

Comparative Study of Federated Learning Vs Centralized Learning

Divya M¹ and Dr. Jasmine K S²

¹⁻²Masters of computer application, R V College of Engineering, Mysore Road, Bangalore-560091
Email: divyam.mca21@rvce.edu.in, jasmineks@rvce.edu.in

Abstract— It is widely accepted that when data privacy is a concern, Federated Learning is preferable than the traditional approach of centralized learning. Deciding between the two approaches necessitates considering computational resources, scalability, and privacy concerns. The comparative study demonstrates that federated learning outperforms centralized learning in accuracy, precision, and recall, effectively handling distributed data and privacy preservation. Further research is recommended to identify specific scenarios where each method excels and understand performance disparities. Federated learning is ideal for privacy-sensitive applications, benefiting from its privacy-preserving advantages and decentralized training, while centralized learning remains suitable for scenarios with centralized data and lower privacy concerns due to its simplicity and effectiveness. The study is conducted using TensorFlow and keras. Federated learning has more performance level when compared to centralized learning, which has a comparison proportion of 85% and 50%, respectively.

Index Terms— Performance evaluation, data privacy, federated learning, centralized learning, data models and TensorFlow

I. INTRODUCTION

The primary idea of federated learning is to get around the privacy problems that centralized learning causes. Instead of centralizing data in one location, federated learning maintains data decentralization and distribution among various devices or servers. This decentralized approach helps to protect user privacy and sensitive data because the raw data is retained locally on the devices rather than being transferred to a central server. Only model changes, like gradients or model parameters, are transmitted between local devices and a central coordination server.

The model training procedure is carried out on this centralized data using the classic machine learning approach known as "centralized learning," in which data is gathered and stored in a central server or data repository. In centralized learning, the central server handles all computations necessary to train the model and has access to the complete dataset. Centralized learning depends on variables such as the dataset's size and composition, privacy needs, available computing power, and the desired amount of control over the learning process.

The objective of this comparative study is to provide insight into the benefits, difficulties, and application scenarios of both federated and centralized learning. Researchers and practitioners can make educated choices when choosing the best method for their unique needs by being aware of the traits and trade-offs of different approaches. The paper also emphasizes the need for additional work to create hybrid systems that integrate the benefits of both paradigms, ensuring privacy protection, scalability, and model accuracy in an increasingly data-driven society.

II. LITERATURE REVIEW

The four stages of the FL model life cycle. Then, we went over the presentation, user services, FL training, and infrastructure layers of the four-layer functional architecture of FL systems [1]. The FedAvg method is enhanced in our suggested model, distributed Fed Avg (D-FedAvg), which enables several clients to work together simultaneously and train a single learning model [2]. The FedAvg method is created, and then it is tested on the MNIST and EMNIST datasets for IID and non-IID data distributions, respectively. The findings demonstrate that the A-Fed Avg algorithm, when compared to the Fed Avg method, can decrease the volume of global communications while maintaining guaranteed accuracy and preserving normal mobile device client operation [3]. The centralized control approach enables the use of optimal routing paths considering the constraint problem of energy efficient routing in the WSN while maximizing the network lifetime. In the centralized control approach, the routing and load balancing decisions of the network are made by the sink since the sensor nodes have no intelligence [5]. Focus on federated learning-based anomaly detection for IoT security attacks. The survey explores the use of federated learning techniques to detect and mitigate security threats in IoT environments. It discusses various federated learning approaches and their applicability in addressing the unique challenges of IoT security [9]. Conducted a survey on federated learning-based multi-task network anomaly detection. They discuss the use of federated learning in detecting anomalies in network traffic and propose a multi-task learning framework for improved anomaly detection. The survey explores the challenges, methodologies, and potential applications of federated learning in enhancing the security of network systems [8].

III. IMPLEMENTATION

A. Centralized Learning Models

1) Linear Regression

Linear Regression is a method of data analysis that forecasts the value of unknown data using the value of related and existing data. The linear regression model uses training data to estimate the values of m and b , and once trained, can be used to predict outcomes for fresh, unusual information.

2) Logistic Regression

Logistic Regression is an approach to statistics that models the correlation between a binary dependent variable and one or more independent variables. The logistic function, a kind of S-shaped curve, converts any continuous input into a probability that ranges from 0 to 1.

3) Decision Trees

Decision Tree is a supervised learning method that can be applied to classification and regression issues; it is most frequently used to address classification issues. The Root Node and Terminal Node are the two nodes that make up a decision tree.

TABLE I: LINEAR REGRESSION VS LOGISTIC REGRESSION VS DECISION TREE

Features	Linear Regression	Logistic Regression	Decision Tree
Explainability	Transparency explainability	Less explainability	More interpretable
Scalability	It can train the large dataset	Requires a sufficiently sized training set	May be trained with a minimal training set
Decision Making Limits	The assumption of linearity between variables	A single, linear decision limitation	Divides the area into two smaller areas.
Overfitting	Tendency of the model	Not easily overfitted	Tendency to overfit
Adaptability to noise	Risky for outliers.	Robust to noise	Significantly impacted by noise

B. Federated Learning Models

1) FedAvg (Federated Averaging)

FedAvg is a crucial algorithm. It involves an iterative procedure in which local devices or clients train their models on their specific sets of data before sending model updates to a central server. In order to combine these updates, the server calculates the weighted average of the model parameters. FedAvg makes sure that the models converge to a global model that represents the information from the scattered data.

2) FedSGD (Federated Stochastic Gradient Descent)

The FedSGD algorithm draws inspiration from stochastic gradient descent. By computing gradients based on a portion of their local input, each device or client in this approach trains a local model locally. The global model is then updated by averaging or aggregating these gradients on the central server. With less communication overhead, FedSGD enables decentralized model training.

3) FedDP (Federated Learning with Differential Privacy)

Differential privacy is a method used in federated learning to safeguard the privacy of individual data samples. FedDP incorporates diverse privacy protections into federated learning. In order to maintain the privacy of specific client information, it adds noise to the model changes during aggregation.

TABLE II: FEDAVG VS FEDSGD VS FEDDP

Features	FedAvg	FedSGD	FedDP
Training Methods	Averaging of Model Updates	Random Gradient Descent	Random Gradient Descent
Privacy Protection	No privacy guarantees	No privacy guarantees	Provides Differential Privacy
Model Updates	Aggregated by Central Server	Sent to Central Server	Sent to Central Server
Communication Efficiency	Lower	Higher	Lower
Model Heterogeneity Handling	Aggregates based on Weights	Aggregates based on Weights	Aggregates based on Weights
Communication Overhead	Moderate	Moderate	Increased due to Privacy Noise

TABLE III: FEDERATED VS CENTRALIZED LEARNING

Features	Federative learning	Centralized learning
Scalability	Highly scalable	Less scalable
Network Dependence	Works in offline mode or limited connectivity	Requires a continuous and reliable network connection
Computational Resources	Utilizes the computational resources available on the client devices	Central server typically handles most of the computational workload
Latency	Low Latency	Increased Latency
Data Distribution	Heterogeneous data distribution	Independently distributed across all clients
Security	Higher security	Less security
Communication Overhead	Less communication overhead	Slightly higher

IV. METHODOLOGY

A. Tools

TensorFlow: TensorFlow is an open-source deep learning library developed by Google. It provides a flexible and efficient framework for building, training, and deploying machine learning models, particularly neural networks.

Keras: Keras is an open-source deep learning library written in Python. It is designed for ease of use and fast prototyping, allowing developers to build and experiment with complex neural networks with minimal code.

B. Technologies

1. Federated Learning (FL)

It is possible to train models on decentralized data without the need for data centralization using federated learning, a distributed machine learning approach. Instead of sharing raw data with a coordinator or central server, federated learning involves the training process taking place on local devices or edge nodes and just updating models. In Federated learning, the privacy and security of the data are maintained by the data remaining on the local devices or edge nodes. Using their respective local data, the local devices train models, and the updated models are delivered to the central server.

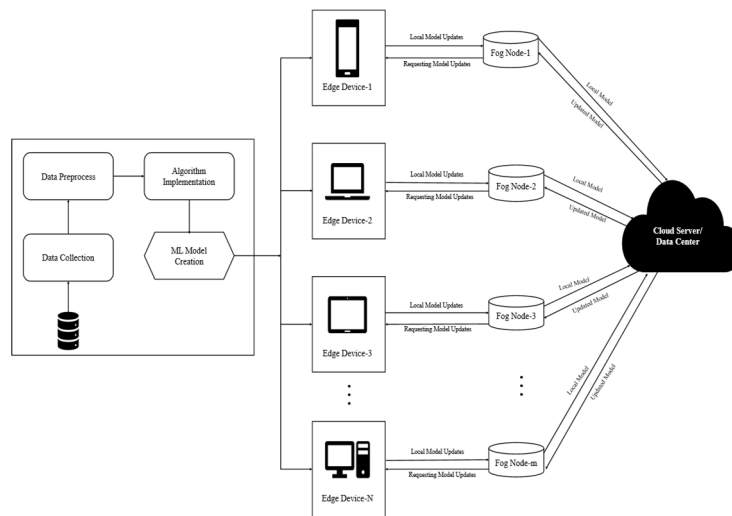


Figure 1. Architecture diagram

2. Federated learning implementation steps

- Step 1: Initialize the global machine learning model with random parameters.
- Step 2: Partition the dataset into subsets, distributing them across multiple data sources
- Step 3: Aggregate the model updates from all data sources using averaging or weighted averaging techniques
- Step 4: Update the global model with the aggregated model updates
- Step 5: Repeat steps 3 to iteratively for a predefined number of rounds or until convergence criteria are met.
- Step 6: Apply privacy-preserving techniques, such as secure multi-party computation
- Step 7: Optimize communication between data sources
- Step 8: Repeat the process to further improve the model's performance
- Step 9: Output the final updated global model

Federated learning is appropriate for scenarios involving a high number of edge devices or IoT devices since it allows for the participation of many devices or clients. The comparative study is to conventional centralized methods, federated learning can make use of the computing capacity of these devices to execute local model training, improving efficiency and lowering communication costs. Federated learning offers a decentralized method for machine learning that enables businesses to make use of distributed data while safeguarding privacy.

C. Result and Discussion

The outcome that shows the accuracy, precision, and recall to determine how well the algorithm performed. To determine performance, we employed the FedAvg algorithm in conjunction with tools like TensorFlow, Keras, and others. To determine the performance, we used the dataset.

```

# Define the Federated Averaging algorithm
def federated_averaging(clients_data, num_epochs=5, batch_size=10):
    global_weights = global_model.get_weights()

    for epoch in range(num_epochs):
        client_weights = []
        for client_data in clients_data:
            local_model = tf.keras.models.clone_model(global_model)
            local_model.compile(loss=loss_fn, optimizer=tf.keras.optimizers.SGD(), metrics=metrics)

            # Train the local model on client data
            x_train, y_train = client_data
            local_model.fit(x_train, y_train, epochs=1, batch_size=batch_size, verbose=0)
            client_weights.append(local_model.get_weights())

        # Aggregate client weights
        global_weights = np.mean(client_weights, axis=0)
        global_model.set_weights(global_weights)

```

Figure 2. Algorithm implementation

```

# Evaluation on a test dataset
x_test = np.random.random((100, 10))
y_test = np.random.randint(0, 2, size=(100, 1))

global_model.compile(loss=loss_fn, optimizer=tf.keras.optimizers.SGD(), metrics=metrics)
_, accuracy, precision, recall = global_model.evaluate(x_test, y_test)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)

```

Figure 3. Evolution of test data

Figure 2 and 3 provides code snippet which demonstrates the basic implementation of federated averaging, a core technique in federated learning. The process begins by initializing a global model with its initial weights. The training process is then conducted for a specified number of epochs, iterating through each client's data. After training, the model weights of each client are stored in a list. Once all clients have completed training, the weights are aggregated by taking the mean, and the global model is updated with the averaged weights. The global model is then evaluated on a test dataset to obtain accuracy, precision, and recall metrics.

```

... 4/4 [=====] - 1s 10ms/step - loss: 0.6932 - binary_ac
Accuracy: 0.49272727966308594
Precision: 0.5050504803657532
Recall: 0.26737967133522034

```

Figure 4. Output screen for performance of Federated Learning

The calculated result indicates a low accuracy of 0.4927, indicating poor overall performance of the model. As may be seen from the precision of 0.5051, there are marginally more true positives than false positives. The model does not successfully detect a sizable proportion of genuine positives, as evidenced by the poor recall of 0.2674, which shows that there are many false negatives.

D. Centralized Learning (CL)

A server or data center is often used as the central repository for data collected and stored as part of the traditional approach to training machine learning models. In centralized learning, all calculations and model training activities are handled by the central server, which has direct access to the training data.

In centralized learning, information from many sources is gathered, combined, and kept on a single server. Since all the data is available in one place, this centralized data repository makes access and analysis simple. The preprocessing of the data, feature engineering, and training of the machine learning models are all done on the central server. Using fresh, unforeseen data, the trained models are then used to make predictions or judgements.

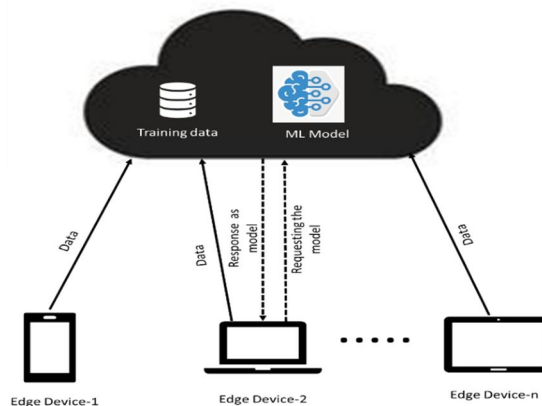


Figure 5. Architecture diagram of centralized learning

1) Centralized Learning Implementation Steps

Step 1. Data Preprocessing: Clean and preprocess the training dataset, handling missing values and scaling features if necessary.

Step 2. Model Initialization: Initialize the machine learning model with random parameters based on the chosen architecture.

Step 3. Model Training: Iterate over the training dataset for a specified number of epochs, updating model parameters through an optimization algorithm like Gradient Descent.

Step 4. Model Evaluation: Use the trained model to make predictions on a separate test dataset and calculate evaluation metrics (e.g., accuracy, precision, recall).

Step 5. Output: Return the trained model and the evaluation metrics on the test dataset.

Centralized learning has drawbacks and difficulties. Since all the data is concentrated in one place, making it more susceptible to breaches or unauthorized access, privacy and data security are key problems. Additionally, because of privacy laws or the amount of the data, the transfer of raw data from spread sources to the central server required for centralized learning may be problematic or illegal.

Centralized learning delivers simplicity, effective computation, and simple model management. However, it encounters issues with privacy, data transfer, and scalability in dispersed systems. Alternative strategies, including federated learning, have therefore been developed to overcome these constraints by permitting decentralized model training on edge devices while maintaining data confidentiality and privacy.

2) Result and Discussion

Results that demonstrate accuracy, precision, and recall are used to assess how effectively the algorithm worked. Performance is measured by using sequential model and the dataset with tools like keras, TensorFlow and others to get the better performance result.

```
# Define the model architecture
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(10,)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Define loss function and optimizer
loss_fn = tf.keras.losses.BinaryCrossentropy()
optimizer = tf.keras.optimizers.SGD(learning_rate=0.1)

# Define evaluation metrics
metrics = [
    tf.keras.metrics.BinaryAccuracy(),
    tf.keras.metrics.Precision(),
    tf.keras.metrics.Recall()
]
```

Figure 6. Model implementation

```
# Evaluation on a test dataset
x_test = np.random.random((100, 10))
y_test = np.random.randint(0, 2, size=(100, 1))

_, accuracy, precision, recall = model.evaluate(x_test, y_test)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
# print("F1-score:", f1_score)
```

Figure 7. Evolution of test data

Figure 6 and 7 provides a basic feedforward neural network model using TensorFlow/Keras for binary classification. The model consists of three dense layers, with the first two layers having 64 neurons and the Rectified Linear Unit (ReLU) activation function. The final layer has one neuron with a sigmoid activation function, suitable for binary classification tasks. The Binary Cross entropy loss function is selected for this task, along with Stochastic Gradient Descent (SGD) as the optimization algorithm with a learning rate of 0.1. The evaluation metrics, including Binary Accuracy, Precision, and Recall, are defined to assess the model's performance on the test dataset.

```
... 4/4 [=====] - 0s 6ms/step - loss: 0.7046
Accuracy: 0.4300000071525574
Precision: 0.38461539149284363
Recall: 0.19607843458652496
```

Figure 8. Output screen for performance of Centralized learning

The result shows that the model's performance is not very good. Only 43% of the situations are correctly predicted by it, meaning that its accuracy is 43%. A low number of accurate positive predictions is indicated by the precision of 38%, but a low percentage of actual positives that the model accurately detected are also indicated by the recall of 19%.

V. PERFORMANCE ASSESSEMENT MECHANISUM

To assess the model's performance confusion matrix is a widely used tool in machine learning and classification tasks that provides a comprehensive summary of the performance of a classification model. The comparison between a model's predicted labels and the actual labels of the data is displayed in a square matrix. When dealing with binary classification issues, the confusion matrix is extremely helpful.

		Actual Values	
		Positive(1)	Negative(0)
Predicted Values	Positive(1)	True Positive (TP)	False Positive (FP)
	Negative(0)	False Negative (FN)	True Negative (TN)

Figure 9. Confusion Matrix

The four basic parts of the confusion matrix are true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The number or frequency of data points that fall into a certain category is represented by each cell of the matrix. To evaluate the effectiveness of the classification model, numerous evaluation metrics can be derived by analyzing the values in the confusion matrix. Accuracy, precision, recall, and F1 score are a few often derived metrics

Formulas to calculate the Accuracy, Precision and Recall are:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

VI. RESULTS

The results obtained by implementing federated learning and centralized learning are represented in figure 10. The experimental results demonstrate that federated learning is the most effective method for achieving a nearly flawless performance with average values for accuracy, precision, and recall of 0.4927272796, 0.505050480, and 0.26737967, respectively. By achieving a performance with an average accuracy of 0.4300000071, a performance with an average precision of 0.384615391, and an average recall of 0.19607843, centralized learning is the least effective technique for this assignment.

	Accuracy (%)	Precision (%)	Recall (%)
Federated Learning	0.4927272796	0.505050480	0.26737967
Centralized Learning	0.4300000071	0.384615391	0.19607843

Figure 10. Federated and centralized learning model performance

The performance of federated learning utilizing the FedAvg algorithm is shown in Fig. 7 above by analyzing the metrics with a random dataset. The effectiveness of centralized learning using the model is assessed using a random dataset, and the results show average accuracy, precision, and recall.

The performance of federated learning utilizing the FedAvg algorithm is shown in Fig. 7 above by analyzing the metrics with a random dataset. The effectiveness of centralized learning using the model is assessed using a random dataset, and the results show average accuracy, precision, and recall.

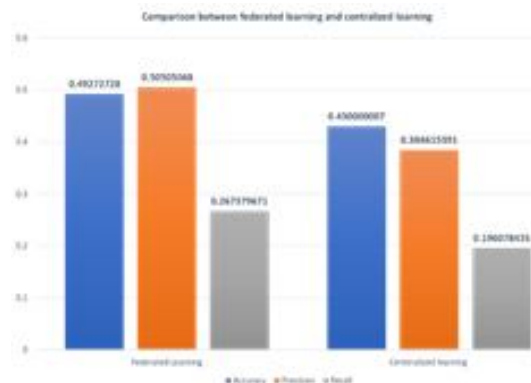


Figure 11. Comparison graph of FL Vs CL

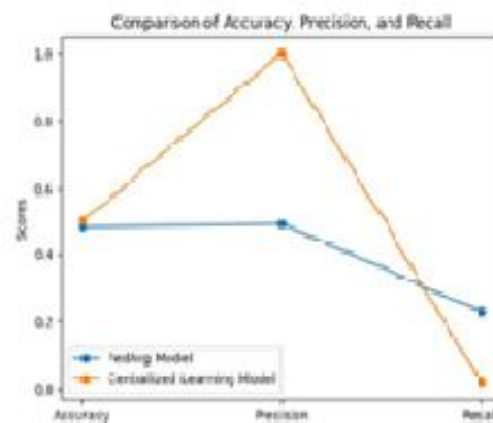


Figure 12. Line chart for comparison of FL & CL

The federated learning model exhibits marginally better outcomes when performance of centralized learning and federated learning are compared based on the specified metrics. As opposed to the centralized learning approach, it achieves superior accuracy, precision, and recall. These results imply that federated learning, which makes use of decentralized model training and distributed data, can be more advantageous for model performance than conventional centralized learning techniques. The results are dependent on the precise evaluation metrics and dataset used, thus it's vital to keep in mind that more investigation and testing may be required to reach more firm conclusions.

VII. CONCLUSION

The comparative study findings suggest that federated learning performed better than centralized learning in terms of accuracy, precision, and recall. This shows that the difficulties of distributed data and privacy preservation are better handled through federated learning. When choosing between the two strategies, it is crucial to take into account additional aspects including computational resources, scalability, and privacy concerns. To study the precise scenarios and contexts where federated learning and centralized learning thrive as well as to identify the underlying causes of the observed performance disparities, more research and experimentation may be done. The decision between centralized learning and federated learning is based on the application's particular needs. Federated learning is appropriate for situations where data privacy is crucial since it provides privacy-preserving advantages and decentralized training. For applications with centralized data and lower privacy concerns, centralized learning is still a feasible choice due to its effectiveness and simplicity of maintenance.

ACKNOWLEDGMENT

We would like to express our sincere gratitude and appreciation to all those who have contributed to the success of the research work. Although it is not possible to mention every individual by name, we extend our gratitude to everyone who participated, supported, and made this research possible.

REFERENCES

- [1] Liu, Ji, et al. "From distributed machine learning to federated learning: A survey." *Knowledge and information systems* volume:64, issue-4, 885-917 (2022)
- [2] Ghimire, A., Asiri, A. N., Hildebrand, B., & Amsaad, F. (2023, April). "Implementation of Secure and Privacy-aware AI Hardware using Distributed Federated Learning". *Dallas Circuits and Systems Conference*, Volume:16, 1-6 (2023)
- [3] Liu, S., & Xu, F. "Adaptive federated learning aggregation strategies based on mobile edge computing". In *Third International Conference on Machine Learning and Computer Application*, Volume: 12636, 65-73 (2022)
- [4] P. Singh, M. K. Singh, R. Singh and N. Singh, "Federated learning: Challenges methods and future directions" in *Federated Learning for IoT Applications*, Springer International Publishing, Volume:37, 199-214 (2022)
- [5] Obi, E., Mammeri, Z., & Ochia, O. E." A Centralized Routing for Lifetime and Energy Optimization in WSNs Using Genetic Algorithm and Least-Square Policy Iteration". *Computer Science Research Institute*, Volume: 12(2), 45-65 (2022)
- [6] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang, and D. Niyato, "Federated learning for 6g communications: Challenges, methods, and future directions," *China Communications*, vol. 17, no. 9, pp. 105–118 (2020)
- [7] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani, "A survey on iot intrusion detection: Federated learning, game theory, social psychology and explainable ai as future directions," *IEEE Internet of Things Journal*, pp. 1–1 (2022)
- [8] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proceedings of the tenth international symposium on information and communication technology*, pp. 273–279 (2019)
- [9] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, vol. 9(4), 2545–2554 (2021)
- [10] T. T. Huong, T. P. Bac, K. N. Ha, N. V. Hoang, N. X. Hoang, N. T. Hung, and K. P. Tran, "Federated learning-based explainable anomaly detection for industrial control systems," *IEEE Access*, vol. 10, 53854–53872 (2022)
- [11] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan and R. Jain, "Machine learning-based network vulnerability analysis of industrial Internet of Things", *IEEE Internet Things J.*, vol. 6(4), 6822-6834 (2019)
- [12] N. Moustafa, M. Keshky, E. Debiez and H. Janicke, "Federated TON_IoT Windows datasets for evaluating AI-based security applications", *Proc. IEEE 19th Int. Conf. Trust Secur. Privacy Comput. Commun. (TrustCom)*, Volume:5, 848-855 (2020)
- [13] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke and L. Shu, "Federated deep learning for cyber security in the Internet of Things: Concepts applications and experimental analysis", *IEEE Access*, vol. 9, 138509-138542 (2021)