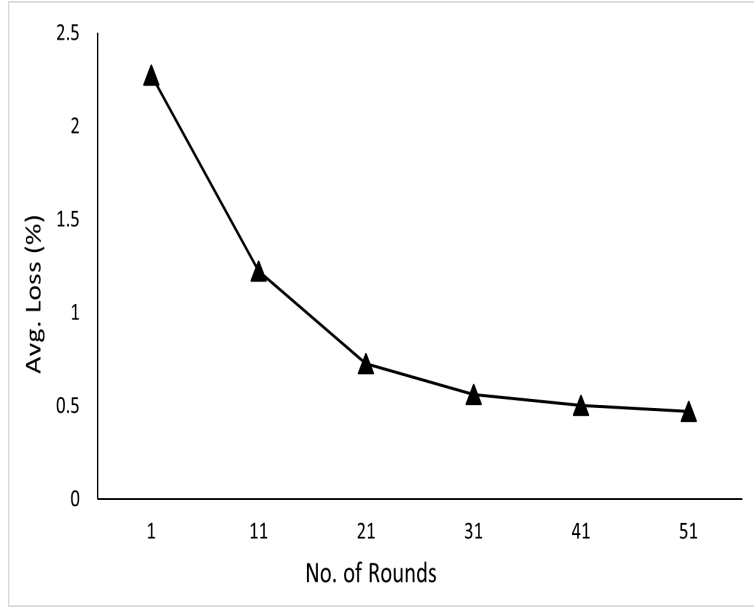


Figure 6: Average Information Loss

We executed the Equation 5 to determine the information loss as a percentage.

$$\text{Information Loss} = 1 - \frac{I(A; B)}{H(A)} * 100\% \quad (5)$$

It was 2.27% at round 1 and begin to decrease with the increment of FL rounds, eventually reaching 0.49%. This demonstrates the efficacy of the suggested model.

6.2.2. Accuracy

The test accuracy is shown in Figure 7 for both the proposed scheme and the previous scheme, along with each FL round. Test accuracy has been calculated via Equation 6.

$$\text{Test Accuracy} = \frac{\text{Number of Correctly Classified Samples}}{\text{Total Number of Samples}} \quad (6)$$

These findings, which were obtained by employing 50 FL rounds and 10 epochs each round, demonstrate that the proposed framework provides more accuracy than the FLaaS that was previously used. The accuracy increased by 8.33%, and the previous FLaaS accuracy was 0.44% at round 1 and ended at 0.72%. The proposed system's accuracy was 0.48% at round 1 and started increasing with the increment of FL rounds, ending at 0.78%.

6.2.3. Execution Time

Figure 8 indicates the amount of time spent on execution for each FL round. Execution time has been calculated via Equation 7.

$$\text{Execution time} = \text{End time} - \text{Start time} \quad (7)$$

Because of the efficient way in which service requests are routed, the New Scheme's operation may be completed in less time than the old schemes. The execution time decreased by 82%, and the previous FLaaS execution time was 0.81 seconds in round 1 and ended at 0.93 seconds. The proposed system's execution time was 0.16 seconds in round 1 and then it fluctuated in the FL rounds and ended at 0.16 seconds.

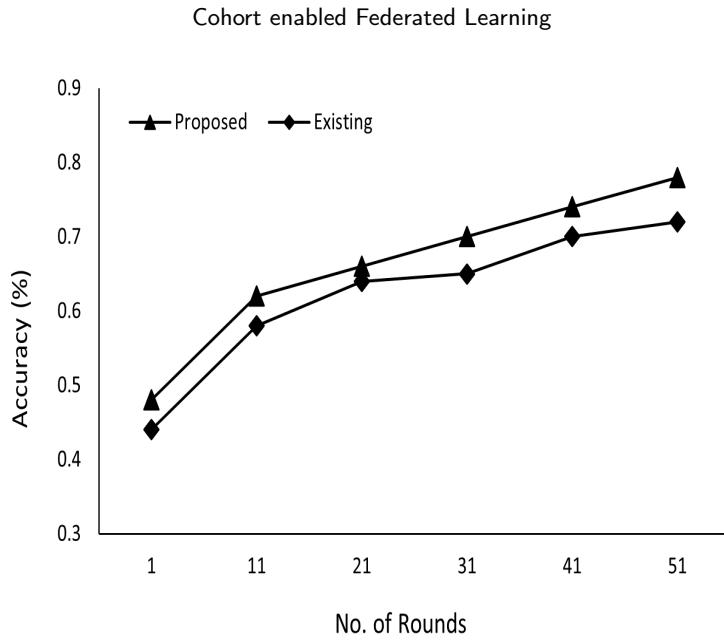


Figure 7: Test Accuracy

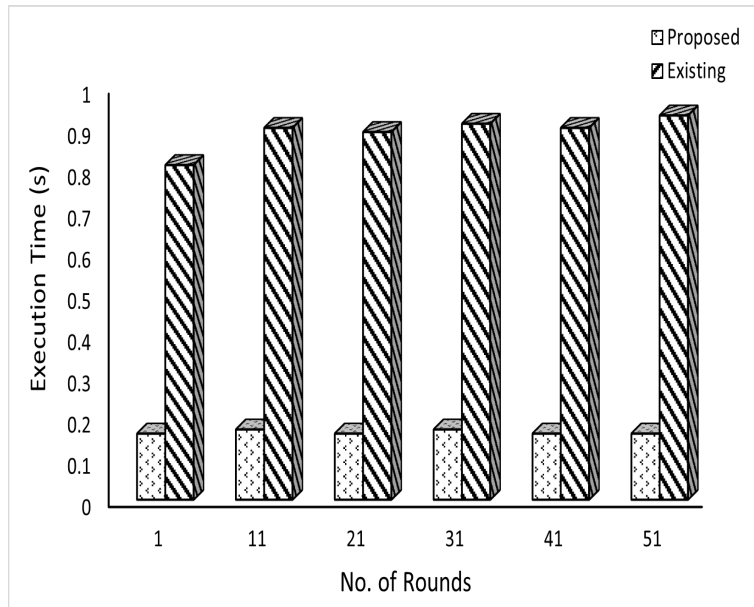


Figure 8: Average Execution Time

6.2.4. Request Waiting Time

In Figure 9, the typical amount of waiting time for each F L round is shown for both the new MSR and the old method. Average waiting time has been calculated via Equation 8.

$$\text{Average Waiting Time} = \frac{\text{Total Wait Time of Requests}}{\text{Number of Requests}} \quad (8)$$

As can be seen in the figure, the amount of waiting time grows proportionately with the number of requests made in each scenario; nevertheless, the waiting time experienced while using MSR is much shorter than when using the

Cohort enabled Federated Learning

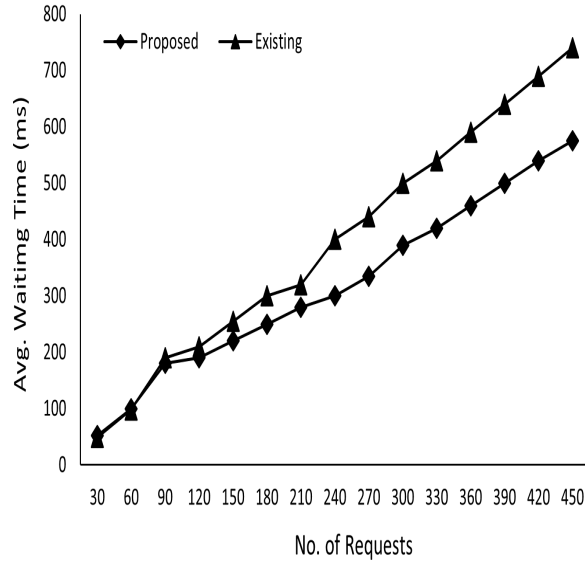


Figure 9: Average Waiting Time

conventional method. The average waiting time of the previous FLaaS was 49 milliseconds in round 1 and ended at 740 milliseconds. The proposed system’s average waiting time was 52 milliseconds in round 1 and then in round 50, it ended at 575 milliseconds.

6.2.5. SLA Violations

Figure 10 depicts our analysis of SLA violations using MSR and the Greedy method. Any service provider’s purpose is to supply the service. Because there are fewer users and a limited number of requests they may create, the percentage of SLA violations is not significant. SLA violations have been calculated via Equation 9.

$$\text{SLAViolations} = \text{Total Number of Requests} - \text{Number of Successful Requests} \quad (9)$$

The SLA violations of the previous FLaaS were 0.42% in round 1 and ended at 0.47%. The proposed system’s violations were 0.0% in round 1 and then in round 50, it ended at 0.24%. However, the statistics show that when the amount of service requests increases, so does the frequency of SLA breaches. Because there are fewer servers than service requests, the aforementioned is true. The MSR appears to do well than that of the Greedy method, and the percentage of creep in MSR violations is much lower than that of the Greedy method.

6.2.6. Memory Consumption

The amount of memory used by each FL round is shown in Figure 11. Memory consumption has been calculated via Equation 10.

$$\text{Memory Consumption} = \text{Number of Parameters or Gradients} * \text{Size of Each Parameter or Gradient} \quad (10)$$

The memory consumed by the previous FLaaS was 1.47 MBs in round 1 then it fluctuated a little bit and ended at 1.48 MBs. The proposed system’s consumed memory was 2.47 MBs in round 1 and then it fluctuated a little bit and in round 50, it ended at 2.48 MBs. The memory requirements of the new Scheme are higher than those of the current Scheme due to the increased amount of storage space required by cohorts. 250 samples per round were used in the previous scheme; on the other hand, only 50 samples were used in the proposed scheme, but if we compare, the memory consumed in the proposed scheme is higher than that of the existing scheme.

6.2.7. Discussion

Cohorts have been used in combination with KPCA in order to achieve security and effectiveness in the federated learning environment. TC-KPCA increased the performance and security of the system by detecting malicious clients

Cohort enabled Federated Learning

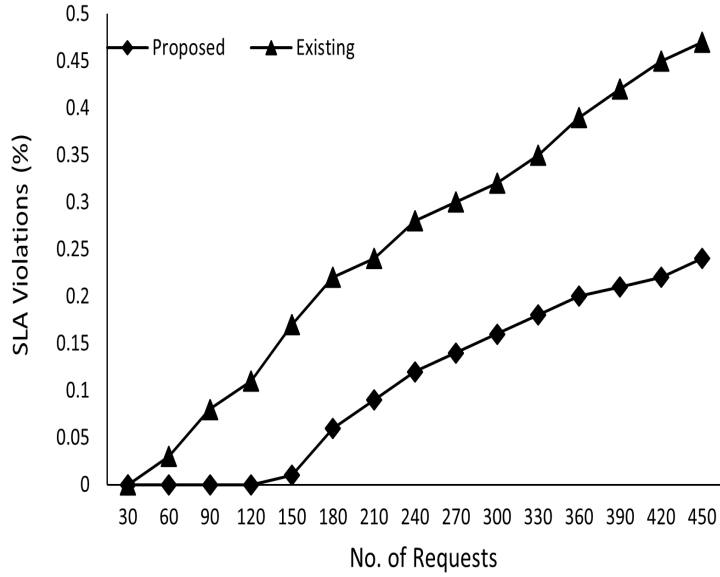


Figure 10: Comparative Analysis of SLA Violations

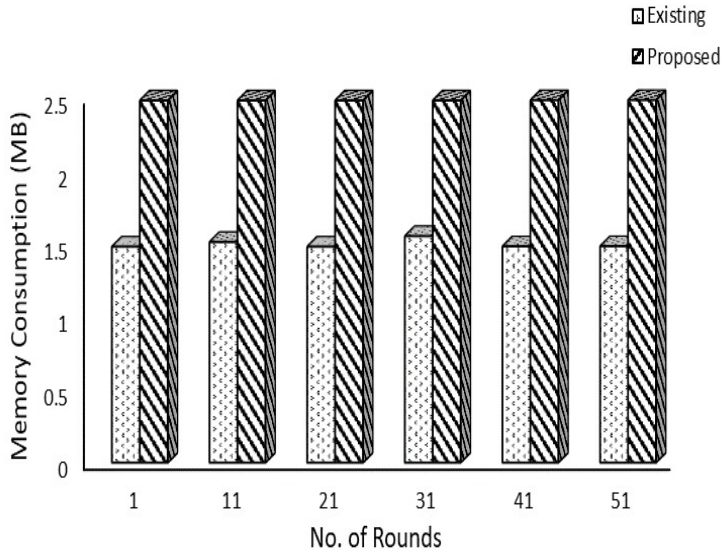


Figure 11: Average Memory Consumption

and blocking them from the network to maximize accuracy, as compared to the [2] nodes, which were compromised due to non-secure mechanisms being used while training and a few requests were being delayed and dropped due to the increased number of service requests at the edge nodes. To contrast, the proposed scheme reduced the delay time of a request and decreased congestion at the edge nodes, which were being used as a medium to send the client's data to the FL server. Hence, this improved the overall efficiency of the whole system. Table 2 shows comparison between proposed and old scheme.

Table 2
Comparison with state-of-the art method

FL Rounds	Accuracy (Old Scheme)%	Memory Consumption (Old Scheme)MB	Accuracy (Proposed Scheme)%	Memory Consumption (Proposed Scheme)MB
1	0.44	1.478	0.48	2.478
11	0.58	1.510	0.62	2.478
21	0.64	1.478	0.66	2.478
31	0.65	1.550	0.70	2.478
41	0.70	1.480	0.74	2.480
50	0.72	1.481	0.78	2.481

7. Conclusion

A Cohort-based kernel principal component analysis was proposed in FLaaS. The efficient and secure framework was proposed based on TC-KPCA, in the given thesis work. An efficient MSR-based solution is provided for the efficient routing of service requests for edge nodes. TC-KPCA provides security against model poisoning attacks and specifically against label-flipping attacks. Security is a major concern for communication in the digital environment which is somehow hostile to different adversarial attacks. So, secrecy is the main function to protect the training data among different FL clients. For this purpose, many protocols and schemes have been developed to provide security for training data. Here we propose a protocol that provides a secure environment for the IoT devices (clients), edge nodes, and FL server. The client's devices are hostile to many attacks to get training information. An efficient and secure mechanism based on type-based cohorts with the combination of kernel principal component analysis and multipath service routing is used to achieve accuracy and reliability for the routing of requests for clients' devices. It is resistant to different model poisoning and data poisoning attacks to provide security during the training phase. For future work, we are looking to minimize the computation overhead and memory consumption caused by clustering algorithms used in cohort building.

Acknowledgment

This research has been supported by the Estonian Research Council grant PRG 1780.

References

- [1] L. Li, Y. Fan, M. Tse and K. Lin, "A review of applications in federated learning", *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020. Available: 10.1016/j.cie.2020.106854 [Accessed 16 February 2022].
- [2] N. Kourtellis, K. Katevas and D. Perino, "FLaaS", *Proceedings of the 1st Workshop on Distributed Machine Learning*, 2020. Available: 10.1145/3426745.3431337 [Accessed 14 February 2022].
- [3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273– 1282.
- [4] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *ICLR*, 2018.
- [5] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "Fedvision: An online visual object detection platform powered by federated learning," in *IAAI*, 2020.
- [6] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *CoRR*, arXiv:1711.10677, 2017.
- [7] C. Wu, F. Wu, R. Liu, L. Lyu, Y. Huang, and X. Xie, "Fedkd: Communication efficient federated learning via knowledge distillation," *arXiv preprint arXiv:2108.13323*, 2021.

- [8] C. Wu, F. Wu, L. Lyu, T. Di, Y. Huang, and X. Xie, “Fedctr: Federated native ad ctr prediction with multi-platform user behavior data,” arXiv preprint arXiv:2007.12135, 2020.
- [9] J. Cui, C. Chen, L. Lyu, C. Yang, and W. Li, “Exploiting data sparsity in secure cross-platform social recommendation,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [10] J. Li, L. Lyu, X. Liu, X. Zhang, and X. Lv, “Fleam: A federated learning empowered architecture to mitigate ddos in industrial iot,” *IEEE Transactions on Industrial Informatics*, 2021.
- [11] C. Wu, F. Wu, Y. Cao, L. Lyu, Y. Huang, and X. Xie, “Fedgmn: Federated graph neural network for privacy-preserving recommendation,” arXiv preprint arXiv:2102.04925, 2021.
- [12] J. Zhou, C. Chen, L. Zheng, X. Zheng, B. Wu, L. Lyu, Z. Liu, and L. Wang, “Privacy-preserving graph neural network for node classification,” arXiv e-prints, pp. arXiv–2005, 2020.
- [13] X. Ni, X. Xu, L. Lyu, C. Meng, and W. Wang, “A vertical federated learning framework for graph convolutional network,” arXiv preprint arXiv:2106.11593, 2021.
- [14] T. Hiessl, S. Lakani, J. Kemnitz, D. Schall and S. Schulte, “Cohort-based federated Learning Services for industrial collaboration on the edge,” 2021.
- [15] L. Lyu, H. Yu, X. Ma, L. Sun, J. Zhao, Q. Yang, and P. Yu “Privacy and robustness in federated learning: Attacks and defenses,” arXiv [cs.CR], 2020.
- [16] P. Liu, X. Xu and W. Wang, "Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives", *Cybersecurity*, vol. 5, no. 1, 2022. Available: 10.1186/s42400-021-00105-6 [Accessed 16 February 2022].
- [17] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, “Protection against reconstruction and its applications in private federated learning,” CoRR, arXiv:1812.00984, 2018.
- [18] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” in *SP*, 2019, pp. 691–706.
- [19] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *NeurIPS*, 2019, pp. 14 747–14 756.
- [20] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, “cpsgd: Communication-efficient and differentially-private distributed sgd,” in *NeurIPS*, 2018, pp. 7564–7575.
- [21] Y. Aono, T. Hayashi, L. Wang, S. Moriai et al., “Privacy-preserving deep learning via additively homomorphic encryption,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333– 1345, 2018.
- [22] B. Zhao, K. R. Mopuri, and H. Bilen, “idlg: Improved deep leakage from gradients,” CoRR, arXiv:2001.02610, 2020.
- [23] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.- y. Sohn, K. Lee, and D. Papailiopoulos, “Attack of the tails: Yes, you really can backdoor federated learning,” *NeurIPS*, 2020.
- [24] C. Xie, K. Huang, P.Chen, and B. Li,” Dba: Distributed backdoor attacks against federated learning”, In *International Conference on Learning Representations*, 2019.
- [25] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” CoRR, arXiv:1807.00459, 2018.
- [26] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, “Analyzing federated learning through an adversarial lens,” CoRR, arXiv:1811.12470, 2018.
- [27] C. Fung, C. J. Yoon, and I. Beschastnikh, “The limitations of federated learning in sybil settings,” in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.

- [28] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" arXiv preprint arXiv:1911.07963, 2019.
- [29] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in ICCD, 2017, pp. 45–48.
- [30] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in NeurIPS, 2018, pp. 8000–8010.
- [31] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in ECCV. Springer, 2020, pp. 182–199.
- [32] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," arXiv preprint arXiv:2101.05930, 2021.
- [33] Y. Li, X. Lyu, N. Korean, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," Advances in Neural Information Processing Systems, vol. 34, 2021.
- [34] P. Blanchard, R. Guerraoui, J. Stainer et al., "Machine learning with adversaries: Byzantine tolerant gradient descent," in NeurIPS, 2017, pp. 119–129.
- [35] G. Damaskinos, E. M. El Mhamdi, R. Guerraoui, A. H. A. Guirguis, and S. L. A. Rouault, "Aggregathor: Byzantine machine learning via robust gradient aggregation," in SysML, 2019.
- [36] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 1, no. 2, p. 44, 2017.
- [37] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantineroobust distributed learning: Towards optimal statistical rates," CoRR, arXiv:1803.01498, 2018.
- [38] S. Malik, T. Kanwal, S. Khan, H. Malik and H. Pervaiz, "A User-Centric QoS-Aware Multi-Path Service Provisioning in Mobile Edge Computing", IEEE Access, vol. 9, pp. 56020-56030, 2021. Available: 10.1109/access.2021.3070104 [Accessed 14 February 2022].
- [39] T. Hiessl, "Cohort-based federated Learning Services for industrial collaboration on the edge," 2021.
- [40] D. Li, W. E. Wong, W. Wang, Y. Yao and M. Chau, "Detection and Mitigation of Label-Flipping Attacks in Federated Learning Systems with KPCA and K-Means," 2021 8th International Conference on Dependable Systems and Their Applications (DSA), 2021, pp. 551-559, doi: 10.1109/DSA52907.2021.00081.
- [41] Z. Ma, J. Ma, Y. Miao, X. Liu, K. -K. R. Choo and R. Deng, "Pocket Diagnosis: Secure Federated Learning against Poisoning Attack in the Cloud," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2021.3090771.
- [42] M. Malekzadeh, B. Hasircioglu, N. Mital, K. Katarya, M. E. Ozfatura, and D. Gündüz, "Dopamine: Differentially private federated learning on medical data," arXiv [cs.LG], 2021.
- [43] S. Truex, L. Liu, K.H. Chow, M. E. Gursoy, and W. Wei, "LDP-Fed: Federated learning with local differential privacy," in Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, 2020.
- [44] J. So, B. Güler and A. S. Avestimehr, "Byzantine-Resilient Secure Federated Learning," in IEEE Journal on Selected Areas in Communications, vol. 39, no. 7, pp. 2168-2181, July 2021, doi: 10.1109/JSAC.2020.3041404.
- [45] G. Xu, H. Li, S. Liu, K. Yang and X. Lin, "VerifyNet: Secure and Verifiable Federated Learning," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 911-926, 2020, doi: 10.1109/TIFS.2019.2929409.
- [46] X. Cao, J. Jia, and N. Z. Gong, "Provably secure federated learning against malicious clients," arXiv [cs.CR], 2021.

- [47] Y. Liu, J. Peng, J. Kang, A. M. Ilyasu, D. Niyato and A. A. A. El-Latif, "A Secure Federated Learning Framework for 5G Networks," in *IEEE Wireless Communications*, vol. 27, no. 4, pp. 24-31, August 2020, doi: 10.1109/MWC.01.1900525.
- [48] L. Burkhalter, H. Lycklama, A. Viand, N. Küchler, and A. Hithnawi, "RoFL: Attestable robustness for secure federated learning," arXiv [cs.CR], 2021.
- [49] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin and V.Chandra, "Federated Learning with Non-IID Data," arXiv [cs.CR], 2018.
- [50] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in 2021 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [51] J. Konecny, H. B. McMahan, F. X. Yu, A.T. Suresh and D. Bacon, "FEDERATED LEARNING: STRATEGIES FOR IMPROVING COMMUNICATION EFFICIENCY," arXiv [cs. CR], 2017.
- [52] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," arXiv [cs.LG], 2019.
- [53] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," arXiv [cs.LG], 2020.
- [54] L. Liu, J. Zhang, S. H. Song and K. B. Letaief, "Client-Edge-Cloud Hierarchical Federated Learning," *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9148862.
- [55] W. Liu, L. Chen, Y. Chen and W. Zhang, "Accelerating Federated Learning via Momentum Gradient Descent," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754-1766, 1 Aug. 2020, doi: 10.1109/TPDS.2020.2975189.
- [56] M. Hao, H. Li, X. Luo, G. Xu, H. Yang and S. Liu, "Efficient and Privacy-Enhanced Federated Learning for Industrial Artificial Intelligence," in *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532-6542, Oct. 2020, doi: 10.1109/TII.2019.2945367.
- [57] M. Chen, N. Shlezinger, H. Poor, Y. Eldar and S. Cui, "Communication-efficient federated learning", *Proceedings of the National Academy of Sciences*, vol. 118, no. 17, p. e2024789118, 2021. Available: 10.1073/pnas.2024789118 [Accessed 18 February 2022].
- [58] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar and H. Ludwig, "HybridAlpha", *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security - AISec'19*, 2019. Available: 10.1145/3338501.3357371 [Accessed 18 February 2022].