iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** IV **Month of publication:** April 2023

DOI: https://doi.org/10.22214/ijraset.2023.50826

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

Mobile Keyword Prediction using Federated Learning

Khushi Udaysingh Chouhan[1], Nikita Pradeep Kumar Jha[2], Roshni Sanjay Jha[3], Shaikh Insha Kamaluddin[4], Dr. Nupur Giri[5]
[1, 2, 3, 4, 5]*Department of Computer Science VESIT Mumbai*

Abstract: ***Federated learning is a decentralized form of Machine Learning in which data subsets are trained on several edge devices aggregated and brought to the centralized server. Here, federated applications store the local copy on all the edge devices such as smartphones where users can use it accordingly. The model gradually learns and trains itself from inputs by the user's virtual keyboard and becomes smarter iteratively.***
Devices transfer the results in the form of parameters to the centralized server where these results are aggregated with the help of federated algorithms. For the purpose of next word prediction using federated learning, we implemented different algorithms such as FedAvg, FedProx, FedSgd. In reference to the traditional ML models, data is collected in a centralized location and is used to train the model. However, unlike ML techniques the Federated Learning techniques give more control over user's data to themselves since the data is not shared with the central server or other devices. Thus, federated learning adheres to the data privacy and feasibility for its users.
Keywords: federated, decentralized, centralized server, FedAvg, FedProx, FedSgd

I. INTRODUCTION

Next-word prediction feature facilitates text entry and plays an important role in improving the user's experience. Today, all smart mobile devices are incorporated with virtual keyboards which support more than 600 languages and emoticons. The current mobile keyword prediction models are constrained in multiple ways, such as limited vocabulary, contextual awareness, personalization and multilingual support.
When it comes to the next word prediction feature, the users expect the visible response within a fraction of second. The users might get uptight about the collection and remote storage of their personal data despite use of data cleaning and strict access protocols as in case of the traditional neural networks or N-gram approaches.
This paper aims at showing a federated learning environment that encrypts the user-sensitive data and fixes privacy and latency issues with more feasibility.
Such an approach is referred to as the Secure aggregation Principle where the server is allowed to securely combine the encrypted results and decrypt only the aggregated results. It applies a federated averaging mechanism that plays a role in aggregating encrypted chunks of results from edge devices. The paper shows the comparative study of different federated algorithms to judge the accuracy rate, loss and speed.

II. DATASET AND RELATED WORK

For our work we have used the Shakespeare dataset. In this dataset there are approximately 337 sonnets by Shakespeare. We have used each sonnet as a user for our proposed system.
FSAs and FSTs have been implemented in the context of mobile keyboard's auto correction and predictions for the next word. LSTM models have also improved the decoding of gestures and provided better results. RNN and N-gram language models optimize the keyword prediction rate and keystroke savings with respect to time latency and memory constraints. With the tremendous increased focus on privacy and government regulation, research into distributed training for neural networks has gained significant momentum.
In particular, federated learning provides a useful extension to server based distributed training, client device based training and computations using locally stored data. Gboard, a mostly used virtual keyboard app, has previously used federated learning to train a model to suggest search engine queries based on typing and speed context.

©IJRASET: All Rights are Reserved

III. MODEL ARCHITECTURE

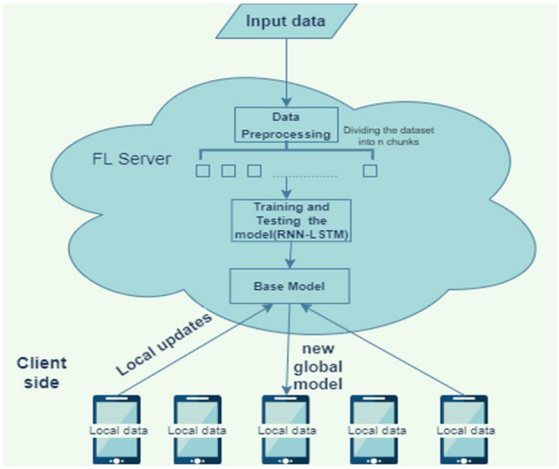


Fig 3.1 Flow of the proposed work

Federated Learning has two main components, the aggregator and remote training parties. The aggregator is the server which averages local updates using federated learning algorithms.Preprocessing of data takes place over local data. Training of models takes place at the server side where the server aggregates model updates from the mobile devices. The remote training parties are the edge devices and often referred to as clients that fetch the global model application from the server and transfer their inputs in the form of local updates to the server.

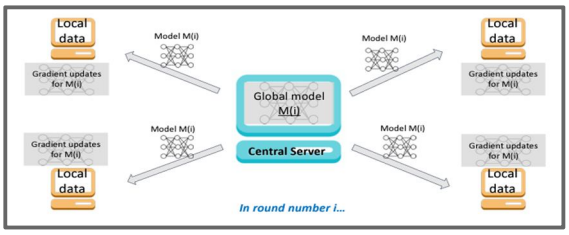


Fig 3.2 Round i in global model

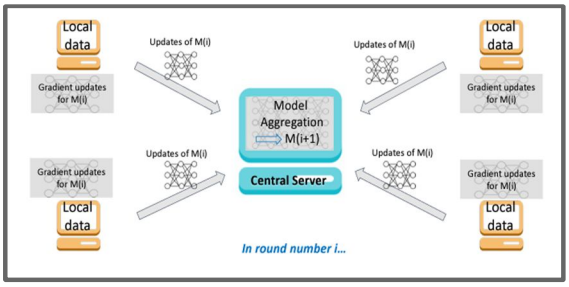


Fig 3.3 Model aggregation

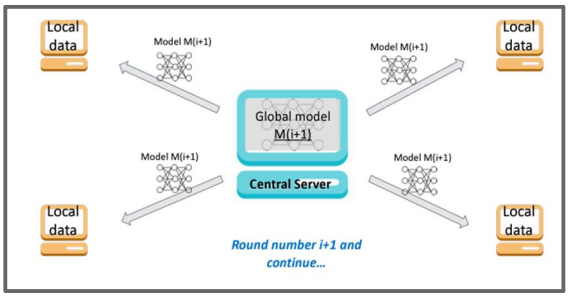


Fig 3.4 global model (i+1)

A typical federated environment has a high density of potentially hundreds of millions of client devices, however only a fraction of them are active and available at all time for training purposes. In general, a training round consists of only a subset of available devices or clients which is randomly selected. This is because it is impractical to coordinate the innumerable amount of clients at once. The initial model and the parameters required for training are distributed by the server to the subset of clients that will participate in a round of training and evaluation. On each client the model is iteratively invoked in an independent and parallel manner on a stream of local data batches to generate a new set of model parameters and a set of local metrics (also known as local aggregations). Thereby the model parameters and locally exported metrics are accumulated across the system by running a distributed aggregation protocol suite.

IV. OPEN SOURCE SYSTEM USED

A. *TensorFlow Federated or TFF framework*

TFF is an open source framework to build ML models on distributed and decentralized data. It allows users to train models using data distributed across multiple devices or servers while maintaining privacy and security. This is achieved by sending code to the device rather than sending the data to any central location, ensuring data stays on the device.

TFF is a high level API that aims at supporting a variety of distributed learning environments, one of which also includes a topic of our interest, Federated Learning. Two important concepts in TFF are 'tff.learning' and 'tff.computation'. The tff.learning package is a higher-level module that provides a set of high level APIs that act as building blocks for federated learning, while tff.computation is a core component of TFF that provides a way to define and execute computations on a distributed system. There are two phases in running a federated computation:

1) *Compile phase:* In this phase, TFF first compiles federated algorithms into an abstract serialized representation of the entire distributed computation.
2) *Execution phase:* In this phase, TFF provides ways to execute these computations via local simulation.

TFF is a functional programming environment, but many processes involved in federated learning are stateful. The state that changes from round to round contains the set of model parameters to be trained and possibly additional states associated with the optimizer. To standardize the processing of simulated federated datasets, TFF can enumerate a set of clients and create datasets containing data for specific clients that feed directly as input to the generated federated datasets. It also provides an abstract interface.

V. FEDERATED LEARNING

Federated learning offers a distribution strategy that can be employed to train a machine learning model. It is also a decentralized technique as it keeps training data localized on mobile devices and never collects them centrally. Mobile devices act as clients and generate big data that is used for training purposes. The client processes local data and exchanges the model updates with the server, rather than uploading data directly to the server as in case of centralized model training. The server aggregates the weights from the sample of clients to generate an improved and updated global model. The model is then sent to the client and this process is done iteratively. Various aggregation algorithms can be used on the server side to combine the client weights and update the model to produce a new, improved and accurate model.

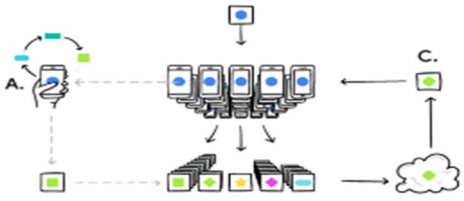


Fig 4.1 Process of federated learning

Despite when the server-hosted data is anonymised, this decentralized on-device processing technique offers some advantages in terms of security and privacy as compared to other server storage methods. Previous research shows that privacy-preserving methods like safe aggregation and differential privacy can both complement from federated learning.Users have direct and hands-on control over their data as the confidential information is kept on client devices only. Each client transmitted model changes are transitory, concentrated and aggregated. The client changes are handled in memory and are instantly deleted after updates become part of the weight vector. They are never saved on the server hence, preserving user privacy and providing secure aggregation.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

VI. METHODS

In this work we have used 3 different federated learning methodologies i.e. FedAvg, FedRep, FedProx. Each method differentiates from one another in terms of aggregation.

At each communication round, the central server selects a subset of clients (S ⊆ [N]) and sends global model information to the subset. Each client then updates the global model with its own local data. The client then sends the updated model back to the central server. The server aggregates and updates the global model based on input from clients. This process repeats several rounds of communication until a termination criterion is reached. B. Validation accuracy is met. A client's Di record with a superscript t represents the t^{th} communication between the central server and the selected client. where t ∈ {1,...,T}.

Input: Client datasets $\{D_i\}_{i=1}^{N}$, T, initialization for $\boldsymbol{w}$
for $t = 1, 2, \cdots T$ **do**
  Orchestrator selects a subset of clients $\mathcal{S} \subseteq [N]$, broadcasts global model $\boldsymbol{w}^t$, or a part of it, to clients in $\mathcal{S}$.
  for each $i \in \mathcal{S}$ **do**
    Clients update model parameters $\boldsymbol{w}_i^{t+1} = client_update\,(\boldsymbol{w}^t, D_i)$
    Clients send updated parameters $\boldsymbol{w}_i^{t+1}$ to server.
  end for
  Orchestrator updates $\boldsymbol{w}^t$ by aggregating client updates $\boldsymbol{w}^{t+1} = server_update\,(\{\boldsymbol{w}_i^t\})$
end for

Fig 5.1 General framework of learning model

A. *FedAvg*

Federated averaging, also known as FedAvg, is a simple and widely-used federated learning algorithm that involves simply averaging the model updates from the client devices to produce the new global model.

It has four hyperparameters; the fraction of clients C for each round, B the local minibatch size, E the number of times each client trains over the local dataset each round (epochs) and the learning rate. We assume that all sample devices complete e epochs and thus it drops stragglers. FedAvg selects clients consecutively prior to each epoch to steer clear of federations with low probability of participating clients. Additionally, the learning rate should be decreased when the objective function is optimized and as the model approaches the global minimum of the loss function to improve the model's accuracy and produce enhanced results.

Server executes:
  initialize w_0
  for each round $t = 1, 2, \dots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return w to server

Fig 5.2 FedAvg

B. *FedSGD*

FedSGD is the distributed version of SGD. It partitions the data across multiple computing nodes. Each node uses a single SGD step to compute the gradient from its local SGD for each round. The weights are then calculated and averaged over all nodes. As a data-parallelization approach, FedSGD utilizes the computation power of multiple compute nodes instead of one. This approach accelerates the normal SGD and has been widely used due to the growing size of datasets collected nowadays. Additionally, since FedSGD only executes one SGD step on a local node, averaging updated weights is equivalent to averaging gradients (η denotes steps size):

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

$$\mathbb{E}_i \left[\boldsymbol{w}^t - \eta \nabla F_i(\boldsymbol{w}^t) \right] = \boldsymbol{w}^t - \eta \mathbb{E}_i \left[\nabla F_i(\boldsymbol{w}^t) \right].$$

Key difference between FedAvg and FedSGD is how they handle Non-IID data. Non-IID refers to data without identical and independent distribution. FedAvg tends to be sensitive to the non-iid data, as it assumes that all client devices have the same data distribution. FedSGD , on the other hand, handles non-iid data by allowing each client device to take multiple steps of SGD on its own, which helps reduce impact of local data variations.

C. FedProx

FedProx is a generalization of FedAvg with some modifications to address heterogeneity of data and systems. The learning is again performed in rounds. At each round, the server samples a set of m clients and sends them the current global model. The FedProx algorithm works by adding a proximal term to the objective function that is being optimized. This term penalizes large deviations of the model parameters from the previous round, encouraging the model to converge to a more stable solution. The proximal term is controlled by a hyperparameter called the "proximal coefficient", which determines the strength of the penalty. Additionally, we perform the local optimization for a variable number of epochs according to the system resources.

Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
Server sends w^t to all chosen devices
Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
Each device $k \in S_t$ sends w_k^{t+1} back to the server
Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

Fig 5.3 FedProx

VII. RESULTS

A. Federated Learning simulation on Shakespeare Dataset

During the training of different FL models, it is observed that the FedProx converges faster and better than the FedAvg and FedSGD.

The partitioning of the Shakespeare dataset is given in the figure 7.1.1.

```
##########################################
DATASET: shakespeare
236 users
1132837 samples (total)
4800.16 samples per user (mean)
num_samples (std): 10943.10
num_samples (std/mean): 2.28
num_samples (skewness): 8.01
0          138
8000       11
10000      6
12000      7
14000      2
16000      4
18000      0
```

Fig 7.1.1 Shakespeare dataset partitioning

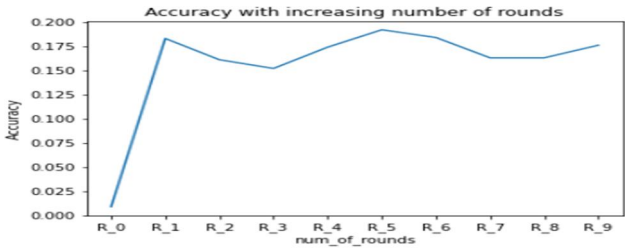


Fig 7.1.2 FedAvg Accuracy for 10 rounds

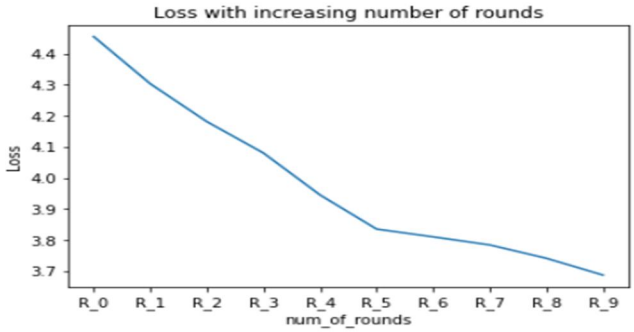


Fig 7.1.3 FedAvg Loss for 10 rounds

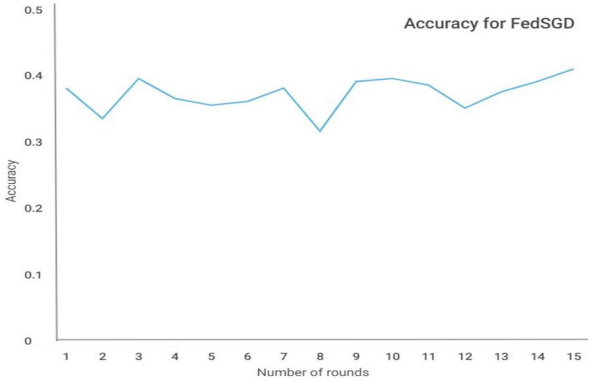


Fig 7.1.4 FedSGD Accuracy for 15 rounds

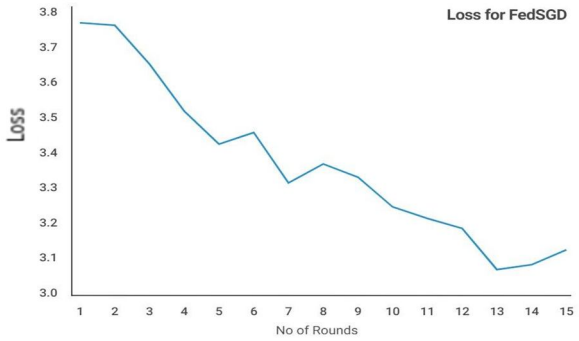


Fig 7.1.5 FedSGD Loss for 15 rounds

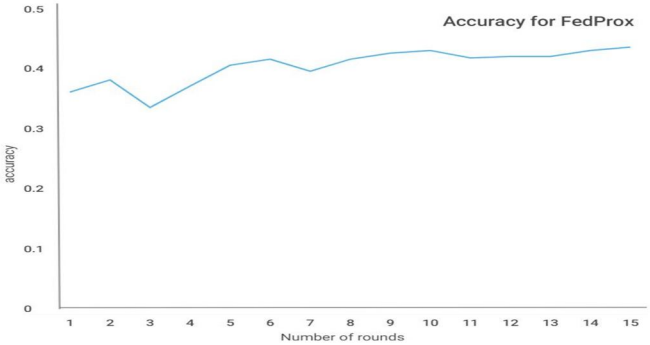


Fig 7.1.6 FedProx Accuracy for 15 rounds

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

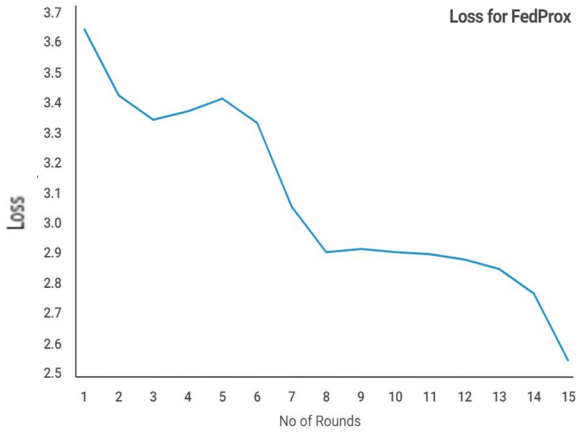


Fig 7.1.7FedProx Loss for 15 rounds

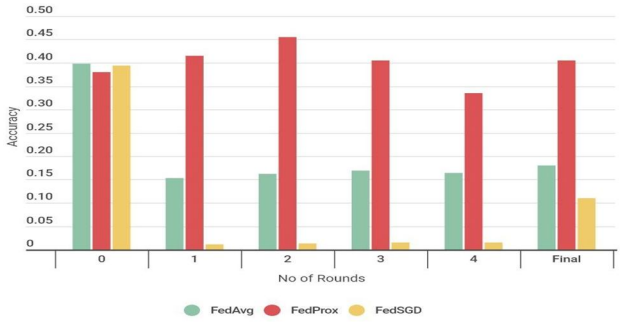


Fig 7.1.8 Comparison of accuracies for various models

With the help of *Fig 7.1.8* it can be verified that FedProx gives better accuracy as compared to FedAvg and FedSGD models. Also it can be observed that for a greater number of rounds the accuracy tends to reach a constant value and the loss tends to decrease continually.

For comparison purposes the learning rate was set to 0.001 for all the models.

VIII. CONCLUSION AND FUTURE SCOPE

The paper discusses Federated Learning and its models in detail. The dataset used in this paper is Shakespeare with about 18500 samples which includes sonnets and plays. Mobile keyword prediction is a model-centric and horizontal federated learning problem. We have used a variety of federated learning techniques, including FedAvg, FedSGD and Fedprox. It can be inferred from the results that the FedProx model not only converges faster but also tends to give better results for this dataset.

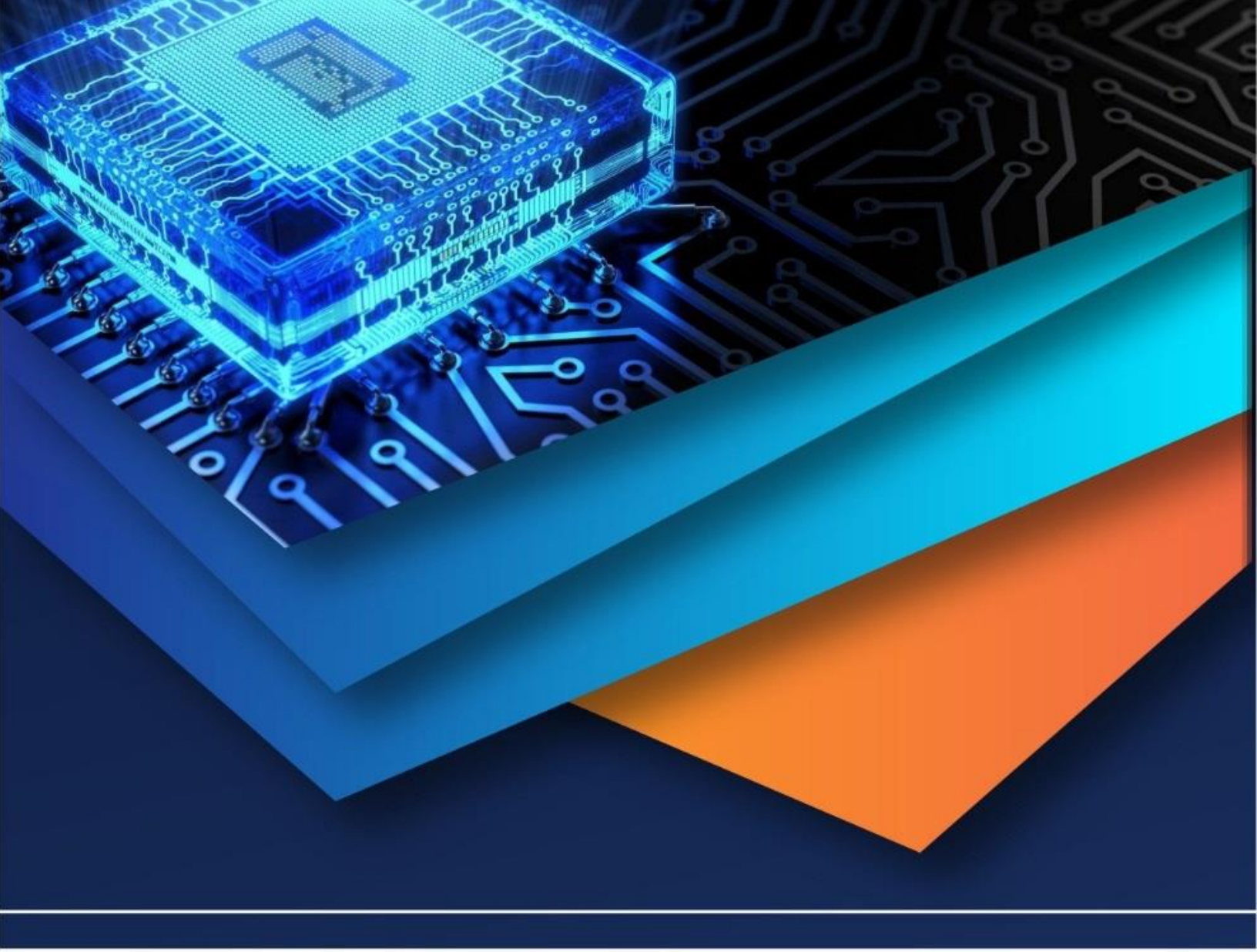
Better results can be achieved by increasing the number of rounds further. In the future, we intend to experiment with other federated learning algorithms and employ various datasets in the same or different languages.

REFERENCES

[1] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Franciose Beaufays,“ Federated Learning for mobile keyboard prediction.”
[2] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, We Li,“ Applied Federated Learning: Improving Google Keyboard Query Suggestions.”
[3] Mingqing Chen, Rajiv Mathews, Tom Ouyang, Francoise Beaufays,“ Federated Learning Of Out-Of-Vocabulary Words.”
[4] Chaoyang He, Conghui Tan, Hanlin Tang, Shuang Qiu, Ji Liu,“ Central Server Free Federated Learning over Single-sided Trust Social Networks.”
[5] Khrystyna Shakhovska, Iryna Dumyn, Natalia Kryvinska,“ An approach for a next-word prediction for Ukrainian Language.”
[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov,“Towards Federated Learning At Scale: System Design.”
[7] Jakub Konecny, H. Brendan McMahan, Daniel Ramage, Peter Richtarik,“ Federated Optimization: Distributed Machine learning for On-Device Intelligence.”
[8] Joel Stremmel, Arjun Singh,“ Pretraining Federated Text Models for Next Word Prediction.”
[9] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan and Yan Zhang,“ Federated Learning for Data Privacy Preservation in Vehicular Cyber-Physical Systems.”

[10] Yiqiang Chen, Xin Qin, Jingong Wang, Chaouhui Yu, Wen Gao," FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare."
[11] Mahfuzzur Rahman, Mahima Rabbi, Annajiat Alim Rasel, Md Tanzim Reza, " A Design and Implementation of Bangla Next Word Predictor Based on Personalized Federated Learning Leveraging Model Agnostic Meta Learning and Semantic Analysis"
[12] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato and Chunyan Miao, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey"
[13] Kaiyue ZHANG, Xuan SONG, Chenhan ZHANG, Shui YU, " Challenges and future directions of secure federated learning: a survey"
[14] Sukanya Bag, "Federated Learning – A Beginners Guide"
[15] George Jeno, "Federated Learning with Python: Design and Implement a Federated Learning System and Develop Applications using existing frameworks"

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)