

# Layer-Wise Weight Statistics for Node Classification and Defense of Federated Large Language Models

Alexander Berns\*, Reon Akai†, Minoru Kuribayashi\*, Rémi Cogranne‡

\* Center for Data-driven Science and Artificial Intelligence, Tohoku University, Japan

E-mail: berns.alexander.p2@dc.tohoku.ac.jp

† Graduate School of Natural Science and Technology, Okayama University, Japan

‡ Troyes University of Technology, Troyes, France.

**Abstract**—We address performance degradation in Federated Learning (FL) caused by malicious nodes and data bias, focusing on federated large language models (FedLLM). Instead of computationally intensive gradient-aggregation defenses, we propose a lightweight method that flags anomalous nodes by analyzing layer-wise weight updates. After each training round, outlier weights that significantly deviate from the expected Gaussian distribution are identified and excluded from federated learning. Subsequently, statistical variables such as the mean, median, and standard deviation are extracted from the weight information and provided as input features for a machine learning detector. Experiments using a small LLM model, DistilBERT, show that this approach substantially outperforms a baseline that relies solely on raw weight values while reducing communication and computation overhead. Further analysis indicates that the closer the Transformer layer of the LLM is to the final output, the more important the statistical variables of the weight values are.

**Index Terms**—Federated Learning, FedLLM, Transformer, Machine Learning, Anomaly Detection

## I. INTRODUCTION

Large Language Models (LLMs) based on the Transformer architecture have demonstrated remarkable performance in various natural language processing tasks through their self-attention mechanism [1]. However, training these models requires vast amounts of data, which often contains sensitive information. Federated Learning (FL) has emerged as a promising solution to this privacy concern.

In FL [2], multiple nodes (clients) train models locally using their own local training dataset, hereby referred to as *local models* and only the model parameters are shared with a central server. The *global model* denotes the aggregated model at the central server which is periodically distributed to each of the participating clients. This approach preserves data privacy since the raw data never leaves the local devices. However, FL faces significant challenges, particularly when applied to LLMs (FedLLMs), as it becomes vulnerable to various attacks.

**Problem setting:** FL is vulnerable to poisoning attacks as trust is assumed towards local clients. If Byzantine-robust aggregation rules are not used, a single malicious device can make the learned global model useless [3].

**Possible solutions:** Previous research [4] has identified several security issues in FL environments. Malicious nodes can compromise the global model by injecting poisoned data or manipulating model updates. The main *drawback* of various

proposed defense mechanisms [5] is their reliance on computationally expensive gradient aggregation methods which are optimized for small to medium-sized Deep Neural Networks (DNNs), that are impractical for LLMs with often billions of weight parameters. Norm-clipping [6] was presented as a more scalable defense approach; however, it is not able to withstand more sophisticated attacks that remain within the norm bounds.

**Our work:** Focusing on FedLLMs, we propose a computationally light method that analyzes the distribution of weight parameters of Transformer models aggregated by the server after each training round, aiming to detect influences caused by malicious attacks and data biases. The approach involves examining statistical metrics of the weight parameters sent from each node and applying machine learning-based binary classification to identify anomalies. Furthermore, we quantitatively analyze the statistical metrics within each Transformer layer of the LLM model that significantly contribute to anomaly detection.

Section II formalizes the context of FL with an overview of the underlying fundamentals and vulnerabilities. In Section III, the proposed method is presented. The computer simulation and the respective results and evaluation are illustrated in Section IV. A conclusion rounds up the scope of this study.

## II. FEDERATED LEARNING

FL [7] is a distributed machine learning approach where multiple nodes collaboratively train a model while keeping their data localized.

### A. FL Overview

As illustrated in Figure 1, we consider a typical FL system comprising one server and  $N$  clients (nodes). Let  $D_i$  denote the local dataset and  $M_i$  the local model held by client  $C_i$ , where  $i \in \{1, 2, \dots, N\}$ . The objective at the server is to train a model based on data distributed among the  $N$  clients. Active clients participating in local training compute the weight vector  $w$  of a LLM that minimizes the loss function. Typically, the server aggregates the weights received from the  $N$  clients as described by Equation (1):

$$w = \sum_{i=1}^N p_i w_i. \quad (1)$$

Here,  $w_i$  denotes the parameter vector trained on the  $i$ -th client, and  $w$  denotes the parameter vector aggregated at the server.  $p_i$  is defined in Equation (2), and  $|D|$  is calculated as shown in Equation (3):

$$p_i = \frac{|D_i|}{|D|} \geq 0 \quad \left( \sum_{i=1}^N p_i = 1 \right), \quad (2)$$

$$|D| = \sum_{i=1}^N |D_i|, \quad (3)$$

$|D_i|$  denotes the total amount of data samples held by the  $i$ -th client. Furthermore, the optimization problem of this weighting can be formulated using  $w^*$  as shown in Equation (4):

$$w^* = \arg \min_w \sum_{i=1}^N p_i F_i(w, D_i). \quad (4)$$

Here,  $F_i$  denotes the local loss function of the  $i$ -th client. In general, the local loss function  $F_i$  is defined by the empirical risk computed locally. A FL process typically follows these three steps. This sequence of steps is referred to as a *round*.

*Step 1:* The server selects a subset of clients to participate and sends the current global model to the selected clients. In the first round, the global model parameters are initialized randomly. *Step 2:* Each client trains the received model using its own local dataset, producing an updated model referred to as the local model. Later, the clients send their locally updated model parameters back to the server. *Step 3:* The server aggregates the received local models and updates the global model.

The steps are repeated a number of times until the loss function converges to a global optimum. In the present study, the Flower FL framework [8] is used.

### B. Threat Model and Related Works

FL is susceptible to various types of attacks, including *data poisoning*, where malicious clients inject manipulated training to distort the model's behaviour, and *model poisoning*, in which direct alteration of the model parameters or gradients is conducted. Further vulnerabilities result due to *backdoor attacks*, where hidden behavior is embedded into the global model, and *Byzantine failures*, describing malicious client behaviour, to disrupt the learning process. For backdoor and targeted attacks in FL [9], [10], various defense methods have been proposed, including: (i) aggregation rules with enhanced Byzantine robustness such as Krum [11], Median/Trimmed Mean [12], and Bulyan [13]; (ii) anomaly detection techniques that sequentially monitor local updates, such as Zeno [14] and Auror [15]; (iii) robust training via gradient sparsification or reweighting, such as SparseFed [16] and Reweighting [10]; and (iv) frameworks for constructing secure model repositories, such as FLSMR [17]. While these methods have demonstrated certain levels of robustness for conventional neural network models, there has been few research into defenses specifically designed for FedLLMs.

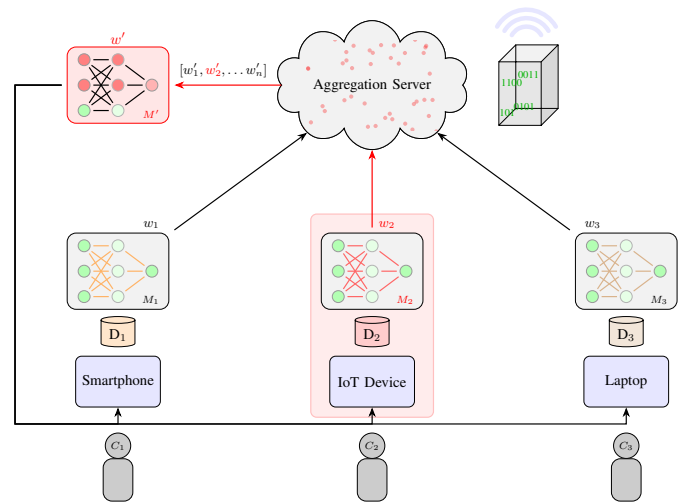


Fig. 1. Federated Learning Overview

## III. PROPOSED METHOD

This section proposes a method to ensure the security of training in the presence of malicious or accidentally faulty nodes (hereafter collectively referred to as biased nodes) during FedLLM training. We propose a method for detecting nodes that perform extremely divergent updates by leveraging statistical indicators in the model parameters. Unlike conventional approaches that monitor all dimensions of the gradients in detail, our method relies on lightweight statistical information and restricts the analysis to specific training layers, aiming for a design that is scalable even to large-scale models. Furthermore, in light of prior observations that it is difficult to determine a universally optimal defense method under diverse conditions such as varying node counts, data distributions, and attack strategies [18], [19], this study explores an approach adaptable to the emerging FedLLMs.

### A. Overview

The FedLLM training scenario assumed in this study is illustrated in Figure 2. Multiple distributed nodes, benign, malicious, or faulty, perform training using their respective local datasets. After each training round, the updated model parameters (typically weight vectors and bias terms) are transmitted to a central server. On the server side, while aggregating the received parameters, the proposed anomaly detection method is used to evaluate whether the current round is normal or includes anomalous updates. If the round is deemed abnormal, appropriate countermeasures such as excluding part of the update information or adjusting the learning rate are applied to minimize the negative impact on the final global model.

Specifically, on the server side, the update parameters from all nodes at the end of each round are collectively observed, and statistical features such as the number of outliers and standard deviation are extracted. These parameters and their corresponding statistics, accumulated over the course of training, are labeled as either normal or abnormal and used to construct

a dataset for binary classification. A machine learning model is then trained to determine whether an anomaly has occurred in the FedLLM training process. This mechanism aims to detect accurately abnormal learning behavior by capturing deviations in the weight distribution, even when malicious or faulty nodes are present in the system.

### B. Assumed Biased Node Conditions

We consider a FL scenario in which the majority of participating nodes are benign, while a small number of biased nodes are present. A biased node refers to a client that, due to intentional or accidental factors, holds data containing incorrect labels and transmits updates to the server that deviate from standard training procedures.

For example, in a sentiment analysis task using a LLM, if certain nodes are configured to continue training with sentiment labels that are intentionally reversed, a form of backdoor or poisoning attack, the resulting model is likely to exhibit biased inference results. In this study, we quantify this biased state using the following two metrics:

- Poisoning Rate ( $R_{\text{poison}}$ ): The proportion of data with incorrect labels among all data held by biased nodes.
- Bias Rate ( $R_{\text{bias}}$ ): The proportion of biased nodes relative to the total number of nodes in the FL framework.

In the primary experimental setting of this study, we assume a fixed configuration with  $R_{\text{poison}} = 0.1$  and  $R_{\text{bias}} = 0.1$ .

### C. Feature Extraction

To enable binary classification (normal vs. abnormal) of the FedLLM training process, this study extracts statistical features from the weight update parameters of each node, which are aggregated by the server at the end of each round. The extraction follows the procedure outlined below. A feature vector composed of multiple statistical metrics is then used to assess the degree of abnormality in the training behavior.

#### (1) Layer-wise Weight Parameter Extraction:

In LLMs such as DistilBERT, numerous parameters are present in the 6 Transformer layers. This study assumes the extraction of weight vectors from multiple layers, with statistical metrics computed for each layer. As the number of layers in even large models, such as GPT-3.5 with 96 layers or OpenAI's o3-mini-high with 32 layers, remains manageable, the proposed layer-wise extraction remains scalable to large-scale LLMs.

#### (2) Outlier Count Calculation

Let the total number of layers be  $L$ . For each layer  $\ell \in \{1, 2, \dots, L\}$ , the set of weight parameters is defined as:

$$W_\ell = \{w_{\ell,1}, w_{\ell,2}, \dots, w_{\ell,N_\ell}\}. \quad (5)$$

Let  $N_\ell$  denote the number of weights in layer  $\ell$ . To capture the presence of extreme values, elements in  $W_\ell$  that deviate significantly from the mean are identified as outliers. Specifically, any weight  $w_{\ell,i}$  is classified as an outlier if its absolute deviation from the mean  $\mu_\ell$ , across all layer weights of a specific client, exceeds  $k$  times the standard deviation  $\sigma_\ell$ ,

i.e.,  $|w_{\ell,i} - \mu_\ell| > k \cdot \sigma_\ell$ . The count of such statistically defined outliers is then computed as:

$$n_{\text{outliers},\ell} = |\{w_{\ell,i} \in W_\ell \mid |w_{\ell,i} - \mu_\ell| > k \cdot \sigma_\ell\}|. \quad (6)$$

This formulation allows the outlier count to adapt to the actual distribution of weights in each layer, offering a distribution-sensitive measure of weight irregularity.

#### (3) Computation of Basic Weight Statistics:

For the weight parameters  $W_\ell$  of each layer  $\ell$  obtained in each training round, the following statistical values are calculated. These metrics are expected to vary significantly in the presence of biased nodes and are therefore considered useful for anomaly detection:

- $\min_\ell$  (minimum),  $\max_\ell$  (maximum)
- Mean:  $\bar{w}_\ell = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} w_{\ell,i}$
- Median:  $\text{median}_\ell$ , standard deviation:  $\sigma_\ell$
- Quartiles:  $Q_{1,\ell}$  (first quartile),  $Q_{3,\ell}$  (third quartile)

Then a characteristic vector is constructed by combining these statistics, including the outlier count and standard deviation, and used to assess the degree of abnormality in training.

### D. Extraction of the Training Dataset

To train the proposed binary classifier, which determines whether a given round includes malicious nodes, a training dataset is constructed using the following procedure:

#### (1) Collection of FedLLM Training Results:

Both normal FedLLM training and anomalous FedLLM training (i.e., training with biased nodes) are executed the same number of times. After each round, the server collects the updated model parameters. The weights obtained from normal training are labeled as class 0 (normal), while those from anomalous training are labeled as class 1 (anomalous).

#### (2) Computation of Layer-wise Statistics:

For each round and each layer, we compute the statistical features  $\Omega$ , including the number of outliers and basic statistics. Specifically, for DistilBERT, parameters are extracted from transformer layers 1 through 6:

$$\Omega = \{n_{\text{outliers},\ell}, \min_\ell, \max_\ell, \bar{w}_\ell, \text{median}_\ell, \sigma_\ell, Q_{1,\ell}, Q_{3,\ell}\}. \quad (7)$$

#### (3) Labeling and Data Aggregation:

Each feature vector obtained from training rounds is assigned a binary label (0 for normal, 1 for anomalous) based on the training condition. As a result, we construct a training dataset where each feature vector representing weight distribution is associated with a corresponding label. Formally, this results in a dataset represented by the matrix  $Z$ , as defined below.

$$Z = \begin{bmatrix} \text{FedLLM}_1 & \text{Round}_1 & \text{Layer}_1 & \Omega_{1,1} & y_1 \\ \text{FedLLM}_1 & \text{Round}_1 & \text{Layer}_2 & \Omega_{1,2} & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{FedLLM}_m & \text{Round}_n & \text{Layer}_p & \Omega_{m,n} & y_m \end{bmatrix}. \quad (8)$$

FedLLM<sub>*i*</sub> denotes the client index, Round<sub>*j*</sub> represents the round number, Layer<sub>*p*</sub> indicates the layer number,  $\Omega_{i,j}$  denotes the feature vector, and  $y_i$  corresponds to the binary label (0;1).

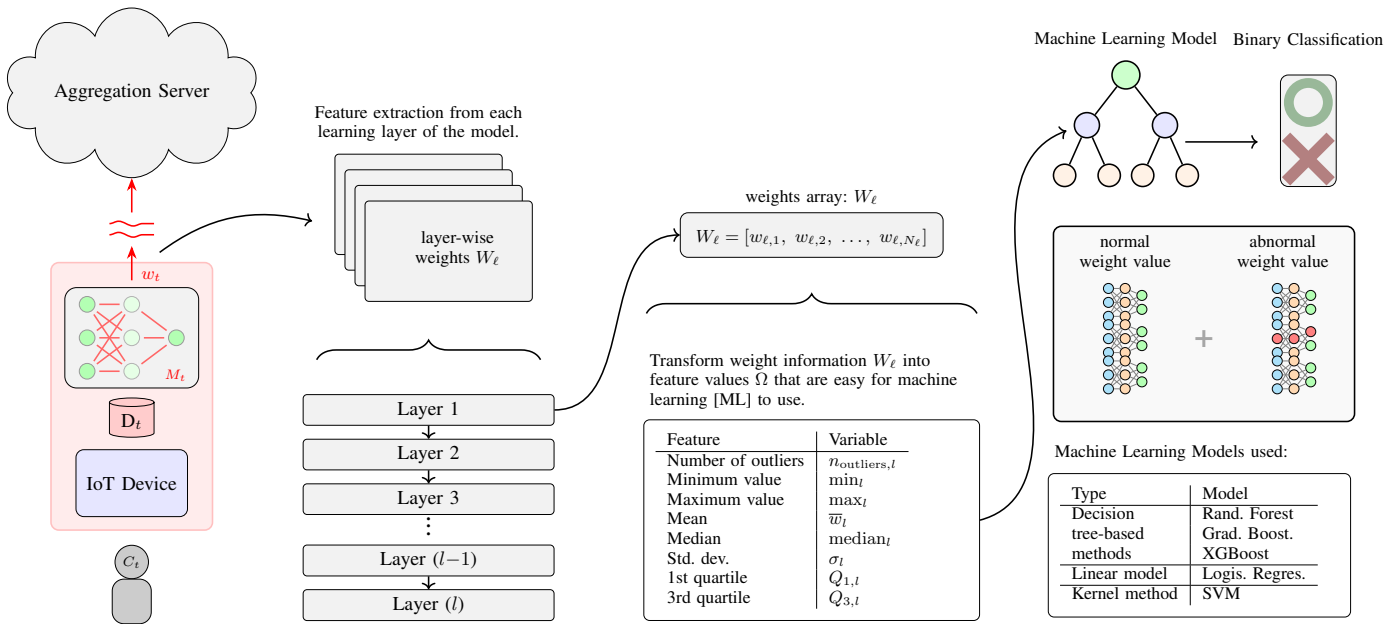


Fig. 2. Malicious node-detection pipeline

### E. Training and Inference of Binary Classifier

Using the labeled training data *normal/abnormal* generated as described in subsections C and D, a binary classifier is trained following the steps below. Five ML-model types are evaluated: Random Forest, Gradient Boosting, Logistic Regression, Radial Basis Function (RBF) SVM, and XGBoost.

(1) *Data Loading and Preprocessing*: The dataset (e.g., CSV files) is loaded, and labels (normal = 0, abnormal = 1) are verified. Data cleaning addresses missing values and outliers. Features are standardized using `StandardScaler`.

(2) *Aggregation and Feature Vector Construction*: For each FedLLM instance, round-level statistics (mean, standard deviation, etc.) are aggregated into a single feature vector. For example, 10 rounds produce one aggregated vector.

(3) *Train/Validation/Test Split*: The dataset is split accordingly. The model is trained on the training set, tuned via validation data, and evaluated on the test set.

(4) *Model Construction and Tuning*: Random Forest, Gradient Boosting, and XGBoost are tree-based ensembles tuned via Grid Search. SVM and Logistic Regression are tuned similarly.

(5) *Evaluation and Saving*: The best-performing model is selected based on Accuracy, Precision, Recall, and F1-Score.

(6) *Inference*: The trained model classifies new FedLLM rounds as normal or abnormal. If abnormal, the server modifies its handling of that update.

## IV. COMPUTER SIMULATION

This section explains the experimental setup. Part A focuses on the conditions for *normal* and *abnormal* training and data processing. In Part B an evaluation of the results portrays the effectiveness of the proposed method.

### A. Experimental Conditions

1) *Definition of Normal and Abnormal FedLLM Conditions*: To compare the presence and absence of biased nodes, two types of FedLLM training environments were constructed.

- Normal FedLLM:
  - Model: DistilBERT (sentiment analysis task)
  - FL Strategy: FedAvg (averaging of weight parameters)
  - Number of Nodes: 10
  - Number of Rounds: 10
  - Dataset: IMDB (50,000 movie review texts)
- Abnormal FedLLM:
  - Model: DistilBERT (sentiment analysis task)
  - FL Strategy: FedAvg
  - Number of Nodes: 10 (one of which is a biased node)
  - Number of Rounds: 10
  - Dataset: For one node, 10% of the assigned IMDB data is mislabeled (Poisoning Rate = 0.1). Consequently, 10% of the total nodes contain mislabeled data (Bias Rate = 0.1).

In the abnormal FedLLM setting, the dataset for the sentiment-analysis task contains mislabeled samples, making extreme bias in portions of the final weight updates highly likely. Both the normal and abnormal configurations are executed for the same number of rounds, and the weight parameters are recorded at the end of each round for comparison.

2) *Dataset Specifications*: We conducted 10 runs each of normal and anomalous FedLLM training, yielding 20 training sessions in total. Across all sessions, with 10 nodes and 10 rounds, weight parameters for every network layer were recorded. For each Transformer layer of DistilBERT, we then computed the outlier count and basic descriptive statistics, collecting the results in CSV format. A threshold of  $k = 1.0$

was adopted, and outliers were defined by deviations exceeding  $k$  times the standard deviation from the layer-wise mean.

3) *Conditions of Machine Learning Models*: A binary classifier was trained on the dataset to distinguish normal from anomalous runs. Five machine-learning models were evaluated:

- Random Forest
- Logistic Regression
- Support Vector Machine (SVM)
- Gradient Boosting
- XGBoost

For each model, we performed hyperparameter tuning using GridSearch combined with five-fold cross-validation ( $k = 5$ ). The data were split into training and testing sets at a ratio of 4:1.

### B. Experimental Results

We compared our proposed approach, which leverages multiple statistics  $\Omega$  (e.g., number of outliers, standard deviation, mean) extracted from the weight parameters, with a baseline that uses individual features derived from the first element of the weight parameters. The baseline uses only the first element  $w_{\ell, \text{first}}$  from each layer's weight matrix  $W_{\ell}$  as a one-dimensional feature. In contrast, the proposed method integrates statistical information such as the outlier count, standard deviation, maximum, and minimum for every layer, forming a multi-dimensional feature vector.

1) *Baseline Results Using Individual Weight Values*: Table I presents the results of evaluating the baseline method, using a individual weight values as input features, across the five models. From this table, the SVM achieves comparatively high performance, with an accuracy of roughly 82% and an F1-score of about 82%. By contrast, tree-based models such as Random Forest, XGBoost, and Gradient Boosting remain at around 69% accuracy, while Logistic Regression reaches approximately 74%. These findings indicate that individual weight elements cannot adequately capture the bias and variance characteristic of anomalous nodes.

TABLE I

EVALUATION METRICS FOR FIVE MODELS USING THE BASELINE METHOD (RAW WEIGHT VALUE)

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.6923	0.6935	0.6923	0.6911
Logistic Regression	0.7436	0.7436	0.7436	0.7436
SVM	<b>0.8205</b>	<b>0.8211</b>	<b>0.8205</b>	<b>0.8203</b>
Gradient Boosting	0.6923	0.6935	0.6923	0.6911
XGBoost	0.6923	0.6952	0.6923	0.6919

### 2) Results of the Proposed Method (Multiple Features):

Next, Table II presents the results of evaluating the proposed method, which constructs a multi-dimensional feature set by combining statistics such as the number of outliers, standard deviation, and maximum and minimum values across the same five models. With every model, metrics such as accuracy and F1-score reached over 95%, achieving a very high discrimination between normal and anomalous training runs. These

findings indicate that integrating diverse weight distribution information allows clear identification of anomalous node characteristics.

TABLE II

EVALUATION METRICS FOR FIVE MODELS USING THE PROPOSED METHOD (MULTIPLE STATISTICAL FEATURES  $\alpha$ )

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.9601	0.9503	0.9702	0.9603
Logistic Regression	0.9706	0.9753	0.9601	0.9672
SVM	0.9801	0.9800	0.9704	0.9755
Gradient Boosting	0.9609	0.9452	0.9607	0.9523
XGBoost	0.9303	0.9402	0.9307	0.9351

3) *Comparison Between Different Feature Sets*: Figure 3 compares the baseline and proposed methods by presenting the accuracy of each model. The proposed method achieves over 95% Accuracy in nearly all cases. This shows that aggregating statistical features, such as measures of dispersion and extreme values, provides significantly more effective signals for anomaly detection than relying on raw weight values extracted from the individual LLM Transformer layers.

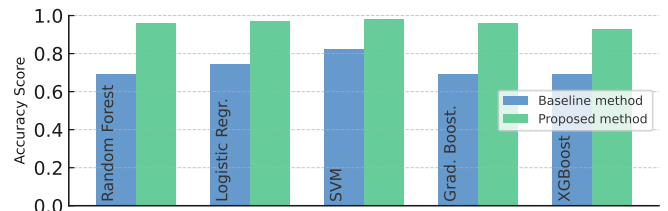


Fig. 3. Accuracy comparison between the baseline and proposed method

4) *Analysis of Feature Importance in the Proposed Method*: To assess how the multiple features  $\Omega$  employed in the proposed method contribute to anomaly node detection, we computed feature importance scores using a Random Forest classifier trained on the extracted statistical features. The total feature set consists of 48 variables, derived from 8 statistical metrics across 6 Transformer layers. The importance of each feature is computed based on the average decrease in Gini impurity across all trees in the forest [20]. These scores are normalized such that their total sums to one, yielding a percentage-like measure of each feature's relative contribution. Figure 4 displays the top 20 most important features according to this measure. The two highest ranked features,

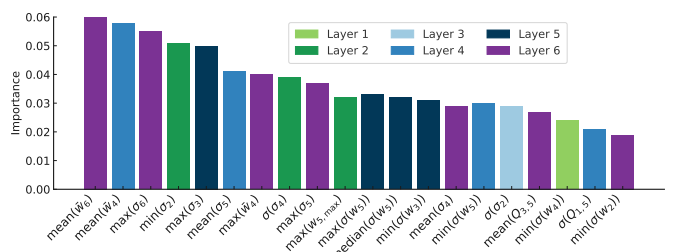


Fig. 4. Feature importance in the proposed method (top 20 features)

$\text{mean}(\bar{w}_6)$  and  $\text{mean}(\bar{w}_4)$ , are both averages of the mean weight values, making them indicators that can easily capture layer specific learning biases. Many other high importance features focus on dispersion for example,  $\max(\sigma(w_6))$  and  $\min(\sigma(\bar{w}_2))$  represent the maximum and minimum standard deviations, respectively. This emphasis on variance reflects the fact that, when biased nodes are present, a small subset of weights changes dramatically, leading to a pronounced increase in variance. Features from layers 1 to 6 appear among the top ranks, indicating that biased node effects may impact any layer. In deep architectures such as LLMs, observing weight distributions across multiple layers is hence effective.

## V. CONCLUSION

In this study, we considered a scenario in FedLLMs involving malicious or faulty nodes. We proposed an anomaly detection method that extracts statistical features, such as the number of outliers and standard deviation, from the weight information parameters collected by the server from the individual LLM Transformer layers after each training round.

Experimental results showed that the baseline method, which uses raw weight values as features, exhibited significant variations in accuracy depending on the model selected. In contrast, our proposed approach, combining multiple statistical features, consistently achieved very high accuracy and F1-scores across various machine learning models, demonstrating excellent anomaly detection performance.

Indeed, feature importance analysis indicated that statistical variables closer to the output layer of the LLM serve as critical indicators for detecting anomalous nodes. Even a limited number of biased nodes can introduce noticeable deviations into the weight updates during each training round, differing significantly from typical training patterns. Therefore, comprehensively analyzing these statistical features enhances the accuracy of distinguishing normal from anomalous behavior.

Future work involves evaluating the effectiveness of the proposed method under non-IID data distributions, diverse attack strategies, and scenarios with a higher proportion of malicious clients, as well as conducting further experiments with varied bias rates and larger node counts to examine scalability.

## ACKNOWLEDGMENT

This study was supported by JSPS KAKENHI (25K15225), JST SICORP (JPMJSC20C3), JST CREST (JPMJCR20D3), Japan.

## REFERENCES

- [1] A. Vaswani, N. Shazeer, and N. e. a. Parmar, "Attention is all you need," in *Advances in Neural Inform. Process. Syst. 30 (NeurIPS)*, 2017, pp. 5998–6008.
- [2] H. B. McMahan, E. Moore, and D. e. a. Ramage, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. AI and Stat. (AISTATS)*, 2017, pp. 1273–1282.
- [3] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-Robust federated learning," in *Proc. 29th (USENIX Security)*, Aug. 2020, pp. 1605–1622. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity20/presentation/fang>.
- [4] P. Kairouz, H. B. McMahan, and B. e. a. Avent, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [5] H. Wang, Y. Yue, and R. e. a. Lu. "Model surgery: Modulating llm behaviour via simple parameter editing." arXiv: 2407.08770 [cs.LG]. (2024).
- [6] X. Zhang, A. Raghunathan, and J. Chen, "Foolsgold: Defending against model poisoning attacks in federated learning," *arXiv preprint arXiv:1911.07963*, 2019.
- [7] K. Wei, J. Li, and M. e. a. Ding, "Fl with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Sec.*, vol. 15, pp. 3454–3469, 2020.
- [8] D. J. Beutel, T. Topal, and A. e. a. Mathur. "Flower: A friendly fl framework." arXiv: 2007.14390. (2022).
- [9] E. Bagdasaryan, A. Veit, and Y. e. a. Hua, "How to backdoor federated learning," in *Proc. International Confer. AI and Statistics (AISTATS)*, 2020, pp. 2938–2948.
- [10] A. N. Bhagoji, S. Chakraborty, and P. e. a. Mittal, "Analyzing fl through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 634–643.
- [11] P. Blanchard, E. M. El Mhamdi, and R. e. a. Guerraoui, "Ml with adversaries: Byzantine-tolerant gradient descent," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017, pp. 119–129.
- [12] D. Yin, Y. Chen, and R. e. a. Kannan, "Byzantine-robust distributed learning: Toward optimal statistical rates," in *Proc. 35th (ICML)*, 2018, pp. 5650–5659.
- [13] R. Guerraoui and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proc. 35th (ICML)*, 2018, pp. 3521–3530.
- [14] C. Xie, O. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault tolerance," in *Proc. 36th (ICML)*, 2019, pp. 6893–6901.
- [15] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep-learning systems," in *Proc. 32nd Annu. Comput. Secur. Appl. Conf. (ACSAC)*, 2016, pp. 508–519.
- [16] A. Panda, S. Mahloujifar, and A. N. e. a. Bhagoji, "Sparsefed: Mitigating model-poisoning attacks in fl with sparsification," in *Proc. Int. Conf. AI and Stat. (AISTATS)*, 2022, pp. 7587–7624.
- [17] D. e. a. Kolasa, "Fl secure model: A framework for malicious-client detection," *SoftwareX*, vol. 27, p. 101765, 2022.
- [18] R. Akai, M. Kuribayashi, and N. Funabiki, "A preliminary study on excluding malicious distr. nodes in fl," *IEICE Tech. Rep.*, vol. 122, no. 412, pp. 89–94, 2023.
- [19] R. Akai, M. Kuribayashi, and N. Funabiki, "A study on eliminating biased nodes in federated learning," in *Proc. APSIPA ASC*, 2023, pp. 620–627.
- [20] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.