scientific reports

Check for updates

OPEN

# A privacy-preserving dependable deep federated learning model for identifying new infections from genome sequences

Sk. Tanzir Mehedi[1], Lway Faisal Abdulrazak[2,3], Kawsar Ahmed[4,5,6]✉, Muhammad Shahin Uddin[1], Francis M. Bui[4], Li Chen[4], Mohammad Ali Moni[7,8] & Fahad Ahmed Al-Zahrani[9]✉

**The traditional molecular-based identification (TMID) technique of new infections from genome sequences (GSs) has made significant contributions so far. However, due to the sensitive nature of the medical data, the TMID technique of transferring the patient's data to the central machine or server may create severe privacy and security issues. In recent years, the progression of deep federated learning (DFL) and its remarkable success in many domains has guided as a potential solution in this field. Therefore, we proposed a dependable and privacy-preserving DFL-based identification model of new infections from GSs. The unique contributions include automatic effective feature selection, which is best suited for identifying new infections, designing a dependable and privacy-preserving DFL-based LeNet model, and evaluating real-world data. To this end, a comprehensive experimental performance evaluation has been conducted. Our proposed model has an overall accuracy of 99.12% after independently and identically distributing the dataset among six clients. Moreover, the proposed model has a precision of 98.23%, recall of 98.04%, f1-score of 96.24%, Cohen's kappa of 83.94%, and ROC AUC of 98.24% for the same configuration, which is a noticeable improvement when compared to the other benchmark models. The proposed dependable model, along with empirical results, is encouraging enough to recognize as an alternative for identifying new infections from other virus strains by ensuring proper privacy and security of patients' data.**

**Keywords** Deep federated learning, Privacy and security, Dependability, Coronavirus, Genome sequence

**Abbreviations**

| | |
|---|---|
| ACTG | Adenine, cytosine, thymine, guanine |
| BC | Blockchain |
| BFL | Blockchain-based federated learning |
| BiLSTM | Bidirectional long-short-term memory |
| BLAST | Basic local alignment search tool |
| BWA | Burrows–Wheeler aligner |
| cDNA | Complementary deoxyribonucleic acid |
| CFL | Centralized federated learning |

[1]Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University, Santosh, Tangail 1902, Bangladesh. [2]Electrical Engineering Technical College, Middle Technical University, Baghdad, Iraq. [3]Department of Computer Science, Cihan University Sulaimaniya, Sulaimaniya, Kurdistan Region 46001, Iraq. [4]Department of Electrical and Computer Engineering, University of Saskatchewan, 57 Campus Drive, Saskatoon, SK S7N 5A9, Canada. [5]Health Informatics Research Lab, Department of Computer Science and Engineering, Daffodil International University, Daffodil Smart City, Birulia, Dhaka 1216, Bangladesh. [6]Group of Bio-Photomatiχ, Information and Communication Technology, Mawlana Bhashani Science and Technology University, Santosh, Tangail 1902, Bangladesh. [7]AI and Digital Health Technology, Artificial Intelligence and Cyber Future Institute, Charles Sturt University, Bathurst, NSW 2795, Australia. [8]AI and Digital Health Technology, Rural Health Research Institute, Charles Sturt University, Orange, NSW 2800, Australia. [9]Department of Computer Engineering, Umm Al-Qura University, 24381 Mecca, Saudi Arabia. ✉email: kawsar.ict@mbstu.ac.bd; k.ahmed.bd@ieee.org; k.ahmed@usask.ca; fayzahrani@uqu.edu.sa

| | |
|---|---|
| CNN | Convolution neural network |
| COVID | Coronavirus disease |
| DeFL | Decentralized federated learning |
| DFL | Deep federated learning |
| DL | Deep learning |
| DNA | Deoxyribonucleic acid |
| DTL | Deep transfer learning |
| FCN | Fully convolutional networks |
| FedSGD | Federated stochastic gradient descent |
| FedAvg | Federated averaging |
| FedDyn | Federated learning with dynamic regularization |
| FL | Federated learning |
| FML | Federated machine learning |
| GDPR | General data protection regulation |
| GPU | Graphics processing unit |
| GS | Genome sequence |
| HFL | Horizontal federated learning |
| HeFL | Heterogeneous federated learning |
| HIPAA | Health insurance portability and accountability act |
| IID | Independently and identically distributed |
| IncepNet | Inception network |
| LeNet | LeCun network |
| ML | Machine learning |
| NCNet | Neighbourhood consensus networks |
| NGS | Next-generation sequencing |
| P2P | Peer-to-peer |
| PCR | Polymerase chain reaction |
| PFL | Personalized federated learning |
| qPCR | Quantitative PCR |
| RAM | Random-access memory |
| ReLU | Rectified linear-unit |
| ResNet | Residual neural network |
| RF | Random forest |
| RNA | Ribonucleic acid |
| ROC AUC | Receiver operating characteristic—area under the curve |
| RT-PCR | Reverse transcription-PCR |
| RT-qPCR | Reverse transcription-quantitative PCR |
| SARS-CoV-2 | Severe acute respiratory syndrome coronavirus |
| SSD | Solid state drive |
| SVM | Support vector machine |
| PFLP | Personalized federated learning by pruning |
| TFL | Transfer federated learning |
| TMID | Traditional molecular-based identification |
| TML | Traditional machine learning |
| VFL | Vertical federated learning |
| VGS | Viral genome sequence |
| WHO | World Health Organization |

The SARS-CoV-2 is a novel human-infecting coronavirus that was first identified in a patient with pneumonia using the next-generation sequencing technique[1–3]. It was declared a world emergency and then a pandemic by the world health organization (WHO)[4,5]. When a pandemic outbreak occurs, it's critical to decide whether it's caused by a new or well-known virus[1,2,6]. This means that early identification of new infections can help scientists to control the transmission rates and limit the risks[6]. Because of the genetic similarities among the viruses, it is challenging to identify new infections from others virus strains[7]. Furthermore, patients suspected of having a new infection may also show symptoms similar to other infections[1,8]. So, it is important to accurately and efficiently identify new infections from genome sequences while ensuring proper privacy and security of the patients' data.

The traditional molecular-based identification (TMID) technique initially failed to identify 35.2% of the 173 samples, resulting in false-negative results[6,9]. So, patients with negative results should repeat the test to avoid misdiagnosis[10,11]. Though the TMID technique has a significant risk of false-negative results, it can detect a small percentage of other similar types of viruses[12]. After considering all the performance parameters of TMID techniques, the overall unsatisfactory identification rate leads to the search for an alternative way to efficiently and accurately identify new infections amongst other viruses from GSs[6,7]. Also, due to the sensitive nature of medical data, the TMID technique of transferring patient data to the central machine or server may create serious privacy and security issues. In particular, direct data transfer increases the risk of privacy breaches and unauthorized access, which can compromise patient confidentiality[13]. To address these concerns, our DFL-based approach ensures that sensitive data remains on local devices, and only encrypted model updates are transmitted to the central server[14]. This mitigates privacy risks by preventing direct exposure of medical data. However, potential security challenges, such as inference attacks or adversarial updates, remain a concern. To safeguard data integrity and confidentiality, secure communication protocols and encryption techniques are employed. Furthermore, TMID techniques require high computational

capabilities[11,12,15]. As a result, TMID techniques cannot be directly deployed for lightweight Medicare applications[15]. On the other hand, several machine learning (ML) and deep learning (DL) models have been proposed to identify new infections to overcome those issues[7]. However, most of the proposed ML and DL-based models did not consider privacy and security issues related to the patient's data[16,17]. Also, most of the previously proposed ML and DL-based models did not consider computational complexity and dependability analysis[18,19].

To address these challenges, this article proposes a privacy-preserving and dependable identification model of new infections amongst other viruses from GSs. Figure 1 shows the motivation on how the proposed model can be applied to identify new infections by ensuring proper privacy and security of patients' data.

The key contributions of this paper are narrated as follows:

- In this work, we propose a privacy-preserving and dependable deep-federated learning-based LeCun Network (LeNet) identification model of new infections amongst other viruses from genome sequences.
- The performance of the proposed model is evaluated using a real-time genome sequence dataset with a different number of clients in terms of the IID distribution of the dataset.
- We consider various deep-transfer learning-based models to compare the performance metrics with the proposed model.
- We also investigate the dependability performance analysis and computational complexity of the proposed model.
- Finally, an overall performance evaluation is considered, where the proposed model is more secure, dependable, and outperforms the existing centralized-based models.

The rest of the paper is organized as follows. First of all, the 'Background and related work' section discusses the overall background and several related works that overlap similar motivations in different ways. In section 'Proposed methodology', we present the proposed model by outlining the complete workflow first, and then the structure of the proposed model, followed by the privacy-preserving model. Additionally, the section 'Model implementation and evaluation' continues with a discussion of environment setup and concludes with an overview of the dataset. Evaluation and experimental result analysis are discussed in the 'Result analysis' section. The 'Limitations and future directions' section highlights the study's constraints and outlines potential areas for future research. Finally, the 'Conclusion' section summarizes the key contributions and findings of this work.

## Background and related work

While investigating the most recent relevant works, we find several works overlap the similar motivation in different ways. The following sections illustrate the overview of federated learning before investigating the related work in this domain.

### Federated learning as a solution

Federated learning (FL) is a distributed ML framework, where multiple devices collaborate to solve traditional ML problems under the coordination of the central server without sharing local private data with other devices[13,14]. This mechanism is completely different from the typical ML setting, which is used in large-scale artificial intelligence technology, medical technology, internet of things technology, and so on for data privacy and security purposes[20–23]. The Google developer team recommends the FL model to protect the privacy and security of the user's data[16,24–26]. According to recent studies, the majority of large-scale service providers have already incorporated FL technology[20–25,27]. The number of medical devices is increasing day by day. As a result, large amounts of medical data are being generated. Analyzing these large-scale datasets requires extensive computing capabilities and processing power. Also, a large number of patients' confidential data must be kept

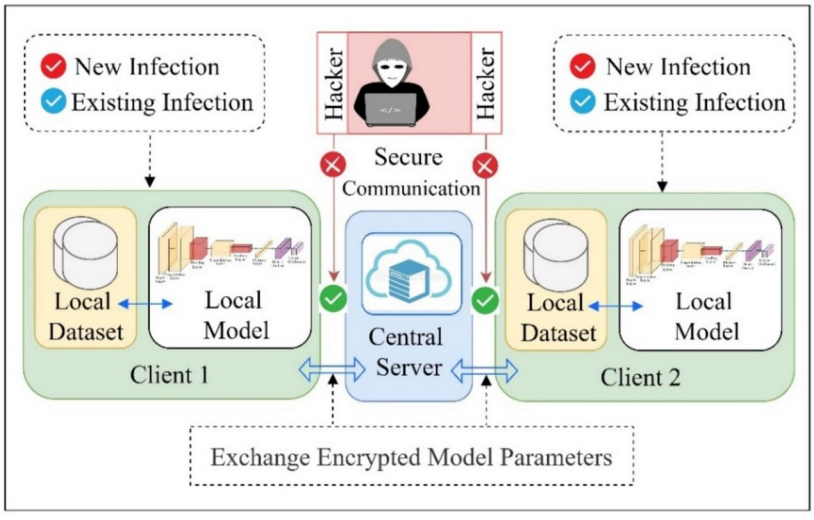



**Fig. 1**. The overall scenario of the proposed dependable identification model of new infections from genome sequences by ensuring proper privacy and security of patients' data.

secure at all times in order to strictly medical laws and regulations (e.g., GDPR sets standards for all sensitive personal data, and HIPAA deals with only protected health information)[28,29].

So, it is recommended to store data locally and perform the computation mechanism on local devices with secure edge computing technology[24,27,30]. As the storage and processing capabilities of local devices improve over time, local devices can be used more efficiently[31]. In this case, the FL mechanism is more suitable[13,23,29]. This technique also enables private and secure collaborative learning between other organizations, where privacy and security of patients' or others' data are the main concerns.

## Structure of federated learning

Figure 2 illustrates the overall architecture and secure communication process between local devices and the central server in a typical FL mechanism. The communication process consists of multiple stages, beginning with system initialization and device selection. During this phase, the central aggregation server determines specific tasks and configures various learning parameters, such as the number of communication rounds, batch size, loss function, and learning rates[27]. Once the initial setup is complete, the server builds a new global model and distributes it to the selected local devices to initiate distributed training[32]. Each local device then trains the model using its private data, calculating updates by minimizing the loss function[28]. Afterward, the local updates are sent back to the central server, which compiles all updates from the devices and resolves the optimization problem to create a new global model[24,27].

A critical step in this process is weight aggregation, where updates from distributed clients are combined to refine the global model. The most commonly used technique is Federated Averaging (FedAvg), which computes a weighted average of model updates based on client data sizes and is widely used due to its simplicity and effectiveness[33]. While more advanced methods exist to address challenges like client heterogeneity and non-IID data—such as FedProx for stabilizing convergence[34] or Scaffold for correcting client drift[35]-FedAvg was chosen for its scalability and established performance in distributed learning scenarios. Adaptive optimizers like FedAdam and methods such as Clustered Federated Learning and robust techniques like Trimmed Mean and Krum are designed for specific challenges, including diverse data distributions and adversarial robustness[36]. However, for our experiments, FedAvg provided an efficient and reliable framework for aggregating updates, ensuring a strong foundation for the analysis.

## Related works

While investigating the most recent relevant works, we find several works overlap the similar motivation in different ways. The following sections discuss the overview of the related works.

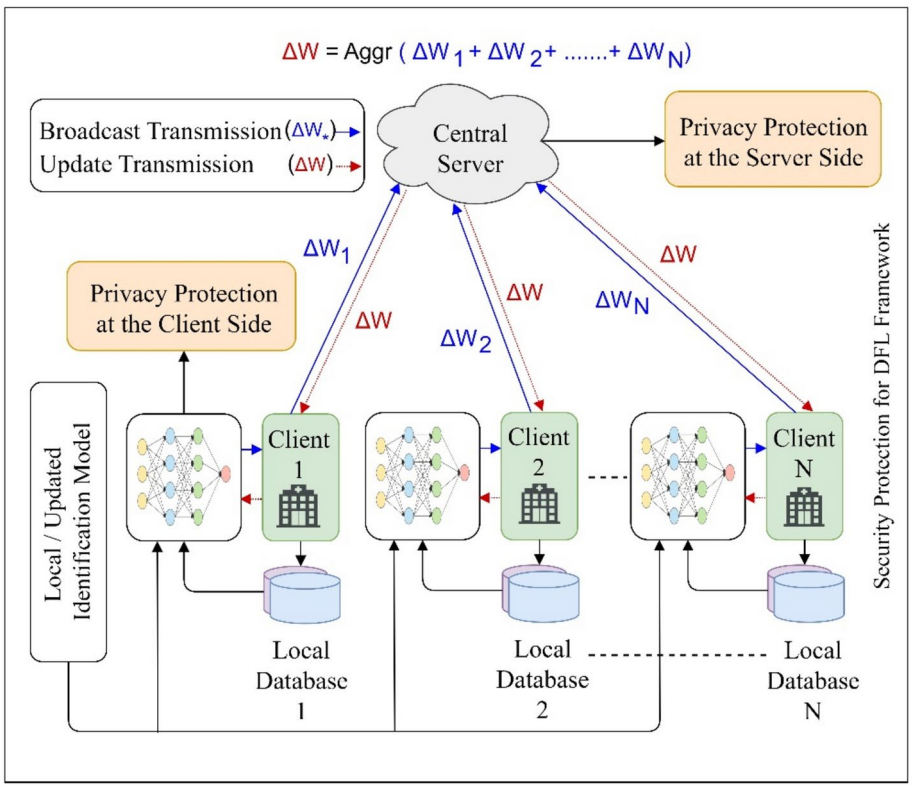



**Fig. 2**. The structure and the communication procedure between a local and a central server in a typical federated learning mechanism.

*ML-based schemes*

Randhawa et al.[37] proposed an ML-based tool for DNA sequence comparison and analysis. The tool was created to address issues related to the alignment of DNA sequences. Zeng et al.[38] proposed an alignment-free CNN architecture for predicting DNA protein binding. The performance of the proposed method does not increase monotonically with the complexity. Zou et al.[39] also provided an alignment-free DL-based technique for genome analysis, which succeeded in the fields of regulatory genomics, variant calling, and pathogenicity score analysis. Furthermore, Seo et al.[40] proposed DeepFam, which is also an alignment-free DL-based technique for protein prediction and modeling. DeepFam uses a feed-forward CNN model, which improves accuracy and reduces runtime. Nguyen et al.[41] also proposed a DL model for classifying DNA sequences. To represent sequences, they used one-hot vectors as input, which preserves the essential position information of each nucleotide in the sequence. The proposed model's performance was evaluated using 12 DNA sequence datasets, and it achieved significant improvements in all these datasets. NCNet is a DL model introduced by Zhang et al.[42] for predicting the function of non-coding DNA sequences. In finding regulatory patterns of motifs, the NCNet model outperforms popular ML methods such as support vector machines (SVM) and random forest (RF). Zhang et al.[43] proposed DeepSite, a combination of bidirectional long-short-term memory (BiLSTM) and convolution neural network (CNN) that was proposed to capture long-term dependencies between DNA sequence motifs. Zhou et al.[44] proposed a model called DeepSEA, which predicts the chromatin effects of sequence modifications with single nucleotide sensitivity. Whata et al.[6] proposed an alignment-free DL-based model for new genome sequences. The proposed CNN-BiLSTM model achieves better performance metrics. Furthermore, Lopez-Rincon et al.[7] also proposed a classification and a specific primer design for the accurate detection of new infections using the DL technique. The suggested methodology has a significant advantage over existing methods. But most of the above-mentioned ML and DL-based techniques did not consider the privacy and security of patients' sensitive data as well as the dependability performance analysis of the model.

*DTL-based schemes*

A DTL paradigm for wearable healthcare systems was proposed by Chen et al.[17]. Experiments show that the proposed technique outperforms traditional methods for wearable healthcare activity recognition, improving accuracy by 5.3%. Hinton et al.[45] and Krizhevsky et al.[46] suggest that the DTL techniques have recently been shown to be superior to the traditional ML and DL techniques, with the majority of applications focusing on discovering patterns and developing those models to make predictions. Mehedi et al.[19] proposed a dependable IDS system for IoT devices based on the DTL approach. Extensive analysis and performance evaluation show that the proposed model is robust, more efficient, and has demonstrated better performance, ensuring dependability, but they did not consider data privacy and security. Furthermore, the DTL technique has also been used in topic categorization, medical systems, and spam detection[47–54]. But most of the proposed DTL models did not consider the privacy and security of data and dependability performance analysis of the proposed models.

*FL-based schemes*

Recently, Kumar et al.[55] proposed a blockchain-based FL structure for new infection image classification. They considered the privacy and security of patient's data, but they did not evaluate the proposed model's dependability. Wu et al.[30] have also proposed a personalized FL technique in the healthcare paradigm based on the cloud edge. The accuracy of the proposed model is high, and they also consider the privacy and security of patient data but not the dependability of the model. Moreover, Roth et al.[56] proposed a breast density classification model based on the FL technique. The proposed model partially takes into account dependability analysis as well as the privacy and security of patient data. Finally, Qayyum et al.[57] proposed a collaborative FL technique for new infection diagnosis at the edge. The accuracy of the proposed model is very high in comparison with the previously proposed model. They considered the privacy and security of the patient's data but not the dependability analysis of the proposed model. A summary of all the existing ML, DL, DTL, and FL mechanisms is given in Table 1. The existing identification and classification models have been categorized according to the proposed algorithm, models' accuracy, dependency, identification coverage, privacy and security of patient's data, and dependability analysis of the proposed model.

## Proposed methodology

In this section, we elaborate the proposed model by outlining the complete workflow first, and then the structure of the proposed model, followed by the privacy-preserving model.

### Workflow of the proposed model

The idea of the proposed model is to develop a dependable and privacy-preserving DFL model. The complete workflow of the proposed model includes model initialization, local model training by medical clients, model parameter encryption by medical clients, model parameter aggregation by the cloud server, and local model update by medical clients. Figure 3 shows the overall workflow of the proposed model with two clients and a key management authority.

*Model initialization*

First, a certified trusted authority $T_a$ generates the public key $PuK$ and private key $PrK$ pair using the key generation method $GeK(k)$, where $k$ is the security parameter. Then, $T_a$ established a secure communication channel between the server $S$ and each client $C$. Once a secure communication has been established, $S$ initializes the learning rate of the model $\alpha$, the rate of exponential decay of moment estimator $m_1, m_2$ where

| References | Algorithm | Accuracy | Dependency | Coverage | Privacy and security | Dependability performance analysis |
|---|---|---|---|---|---|---|
| [37] | MLDSP | 92.0% | Alignment-free | DNA sequence comparison | ✗ | ✗ |
| [38] | CNN | 88.6% | Alignment-free | Predicting DNA–protein binding | ✗ | ✗ |
| [39] | DL | N/A | Alignment-free | Discover DNA binding motifs | ✗ | ✗ |
| [40] | DeepFam | 97.17% | Alignment-based | Protein modeling and prediction | ✗ | ✗ |
| [41] | CNN | 96.23% | Alignment-based | DNA sequence classification | ✗ | ◐ |
| [42] | NCNet | 94.89% | Alignment-free | Predicting of non-coding DNA | ✗ | ✗ |
| [43] | DeepSite | 89.19% | Alignment-free | Predicting DNA–protein binding | ✗ | ✗ |
| [44] | DeepSEA | 95.80% | N/A | Predicting of non-coding variants | ✗ | ✗ |
| [6] | DL | 98.70% | N/A | Genome sequences classification | ✗ | ◐ |
| [7] | DL | 98.73% | Alignment-free | Classification of new infection | ✗ | ◐ |
| [55] | BFL | 98.68% | N/A | New infection image classification | ✓ | ✗ |
| [30] | PFL | 95.37% | N/A | Healthcare paradigm on the cloud | ✓ | ✗ |
| [56] | FL | N/A | N/A | Breast density classification | ✓ | ◐ |
| [57] | CFL | 99.05% | N/A | New Infection diagnosis at the edge | ✓ | ✗ |
| Proposed | DFL | 99.12% | Alignment-free | Identification of new infection | ✓ | ✓ |

**Table 1**. An overview of recent research. The symbols '✓', '◐', and '✗' indicate that the topic is covered, partially, and not covered, respectively.

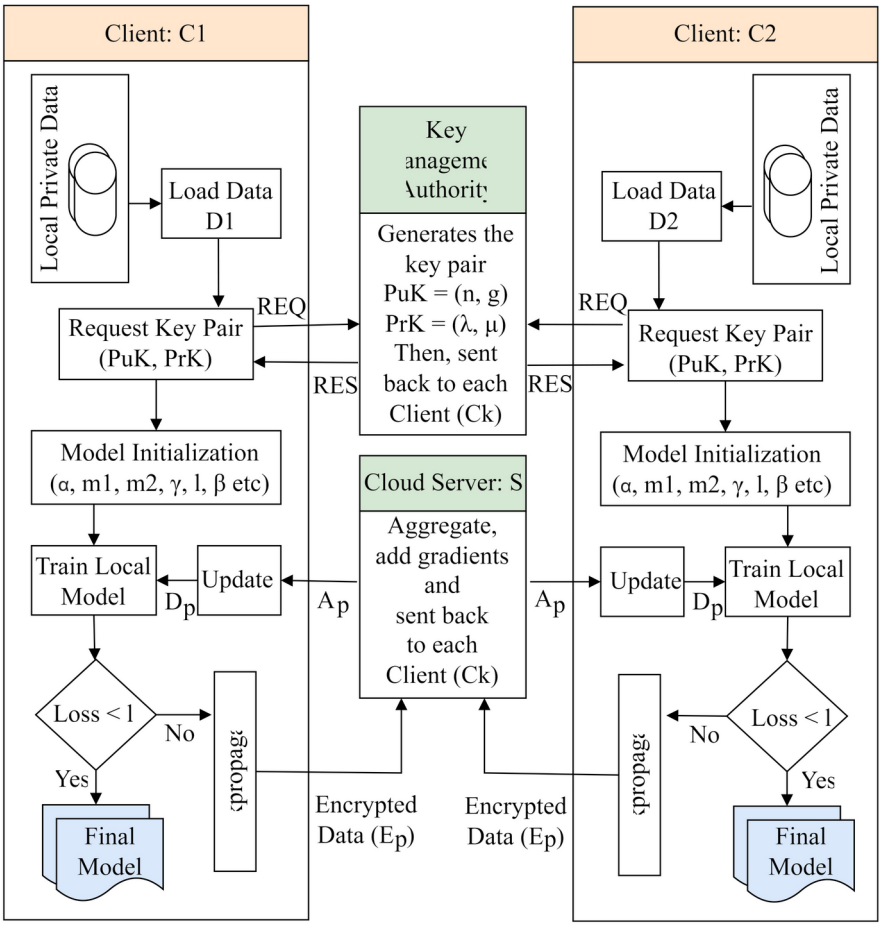



**Fig. 3**. The comprehensive workflow of the proposed model with two client devices, a server, and a key management authority.

$0 \leq m_1, m_2 < 1$, numerical stabilization $\gamma$, loss function of the model , and batch size $\beta$. Then, $S$ selects an array and set its initial parameters to $W^0$. Furthermore, each medical client $C_k$ reports a size $N_k$ of its personal data resource $D_k$ to the $S$, where $k \in K = \{1, 2, 3, 4, ..., K\}$, and then, $S$ computes each $C_k$ contribution ratio by calculating $\eta_k = N_k/(N_1 + N_2 + N_3 + ... + N_K)$;. Finally, set the index of the first communication round $r$ to 1. Algorithm 1 shows the details of the privacy-preserving DFL model.

**Input:** Security parameter: $k$
Medical clients set: $C$
Data resources of all clients: $\{D_k | k \in K\}$
Number of communication rounds: $R$
**Output:** Efficient deep federated learning model.
**Initialization:**
→ $T_a$ generates the key pair by $\{PuK, PrK\} = GeK(k)$;
→ $S$ initializes $\alpha, m_1, m_2, \gamma, l, \beta$;
→ $S$ set initial model parameters as $W^0$;
→ $C_k$ reports a size $N_k$ to the $S$, where $k \in K$;
→ $S$ computes each contribution ratio by $\eta_k = N_k/(N_1 + N_2 + N_3 + ..... + N_K)$;
→ $S$ set the index of first communication round $r$ to 1;
**Procedure:**

```
for r ≤ R do
    (a) Medical clients:
    for ∀k ∈ K do
        C_k calculates the r^th round model parameters
        W_k^r as per local DL model training with inputs: α,m_1,m_2,γ,l,β,W^{r−1},C,D_k;
        for ∀i ∈ δ do
            E_P(w_{k,i}^r) = P_E(w_{k,i}^r, PuK);
        end for
        C_k uploads E_P(w_{k,i}^r) | i ∈ δ to S;
    end for
    (b) Cloud server:
    for ∀i ∈ δ do
        A_P = P_A(w_{1,i}^r, w_{2,i}^r, ..., w_{K,i}^r, η_1, η_2, ..., η_K);
    end for
    S distributes A_P | i ∈ δ to all C_k(k ∈ K);
    (c) Medical clients:
    for ∀k ∈ K do
        for ∀i ∈ δ do
            D_P(w̄_{k,i}^r) = P_D(A_P, PrK);
        end for
        C_k updates local model by D_P(w̄_{k,i}^r) | i ∈ δ;
    end for
    r ← r + 1;
end for
return Efficient DFL model with parameters W^R.
```

**Algorithm 1**. Privacy-preserving DFL model

*Local model training by medical clients*

Each $C_k$ trains a DL-based model locally, using its own private data resource $D_k$ after receiving initial model parameters $w^0$ as well as $\alpha, m_1, m_2, \gamma, l, \beta, W^{r-1}, C, D_k$ from $S$. The training procedure of the local models continues until the loss function converges. First, initialize the first-moment variable $m_1$ and the second-moment variable $m_2$ by 0. Then, split $D_k$ into batches with equal size $\beta$ and set the initial local model parameters by $W_k^r \leftarrow \overline{W}^{r-1}$. Furthermore, for each batch of data resource compute the gradient $g$, biased first and second moment estimate $n_1, n_2$, respectively. Also, compute bias-corrected first moment estimate $\overline{n}_1$, second moment estimate $\overline{n}_2$, and the bias-corrected learning rate $\alpha$. Finally, update the local model parameter $W_k^r$ and return the model parameters. Algorithm 2 summarizes the details of the training procedure of the local model.

**Input :** Learning rate of the model $\alpha$
Numerical stabilization $\gamma$
Loss function of the model $l$
Batch size $\beta$
Security parameter $k$
Medical clients set $C$
Data resources of all clients $\{D_k | k \in K\}$
Moment estimator $m_1, m_2$
**Output:** Model parameters $W_k^r$
**Initialization:**
→ Initialize momentum terms $n_1 = 0$ and $n_2 = 0$;
→ Split $D_k$ into batches with equal size $\beta$;
→ Set initial local model parameters by $W_k^r \leftarrow \overline{W}^{r-1}$;
**Procedure:**

**repeat**
  **for** each batch of data resource **do**
    Compute gradient: $g \leftarrow \Delta W_k^r.l$;
    Compute and update biased $1^{st}$ moment estimate: $n_1 \leftarrow m_1.n_1 + (1 - m_1).g$;
    Compute and update biased $2^{nd}$ moment estimate: $n_2 \leftarrow m_2.n_2 + (1 - m_2).g^2$;
    Compute bias-corrected $1^{st}$ moment estimate: $\overline{n}_1 \leftarrow n_1/(1 - m_1^e)$;
    Compute bias-corrected $2^{nd}$ moment estimate: $\overline{n}_2 \leftarrow n_2/(1 - m_2^e)$;
    Compute bias-corrected learning rate: $\alpha \leftarrow \alpha.\sqrt{(1 - m_2)}/(1 - m_1)$;
    Update local model parameters: $W_k^r \leftarrow W_k^r - \alpha.\overline{n}_1/(\sqrt{\overline{n}_2 + \gamma})$ ;
  **end for**
**until** Loss function $l$ converges: $l = |l_i - l_{i-1}| < \epsilon$, where $\epsilon$ is a small positive value;
**return** Model parameters $W_k^r$.

**Algorithm 2**. Training of the local DL model

*Encryption of model parameters by medical clients*
When the local model is trained and finally returns the model parameters $W_k^r$, then each $C_k$ encrypts $W_k^r$ using the method $P_E(w_{k,i}^r, PuK)$, where $W_k^r = (w_{k,1}^r, w_{k,2}^r, w_{k,3}^r, ..., w_{K,i}^r)$ and $i \in \delta = (1, 2, 3, ..., \delta)$. Then, the encrypted parameters $E_P(w_{k,i}^r) | i \in \delta$ of the local model are uploaded to the $S$ by each $C_k$ via the secure communication channel, where $\delta$ is the total number of parameters in a local model.

*Aggregation of model parameters by cloud server*
Cloud server $S$ computes each $C_k$ contribution ratio by calculating $\eta_k = N_k/(N_1 + N_2 + N_3 + .... + N_K)$. Then, the contribution ratios $\eta_k$ and encrypted parameters $E_P(w_{k,i}^r) | i \in \delta$ from all $C_k$ are aggregated by $S$, where aggregated parameters $A_P = P_A(w_{1,i}^r, w_{2,i}^r, w_{3,i}^r, ..., w_{K,i}^r, \eta_1, \eta_2, \eta_3, ..., \eta_K)$. Finally, $S$ sent back the aggregated parameters $A_P | i \in \delta$ to all $C_k (k \in K)$ by the secured communication channel.

*Local model updating by medical clients*
The encrypted aggregated parameters $A_P|i \in \delta$ are decrypted by using the method $P_D(A_P, PrK)$. Then, $C_k$ updates the local model by the decrypted parameters $D_P(\overline{w}^r_{k,i})|i \in \delta$. After the completion of each successful update operation, the index value of $r$ increases one by one, where $r = (1, 2, 3, 4, ..., R)$. Finally, an efficient DL-based model has been obtained after R rounds of interactions between $S$ and $C_k$.

*Overall computational complexity of the model*
For each successful communication round, each $C_k$ needs to conduct parameter encryption method $P_E$ and parameter decryption method $D_P$. So, total requires $\tau$ number of exponentiation operations in $\mathbb{Z}^*_{n^2}$ and a line of multiplication operations in $\mathbb{Z}^*_{n^2}$ for each successful communication round. Here, $\mathbb{Z}^*_{n^2}$ is treated as a multiplicative group, where $\mathbb{Z}_{n^2}$ is the set of integers $\{0, 1, 2, 3, ..., n-1\}$ with arithmetic being done modulo $n^2$. As a result, the computational complexity of each $C_k$ is approximately linearly proportional to $\delta$ ($C_k \propto \delta$) in a local DL model. Furthermore, $S$ must perform $K$ times of multiplication operations in $\mathbb{Z}^*_{n^2}$ when $\widetilde{C_k}$ aggregates all model parameters and contribution rations for each communication round.

## Proposed DFL model

In this section, we first discuss the proposed model's architecture, and then we thoroughly explain the local model training procedure.

*Model overall architecture*
The block diagram of the proposed model is shown in Fig. 4a, which contains two parts: the local part and the cloud server part. The local part is further divided into two stages: model training and model validation. After preprocessing the raw data, we split the whole dataset into multiple datasets and applied it to the local model for training.

The local model is configured with two trainable layers, containing 32 and 64 units, respectively. It uses a batch size of 64 and trains over 1000 epochs. The *ReLU* activation function is employed in the hidden layers, while the *Softmax* function is used in the output layer to ensure probabilistic outputs. To mitigate overfitting, a dropout rate of 0.10 is incorporated. Additionally, we used early stopping based on validation loss to monitor the model's performance and halt training when further improvement was minimal. The model is optimized using the *Adam* optimizer with a learning rate of 0.001, and its performance is assessed using the *Categorical Crossentropy* loss function. These hyperparameters were determined empirically through iterative experimentation.

For training, a randomly selected subset of the data is used, while another subset is reserved for validation. The process iteratively continues until the loss converges to a minimum threshold, which is determined by the reduction in loss between consecutive iterations becoming smaller than a set tolerance level (e.g., $|l_i - l_{i-1}| < \epsilon$, where $\epsilon$ is a small positive value). If this convergence criterion is not met during validation, the local model parameters are back-propagated to the server. The server aggregates and averages the parameters from all local models to update the global model. These updated global parameters are then redistributed to the local clients, and the training process is repeated. Finally, the global model with the best validation performance metrics is selected, ensuring optimal accuracy and reliability.

*Internal structure of the local models*
The internal structure and the working mechanism of each layer of the client devices have been discussed in this section.

- *Input layer*: The local model consists of eight layers, including the input layer, as shown in Fig. 4b. The input layer receives preprocessed, encoded DNA data batches $B_k \subseteq D_k$, where $D_k$ is the local dataset for client $C_k$. The input is uniformly formatted and passed to the subsequent layers for feature extraction and classification.

$$X_{\text{input}} = \{x_1, x_2, \ldots, x_\beta\} \tag{1}$$

where $\beta$ is the batch size.

- *Convolution layer*: The second layer of the model is the convolution layer, directly connected to the input layer. The convolution layer applies a trainable kernel $W$ to the input sequence to transform it into feature maps[58]. This layer uses a sliding window mechanism to extract local features from the encoded data[47].

$$h_{i,j} = \sum_{k=1}^{K} W_k \cdot X_{\text{input},i+k-1} + b \tag{2}$$

where $W_k$ weights of the kernel, $b$ bias, and $K$ kernel size.

- *Max pooling layer*: The max pooling layer is non-trainable and reduces the size of the feature map by selecting the maximum value in a sliding window. This process retains the most relevant features. We used a total of two max pooling layers after the convolution layer, which selects the maximum activated value among neurons.

$$p_j = \max\left(h_{j:j+s}\right) \tag{3}$$

where $p_j$ pooled feature, $s$ pool size, and $h_{j:j+s}$ window of feature map values.

- *Dropout layer*: The dropout layer prevents overfitting by randomly deactivating neurons during training[19]. This is controlled by a predefined probability $p$, known as the dropout rate. In this model, a dropout layer with a ratio of $0.10(10\%)$ is applied after the second max pooling layer, ensuring effective regularization and enhanced generalization[59].

$$y = \begin{cases} 0, & \text{with probability } p \\ x, & \text{with probability } 1-p \end{cases} \tag{4}$$

- *Flatten layer*: In order to feed the data into the following layer, it must be flattened or transformed into a one-dimensional array. In this layer, the output of the preceding layers is flattened into a single lengthy feature vector, which is connected to the next layer[19].

$$X_{\text{flat}} = \text{Flatten}\left(P\right) \tag{5}$$

where $P$ is the pooled feature map.

- *Dense layer*: A dense layer is used to classify infections based on output from previous layers. This layer is deeply connected with its preceding layer. This fully connected layer combines features extracted from the sequence and identifies high-order patterns. To enhance non-linearity and improve feature representation, the ReLU activation function is employed in this layer[6].

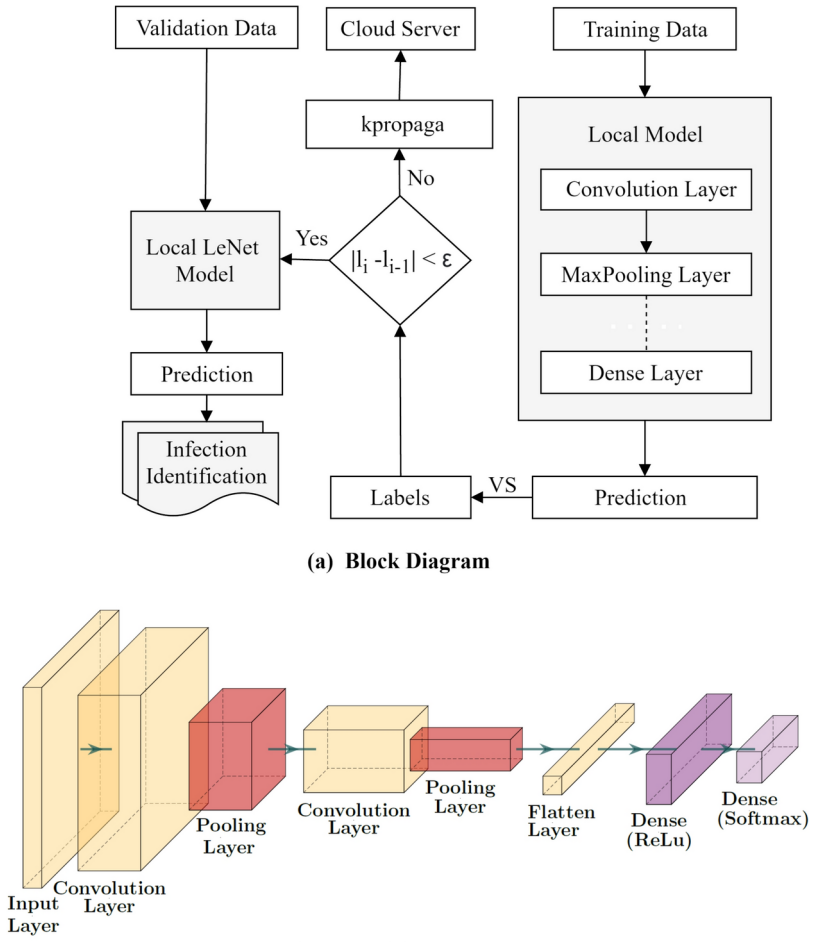



**Fig. 4**. (**a**) The block diagram of the proposed model with a local client device and a cloud server, and (**b**) the internal structure of the local model including the input layer.

$$y = \text{ReLU}\left(W \cdot X_{\text{flat}} + b\right) \tag{6}$$

where the ReLU activation function is defined as:

$$\text{ReLU}\left(x\right) = \max\left(0, x\right) \tag{7}$$

- *Softmax layer*: The final layer of the model is the softmax layer, which is connected to the dense layer. This layer computes the probability distribution over the output classes, determining the likelihood of each class or label. In this case, it predicts whether the input corresponds to a new infection or not[6].

$$\text{softmax}\left(z_i\right) = \frac{\exp\left(z_i\right)}{\sum_{j=1}^{K} \exp\left(z_j\right)}, \tag{8}$$

where $z_i$ is the input to the softmax layer, and $K$ is the number of output classes.

*Paillier cryptosystem-based secure communication protocol*
In this section, we discussed the Paillier cryptosystem-based secure communication protocol[60], including key generation, parameter encryption, parameter aggregation, and decryption techniques.

Key generation The encryption algorithm utilized in this work builds upon principles derived from RSA but incorporates key modifications to enhance security and adaptability for distributed learning systems. First of all, select two large equal bit-size prime numbers $p$ and $q$ randomly and independently of each other that also satisfy the following condition 9.

$$gcd\left((p \times q), (p-1)(q-1)\right) = 1 \tag{9}$$

Then, calculate $n$ and $\lambda$ by the following Eqs. 10 and 11.

$$n = (p \times q) \tag{10}$$

$$\lambda = lcm\left(p-1, q-1\right) \tag{11}$$

An integer $g$ is selected randomly, where $g \in \mathbb{Z}^*_{n^2}$ and also ensures $n$ divides the order of $g$ by checking the existence of the modular multiplicative inverse by the following Eq. 12.

$$\mu = \left(L\left(g^{\lambda} \bmod n^2\right)\right)^{-1} \bmod n \tag{12}$$

where function $L(\vartheta)$ is defined as $L(\vartheta) = (\vartheta - 1)/n$, here, $\vartheta$ refers to the input to the $L$ function.

Thus, $T_a$ generates the key pair $(PuK, PrK)$ by the method of $GeK(k)$, where security parameter $k \in \mathbb{Z}^+$. The generated keys are $PuK = (n, g)$ and $PrK = (\lambda, \mu)$. Finally, $T_a$ publishes $PuK$ and distributes $PrK$ to all the $C_k$ as well as generates a symmetric key $s_i$ for $S$ asnd each $C_i$, where $i \in \{1, 2, 3, 4, ..., K\}$ to establish a secure channel between $S$ and each $C_k$.

Inspired by RSA, this algorithm introduces key deviations from traditional implementations by incorporating the Carmichael function (λ) instead of Euler's totient function (ϕ(n)), incorporating the modular multiplicative inverse, and utilizing the $L$ function to enhance security for distributed systems. Similar advancements have been explored in the literature, such as Jamgekar and Joshi's secure RSA variant for file encryption and decryption[61], Thangavel's Enhanced and Secured RSA Key Generation Scheme (ESRKGS) for improved key generation efficiency[62], and Balasubramanian's study on RSA variants and their cryptanalysis[63]. Building on these innovations, this algorithm integrates the Paillier cryptosystem to achieve additive homomorphic properties essential for secure aggregation in federated learning frameworks[60]. The Paillier cryptosystem strikes a balance between security and computational efficiency, outperforming fully homomorphic encryption (FHE) schemes in real-world applications where latency and resource constraints are critical[60,64]. Furthermore, its robustness has been extensively validated through cryptanalysis in existing literature, making it a reliable and effective choice for secure communication in distributed learning environments[64].

Parameter encryption The model parameters $w^r_{k,i}$ are values associated with a specific client $k$ and an index $i$. These parameters are bounded by $0 \leq w^r_{k,i} < n$, where $n$ is the modulus.

To securely encrypt these parameters:

- A random number $R$ is chosen such that $0 < R < n$ or equivalently $R \in \mathbb{Z}^*_n$.
- A public key $PuK$ is used to perform the encryption.

The encryption of a model parameter $w^r_{k,i}$ is defined by the Eq. 13:

$$E_P\left(w^r_{k,i}\right) = g^{w^r_{k,i}} \cdot R^n \bmod n^2 \tag{13}$$

Here, $g$ is the base used for exponentiation, $w^r_{k,i}$ is the model parameter, $R^n$ is a random masking factor to ensure security, and $n^2$ is the modulus. This implies that $E_P\left(w^r_{k,i}\right)$ encodes the model parameter $w^r_{k,i}$ as an exponent of $g$, with an additional random factor $R^n$.

Parameter aggregation Once encrypted, the parameters are aggregated using weighted contributions from multiple clients. Each client $C_k$ provides a contribution ratio $\eta_k$, which represents the importance of the client's data in the aggregation process. The ratios $\eta_k$ are scaled by a factor of $10^3$ to ensure they are positive integers, resulting in amplified ratios $\eta'_k$. Then, the encrypted parameters $E_P\left(w^r_{k,i}\right)$ from all clients are combined using the scaled contribution ratios $\eta'_k$. The aggregation process is defined by the Eq. 14:

$$\Lambda_P = \prod_{i=1}^{K} E_P\left(w^r_{k,i}\right)^{\eta'_i} \bmod n^2 \tag{14}$$

Using Eq. (13), substituting $E_P\left(w^r_{k,i}\right) = g^{\left(w^r_{k,i}\right)} \cdot R^n \bmod n^2$, we have:

$$\left[E_P\left(w^r_{k,i}\right)\right]^{\eta'_i} = \left[g^{\left(w^r_{k,i}\right)} \cdot R^n \bmod n^2\right]^{\eta'_i}$$

Applying the properties of exponents under modular arithmetic:

$$\begin{aligned}\left[E_P\left(w^r_{k,i}\right)\right]^{\eta'_i} &= g^{\left(w^r_{k,i}\right)\cdot\eta'_i} \cdot (R^n)^{\eta'_i} \bmod n^2 \\ &= \left(g^{w^r_{k,i}\cdot\eta'_i} \cdot R_i^{n\eta'_i}\right) \bmod n^2\end{aligned}$$

Now, substituting $\left[E_P\left(w^r_{k,i}\right)\right]^{\eta'_i} = \left(g^{w^r_{k,i}\cdot\eta'_i} \cdot R_i^{n\eta'_i}\right) \bmod n^2$, into Eq. 14 , we have:

$$\Lambda_P = \prod_{i=1}^{K} \left[\left(g^{w^r_{k,i}\cdot\eta'_i} \cdot R_i^{n\eta'_i}\right) \bmod n^2\right] \bmod n^2$$

The product of terms can be written as a single-modulus operation:

$$\begin{aligned}A_P &= \prod_{i=1}^{K} \left[g^{w^r_{k,i}\cdot\eta'_i} \cdot R_i^{n\eta'_i}\right] \bmod n^2 \\ &= \prod_{i=1}^{K} g^{w^r_{k,i}\cdot\eta'_i} \cdot \prod_{i=1}^{K} R_i^{n\eta'_i} \bmod n^2\end{aligned}$$

Using the property of exponents for products:

$$\prod_{i=1}^{K} g^{w^r_{k,i}\cdot\eta'_i} = g^{\sum_{i=1}^{K} w^r_{k,i}\cdot\eta'_i}$$

Substitute the combined terms back into the equation:

$$\Lambda_P = \left(g^{\sum_{i=1}^{K} w^r_{k,i}\cdot\eta'_i} \cdot \prod_{i=1}^{K} R_i^{n\eta'_i}\right) \bmod n^2 \tag{15}$$

Parameter decryption When each client $C_k$ receives the encrypted aggregated model parameter $A_P \in \mathbb{Z}^*_{n^2}$ from the server $S$, the client decrypts the parameter using its private key as follows:

$$DP_{w^r_{k,i}} = L\left(\Lambda_P \bmod n^2\right) \cdot \mu \bmod n$$

where $\mu = L\left(g^{\lambda} \bmod n^2\right)^{-1} \bmod n$, $\lambda$ is the private key parameter.

Using the definition of $\Lambda_P$ from Eq. 15:

$$DP_{w_{k,i}^r} = L\left(g^{\sum_{i=1}^{K} w_{k,i}^r \cdot \eta_i'} \cdot \prod_{i=1}^{K} R_i^{n\eta_i'} \mathrm{mod} n^2\right) \cdot \mu \mathrm{mod}\ n$$

Using the additive property under modular arithmetic:

$$L\left(\Lambda_P \bmod n^2\right) = L\left(g^{\sum_{i=1}^{K} w_{k,i}^r \cdot \eta_i'} \bmod n^2\right) + L\left(\prod_{i=1}^{K} R_i^{n\eta_i'} \bmod n^2\right) \bmod n$$

We split into terms, and the first term:

$$L\left(g^{\sum_{i=1}^{K} w_{k,i}^r \cdot \eta_i'} \bmod n^2\right) = \sum_{i=1}^{K} w_{k,i}^r \cdot \eta_i' \cdot L(g)$$

The second term:

$$L\left(\prod_{i=1}^{K} R_i^{n\eta_i'} \bmod n^2\right) = \sum_{i=1}^{K} L\left(R_i^{n\eta_i'}\right)$$

Now, substitute these into the equation for $DP_{w_{k,i}^r}$:

$$DP_{w_{k,i}^r} = \left(\sum_{i=1}^{K} w_{k,i}^r \cdot \eta_i' \cdot L(g) + \sum_{i=1}^{K} L\left(R_i^{n\eta_i'}\right)\right) \cdot \mu \bmod n$$

From the equation 12, $\mu = L\left(g^{\lambda} \mathrm{mod}\ n^2\right)^{-1}$ where $L\left(g^{\lambda}\right) = 1$. Thus, the equation simplifies:

$$DP_{w_{k,i}^r} = \sum_{i=1}^{K} w_{k,i}^r \cdot \eta_i' \bmod n \tag{16}$$

Now, compute the average value and update the local model by the following Eq. 17. Here, $10^3$ is a scalar used to transform the contribution ratios to positive integers.

$$D_P(\overline{w}_{k,i}^r) = \frac{D_P(\overline{w}_{k,i}^r)}{10^3} \tag{17}$$

## Model implementation and evaluation

The section continues with a discussion of environment setup, then an overview of the dataset with the data preparation process and concludes with the training and testing process of the selected DTL and DFL models.

### Environmental setup

The implementation of the proposed model is carried out using *PySyft* and *PyTorch*. *PySyft* is an open-source library that combines different tools for building secure and private machine learning models, especially for the FL model[65]. The main idea behind the *PySyft* library is to extend the APIs of popular DL mechanisms such as *PyTorch*, *Keras*, and *TensorFlow*[24]. Furthermore, *PyTorch* is another *Python* library that provides high-level features, including tensor computation (like *NumPy*) with strong GPU acceleration[66].

First, we installed the *PySyft* and *PyTorch* libraries along with other necessary libraries. Then, we created the virtual client by the *VirtualWorker* method with the help of the *TorchHook* mechanism that simulates each client. We have also initialized the central server called *"secure_worker"*. Each client and the central server has a unique pointer tensor (e.g., client 1 pointer tensor is *45140214847*). Finally, the central server (*PT*: *61532244495*) established secure communications among all the clients. After successful secure communication was established between client devices and the central server, we stored the data into a tensor and then divided the data into features and targets. Furthermore, we calculated the number of samples per client and divided the features and targets, and finally sent them to all clients. Next, we created the model and sent a copy of the model to all the clients. We have performed the *copy()* and *send()* operation by *model.copy().send(PT)* mechanism. Then the model is trained by its own datasets and calculates the performance metrics by 1000 iterations. We defined several functions for training the model while keeping track of the training loss and training accuracy for each client individually. Each local client model improves a little bit in its own way and calculated the local losses and accuracies. If the local loss is not minimized, encrypt the model parameter and send it to the central server parallelly by *backward()* method. The central server calculated the average value of the parameters by the *avg_weight.get()* method and distribute them to all the clients. Then, each client decrypted the model parameters, updated the local model, and retrained the local model using its private dataset. We performed 8 rounds and finally got the final model and computed the testing accuracy with the testing dataset. Therefore, we were able to train the client without exposing each training data to others. We were also able to aggregate the updated model

parameters from each client by a trusted central server to prevent privacy and security of patients' private and sensitive data.

### Dataset description

The dataset has been downloaded from the genes database at the National Center for Biotechnology Information (NCBI) which contains 583 sequences[6]. There were 553 unique sequences after removing all repetitive sequences and then performing our next analysis. The coronavirus family contained all of the virus genes. If a gene contained the SARS CoV-2 gene, we assigned it a label of 0; otherwise, we assigned it a label of 1. The dataset was unbalanced with 16.47% SARS CoV-2 positive samples and 83.53% SARS CoV-2 negative samples.

### Data preparation

It is necessary to clean and prepare the raw DNA sequence dataset before applying DTL and DFL methods to achieve optimal performance. Data preparation of DNA sequence is normally done by removing duplicate samples, converting non-numerical features into numeric features, and dealing with missing values. The DNA sequences are made up of consecutive letters (e.g., A, C, G, and T) with no spaces between them. So, the DNA sequence does not contain any words. We used a unique representation technique to convert DNA sequences into word sequences without losing position information. First, we remove duplicate samples from the dataset by applying the *unique_everseen()* method. Then, we used the *lower()* method to convert the uppercase sequence to the lowercase sequence for further processing. Now translate the lowercase DNA sequences into short overlapping k-mers of length 3 using *getKmers(sequences, size = 3)* method[67]. Figure 5 shows the process of translating the DNA sequence into a sequence of words with window size 3 and slide stride 1.

Figure 6a shows a dictionary with 64 ($4^3$, 4 chars = 'ACGT' and word size = 3) distinct words, and each word is then represented by a one-hot vector of size 64. Then, we set the size of the region and convert the word sequence into a one-hot 2D vector. Now, we have a 2D numerical matrix that contains information on the specific position of each nucleotide in the sequence. Figure 6b shows a one-hot 2D vector representation of generated DNA sequence words with region size 2. This matrix is then used as input for the selected DTL and DFL models for further analysis.

### Model training and testing

First, the pre-processed DNA sequence dataset has been split into the training (80%) and testing (20%) datasets. Then, split the training dataset again into a new training dataset (80%) for training the selected DTL and DFL models and a validation dataset (20%) for tuning the hyperparameters. This splitting ratio is considered as an optimal ratio to reduce overfitting[19,67]. For the proposed DFL model, we first considered the number of client devices and calculated the total number of samples per client device. Then divided the dataset and sent it to all the client devices. We have a total of 553 unique pre-processed DNA sequence datasets and divided them into 4, 6, 10, 15, and 20 client devices, where each client device got 138, 92, 53, 37, and 28 pre-processed DNA sequence datasets, respectively. Afterward, we selected the final model after a series of operations. We have also considered DTL algorithms because some of their variants have been successfully applied to solve classification tasks related to DNA sequencing[68]. The overall performance of the DTL models has been evaluated with a wide range of tested hyperparameters. Specifically, we considered five models based on previous performance in different datasets—CNN, FCN, IncepNet, ResNet, and LeNet, with varying architectures and configurations. These models were tested using different numbers of trainable hidden layers, neuron configurations, activation functions, and dropout settings. For instance, CNN has 4 trainable hidden layers (32, 64, 128, 64 neurons), while FCN, IncepNet, and ResNet have 3 layers each, with FCN and ResNet using 128, 256, and 128 neurons, and IncepNet using 32, 64, and 32 neurons. LeNet has 2 layers (32, 64 neurons). All models use a batch size of 64 and the Adam optimizer, with ReLU activation in CNN, ResNet, and LeNet; tanh in FCN; and linear in IncepNet. The output layers use sigmoid for CNN and IncepNet, and softmax for FCN, ResNet, and LeNet. Dropout layers were applied in CNN at 20%, which is double that of ResNet and LeNet, both set at 10% percent, to effectively mitigate overfitting and improve model generalization. This comprehensive evaluation highlights the adaptability of DTL models for genome sequence classification tasks.

## Result analysis

This section discusses the overall performance of the selected models. We implemented a wide range of analysis scenarios using various performance indicators and compared them. We also analyzed the computational complexity and dependability performance of the selected models.

### Performance indicators

We have used several performance indicators to analyze the results. The following performance indicators were used to evaluate the selected models.

- *True positive* ($T_{pos}$) refers to the number of new infections that are correctly detected as new infections.
- *True negative* ($T_{neg}$) is the number of existing infections that are correctly detected as existing infections.
- *False positive* ($F_{pos}$) is the number of existing infections that are incorrectly detected as new infections.
- *False negative* ($F_{neg}$) refers to the number of new infections that are incorrectly detected as existing infections.

The model's performance across all classes is often measured by its accuracy metric, where higher accuracy means better performance[19]. Equation 18 defines the mathematical representation of accuracy.

$$Accuracy = \frac{T_{pos} + T_{neg}}{T_{pos} + T_{neg} + F_{pos} + F_{neg}} \tag{18}$$

The precision shows how accurate the model is in identifying positive samples[69]. Equation 19 defines the mathematical representation of precision.

$$Precision = \frac{T_{pos}}{T_{pos} + F_{pos}} \tag{19}$$

The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected[69]. Recall can be defined by the Eq. 20.

$$Recall = \frac{T_{pos}}{T_{pos} + F_{neg}} \tag{20}$$

The F1-score computes the harmonic mean of precision and recall, respectively. The F1-score has a range of 0.0–1.0, with 1.0 denoting model perfection[69]. Equation 21 defines the mathematical representation of the F1-score.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{21}$$

Cohen's kappa ($\kappa$) measures the inter-rater agreement between observed and predicted classifications while accounting for the agreement occurring by chance. It provides a range of values between − 1 and 1, where 1 indicates perfect agreement, 0 represents random chance, and negative values indicate systematic disagreement. It is particularly useful for imbalanced datasets, where accuracy might be misleading[69]. Equation 22 shows the mathematical representation of Cohen's kappa.

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{22}$$

where $P_o$ is the observed agreement, and $P_e$ is the expected agreement by chance. A detailed evaluation of Cohen's kappa in our model ensures that we assess the reliability of predictions beyond mere accuracy.

The receiver operating characteristic—area under the curve (ROC-AUC) evaluates the trade-off between true positive and false positive rates across different thresholds[69]. A higher ROC-AUC value indicates better model performance, as it reflects the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance. The ROC curve plots the Recall against the False Positive Rate (FPR). Equation 23 shows the mathematical representation of FPR.

$$\text{FPR} = \frac{F_{pos}}{F_{pos} + T_{neg}} \tag{23}$$

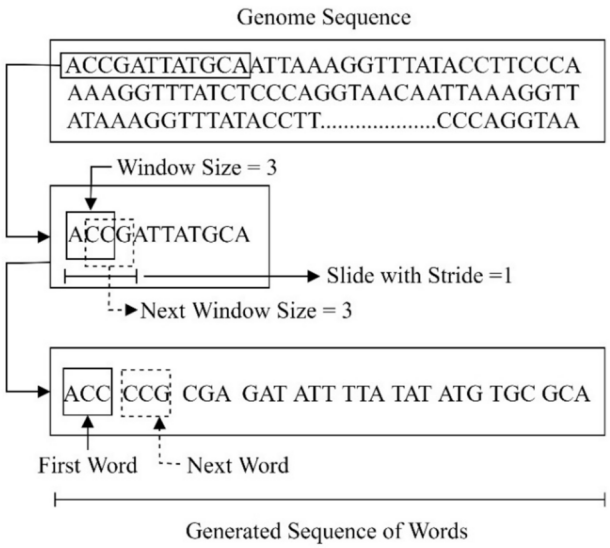
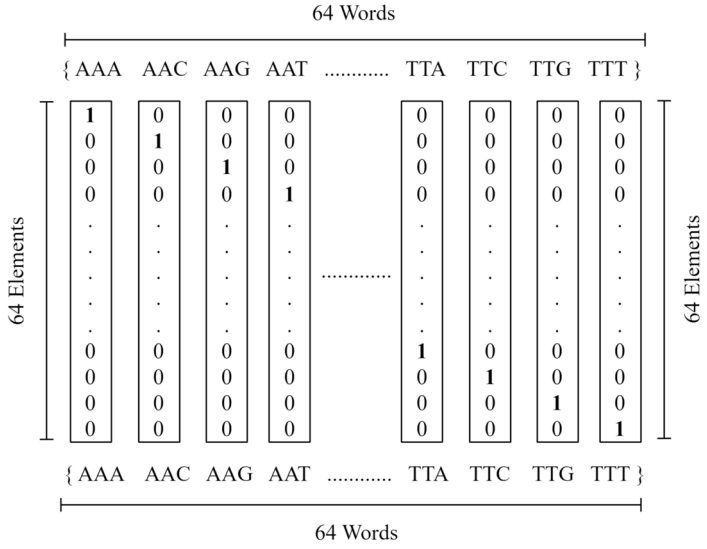
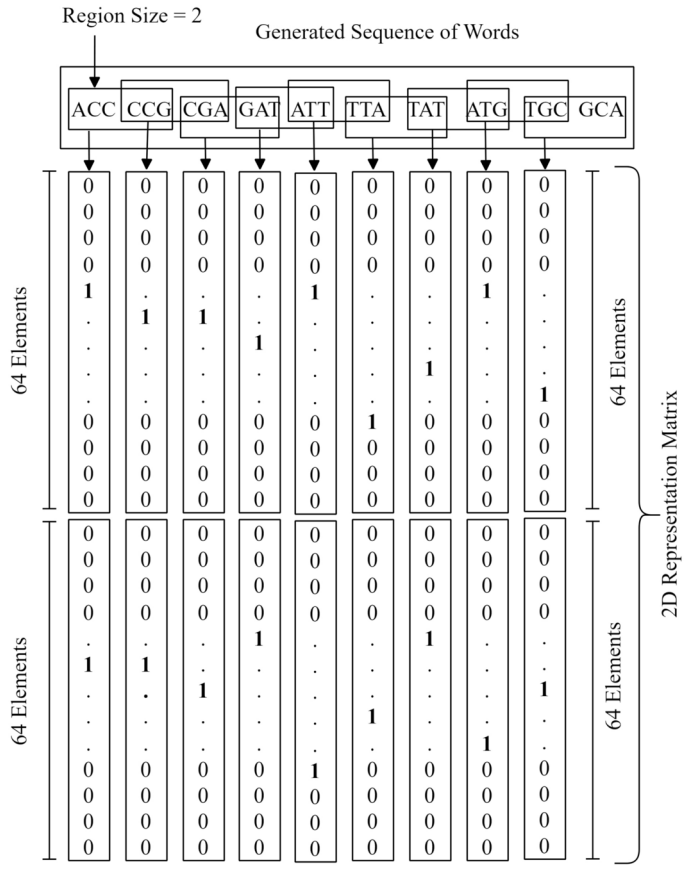



**Fig. 5**. The process of translating the deoxyribonucleic acid sequence into a sequence of words with window size = 3 and slide stride = 1.




**(a) Dictionary representation**




**(b) One-hot 2D vector representation**

**Fig. 6**. (**a**) Dictionary representation of the generated word sequence with 64 (43, 4 chars = 'ACGT' and word size = 3), and (**b**) One-hot 2D vector representation of the generated distinct word sequence with region size = 2.

AUC values range from 0.5 (random guessing) to 1.0 (perfect classification). ROC-AUC is critical in our evaluation as it highlights the model's ability to handle imbalanced classes effectively.

## DTL metrics analysis

In recent years, DTL models have advanced features, and some of these variants have been effectively applied to solve genome sequence classification problems[53,69]. Therefore, we considered five DTL models (e.g., CNN, FCN, IncepNet, LeNet, and ResNet) because of their optimal performance. Table 2 shows the performance comparison metrics of the selected DTL models.

In this analysis, we trained all models for 1000 epochs, and the LeNet model demonstrated the best overall performance, achieving the highest accuracy of 0.9907, a precision score of 0.9903, recall of 0.9907, F1-score of 0.9901, and a ROC AUC score of 0.8598. Its metrics highlight its ability to consistently classify new and existing infections with high reliability. The CNN model closely followed LeNet, achieving an accuracy score of 0.9906, with similar performance across precision, recall, and F1 metrics but slightly lower ROC AUC. The IncepNet and ResNet models delivered competitive accuracy scores of 0.9900 each, with nearly identical precision and recall values. However, both models achieved a lower ROC AUC score (0.8820 for IncepNet and 0.8430 for ResNet), indicating a slightly lower ability in distinguishing between classes compared to the top-performing LeNet and CNN models.

In contrast, the FCN model exhibited the lowest performance among all evaluated models. It achieved an accuracy of 0.9860, with a precision score of 0.9859, recall of 0.9860, and F1-score of 0.9859. The lower precision score suggests that the FCN model had a higher proportion of misclassifications compared to the other models, underscoring its relative limitations in this analysis.

Taking into account all performance indicators, the LeNet model emerges as the most effective among the selected DTL models, followed closely by CNN. Both models exhibited the best stability in classification tasks. The competitive performance of IncepNet and ResNet highlights their potential but shows room for improvement in distinguishing class boundaries. On the other hand, the FCN model's lower performance metrics suggest it may not be as robust for genome sequence classification tasks.

## DFL metrics analysis

In this section, we have analyzed the DFL models with different numbers of client devices. We have considered 4, 6, 10, 15, and 20 client devices and 8 rounds for this analysis because the flattening characteristics of the curve and the performance metrics were not increasing literally. Table 3 shows the quantitative performance metrics summary of the selected DFL models.

First, we analyzed the performance of the model using 4 client devices. Figure 7a shows the classification performance of the DFL model under this configuration. The model's accuracy starts at 0.8350 and increases to 0.9869 by round 6, stabilizing through round 8. Similarly, the precision score begins at 0.5350 and gradually rises to 0.9785 by round 5, remaining stable until the final round. The recall score of the model increases rapidly in round 4, reaching a peak of approximately 0.9191 at round 6, and remains nearly stable up to round 8. The F1 score starts at its lowest value of 0.2702 in round 1, improves significantly to 0.9189 in round 3, and reaches 0.9393 in round 6, remaining constant thereafter. Lastly, the ROC AUC score begins at 0.4502, increases gradually with each round, reaching 0.9440 by round 5, and remains stable through the final round.

Next, we evaluated the model's performance with 6 client devices. Figure 7b shows the classification results for this configuration. The accuracy starts at 0.8830 and rises to 0.9912 by round 5, stabilizing through round 8. This demonstrates that the model provides more reliable classification performance than the configuration with 4 client devices. The precision score begins at 0.5490 and steadily increases to 0.9823 by round 5, remaining constant until the final round. This indicates that the model is more effective at identifying positive samples compared to the previous configuration. The recall score jumps rapidly in round 5 and peaks at approximately 0.9804 in round 6, remaining stable up to round 8. Similarly, the F1 score begins at its lowest value but improves significantly, reaching 0.9288 in round 4 and 0.9624 in round 7, and remains constant thereafter, reflecting the model's enhanced performance. Lastly, the ROC AUC score starts at 0.5508, rises consistently with each round, reaching 0.9824 by round 5, and remains stable until the final round. These results indicate that the 6-client configuration is more reliable, accurate, and efficient than the 4-client configuration.

To further analyze the impact of increasing client devices, we evaluated the model's performance with 10, 15, and 25 client devices. Figure 7c–e show the classification performance under these configurations. The accuracy scores were 0.9899, 0.7931, and 0.6646 for 10, 15, and 25 client devices, respectively, highlighting the reduction in classification reliability compared to the 4 and 6-client models. Similarly, the precision scores decreased as the number of clients increased, further demonstrating reduced accuracy in identifying positive samples.

The recall scores also decreased, with values of 0.8620, 0.7706, and 0.6280 for 10, 15, and 25 client devices, respectively. This indicates that these configurations are less effective at detecting positive samples compared to the 4 and 6-client models. Furthermore, the F1 scores were the lowest among all configurations, reflecting

| Model | Accuracy | Precision | Recall | F1Score | ROC |
|---|---|---|---|---|---|
| CNN | 0.9906 | 0.9903 | 0.9907 | 0.9902 | 0.8711 |
| FCN | 0.9860 | 0.9859 | 0.9860 | 0.9859 | 0.8687 |
| IncepNet | 0.9900 | 0.9896 | 0.9900 | 0.9897 | 0.8820 |
| LeNet | 0.9907 | 0.9903 | 0.9907 | 0.9901 | 0.8598 |
| ResNet | 0.9900 | 0.9901 | 0.9903 | 0.9898 | 0.8430 |

**Table 2**. Deep transfer learning models' different performance comparison metrics.

reduced overall performance. Lastly, the ROC AUC scores also decreased with more clients, further emphasizing the reduction in reliability and accuracy.

We have also calculated the confusion matrices for the DFL models across different client settings to analyze the classification performance in detail. Figure 8 shows the confusion matrices for the DFL models with different numbers of clients. The model with K = 6 clients demonstrated the best performance, achieving the highest true positives and true negatives, while maintaining minimal false positives and false negatives. In contrast, the performance started to decrease as the number of client devices increased. For instance, with K = 10, K = 15, and K = 20 clients, false positives and false negatives increased significantly, reducing the model's reliability. As the amount of training data on each local device decreased as the number of client devices increased. Therefore, models with 10, 15, and 20 client devices are less reliable, accurate, and efficient compared to those with 4 or 6-client devices.

*Theoretical analysis of accuracy trends*

We observe an initial increase in accuracy as the number of clients increases from k = 4 to k = 6. This trend suggests that incorporating more clients initially helps the model learn better by providing a more diverse set of data points for training, which improves generalization[70]. However, as the number of clients increases further k = 10, 15, 20, accuracy begins to decrease. This can be attributed to two main factors:

- *Smaller local datasets*: As the number of clients increases, the amount of data per client decreases. This reduction in local training data may hinder each client's ability to learn robust representations, leading to less effective updates sent to the central server[71].
- *Distributed learning challenges*: The aggregation process in distributed learning may fail to preserve certain learned patterns when too many clients participate, especially if data distributions are heterogeneous. This can result in a loss or superimposition of knowledge during the parameter aggregation process[72].

*Impact of distributed learnings*

Distributed learning can indeed affect learning patterns, especially when clients are provided with smaller datasets[71]. The models trained on limited, possibly biased datasets at each client may fail to capture sufficient diversity, leading to suboptimal contributions during the aggregation process. This challenge is amplified in cases with a high number of clients and non-IID data, where conflicting updates may reduce the overall model performance[71].

## Performance comparison of the models

We have considered different DTL and DFL models for their optimal performance. First, we applied different DTL models and selected the best model based on their performance. But, due to the sensitive nature of the patient's data, the DTL technique of transferring data to the central machine or server may create serious privacy and security issues. As a result, to ensure optimal performance, we compared the models across various metrics such as accuracy, precision, recall, and F1 score, where the proposed model performed significantly better in most cases. The overall result demonstrated that the proposed model has adequate efficiency, scalability, low loss, and better classification accuracy than others. Figure 9 shows the comparison of proposed DFL (K = 6) and DTL (LeNet) models in terms of (a) classification performance, (b) training time (sec), and (c) testing time (sec).

## Model training and testing time analysis

The training and testing time for each DTL and DFL model has been calculated and summarized in Table 4, which compares the total number of parameters, training time (in sec), and testing time (in sec) for each model. Among the DTL models, the CNN model achieves the minimum training and testing time, requiring 33,869.838 s and 2.718 s, respectively. In contrast, the FCN model demonstrates the maximum training and testing time, with a total of 265,986 trainable parameters, a training time of 46,756.338 s, and a testing time of 7.722 s. Additionally, the IncepNet model shows the second-highest training and testing time, requiring 36,657.213 s for training and 4.229 s for testing. Meanwhile, the LeNet model has a training time of 38,936.911 s and a testing time of 3.490 s, which is the second-longest testing time after the CNN model. Finally, the ResNet model, despite having more parameters (506,818), requires a slightly lower training and testing time than the FCN and IncepNet models, with 44,401.586 s for training and 4.014 s for testing.

Next, we analyzed the training and testing times of the DFL models, where all models share the same number of trainable parameters (350,985) and the same number of communications rounds but differ in the number of clients. In this comparison, the DFL model with 4 clients has the smallest training and testing times, requiring 38,242.180 s and 3.660 s, respectively. In contrast, the highest training and testing times are observed in the DFL

| DFL Model | Accuracy | Precision | Recall | F1Score | ROC AUC |
|---|---|---|---|---|---|
| K = 4 | 0.9869 | 0.9785 | 0.9191 | 0.9393 | 0.9440 |
| K = 6 | 0.9912 | 0.9823 | 0.9804 | 0.9624 | 0.9824 |
| K = 10 | 0.9899 | 0.9808 | 0.8620 | 0.8821 | 0.9422 |
| K = 15 | 0.7931 | 0.6783 | 0.7706 | 0.7543 | 0.7365 |
| K = 20 | 0.6646 | 0.6071 | 0.6280 | 0.6087 | 0.5692 |

**Table 3**. Deep federated learning models' performance comparison metrics in terms of client numbers.

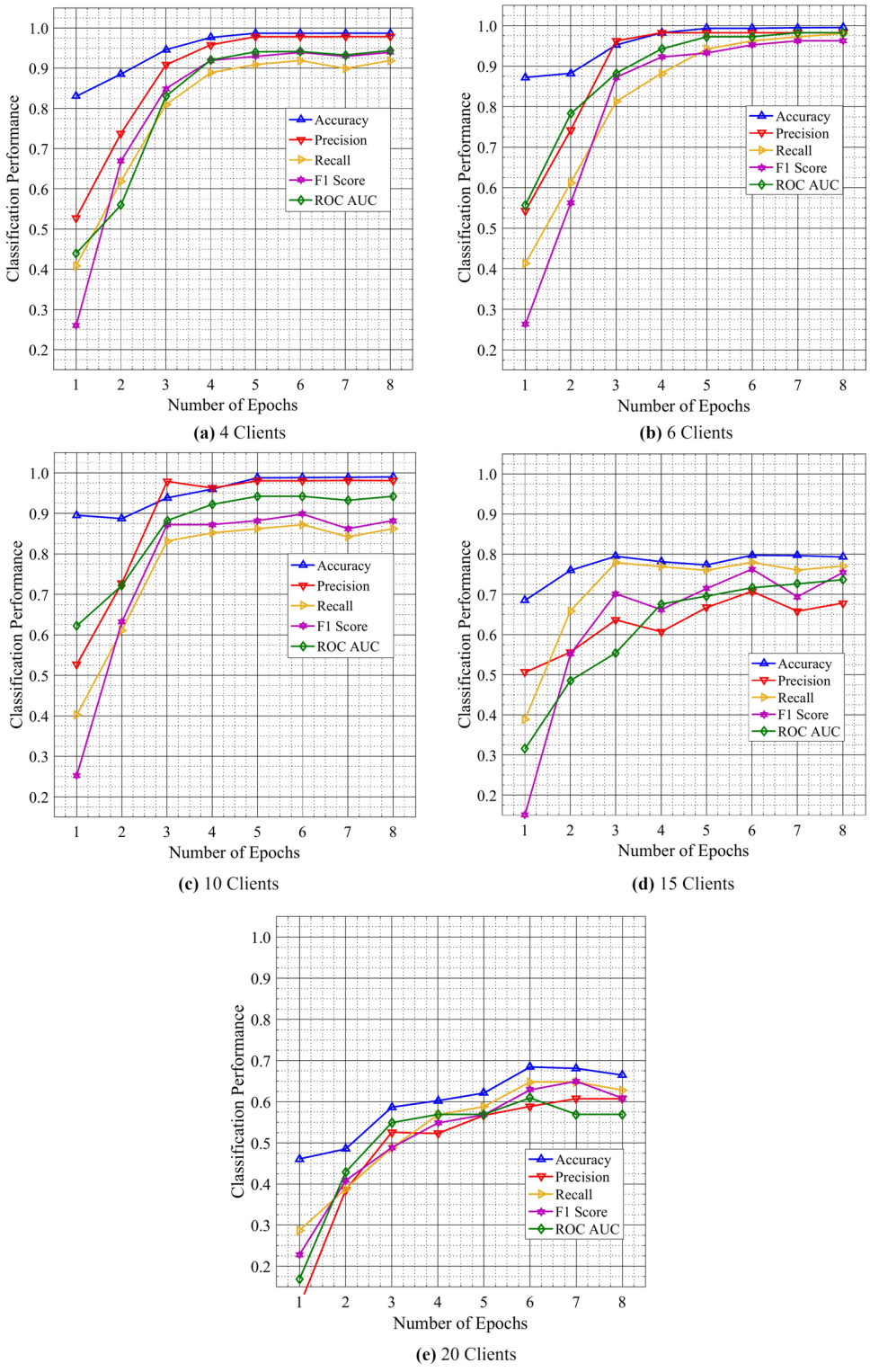



**Fig. 7**. Classification performance metrics comparison of the DFL model using different number of clients.

model with 20 clients, where training takes 49,921.078 s, and testing takes 6.940 s. This analysis reveals that the training and testing time of DFL models increases with the number of clients, as the communication and processing overhead grow. Among the configurations, the proposed DFL model with K = 6 clients achieve the best balance between performance and computational efficiency, requiring 41,928.245 s for training and 5.105 s for testing.

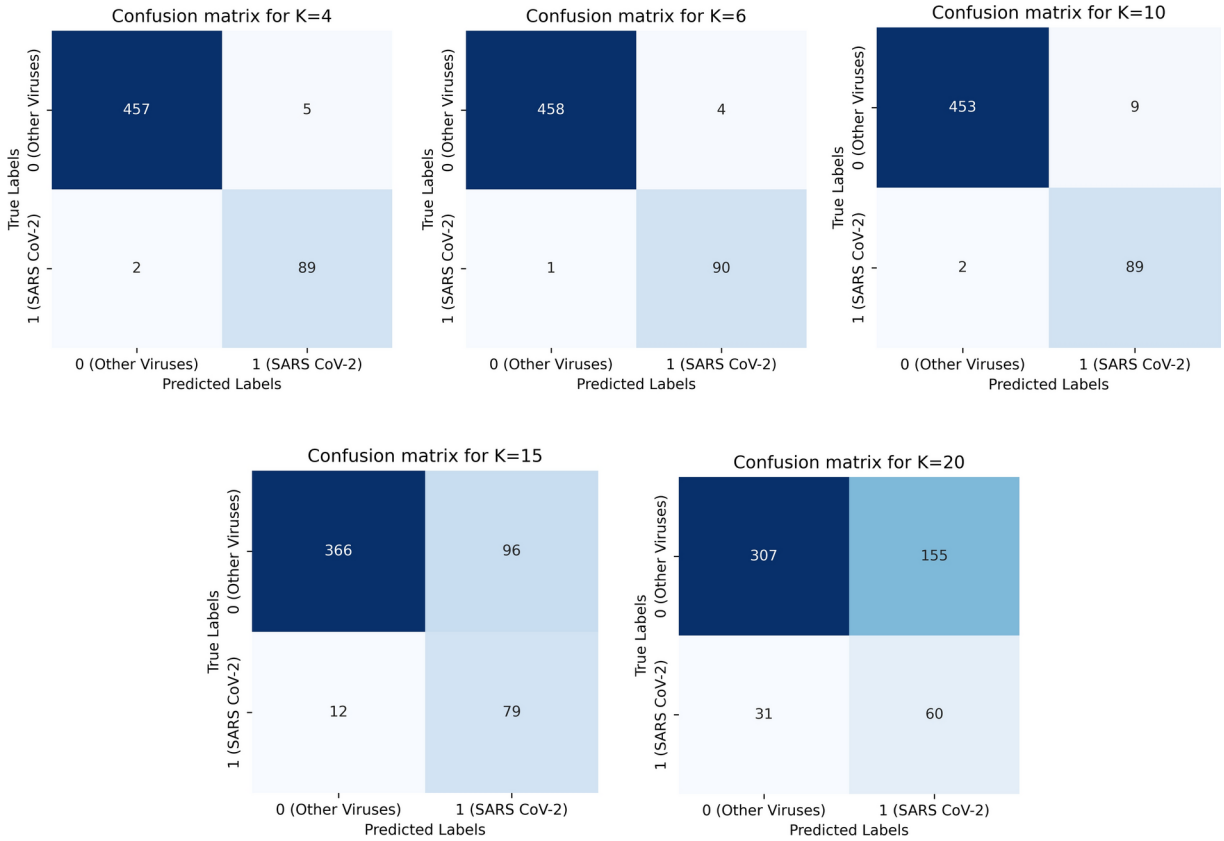



**Fig. 8**. Confusion matrices for the DFL models with different numbers of clients.

## Dependability performance analysis

In this section, we analyzed the dependability performance of the proposed model. The model's efficiency, availability, and scalability have been considered in the dependability performance analysis[18,19]. Figure 10 shows the scalability performance of the proposed model.

The proposed model outperforms several models with low training and testing time in terms of performance evaluations (e.g., accuracy, precision, recall, etc.), which ensures the efficiency of the model. Additionally, we have taken various strategies for choosing important features before implementing the proposed model to correctly classify both new and existing infections without experiencing any defects or requiring a repair technique, thus maintaining the availability feature of the proposed model. Finally, we observed an increase in the scalability properties of the proposed model by dividing the dataset among the different numbers of clients with maximum consistency that has been acquired from a wider range of DNA sequence data. As a result, the proposed model's accuracy remained almost the same with an increase in the round number from 3 to 8, which indicates the model's scalability.

## Limitations and future directions

While this study presents a privacy-preserving and dependable DFL-based model for identifying new infections from genome sequences, several limitations must be acknowledged. The computational overhead of federated learning, especially as the number of clients increased from 4 to 20, resulted in higher communication costs and longer training and testing times. The current study employs the FedAvg aggregation method, which, while effective, may not fully account for data heterogeneity or robustness against noisy updates. Future work will explore adaptive aggregation techniques to enhance performance metrics and address these challenges. Moreover, the study's results are based on the available genome sequence datasets, and further validation on larger and more diverse datasets will be considered.

Additionally, bounding the weights and operating in finite precision systems introduce further challenges. Constraining the range of weights, particularly by enforcing positivity, affects the statistical properties of the weight distribution, such as the mean and standard deviation. These constraints could lead to truncation errors, reduced model accuracy, or unexpected effects, especially when dealing with diverse datasets or boundary cases. Such challenges are particularly relevant in heterogeneous data environments and resource-constrained devices. Investigating these boundary scenarios and developing solutions like adaptive range constraints, re-projection techniques, or error correction mechanisms will be essential for ensuring robust and accurate aggregation. Recent studies have explored various strategies to address these issues. Ouyang et al.[73] introduced a dynamic weight adjustment mechanism using particle swarm optimization to improve aggregation efficiency and model

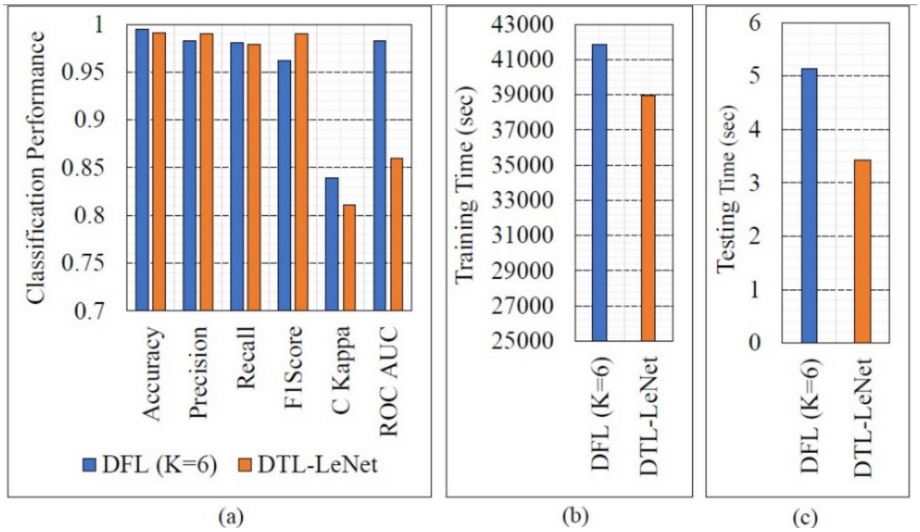



**Fig. 9**. Comparison of proposed deep federated learning (K = 6) and deep transfer learning (LeNet) models in terms of (**a**) classification performance, (**b**) training time (s), and (**c**) testing time (s).

| Model | Parameter | Train time (s) | Test time (s) |
|---|---|---|---|
| DTL | | | |
| CNN | 202,686 | 33,869.838 | 2.718 |
| FCN | 265,986 | 46,756.338 | 7.722 |
| IncepNet | 233,074 | 36,657.213 | 4.229 |
| LeNet | 360,052 | 38,936.911 | 3.490 |
| ResNet | 506,818 | 44,401.586 | 4.014 |
| DFL | | | |
| DFL (K = 4) | 350,985 | 38,242.180 | 3.660 |
| DFL (K = 6) | 350,985 | 41,928.245 | 5.105 |
| DFL (K = 10) | 350,985 | 42,763.788 | 5.667 |
| DFL (K = 15) | 350,985 | 47,921.078 | 6.140 |
| DFL (K = 20) | 350,985 | 49,921.078 | 6.940 |

**Table 4**. Comparison of parameters, training time (sec), testing time (sec) of deep transfer learning and deep federated learning model.

accuracy in federated learning systems. Furthermore, studies on model compression and quantization methods, such as hierarchical aggregation techniques[74], demonstrate promising approaches to mitigating communication overhead while maintaining model performance. These findings highlight the importance of designing adaptive algorithms for addressing precision-related challenges in federated systems.

Future work will focus on exploring these advanced techniques to enhance the proposed model. In addition to validating the framework on larger and more diverse datasets, we plan to implement the model on decentralized devices or servers, ensuring proper privacy and security of patient data through advanced techniques like differential privacy. By addressing these limitations and incorporating recent insights from the literature, we aim to improve the robustness, scalability, and practical applicability of the proposed privacy-preserving federated learning model for genome-based infection identification.

## Conclusion

In this research, a privacy-preserving and dependable deep-federated learning-based model for identifying new infections from genome sequences has been proposed. Our model demonstrates improved performance metrics compared to other existing methods. The proposed model has an overall accuracy of 99.12% after distributing the dataset independently and equally among six clients. In addition, the proposed model outperforms the existing models in terms of other performance metrics for a similar dataset. This elucidates that the proposed model can effectively and efficiently classify genome sequences by ensuring dependability, minimum computational complexity, and proper privacy and security of patients' data. The proposed paradigm has also demonstrated its potential to protect other critical medical infrastructures where dependable identification models and secure data processing are the major challenges. More specifically, the proposed model provides a complete guide for future researchers in this domain.

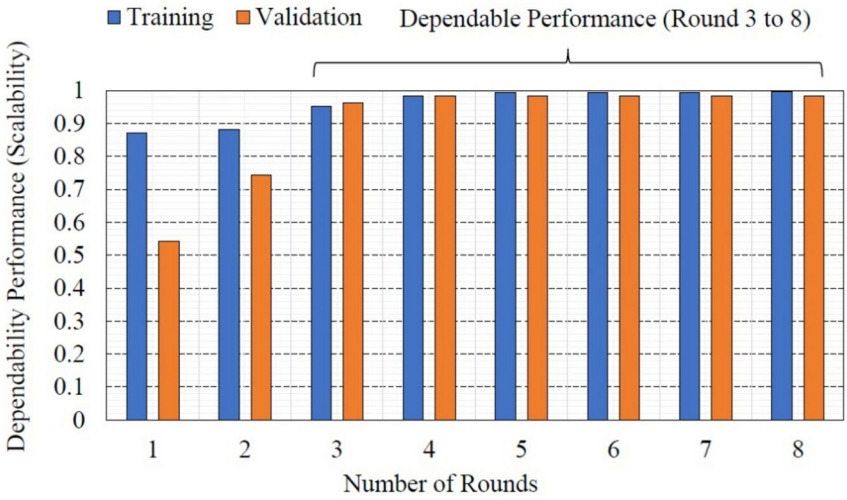



**Fig. 10**. Dependability performance analysis (scalability) of the proposed deep federated learning model with 6 client devices.

## Data availability

The datasets and code generated and/or analysed during the current study are available in the GitHub repository. Link: https://github.com/tanzirmehedi/A-Privacy-Preserving-and-Dependable-DFL-Based-Model-for-Identifying-New-Infections-from-GS.

Received: 24 September 2024; Accepted: 6 February 2025
Published online: 01 March 2025

## References

1. Wu, F. et al. A new coronavirus associated with human respiratory disease in China. *Nature.* **579**, 265–269. https://doi.org/10.1038/s41586-020-2008-3 (2020).
2. Lu, R. et al. Genomic characterisation and epidemiology of 2019 novel coronavirus: implications for virus origins and receptor binding. *The Lancet* **395**, 565–574. https://doi.org/10.1016/S0140-6736(20)30251-8 (2020).
3. Marston, D. A. et al. Next generation sequencing of viral RNA genomes. *BMC Genomics.* **14**, 444. https://doi.org/10.1186/1471-2164-14-444 (2013).
4. Sohrabi, C. et al. World Health Organization declares global emergency: A review of the 2019 novel coronavirus (COVID-19). *Int. J. Surg.* **76**, 71–76. https://doi.org/10.1016/j.ijsu.2020.02.034 (2020).
5. Cucinotta, D. & Vanelli, M. WHO declares COVID-19 a pandemic. *Acta Bio Medica Atenei Parm.* **91**, 157–160. https://doi.org/10.23750/abm.v91i1.9397 (2020).
6. Whata, A. & Chimedza, C. Deep learning for SARS COV-2 genome sequences. *IEEE Access.* **9**, 59597–59611. https://doi.org/10.1109/ACCESS.2021.3073728 (2021).
7. Lopez-Rincon, A., Tonda, A. & Mendoza-Maldonado, L. OPEN classification and specific primer design for accurate detection. *Sci. Rep.* (n.d.).
8. Metsky, H. C., Freije, C. A., Kosoko-Thoroddsen, T.-S.F., Sabeti, P. C. & Myhrvold, C. CRISPR-based surveillance for COVID-19 using genomically-comprehensive machine learning design. *Genomics* https://doi.org/10.1101/2020.02.26.967026 (2020).
9. Yang, Y. et al. Laboratory diagnosis and monitoring the viral shedding of SARS-CoV-2 infection. *The Innovation* **1**, 100061. https://doi.org/10.1016/j.xinn.2020.100061 (2020).
10. Zhao, J. et al. Antibody responses to SARS-CoV-2 in patients with novel coronavirus disease 2019. *Clin. Infect. Dis.* **71**, 2027–2034. https://doi.org/10.1093/cid/ciaa344 (2020).
11. Long, C. et al. Diagnosis of the coronavirus disease (COVID-19): rRT-PCR or CT?. *Eur. J. Radiol.* **126**, 108961. https://doi.org/10.1016/j.ejrad.2020.108961 (2020).
12. Arevalo-Rodriguez, I. et al. False-negative results of initial RT-PCR assays for COVID-19: A systematic review. *PLOS ONE.* **15**, e0242958. https://doi.org/10.1371/journal.pone.0242958 (2020).
13. Neranjan Thilakarathne, N. et al. Federated learning for privacy-preserved medical internet of things. *Intell. Autom. Soft Comput.* **33**, 157–172. https://doi.org/10.32604/iasc.2022.023763 (2022).
14. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D. & Chandra, V. Federated learning with non-IID data (2018). https://doi.org/10.48550/arXiv.1806.00582.
15. Bustin, S. A. & Nolan, T. RT-qPCR testing of SARS-CoV-2: A primer. *Int. J. Mol. Sci.* **21**, 3004. https://doi.org/10.3390/ijms21083004 (2020).
16. Yuan, B., Ge, S. & Xing, W. A federated learning framework for healthcare IoT devices. (2020). http://arxiv.org/abs/2005.05083 (Accessed February 21, 2023).
17. H. Chen, H. Li, G. Xu, Y. Zhang, X. Luo, Achieving privacy-preserving federated learning with irrelevant updates over E-health applications. In *ICC 2020–2020 IEEE International Conference on Communication ICC* 1–6 (IEEE, Dublin, 2020). https://doi.org/10.1109/ICC40277.2020.9149385.
18. Bhuiyan, M. Z. A., Kuo, S., Lyons, D. & Shao, Z. Dependability in cyber-physical systems and applications. *ACM Trans. Cyber-Phys. Syst.* **3**, 1–4. https://doi.org/10.1145/3271432 (2019).
19. Mehedi, S. K. T., Anwar, A., Rahman, Z., Ahmed, K. & Islam, R. Dependable intrusion detection system for IoT: A deep transfer learning based approach. *IEEE Trans. Ind. Inform.* **19**, 1006–1017. https://doi.org/10.1109/TII.2022.3164770 (2023).
20. Billah, M., Mehedi, S. T., Anwar, A., Rahman, Z. & Islam, R. A systematic literature review on Blockchain enabled federated learning framework for internet of vehicles (2022). http://arxiv.org/abs/2203.05192 (Accessed February 21, 2023).

21. Lo, S. K., Lu, Q., Wang, C., Paik, H.-Y. & Zhu, L. A systematic literature review on federated machine learning: From a software engineering perspective. *ACM Comput. Surv.* **54**, 1–39. https://doi.org/10.1145/3450288 (2022).
22. Kaissis, G. A., Makowski, M. R., Rückert, D. & Braren, R. F. Secure, privacy-preserving and federated machine learning in medical imaging. *Nat. Mach. Intell.* **2**, 305–311. https://doi.org/10.1038/s42256-020-0186-1 (2020).
23. Zerka, F., Barakat, S., Walsh, S., Bogowicz, M., Leijenaar, R.T.H., Jochems, A., Miraglio, B., Townend, D. & Lambin, P. Systematic review of privacy-preserving distributed machine learning from federated databases in health care. *JCO Clin. Cancer Inform.* 184–200 https://doi.org/10.1200/CCI.19.00047 (2020).
24. Li, T., Sahu, A. K., Talwalkar, A. & Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **37**, 50–60. https://doi.org/10.1109/MSP.2020.2975749 (2020).
25. Xu, J. et al. Federated learning for healthcare informatics. *J. Healthc. Inform. Res.* **5**, 1–19. https://doi.org/10.1007/s41666-020-00082-4 (2021).
26. Yang, Q., Liu, Y., Chen, T. & Tong, Y. Federated machine learning: concept and applications (2019). http://arxiv.org/abs/1902.04885 (Accessed February 21, 2023).
27. Li, B. et al. DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems. *IEEE Trans. Ind. Inform.* **17**, 5615–5624. https://doi.org/10.1109/TII.2020.3023430 (2021).
28. Khan, L. U., Saad, W., Han, Z., Hossain, E. & Hong, C. S. Federated learning for internet of things: Recent advances, taxonomy, and open challenges (2021). http://arxiv.org/abs/2009.13012 (Accessed February 21, 2023).
29. Fang, H. & Qian, Q. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet* **13**, 94. https://doi.org/10.3390/fi13040094 (2021).
30. Wu, Q., He, K. & Chen, X. Personalized federated learning for intelligent IoT applications: A cloud-edge based framework. *IEEE Open J. Comput. Soc.* **1**, 35–44. https://doi.org/10.1109/OJCS.2020.2993259 (2020).
31. Aledhari, M., Razzak, R., Parizi, R. M. & Saeed, F. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access* **8**, 140699–140725. https://doi.org/10.1109/ACCESS.2020.3013541 (2020).
32. Dou, Q. et al. Federated deep learning for detecting COVID-19 lung abnormalities in CT: A privacy-preserving multinational validation study. *Npj Digit. Med.* **4**, 60. https://doi.org/10.1038/s41746-021-00431-6 (2021).
33. McMahan, H. B., Moore, E., Ramage, D., Hampson, S. & Arcas, B. A. Y (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)* 1273–1282 (Fort Lauderdale, FL, USA).
34. Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A. & Smith, V. Federated optimization in heterogeneous networks. In *Proceedings of the 4th MLSys Conference* (San Francisco, CA, USA, 2020).
35. Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. & Suresh, A. T. SCAFFOLD: Stochastic controlled averaging for federated learning. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)* 5132–5143 (Vienna, Austria, 2020).
36. Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S. & McMahan, H. B. Adaptive federated optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)* (Addis Ababa, Ethiopia, 2021).
37. Randhawa, G. S. et al. Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens: COVID-19 case study. *PLOS ONE.* **15**, e0232391. https://doi.org/10.1371/journal.pone.0232391 (2020).
38. Zeng, H., Edwards, M. D., Liu, G. & Gifford, D. K. Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics* **32**, i121–i127. https://doi.org/10.1093/bioinformatics/btw255 (2016).
39. Zou, J. et al. A primer on deep learning in genomics. *Nat. Genet.* **51**, 12–18. https://doi.org/10.1038/s41588-018-0295-5 (2019).
40. Seo, S., Oh, M., Park, Y. & Kim, S. DeepFam: Deep learning based alignment-free method for protein family modeling and prediction. *Bioinformatics* **34**, i254–i262. https://doi.org/10.1093/bioinformatics/bty275 (2018).
41. Nguyen, N. G. et al. DNA sequence classification by convolutional neural network. *J. Biomed. Sci. Eng.* **09**, 280–286. https://doi.org/10.4236/jbise.2016.95021 (2016).
42. Zhang, H., Hung, C.-L., Liu, M., Hu, X. & Lin, Y.-Y. NCNet: Deep learning network models for predicting function of non-coding DNA. *Front. Genet.* **10**, 432. https://doi.org/10.3389/fgene.2019.00432 (2019).
43. Zhang, Y., Qiao, S., Ji, S. & Li, Y. DeepSite: bidirectional LSTM and CNN models for predicting DNA–protein binding. *Int. J. Mach. Learn. Cybern.* **11**, 841–851. https://doi.org/10.1007/s13042-019-00990-x (2020).
44. Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning–based sequence model. *Nat. Methods.* **12**, 931–934. https://doi.org/10.1038/nmeth.3547 (2015).
45. Hinton, G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **29**, 82–97. https://doi.org/10.1109/MSP.2012.2205597 (2012).
46. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Commun. ACM.* **60**, 84–90. https://doi.org/10.1145/3065386 (2017).
47. Roy, P. K., Singh, J. P. & Banerjee, S. Deep learning to filter SMS spam. *Future Gener. Comput. Syst.* **102**, 524–533. https://doi.org/10.1016/j.future.2019.09.001 (2020).
48. Ordóñez, F. & Roggen, D. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors* **16**, 115. https://doi.org/10.3390/s16010115 (2016).
49. Johnson, R. & Zhang, T. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference on North American Chapter Associate Computer Linguistics Human Language Technology* 103–112 (Association for Computational Linguistics, Denver, Colorado, 2015). https://doi.org/10.3115/v1/N15-1011.
50. Ravi, D. et al. Deep learning for health informatics. *IEEE J. Biomed. Health Inform.* **21**, 4–21. https://doi.org/10.1109/JBHI.2016.2636665 (2017).
51. Schneble, W. & Thamilarasu, G. Attack detection using federated learning in medical cyber-physical systems (n.d.).
52. Lu, Y., Huang, X., Dai, Y., Maharjan, S. & Zhang, Y. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. *IEEE Trans. Ind. Inform.* **16**, 4177–4186. https://doi.org/10.1109/TII.2019.2942190 (2020).
53. Mehedi, S. K. T. et al. MLBioIGE: Integration and interplay of machine learning and bioinformatics approach to identify the genetic effect of SARS-COV-2 on idiopathic pulmonary fibrosis patients. *Biol. Methods Protoc.* **7**, 013. https://doi.org/10.1093/biomethods/bpac013 (2022).
54. Ren, J. et al. Identifying viruses from metagenomic data using deep learning. *Quant. Biol.* **8**, 64–77. https://doi.org/10.1007/s40484-019-0187-4 (2020).
55. Kumar, R., Khan, A. A., Zhang, S., Kumar, J., Yang, T., Golalirz, N. A., Zakria, Ali, I., Shafiq, S. & Wang, W. Blockchain-federated-learning and deep learning models for COVID-19 detection using CT imaging. *IEEE Sens. J.* **21**, 16301–16314. https://doi.org/10.1109/JSEN.2021.3076767 (2021).
56. Roth, H. R., Chang, K., Singh, P., Neumark, N., Li, W., Gupta, V., Gupta, S., Qu, L., Ihsani, A., Bizzo, B. C., Wen, Y., Buch, V., Shah, M., Kitamura, F., Mendonça, M., Lavor, V., Harouni, A., Compas, C., Tetreault, J., Dogra, P., Cheng, Y., Erdal, S., White, R., Hashemian, B., Schultz, T., Zhang, M., McCarthy, A., Yun, B. M., Sharaf, E., Hoebel, K. V., Patel, J. B., Chen, B., Ko, S., Leibovitz, E., Pisano, E. D., Coombs, L., Xu, D., Dreyer, K. J., Dayan, I., Naidu, R. C., Flores, M., Rubin, D. & Kalpathy-Cramer, J. Federated learning for breast density classification: A real-world implementation. In 181–191. https://doi.org/10.1007/978-3-030-60548-3_18 (2020).
57. Qayyum, A., Ahmad, K., Ahsan, M. A., Al-Fuqaha, A. & Qadir, J. Collaborative federated learning for healthcare: Multi-modal COVID-19 diagnosis at the edge (2021). http://arxiv.org/abs/2101.07511 (accessed February 21, 2023).

58. Tampuu, A., Bzhalava, Z., Dillner, J. & Vicente, R. ViraMiner: Deep learning on raw DNA sequences for identifying viral genomes in human samples. *PLOS ONE* **14**, e0222271. https://doi.org/10.1371/journal.pone.0222271 (2019).
59. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
60. Paillier, P. Paillier encryption and signature schemes. In *Encyclopedia of Cryptography and Security* (eds van Tilborg, H. C. A., Jajodia, S.) (Springer, Boston, MA 2011). https://doi.org/10.1007/978-1-4419-5906-5_488
61. Jamgekar, R. S. & Joshi, G. S. File encryption and decryption using secure RSA. *Int. J. Emerg. Sci. Eng. (IJESE)* **1**, 11–14 (2013).
62. Thangavel, M., Varalakshmi, P., Murrali, M. & Nithya, K. An enhanced and secured RSA key generation scheme (ESRKGS). *J. Inf. Sec. Appl.* **20**, 3–10 (2015).
63. K. Balasubramanian, Variants of RSA and their cryptanalysis. In *2014 International Conference on Communication and Network Technologies* 145–149 (Sivakasi, India, 2014).
64. Jost, C., Lam, H., Maximov, A. & Smeets, B. Encryption performance improvements of the paillier cryptosystem. Cryptol. ePrint Arch. (2015). https://eprint.iacr.org/2015/864.pdf
65. Abdul Salam, M., Taha, S. & Ramadan, M. COVID-19 detection using federated machine learning. *PLOS ONE* **16**, e0252573. https://doi.org/10.1371/journal.pone.0252573 (2021).
66. Li, W., Milletarì, F., Xu, D., Rieke, N., Hancox, J., Zhu, W., Baust, M., Cheng, Y., Ourselin, S., Cardoso, M. J. & Feng, A. Privacy-preserving federated brain tumour segmentation (2019). http://arxiv.org/abs/1910.00962 (Accessed February 21, 2023).
67. Guyon, I. & Laboratories, T. B. A scaling law for the validation-set training-set size ratio (n.d.).
68. Wu, X. et al. Top 10 algorithms in data mining. *Knowl. Inf. Syst.* **14**, 1–37. https://doi.org/10.1007/s10115-007-0114-2 (2008).
69. Mehedi, S. K. T., Anwar, A., Rahman, Z. & Ahmed, K. Deep transfer learning based intrusion detection system for electric vehicular networks. *Sensors* **21**, 4736. https://doi.org/10.3390/s21144736 (2021).
70. Ji, S. et al. Emerging trends in federated learning: from model fusion to federated X learning. *Int. J. Mach. Learn. Cyber.* **15**, 3769–3790. https://doi.org/10.1007/s13042-024-02119-1 (2024).
71. Zhu, H., Xu, J., Liu, S. & Jin, Y. Federated learning on non-IID data: A survey. *Neurocomputing* **465**, 371–390. https://doi.org/10.1016/j.neucom.2021.07.098 (2021).
72. Lu, Z., Pan, H., Dai, Y., Si, X. & Zhang, Y. Federated learning with non-IID data: A survey. *IEEE Internet Things J.* **11**(11), 19188–19209 (2024).
73. Ouyang, J., Wang, L. & Zhou, H. Dynamic weight adjustment for federated learning using particle swarm optimization. *Springer J. Intell. Inf. Syst.* https://doi.org/10.1007/s10791-024-09478-x (2024).
74. Han, Y., Chen, S. & Yang, F. Efficient hierarchical aggregation in federated learning systems. *Springer Int. J. Data Sci. Anal. (IJDS)* https://doi.org/10.1007/s41060-024-00691-x (2024).


## Acknowledgements

The authors extend their appreciation to Umm Al-Qura University in Saudi Arabia for funding this research work through the Project number: 4170008.

## Author contributions

Conceptualization was done by K.A., M.S.U., S.T.M.; Data curation, Formal analysis, Investigation, and Methodology by S.T.M., K.A.; Funding acquisition by F.M.B., L.C., F.A.A.; Project administration by M.S.U., K.A., F.M.B., M.A.M.; Resources, Software by S.T.M., K.A.; Supervision by M.S.U., K.A.; Validation by S.T.M., K.A., L.F.A., M.A.M.; Visualization and Writing of the original draft by S.T.M.; Writing: review editing by S.T.M., L.F.A., K.A., M.S.U., F.M.B., L.C. F.A.A., and M.A.M.

## Funding

This work funded by Umm Al-Qura University, Makkah, Saudi ArabiaProject Number: 4170008.

## Declarations

## Competing interests

The authors declare no competing interests.

## Additional information

**Correspondence** and requests for materials should be addressed to K.A. or F.A.A.-Z.

**Reprints and permissions information** is available at www.nature.com/reprints.

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.