Privacy-Preserving and Dependable Identification Model of New Infections

Sk. Tanzir Mehedi

A thesis in fulfillment of the requirements for the degree of

Master of Science in Engineering (M.Sc (Engg.))

Department of Information and Communication Technology (ICT)

Faculty of Engineering

Mawlana Bhashani Science and Technology University (MBSTU)

Santosh, Tangail-1902, Bangladesh

May 2023

MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY (MBSTU)
Thesis Sheet

Name: **Sk. Tanzir Mehedi**

Student ID: **IT18612**

Abbreviation for degree as given in the University calendar: **M.Sc (Engg.)**

Department: **Information and Communication Technology (ICT)** Faculty: **Faculty of Engineering**

Title: **Privacy-Preserving and Dependable Identification Model of New Infections**

Abstract

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 serious 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, this thesis work proposes 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 the identification of 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. The dependable proposed model has an overall accuracy of 99.12% after independently and identically distributing (IID) the dataset among 6 clients. Moreover, the proposed model has a precision of 98.23%, recall of 98.04%, f1-score of 96.24%, Cohens 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 the identification of new infections amongst other virus strains from genome sequences by ensuring proper privacy and security of patients' data.

Declaration relating to disposition of thesis

I hereby declare that the work which is being presented in the thesis, entitled "Privacy-Preserving and Dependable Identification Model of New Infections", in partial fulfillment of the requirement for the award of the degree of Master of Science in Engineering (M.Sc (Engg.)) in Information and Communication Technology (ICT), and submitted to Mawlana Bhashani Science and Technology University (MBSTU), Tangail, Bangladesh, is an original piece of research work under the guidance of **Dr. Muhammad Shahin Uddin**, Professor, Department of ICT, MBSTU. The matter embodied in this thesis has not been submitted by me for the award of any other degree from any other university or institute.

Signature ..... Date .....

FOR OFFICE USE ONLY Official seal of the department and date of completion of requirements for Award

Originality Statement

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

To the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights, and any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee, and that this thesis has not been submitted for a higher degree to any other university or institution.

Signed .....

Date .....

Copyright Statement

Signed .....

Date .....

Authenticity Statement

I herewith declare that I wrote this thesis on my own and did not use any unnamed sources or aid. Thus, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made by correct citation. This includes any thoughts taken over directly or indirectly from printed books and articles as well as all kinds of online material. It also includes my own translations from sources in a different languages.

The work contained in this thesis has not been previously submitted for examination.

I also agree that the thesis may be tested for plagiarized content with the help of plagiarism software. I am aware that failure to comply with the rules of good scientific practice has grave consequences and may result in expulsion from the program.

Finally, I certify that the library deposit digital copy is a direct equivalent of the final, officially approved version of my thesis.

Signed .....

Date .....

Abstract

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 serious 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, this thesis work proposes 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 the identification of 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. The dependable proposed model has an overall accuracy of 99.12% after independently and identically distributing (IID) the dataset among 6 clients. Moreover, the proposed model has a precision of 98.23%, recall of 98.04%, f1-score of 96.24%, Cohens 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 the identification of new infections amongst other virus strains from genome sequences by ensuring proper privacy and security of patients' data.

Keywords

- Computational Complexity
- Coronavirus
- Dependability
- Deep Federated Learning
- Deep Transfer Learning
- Encryption and Decryption
- Federated Learning
- Genome Sequence
- Machine Learning
- Privacy and Security
- Scalability and Efficiency

Dedication

Every challenging work needs self efforts as well as guidance of elders especially those who were very close to our heart.

My humble effort I dedicated to my sweet and loving

Father & Mother

Whose affection, love, encouragement, and prays of day and night make me able to get such success and honour.

Along with all hard working, source of inspiration, and respected

Teachers

Whose guidance, wisdom, and prays that make me able to get such achievements.

Acknowledgement

All the praise and gratefulness go to the Almighty Allah for giving me perseverance in work, and the ability and intelligence to complete it, as well as submit my thesis work successfully without any major problems. I am extremely grateful to my parents for their love, prayers, care, and sacrifices in educating and preparing me for my future.

I would like to express my deep and sincere gratitude to my research supervisor, **Dr. Muhammad Shahin Uddin**, Professor, Department of Information and Communication Technology (ICT), Mawlana Bhashani Science and Technology University (MBSTU), Tangail, Bangladesh, for giving me the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity, and motivation have deeply inspired me. He has taught me the methodology to carry out the research and to present the research work as clearly as possible. It was a great privilege and honor to work and study under his guidance. I am extremely grateful for what he has offered me. I would also like to thank him for his friendship, empathy, and great sense of humor. His guidance, punctuality, judgment, and simplicity will be remembered in my memory.

I also want to express my gratitude to Kawsar Ahmed, Associate Professor, Department of ICT, MBSTU, for his invaluable support and guidelines.

Moreover, I would like to acclaim all the other respected teachers who have helped me directly or indirectly by providing their valuable support in completing this work.

I would like to extend my deepest gratitude to the Information and Communication Technology (ICT) division of the Government of the Peoples Republic of Bangladesh for granting me a fellowship (Grant number-20FS24067) for this research.

Also, I would like to thank everyone who has helped me in my research journey. I am indebted to those who have helped me directly or indirectly in completing this work.

Finally, I forward my special regards to all my friends who have helped, inspired, and given me mental support at different points during the successful completion of my research work.

List of Publications

Journals

[1] **S. T. Mehedi**, A. Anwar, Z. Rahman, K. Ahmed and I. Rafiqul, Dependable Intrusion Detection System for IoT: A Deep Transfer Learning-based Approach, *IEEE Transactions on Industrial Informatics*, 2022.

Impact Factor: 11.648, **Cite Score**: 21.30, **Category**: Q1 (Engineering)
DOI: *https://doi.org/10.1109/TII.2022.3164770*

[2] **S. T. Mehedi**, A. Anwar, Z. Rahman, and K. Ahmed, Deep Transfer Learning-based Intrusion Detection System for Electric Vehicular Networks, *Sensors*, vol. 21, no. 14, pp. 4736, 2021.

Impact Factor: 3.847, **Cite Score**: 6.40, **Category**: Q1 (Analytical Physics)
DOI: *https://doi.org/10.3390/s21144736*

[3] M. Billah, **S. T. Mehedi**, A. Anwar, Z. Rahman, and I. Rafiqul, A Systematic Literature Review on Blockchain Enabled Federated Learning Framework for Internet of Vehicles, *IEEE Access*, 2022.

Impact Factor: 3.847, **Cite Score**: 6.70, **Category**: Q1 (Computer Science)
DOI: *https://doi.org/10.48550/arXiv.2203.05192*

[4] **S. T. Mehedi**, Z. Rahman, X. Yi, I. Rafiqul, and A. Kelarev, Blockchain Applicability for the Internet of Things: Performance and Scalability Challenges and Solutions, *Electronics*, vol. 11, no. 2, pp. 1416, 2022.

Impact Factor: 2.690, **Cite Score**: 3.70, **Category**: Q2 (Computer Science)
DOI: *https://doi.org/10.3390/electronics11091416*

[5] **S. T. Mehedi**, K. Ahmed, F. M. Bui, M. Rahaman, I, Hossain, T. M. Tonmoy, R. A. Limon, S. M. Ibrahim, and M. A. Moni, MLBioIGE: Integration and Interplay of Machine Learning and Bioinformatics Approach to Identify the Genetic Effect of SARS-COV-2 on Idiopathic Pulmonary Fibrosis Patients, *Biology Methods and Protocols*, vol. 7, no. 1, 2022.

Impact Factor: 2.110, **Cite Score**: 2.40, **Category**: Q2 (Biological Sciences)
DOI: *https://doi.org/10.1093/biomethods/bpac013*

[6] **S. T. Mehedi**, A. A. M. Shamim, B. K. Paul, K. Ahmed, Graphene Injected D-Shape Photonic Crystal Fiber for Nonlinear Optical Applications, *Silicon*, vol. 12, pp. 2293-2300, 2020.

Impact Factor: 2.941, **Cite Score**: 4.60, **Category**: Q2 (Materials Science)
DOI: *https://doi.org/10.1007/s12633-019-00323-1*

[7] **S. T. Mehedi**, A. A. M. Shamim, M. B. A. Miah, Blockchain-based Security Nanagement of IoT Infrastructure with Ethereum Transactions, *Iran Journal of Computer Science*, vol. 2, pp. 189-195, 2019.

Impact Factor: 1.590, **Cite Score**: 2.30, **Category**: Q3 (Computer Science)
DOI: *https://doi.org/10.1007/s42044-019-00044-z*54RX

Conferences

[1] J. Ferdows, **S. T. Mehedi**, A. S. M. Hossain, A. A. M. Shamim, G. M. R. I. Rasiq, A Comprehensive Study of IoT Application Layer Security Management, *IEEE International Conference for Innovation in Technology (INOCON)*, pp. 1-7, 2020.

Conference Rank: B, **Location**: Bengaluru, India
DOI: *https://doi.org/10.1109/INOCON50539.2020.9298245*

Book Chapters

[1] A. A. Sefat, G. M. R. I. Rasiq, N. Nawjis, **S. T. Mehedi**, CPU Performance Prediction Using Various Regression Algorithms, *Proceedings of International Conference on Machine Intelligence and Data Science Applications, Algorithms for Intelligent Systems (AIS)*, Springer, Singapore, pp. 163-171, 2021.

Conference Rank: B, **Location**: Singapore
DOI: *https://doi.org/10.1007/978-981-33-4087-9_14*

N.B. Journal Impact Factor, Journal Cite Score, Journal Category, and Conference Rank are based on Scopus and Conference Ranks website (Accessed: May 2023).

Honors and Awards

List of Honors and Awards

- Certificate of Appreciation: Trainer, Ethical Hacking and Cybersecurity in Power Sector, Operational Perspectives, Bangladesh Power Development Board, 2022.
- Certificate of Appreciation: Coach, BUET Inter-University Programming Contest 2022, BUET CSE Fest, 2022.
- Certificate of Appreciation: Coach, Cefalo CodeFiesta 2022: AUST Inter-University Programming Contest, 2022.
- ICT Fellowship: M.Sc (Engg.) Fellowship, ICT Division, Bangladesh - Government of The People's Republic of Bangladesh, 2020
- Best Trainee Award: Selected as one of the top 5 trainees among 50 IT Engineers, Bangladesh Hi-Tech Park Authority (BHTPA) and ICT Division, Bangladesh - Government of The People's Republic of Bangladesh, 2019
- Merit Scholarships: Faulty of Engineering, Mawlana Bhashani Science and Technology University (MBSTU), B.Sc (Engg.) exam in every year from 2015 to 2018.
- Award: Project Show (Best Innovative Idea) event at Digital Innovation Fairs, 2018.
- Appreciation Certificate: EEE Day Inter-University Project/Thesis Presentation, Bangladesh University of Engineering and Technology (BUET), 2018.
- Certificate of Achievement: Inter-University Programming Contest in the 1st MBSTU CSE Carnival, 2017.
- Certificate of Achievement: AB Bank - Islamic University of Technology (IUT) 8th ICT Fest Programming Contest, 2016.
- Memorandum: Letter of Congratulatory, Science Olympiad Event by the Boson Science Olympiad, Tangail, 2015.
- Award: Letter of Congratulatory for SSC Scholarship Result, Event by Prothom Alo-Robi Award Ceremony, 2010.

Contents

List of Figures

List of Tables

List of Acronyms

A

ACTG	Adenine, Cytosine, Thymine, Guanine

B

BC	Blockchain
BFL	Blockchain-based Federated Learning
BiLSTM	Bidirectional Long-Short-Term Memory
BLAST	Basic Local Alignment Search Tool
BWA	Burrows-Wheeler Aligner

C

cDNA	Complementary Deoxyribonucleic Acid
CFL	Centralized Federated Learning
CNN	Convolution Neural Network
COVID	Coronavirus Disease
CoX	Computation Complexity

D

DeFL	Decentralized Federated Learning
DFL	Deep Federated Learning
DL	Deep Learning
DNA	Deoxyribonucleic Acid
DTL	Deep Transfer Learning

F

FCN	Fully Convolutional Networks
FedSGD	Federated Stochastic Gradient Descent
FedAvg	Federated Averaging

FedDyn	FL with Dynamic Regularization
FL	Federated Learning
FML	Federated Machine Learning

G

GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
GS	Genome Sequence

H

HFL	Horizontal Federated Learning
HeFL	Heterogeneous Federated Learning
HIPAA	Health Insurance Portability and Accountability Act

I

IID	Independently and Identically Distribution
IncepNet	Inception Network

L

LeNet	LeCun Network

M

ML	Machine Learning

N

NCNet	Neighbourhood Consensus Networks
NGS	Next-Generation Sequencing

P

P2P	Peer-to-Peer
PCR	Polymerase Chain Reaction
PFL	Personalized Federated Learning

Q

qPCR	Quantitative PCR

R

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

S

SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus
SSD	Solid State Drive
SVM	Support Vector Machine
Sub-FedAvg	Personalized FL by Pruning

T

TFL	Transfer Federated Learning
TMID	Traditional Molecular-based Identification
TML	Traditional Machine Learning

V

VFL	Vertical Federated Learning
VGS	Viral Genome Sequence

W

WHO	World Health Organization

List of Symbols

A_P	Aggregated Parameter
β	Batch Size
n_1, n_2	Biased Moment Estimator
α_k	Contribution Ratio
r	Communication Round
S	Cloud Server
D_k	Data Resource
N_k	Data Size
D_P	Decrypted Parameter
E_P	Encrypted Parameter
τ	Exponentiation Operations
F_{neg}	False Negative
F_{pos}	False Positive
g	Gradient
$GeK(k)$	Key Generation Method
η	Learning Rate
l	Loss Function
C	Medical Client
C_k	Medical Client
W_k^r	Model Parameters
m_1, m_2	Moment Estimator
δ	Natural Number
γ	Numerical Stabilization
P_E	Parameter Encryption
P_A	Parameter Aggregation
P_D	Parameter Decryption
p, q	Prime Numbers
PrK	Private Key
PuK	Public Key
k	Security Parameter
s_i	Symmetric Key
T_a	Trusted Authority
T_{neg}	True Negative
T_{pos}	True Positive

Chapter 1

Introduction

Why should we develop a privacy-preserving method to identify new infections amongst other virus strains from genome sequences? The ultimate answer is, of course, to ensure the proper privacy and security of patients data. However, some readers may expect a more detailed answer. They have many reasons and consequences for expecting detailed answers. To investigate those reasons, we first identified our research questions and tried to address those questions throughout this thesis. The first question is whether the previously developed methods for the identification of new infections amongst other virus strains from genome sequences are capable of ensuring data privacy and security. Then, which methods and tools should be developed to ensure proper privacy and security of patients' data? Furthermore, in comparison to other relevant work, how will the proposed model ensure privacy and security? Finally, will the proposed model be dependable? Therefore, it is a major concern how we develop a dependable model that can efficiently identify new infections amongst other virus strains from genome sequences by ensuring proper privacy and security of patients' data. We have discussed the whole procedure in successive chapters throughout this book. In that continuity, this chapter typically describes the scope of this thesis and gives a brief explanation and summary of this thesis. We also thoroughly discussed the problem statements and expected contributions of this thesis. Finally, we discussed the reader's background, who will feel more comfortable with this paradigm, and the overall idea of this thesis before starting to read the full document.

1.1 General Background

The SARS-CoV-2 virus, also known as COVID-19, is a novel human-infecting coronavirus that was first identified in a patient with pneumonia in Wuhan, China, in December 2019 using next-generation sequencing techniques [1–3]. The new variant of this virus has since spread around the world, affecting millions of people every day [2, 4]. COVID-19 was declared a world emergency on January 30, 2020, and a pandemic on March 11, 2020, by the World Health Organization (WHO) [5, 6]. When a pandemic outbreak occurs, it's critical to decide whether the infection is caused by a new virus or a well-known virus [1, 2, 7]. This means that early identification of new infections from other virus strains can help scientists control the transmission rates and limit the risks [7]. Because of the genetic similarities among the viruses, it is challenging to identify new infections from others [8]. Furthermore, patients suspected of being infected with this infection may also show symptoms that are similar to other types of respiratory infections [1, 9]. So, it is important to accurately and efficiently identify this new infection from similar types of viruses to control the outbreak of the new virus while ensuring proper security and privacy of the patients' data.

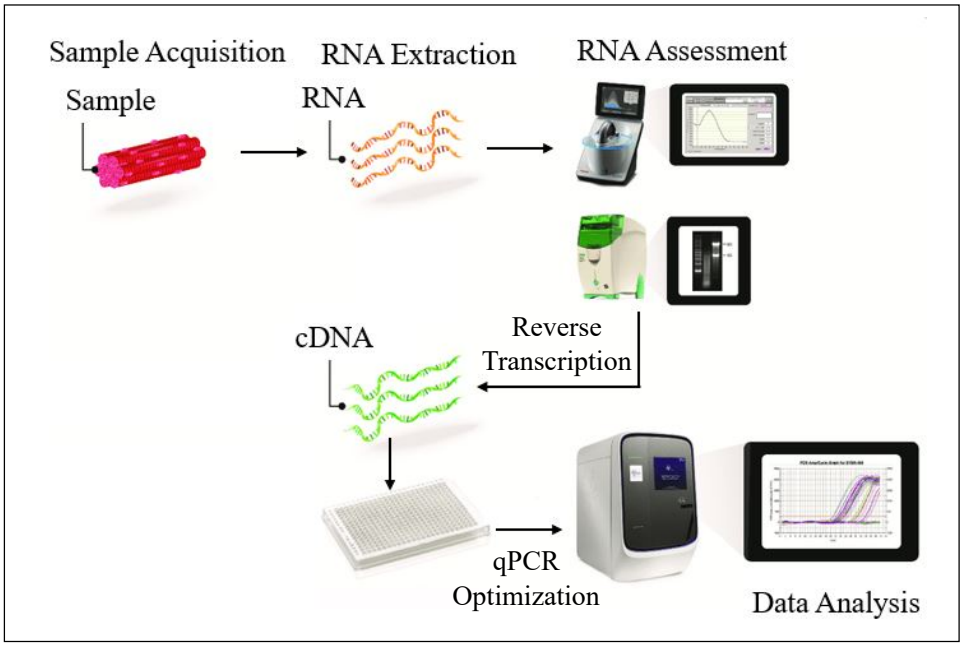


Figure 1.1: The overall scenario of the TMID-based RT-qPCR technique.

The traditional molecular-based identification (TMID) technique of new infections from viral genome sequences (VGSs) is quantitative fluorescence-based reverse transcription-polymerase chain reaction (RT-qPCR) [10–12]. The technique combines reverse transcription-polymerase chain reaction (RT-PCR) and quantitative polymerase chain reaction (qPCR) to determine the number of RNA levels in a qPCR reaction using complementary deoxyribonucleic acid (cDNA) [10, 13]. The ORF1ab and N genes were used in the RT-qPCR technique to identify new infections from VGSs. [14, 15]. Figure 1.1 shows the overall scenario of the traditional molecular-based RT-qPCR technique.

1.2 Specific Background

The background of developing a privacy-preserving and dependable method to identify new infections among other virus strains from genome sequences encompasses several key aspects. Here are the specific background considerations for such a method:

- **Privacy Concerns:** Genome sequences contain highly sensitive and personal information that can be used to identify individuals and potentially disclose their health conditions. Preserving patient privacy is of utmost importance to ensure ethical and legal compliance, as well as to maintain public trust in genomic research and healthcare systems.
- **Data Security:** The genomic data used for identifying new infections needs to be stored, processed, and transmitted securely. Robust security measures are essential to prevent unauthorized access, data breaches, and potential misuse of genomic information. Protecting data integrity and confidentiality is crucial for maintaining trust in the identification process.
- **Distributed Data Sources:** Genome sequences are often distributed across multiple institutions, laboratories, or databases. Collaborative identification of new infections requires a method that can leverage the collective knowledge from diverse data sources while respecting the autonomy and ownership of each dataset. The

method should allow for decentralized analysis without the need to centralize or share raw genomic data.

- **Dependability and Accuracy:** The identification method must be dependable, ensuring accurate and reliable results. False positives or false negatives can have significant consequences in public health and disease management. The method should be rigorously tested, validated, and optimized to provide high accuracy and consistency in identifying new infections among various virus strains.
- **Efficient and Scalable Solutions:** As the volume of genomic data grows rapidly, the identification method should be designed for scalability. It should handle large datasets and be computationally efficient to process and analyze the genome sequences within reasonable time-frames. Scalability is crucial for timely identification and response to emerging infections.
- **Regulatory and Ethical Considerations:** Compliance with regulatory guidelines, such as data protection laws and ethical guidelines for human subject research, is essential. The method should adhere to these regulations to ensure legal and ethical use of genomic data. It should also address potential biases or disparities in the identification process to avoid unintended consequences.

Developing a privacy-preserving and dependable method to identify new infections among other virus strains from genome sequences requires addressing these background considerations. By incorporating robust privacy-preserving techniques, ensuring data security, leveraging distributed data sources, emphasizing dependability and accuracy, enabling efficiency and scalability, and complying with regulatory and ethical guidelines, a comprehensive and effective method can be developed to address the challenges of identifying new infections while protecting patient privacy.

1.3 Motivation of the Research Work

The TMID technique achieved a negative rate of 17.84% when sputum instances were used in moderate illness cases and 11.15% for severe illness cases [7, 16]. Also, when

used on nasal swabs, the technique achieved negative rates of 26.70% and 27.04% in moderate and severe cases, respectively [16]. Furthermore, when used on throat swabs, the technique achieved a negative rate of 40.0% in severe cases and 38.7% in moderate cases [16, 17]. These results may cause variations due to variances in viral species in the RNA sequences [7]. Moreover, in the real-time TMID technique, about 35.2% of 173 samples did not test positive initially, resulting in false-negative findings [17]. As a result, patients with negative results should repeat the test to avoid misdiagnosis [13,18]. Though the TMID technique has a significant risk of false-negative results, it can detect a small percentage of other similar types of viruses, which may lead to the positive and reliable identification of new infections amongst other viruses from viral genome sequences [13,19].

The overall unsatisfactory identification rate of TMID techniques leads to the search for an alternative way to efficiently and accurately identify new infections amongst other viruses from VGSs [7, 8]. Also, due to the sensitive nature of medical data, the TMID technique of transferring the patient's data to the central machine or server may create serious privacy and security issues. Furthermore, TMID techniques require high computational capabilities [11, 13, 19]. As a result, TMID techniques cannot be directly deployed for lightweight medicare applications [11]. On the other hand, several machine learning (ML) and deep learning (DL) models have been proposed to identify new infections to overcome those issues [8]. However, most of the proposed ML and DL-based models did not consider privacy and security issues related to the patient's data [20, 21]. Furthermore, most of the previously proposed ML and DL-based models did not consider computational complexity and dependability performance analysis [22, 23].

In recent years, the progression of deep federated learning (DFL) and its remarkable success in various domains has been guided as a potential solution in this field [20, 24–26]. Specifically, in the Industry 4.0 revolution, medicare applications require a high level of dependable and privacy-preserving structure [23]. Therefore, there is a need for a dependable and privacy-preserving identification model for new infections from VGSs. Generally, dependability performance analysis includes availability, efficiency, and scalability features [22, 23]. An identification model's availability is determined by how often it is available, and its efficiency is how well it performs with low computational complex-

ity [23]. Also, scalability is the capability to adapt easily to the increased number of data sources [22]. On the other hand, privacy-preserving techniques help to keep the data of patients safe and private [25]. Specifically, when any private information is disclosed against a patient's wishes, privacy becomes far more important because privacy-preserving is a fundamental requirement for maintaining the positive reputation of any medicare or other heterogeneous application [24].

1.4 Expected Contributions

Therefore, this article proposes a dependable and privacy-preserving DFL-based LeCun Network (LeNet) identification model of new infections amongst other viruses from VGSs. The proposed model has an overall accuracy of 99.12% after independently and identically distributing (IID) the dataset among 6 clients. Moreover, the proposed model also increases other performance metrics for the same configuration, which is a noticeable improvement when compared to the other benchmark models. Finally, we have analyzed the dependability performance to ensure the availability, efficiency, and scalability features of our proposed model. The proposed dependable model, along with empirical results, is encouraging enough to recognize as an alternative for the identification of new infections amongst other viruses from VGSs by ensuring proper privacy and security of the patient's data. Figure 1.1 shows the motivation on how the proposed dependable model can be applied to identify the new infections amongst viruses from VGSs by ensuring proper privacy and security of patients' data.

The key contributions of this thesis are narrated as follows:

- In this work, we propose a dependable and privacy-preserving deep federated learning -based LeCun Network (LeNet) identification model of new infections amongst other viruses from viral genome sequences.
- The performance of the proposed model is evaluated using a real-time viral genome sequence dataset with a different number of clients in terms of independent and

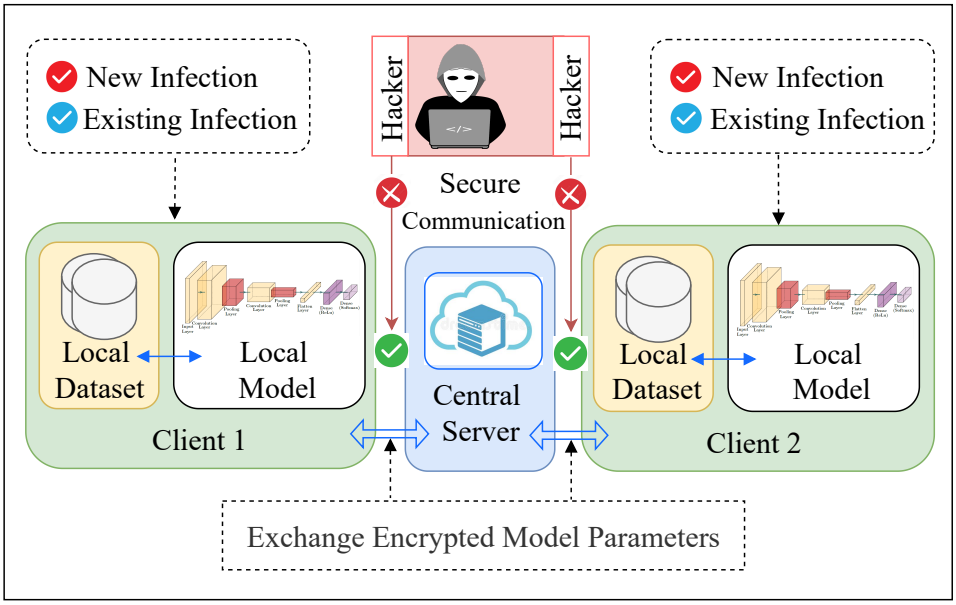


Figure 1.2: The overall scenario of the proposed dependable identification model of new infections from VGSs by ensuring proper privacy and security of patients' data.

identically distributed (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 and computational complexity of the proposed model.
- Finally, overall performance evaluation is considered, where the proposed model is more dependable, efficient, and outperforms the existing traditional centralized-based models with ensuring proper privacy and security of patients' data.

1.5 Reader Background

In this thesis, we assume that the readers have a working knowledge of both bioinformatics and deep federated learning. We do not attempt to explain the basics of any of these paradigms. As in much multi-disciplinary work, most readers will feel more comfortable with this implementation procedure. We hope to convince them that much is to be gained by integrating these two paradigms.

It is not an easy to implement DFL with DNA sequences. Therefore, sometimes we will need to refer to concepts that will only be fully explained in a later chapter. However, a basic knowledge of both paradigms should be enough to get the reader through this thesis.

1.6 Summary

The intent of this chapter is to get you familiar with the foundations of the proposed methodology before taking a deep dive into the main thesis pipelines. We have discussed the overall scenario of the traditional molecular-based RT-qPCR techniques with limitations in this chapter. The need for a privacy-preserving and dependable identification model of new infections from genome sequences in todays world is introduced in the chapter. Finally, the overall scenario of the proposed dependable identification model of new infections from genome sequences by ensuring proper privacy and security of patients' data has been discussed in this chapter.

Next up, we explored the federated learning landscape, starting from the formal definition to the various domains and fields associated with federated learning. Concepts relevant to traditional machine learning, deep learning, and deep federated learning have also been covered in the next chapter. Finally, an overview of recent research on the identification of new infections in this domain has been discussed.

1.7 Organization of the Book

The remaining of this book is structured as follows:

Chapter 2 discusses the background and related works. While investigating most recent relevant works, we find several works overlaps the similar motivation in different ways. Therefore, this chapter illustrates the overview of federated learning, application of federated learning, categories of federated learning, variations of federated learning, and privacy-preservation techniques after investigating the related work in this domain.

Chapter 3 describes the philosophy behind using federated learning to identify new infections from genome sequences. It also elaborates on the key contributions of the proposed model and discusses the overview of the proposed model by outlining the complete workflow. Finally, it thoroughly explains the structure of the proposed model, followed by the privacy-preserving module.

Chapter 4 continues with a discussion of environment setup, then an overview of the dataset with data preparation process, and concludes with the training and testing process of the selected deep transfer learning and deep federated learning models.

Chapter 5 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 of the models.

Chapter 6 summarizes the major contributions of this thesis. Also, discusses the pitfalls and limitations which impacted the interpretation of the findings from our research and presents a road-map for future development.

Appendix A shows the preprocessing steps of a DNA sequence dataset using the Python programming language. We have also implemented the pre-processed dataset in traditional machine learning to check the consistency of the dataset.

Appendix B contains a complete implementation procedure of all the deep transfer learning models using the Python programming language.

Appendix C contains a complete implementation procedure of the proposed deep federated learning model with six client devices using the Python programming language.

Chapter 2

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 investigate the related work in this domain before illustrating the overview of federated learning, categories of federated learning schemes, variations of federated learning, and privacy-preservation techniques.

2.1 Related Works

In this Section, we conduct an extensive review of state-of-the-art works in identifying new infections from genome sequences. We analyze their strengths and weaknesses, focusing on efficiency and privacy concerns. This section sets the foundation for our proposed privacy-preserving model by identifying the gaps that need to be addressed.

2.1.1 TMID-based Schemes

First of all, to categorize genome sequences, alignment-based techniques such as the basic local alignment search tool (BLAST) and the burrows-wheeler aligner (BWA) have been used, which depend on annotating the viral genes [27–29]. These alignment-based approaches have been used only in identifying sequence similarities, which suffer from

the necessity of needing base sequences for their detection [27, 30–32]. However, these alignment-based techniques require more computational time and memory when they are implemented to analyze thousands of genomes [33, 34]. As a result, ML-based algorithms for the classification of DNA sequences have been proposed as an alternative. These techniques have the advantage of not requiring pre-selected features in order to classify DNA sequences. Using one-hot label encoding and ML-based algorithms, DNA sequences have been efficiently identified and classified [35].

2.1.2 ML-based Schemes

Randhawa et al. [36] proposed an ML-based tool for DNA sequence comparison and analysis that does not require alignment. The tool was created to address issues related to the alignment of DNA sequences. Zeng et al. [37] 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. [38] 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. [39] 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 run-time. Nguyen et al. [40] 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 of these datasets. NCNet is a DL model introduced by Zhang et al. [41] 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. [42] 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. [43] proposed a model called DeepSEA, which predicts the chromatin effects of sequence modifications with single nucleotide sensitivity.

Whata et al. [7] proposed an alignment-free DL-based model for new genome sequences. The proposed CNN-BiLSTM model achieves better performance metrics. Furthermore, LopezRincon et al. [8] 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 in that it can both automatically discover new viral primer sets from a small quantity of data and give effective results in a short amount of time. But, most of the above-mentioned ML and DL-based techniques did not consider the privacy and security of patients' sensitive data and the proposed model's dependability performance analysis with lower computational complexity.

2.1.3 DTL-based Schemes

A DTL paradigm for wearable healthcare systems was proposed by Chen et al. [21]. Experiments show that the proposed technique outperforms traditional methods for wearable healthcare activity recognition, improving accuracy by 5.3%. Hinton et al. [44] and Krizhevsky et al. [45] 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. [46] proposed a DTL-based IDS system for electric vehicular networks. The proposed method greatly improves accuracy over the mainstream ML, DL, and benchmark DTL models and has demonstrated better performance. Mehedi et al. [23] also 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.

2.1.4 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. [56] 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. [57] 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. [25] 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.

2.1.5 Overview of the Schemes

The existing identification and classification models have been categorized according to the algorithm, accuracy, dependency, identification coverage, privacy and security of patient data, and dependability analysis of the proposed model. In summary, in the real-time TMID technique, about 35.2% of 173 samples did not test positive initially, resulting in false-negative findings. As a result, patients with negative results should repeat the test to avoid misdiagnosis. Also, due to the sensitive nature of the patients data, the TMID technique of transferring data to the central machine or server may create serious privacy and security issues. Furthermore, the TMID technique requires high computational capabilities. As a result, these techniques cannot be directly deployed for lightweight Medicare applications. Moreover, several ML and DL models have been proposed. But, most of the models did not consider privacy and security issues related to the patients data [4]. Also, they did not consider the dependability performance analysis of the proposed models. A summary of all the existing mechanisms is given in Table 2.1. Since the success of the FL technique in different fields has been recognized by researchers, this technique has now been incorporated into the bioinformatics area to increase the performance of prediction or classification tasks. Therefore, in this paper, we propose a privacy-preserving DFL model, which is an alignment-free method to improve the accuracy of new infection detection rates with dependability analysis from other virus strains.

Table 2.1: An overview of the recent research on different medicare applications.

Algorithms	**Accuracy**	**Dependency**	**Coverage**	**Privacy and Security**	**Dependability**
MLDSP [36]	92.0%	Alignment free	DNA sequence comparison	✗	✗
CNN [37]	N/A	Alignment free	Predicting DNA-protein binding	✗	✗
DL [38]	N/A	Alignment free	Discover DNA binding motifs	✗	✗
DeepFam [39]	N/A	Alignment based	Protein family modeling and prediction	✗	✗
CNN [40]	96.23%	Alignment based	DNA sequence classification	✗	✓
NCNet [41]	N/A	Alignment free	Predicting function of non-coding DNA sequence	✗	✗
DeepSite [42]	N/A	Alignment free	Predicting DNA protein binding	✗	✗
DeepSEA [43]	95.80%	N/A	Predicting effects of non-coding variants	✗	✗
DL [7]	98.70%	N/A	New genome sequences classification	✗	✓
DL [8]	98.73%	Alignment free	Classification of new infection	✗	✓
BFL [55]	98.68%	N/A	New Infection image classification	✓	✗
PFL [56]	95.37%	N/A	Healthcare based on the cloud edge	✓	✗
FL [57]	N/A %	N/A	Breast density classification	✓	✗
CFL [25]	99.05 %	N/A	New Infection diagnosis at the edge	✓	✗
DFL [Proposed] * [23, 46, 53]	99.12%	Alignment free	Identification of new infection	✓	✓

* Indicate the published research article using various data sets.

2.2 Federated Learning

Training a model at each local device where data is stored, and then each device exchanges their models' parameters in order to converge on a single global model, where no device is able to snoop on the private data of any other device because of the encryption engineering of the communication mechanism [26, 58]. This is the concept called "Federated Machine Learning (FML)," sometimes referred to simply as "Federated Learning (FL)" [58].

In other words, FL is a distributed machine learning (ML) framework, where multiple devices collaborate to solve traditional distributed ML problems under the coordination of the central server without sharing their local private data with other devices [26, 59].

2.2.1 Application of Federated Learning

Federated Learning (FL) has various applications across different domains and industries. Here are some examples of how federated learning can be applied:

- **Healthcare:** FL enables collaborative model training on distributed healthcare data while maintaining patient privacy. It can be used for tasks such as disease diagnosis, personalized treatment recommendation, predictive analytics, and drug discovery, by leveraging data from multiple hospitals or healthcare institutions.
- **Internet of Things:** FL can be applied to IoT devices, allowing them to collaboratively learn and improve their models without sending raw data to a central server. This is beneficial for applications like smart homes, smart cities, and industrial IoT, where privacy and low-latency communication are crucial.
- **Financial Services:** FL can be employed in the financial industry to analyze customer behavior, detect fraudulent activities, and improve risk assessment models. Banks and financial institutions can collaborate while ensuring the privacy and security of sensitive financial data.
- **Mobile Applications:** FL enables on-device model training for mobile applications. Instead of sending user data to a central server, the models are trained locally

on user devices, providing personalized experiences while maintaining data privacy. This can be used for tasks like predictive text input, voice recognition, and personalized recommendations.

- **Autonomous Vehicles:** FL allows autonomous vehicles to learn collectively from their individual experiences. Models can be trained on vehicle-specific data without compromising user privacy, and the knowledge gained can be shared across the fleet, improving safety, performance, and efficiency.
- **Federated Analytics:** FL can be applied to aggregate and analyze data from multiple sources, such as social media platforms or online marketplaces. This enables insights generation while preserving user privacy and data ownership.
- **Edge Computing:** FL can be combined with edge computing to train models on edge devices or edge servers. This reduces the need for transferring large amounts of data to a centralized server, enabling faster and more efficient model training while addressing latency and bandwidth limitations.
- **Natural Language Processing:** FL can be used for collaborative NLP tasks, such as language translation, sentiment analysis, and speech recognition. Models can be trained using data from multiple sources to improve accuracy and performance.

The flexibility and privacy-preserving nature of FL make it a promising approach for collaborative machine learning across various domains, allowing organizations to leverage collective intelligence while protecting data privacy and security.

2.2.2 Federated Learning as a Solution

The Google developer team recommends the FL model to protect the privacy and security of the user's data, which solves the problem of using a central server or device to train a shared global model [20, 58, 60, 61]. At first, they deployed the FL model to train the ML model based on globally dispersed mobile devices while keeping user data secure [58, 61]. Because of the continuous increase in the number of medical devices in recent years, a large amount of data is generated, and because of its ever-increasing extensive computing

and processing capabilities, as well as the privacy and security issues, it is recommended to store data locally and perform the computation mechanism on the local devices with edge computing technology [56, 58, 62]. As the storage and processing capabilities of local devices improve over time, local devices can be used more efficiently [63].

Therefore, to implement this mechanism, there is a need for the FL setting, which makes it possible to directly investigate the local or remote devices [20]. 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 [64–67]. According to recent studies, the majority of large-scale service providers have already incorporated FL technology [58, 61, 62, 64–67]. Some medical devices and non-medical devices are equipped with a large number of sensors that allow them to acquire, aggregate, respond, and adapt to new sensitive medical data in a real-time environment [68, 69].

For example, in order to accurately predict the risk of cardiovascular disease, medical devices need the most up-to-date models with a wide range of pathological data in order to operate safely and make predictions in real-time [7, 26]. However, privacy and security concerns over highly sensitive medical data and the limited connectivity of devices make it difficult to build aggregate models in such situations [26]. Also, genome sequence data is more sensitive, so privacy and security of patient data must be ensured to analyze that data [7, 70]. As a result, FL technology is used to train models in this context, allowing them to adapt quickly to changes while maintaining users' privacy and security [65–67].

A large number of patients' confidential data must be kept on the local device 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) [71, 72]. In this case, the FL mechanism is more suitable for healthcare applications in the context of medical organizations, which often store significant volumes of patients' confidential data [26, 67, 72]. This FL also enables private and secure collaborative learning between other organizations, where privacy and security of patients' data are the main concerns.

2.2.3 Structure of Federated Learning

Figure 2.1 shows the overall architecture and the secure communication procedure between local devices and the central server in a typical FL mechanism training process. In this case, the local devices and the central aggregation server are located at the medical service access point. There are a number of stages involved in the typical FL communication process between local devices and the central aggregation server [62, 73, 74].

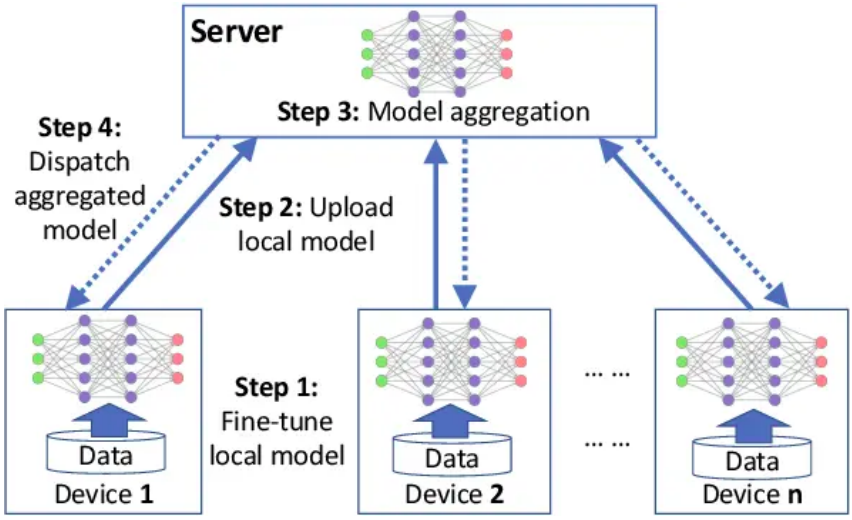


Figure 2.1: The overall architecture and the secure communication procedure between local devices and a central server in a typical FL mechanism training process.

During the system initialization and device selection phases, the central aggregator server chooses specific tasks and sets different learning parameters [62]. These different learning parameters include the number of communication rounds, batch size, loss function, learning rates required to achieve an optimized single global model, etc. After the initial setup is complete, the central aggregation server builds a new model and distributes it to all local devices or clients in order to begin distributed training [24]. Then, the local model is trained using the client's or device's own private data, and the update is calculated by minimizing the loss function [71]. Finally, a new global model is created by compiling all model modifications from local devices and resolving the optimization issue on the server [58, 62].

2.3 Categories of Federated Learning

There are two main categories in the FL mechanism, the first one is the dataset partition FL mechanism, and the last one is the network-structured FL mechanism [26].

2.3.1 Dataset Partition FL Mechanism

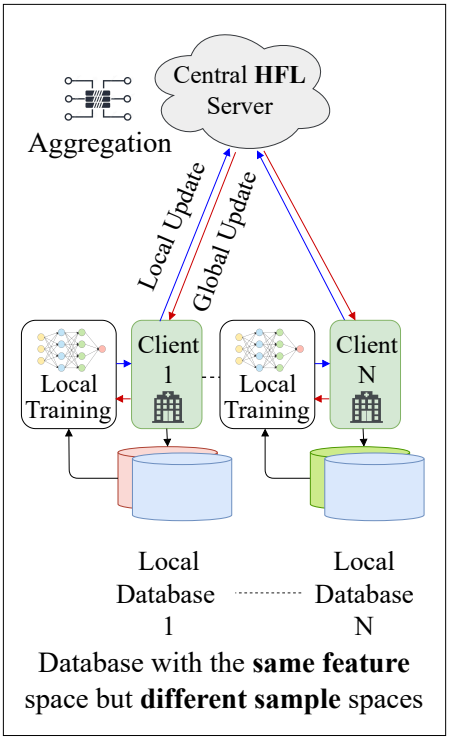
The dataset partition FL mechanism can be further divided into three different categories based on the distribution of training data in the sample and feature space [26, 75]. They are the horizontal federated learning (HFL) mechanism, vertical federated learning (VFL) mechanism, and transfer federated learning (TFL) mechanism.

2.3.1.1 Horizontal Federated Learning

Figure 2.2 shows the HFL mechanism, where all local client devices train the same single global model using the local private dataset from the local client device, which has the same feature space but different sample spaces [26, 75, 76]. However, because of the common feature space, local client devices can train their local models using the same model [26]. Then, each local client device uses the local private dataset to train in order to calculate local updates that can be encrypted through an encryption technique [26]. Then, the central aggregation server collects all local updates from all client devices [26, 75, 76]. In addition, until the goal accuracy is met, the central aggregation server calculates the subsequent global update without accessing the local private dataset and then sends the global update back to the local client devices for the subsequent round of local training [77].

2.3.1.2 Vertical Federated Learning

In the VFL mechanism, it solves the problem of training models across local client device networks on the same sample set with different feature sets [26, 75, 76]. Furthermore, local private data samples are used to train a shared model, which uses a variety of encryption



Local Dataset 1

Sample1	Sample2	Sample3
FeatureA	FeatureA	FeatureA
FeatureB	FeatureB	FeatureB
FeatureC	FeatureC	FeatureC
FeatureD	FeatureD	FeatureD
FeatureE	FeatureE	FeatureE
FeatureF	FeatureF	FeatureF
FeatureG	FeatureG	FeatureG

Local Dataset 2

Sample4	Sample5	Sample6
FeatureA	FeatureA	FeatureA
FeatureB	FeatureB	FeatureB
FeatureC	FeatureC	FeatureC
FeatureD	FeatureD	FeatureD
FeatureE	FeatureE	FeatureE
FeatureF	FeatureF	FeatureF
FeatureG	FeatureG	FeatureG

Figure 2.2: The horizontal federated learning (HFL) mechanism is based on the dataset samples and features partition.

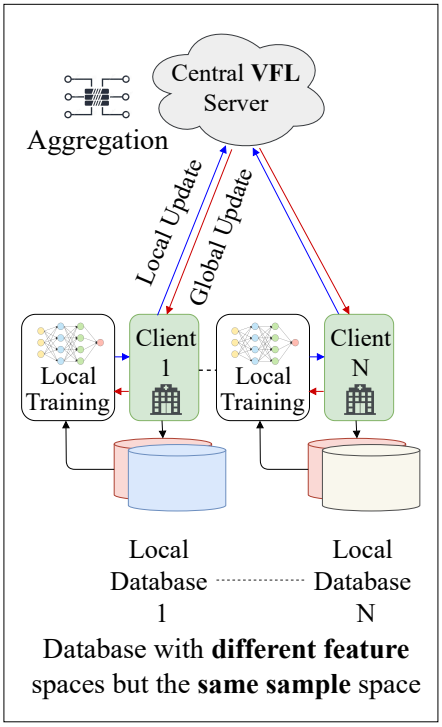
mechanisms in communication between local devices and a central server to enhance the privacy and security of patients' sensitive data [77]. Figure 2.3 shows the VFL mechanism.

2.3.1.3 Transfer Federated Learning

The TFL mechanism, a VFL extension, allows models to interact with different sample spaces and different feature spaces while participating in the learning process [76, 78]. Furthermore, to protect privacy and provide security of patients' sensitive data while learning the model in this environment, secure encryption mechanisms must be deployed [26]. Figure 2.4 shows the TFL mechanism.

2.3.2 Network Structured FL Mechanism

The network structured FL mechanism can be further divided into three categories based on the structure of the network topology [26, 75–77]. The first one is the centralized



Local Dataset 1

Sample1	Sample2	Sample3
FeatureA	FeatureA	FeatureA
FeatureB	FeatureB	FeatureB
FeatureC	FeatureC	FeatureC
FeatureD	FeatureD	FeatureD
FeatureE	FeatureE	FeatureE
FeatureF	FeatureF	FeatureF
FeatureG	FeatureG	FeatureG

Local Dataset 2

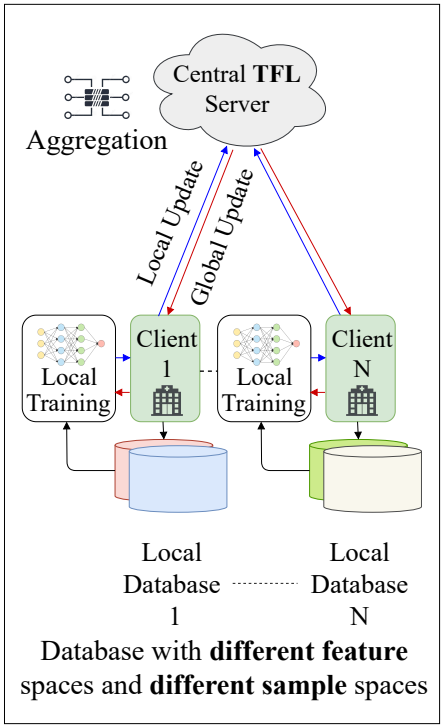
Sample1	Sample2	Sample3
FeatureH	FeatureH	FeatureH
FeatureI	FeatureI	FeatureI
FeatureJ	FeatureJ	FeatureJ
FeatureK	FeatureK	FeatureK
FeatureL	FeatureL	FeatureL
FeatureM	FeatureM	FeatureM
FeatureN	FeatureN	FeatureN

Figure 2.3: The vertical federated learning (VFL) mechanism is based on the dataset samples and features partition.

federated learning (CFL) mechanism, the second one is the decentralized federated learning (DFL) mechanism, and the last one is the heterogeneous federated learning (HFL) mechanism.

2.3.2.1 Centralized Federated Learning

Figure 2.5 shows the CFL mechanism, where a central server is used to control and manage the different steps of the mechanism and coordinate all the client devices during the learning phase [26, 58]. Each client device uses its own local private dataset to train the model [26]. After training the local model parameters, the central server aggregates the trained parameters using a weighted average approach from each client device. As a result, each client device will have a single global model at the end of the training process [76]. The central server is responsible for the client device selection at the beginning of the training phase and for the aggregation of the received model updates [61]. Since all the selected client devices have to send updates to a single global model and to protect the



Local Dataset 1

Sample1	Sample2	Sample3
FeatureA	FeatureA	FeatureA
FeatureB	FeatureB	FeatureB
FeatureC	FeatureC	FeatureC
FeatureD	FeatureD	FeatureD
FeatureE	FeatureE	FeatureE
FeatureF	FeatureF	FeatureF
FeatureG	FeatureG	FeatureG

Local Dataset 2

Sample4	Sample5	Sample6
FeatureH	FeatureH	FeatureH
FeatureI	FeatureI	FeatureI
FeatureJ	FeatureJ	FeatureJ
FeatureK	FeatureK	FeatureK
FeatureL	FeatureL	FeatureL
FeatureM	FeatureM	FeatureM
FeatureN	FeatureN	FeatureN

Figure 2.4: The transfer federated learning (TFL) mechanism is based on the dataset samples and features partition.

privacy and security of training data, the central server may become a key element of the mechanism [75–77].

2.3.2.2 Decentralized Federated Learning

The DFL technique uses a decentralized network architecture [26, 58]. Figure 2.6 shows the DFL technique's overview diagram, where each client device uses its own private dataset for local model training during each communication round, and all client devices are connected to one another as in a peer-to-peer (P2P) network architecture [78, 79]. In this technique, each client device aggregates model updates that are received from nearby client devices via P2P communication to create a single global model. This configuration makes the DFL technique more scalable than the CFL technique [21]. In addition, the DFL technique can further expand its capabilities through P2P-based blockchain (BC) technology, where model updates will be uploaded to the BC ledger for secure model aggregation and distribution mechanism [78–81].

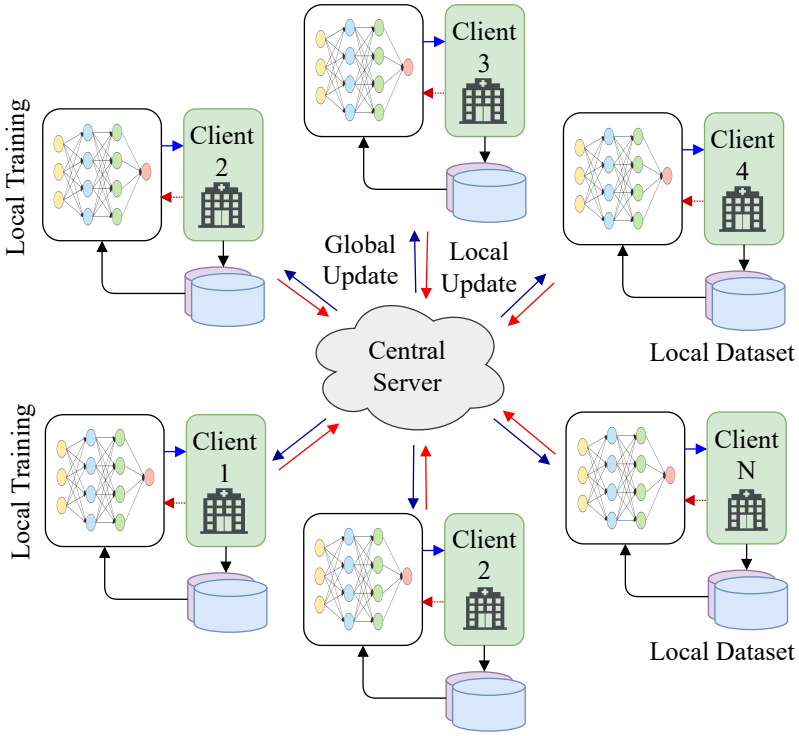


Figure 2.5: The centralized federated learning (CFL) mechanism.

2.3.2.3 Heterogeneous Federated Learning

An increasing number of application domains involve a large set of heterogeneous clients, e.g., mobile phones, medical devices, and industrial IoT devices [81]. The heterogeneous FL (HeteroFL) technique is the most efficient in terms of computation complexity and secure communication protocol for these heterogeneous clients. At present, the majority of existing FL techniques share the same single global model with the local client devices with independent and identically distributed (IID) data configuration [59, 81]. But HeteroFL, a new FL technique, was recently developed to handle these heterogeneous local client devices with a secure communication protocol and different computation capabilities of both IID and non-IID data configuration [59].

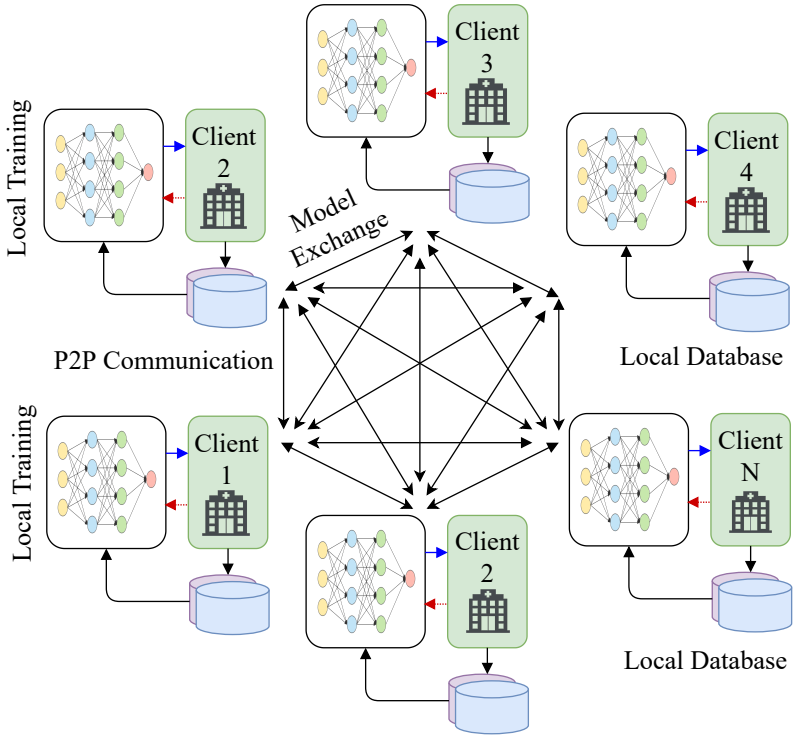


Figure 2.6: The decentralized federated learning (DFL) mechanism.

2.4 Federated Learning Variations

There are many types of variations in the FL setting. Some of the variations are federated stochastic gradient descent (FedSGD), federated averaging (FedAvg), FL with dynamic regularization (FedDyn), and personalized FL by pruning (Sub-FedAvg). Also, some of the variations are in the development phase by various large organizations [82].

2.4.1 Federated Stochastic Gradient Descent (FedSGD)

The majority of SGD variations are used in the deep learning model training phase to generate gradients on a randomly chosen portion of the whole dataset and utilize those results to complete a gradient descent step [82,83]. The FedSGD technique is the straightforward transposition of the SGD technique to the FL setting, which uses all the data of client [82]. Finally, the gradients are averaged by the central server in proportion to the number of training samples on each client and used to create a gradient descent step.

2.4.2 Federated Averaging (FedAvg)

The FedAvg technique is a generalization of the FedSGD technique that enables local clients to execute several batch updates on local data and share the updated weights rather than the gradients. This generalization technique is justified by the fact that when all local clients begin with the same initialization, averaging the gradients is strictly equivalent to averaging the weights themselves [26]. Additionally, the performance of the final averaged model is not always negatively impacted by averaging weights from the same initialization. The implementation process of the FedAvg is simple and easy.

2.4.3 FL with Dynamic Regularization (FedDyn)

The FL techniques suffer when the local client device datasets are heterogeneously distributed. The fundamental problem with heterogeneously distributed device settings is that decreasing device loss functions does not equate to minimizing the central model loss. Recently, the FedDyn technique has been proposed as a solution to the heterogeneous dataset setting [84]. In order to adjust the central global device loss, the FedDyn technique dynamically regularizes each client device's loss. Also, the benchmark for comparison in the FL technique is reducing the number of communications rounds. The FedDynOneGD technique is a FedDyn technique extension with fewer local computation round requirements [83, 84]. So, the FedDynOneGD technique only creates one gradient for each device in each round and uses a regularized gradient to update the central model.

2.4.4 Personalized FL by Pruning (Sub-FedAvg)

The FL techniques cannot achieve good central global performance under the Non-IID configuration of the dataset, which motivates the participating client devices to yield personalized models in the federation. Recently, in Sub-FedAvg, a new personalized FL technique was proposed in which hybrid pruning of sub-networks handles the secure communication efficiency, resource constraints, and final model accuracy [58].

2.5 Security Studies of Federated Learning

The FL has emerged as a promising paradigm for collaborative machine learning, enabling multiple parties to jointly train models without sharing their raw data. While FL offers numerous advantages, it also introduces unique security challenges that need to be addressed.

2.5.1 Threat Models and Adversarial Scenarios

Understanding the potential threats and adversarial scenarios is crucial for designing robust security mechanisms in federated learning. Researchers have identified various threat models, including Byzantine attacks, model poisoning, and inference attacks, which can compromise the integrity and privacy of FL systems [58]. Studying these threat models helps in developing countermeasures and mitigation strategies.

2.5.2 Secure Aggregation Protocols

Secure aggregation is a critical component in FL, enabling participants to securely combine their locally computed model updates without exposing sensitive information [26]. Several cryptographic protocols, such as secure multi-party computation (MPC), homomorphic encryption, and differential privacy, have been investigated to achieve secure aggregation while preserving data privacy [26, 58].

2.5.3 Privacy-Preserving Techniques

Preserving data privacy is a paramount concern in FL. Various techniques, such as federated learning with differential privacy (FL-DP), secure aggregation, and cryptographic protocols, have been explored to ensure that sensitive information remains protected during the training process [59–61]. Understanding the trade-offs and limitations of these techniques helps researchers design privacy-preserving FL algorithms that strike a balance between utility and privacy.

2.5.4 Robustness Against Attacks

FL systems need to be robust against adversarial attacks that aim to compromise the integrity or quality of the trained models [76]. Studies have investigated the vulnerability of FL models to poisoning attacks, backdoor attacks, and model extraction attacks [53, 71]. Analyzing these attack vectors aids in the development of robust defenses, including anomaly detection mechanisms, robust optimization algorithms, and data sanitization techniques [23, 55].

2.5.5 Secure Model Updates and Parameter Exchange

The exchange of model updates and parameters in FL introduces potential security risks [62, 80]. Research has focused on secure and efficient methods for transmitting model updates, leveraging cryptographic techniques, such as secure channel protocols and secure enclaves, to protect the confidentiality and integrity of the exchanged data [62].

2.6 Summary

This chapter explored the federated learning concept, starting from the formal definition to the various domains and fields associated with federated learning. In this chapter, we also surveyed work related to our own, both to point out the many contributions of previous researchers and to place our contributions in the proper context. Furthermore, the performance of traditional machine learning over deep learning has been well-studied, whereas most of the previous methods did not consider dependability performance analysis and users' privacy and security. Finally, an overview of recent research on the identification of new infections in this domain has been discussed in this chapter.

In the following chapters, we develop a privacy-preserving model that can enhance the efficiency of identifying new infections from genome sequences. This model can diminish the challenges of state-of-the-art works by providing a more secure and efficient way to share and analyze genome data.

Chapter 3

Proposed Methodology

In this chapter, we discuss the philosophy behind using federated learning to identify new infections from genome sequences. We then elaborate on the key contributions of our proposed model. We also discuss the overview of our proposed model by outlining the complete workflow. Finally, we thoroughly explain the details structure of our proposed model, followed by the privacy-preserving module.

3.1 Introduction

Federated learning is a machine learning technique that allows multiple parties to train a shared model without sharing their data. This is done by having each party train a local model on their own data and then sending updates to a central server. The central server then aggregates the updates and updates the shared model. This process is repeated until the model converges. Federated learning has several advantages over traditional machine learning techniques. First, it allows for the training of models on sensitive data without the need to share the data. This is important for protecting privacy and security. Second, federated learning can be used to train models on data that is distributed across multiple parties. This can be useful for training models on large datasets or for training models on data that is difficult to share.

The philosophy behind using a federated learning-based identification model for new infection from genome sequences encompasses several key principles and objectives. Federated learning is an approach that allows collaborative model training across multiple decentralized data sources, while preserving data privacy and security. When applied to the identification of new infections from genome sequences, this philosophy aims to achieve the following:

1. **Privacy Protection:** Genome sequences contain sensitive and personal information. By utilizing a federated learning approach, the privacy of individual genome data can be safeguarded. Instead of centralizing the data, the model is trained locally on each data source, ensuring that sensitive genetic information remains secure within its respective source.

2. **Data Diversity and Scalability:** The federated learning framework allows the aggregation of diverse genome sequences from multiple locations or institutions. By leveraging the collective knowledge and genetic diversity from different sources, the model can be trained on a larger and more representative dataset, enhancing its ability to identify new infections accurately. This approach also enables scalability as new data sources can be seamlessly incorporated into the federated learning system.

3. **Real-Time Adaptability:** The identification of new infections requires continuous monitoring and analysis of genome sequences. A federated learning-based model can be updated and refined in real-time as new data becomes available. This adaptability allows the model to stay up-to-date with emerging infections, rapidly incorporating new information and improving its accuracy over time.

4. **Collaboration and Knowledge Sharing:** Federated learning promotes collaboration among different institutions, researchers, and experts. By sharing model updates, insights, and expertise while preserving data privacy, collective efforts can be harnessed to enhance the accuracy and effectiveness of the identification model. This philosophy encourages a cooperative approach in addressing the challenges posed by new infections, fostering a broader understanding and collaborative response to emerging threats.

5. **Ethical Considerations:** The philosophy of using a federated learning-based identification model aligns with ethical principles by respecting the privacy and autonomy of individuals. It ensures that data is used in a responsible and secure manner, reducing the risk of unauthorized access or misuse. By prioritizing privacy and ethical considerations, this approach fosters trust and promotes responsible use of genomics data for the benefit of public health.

In summary, the philosophy behind using a federated learning-based identification model for new infection from genome sequences revolves around privacy protection, data diversity, scalability, real-time adaptability, collaboration, and ethical considerations. By combining the power of decentralized data sources, advanced machine learning techniques, and collaborative efforts, this philosophy aims to enhance the accuracy, privacy, and efficacy of identifying new infections, ultimately contributing to the broader understanding and management of emerging health threats.

3.1.1 Expected Contributions

Therefore, this article proposes a dependable and privacy-preserving DFL-based LeCun Network (LeNet) identification model of new infections amongst other viruses from VGSs. The proposed model has an overall accuracy of 99.12% after independently and identically distributing (IID) the dataset among 6 clients. Moreover, the proposed model also increases other performance metrics for the same configuration, which is a noticeable improvement when compared to the other benchmark models. Finally, we have analyzed the dependability performance to ensure the availability, efficiency, and scalability features of our proposed model. The proposed dependable model, along with empirical results, is encouraging enough to recognize as an alternative for the identification of new infections from VGSs by ensuring proper privacy and security of the patient's data.

The key contributions of this thesis are narrated as follows:

- In this work, we propose a dependable and privacy-preserving deep federated learning -based LeCun Network (LeNet) identification model of new infections amongst other

viruses from viral genome sequences.

- The performance of the proposed model is evaluated using a real-time viral genome sequence dataset with a different number of clients in terms of independent and identically distributed (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 and computational complexity of the proposed model.
- Finally, overall performance evaluation is considered, where the proposed model is more dependable, efficient, and outperforms the existing traditional centralized-based models with ensuring proper privacy and security of patients' data.

The following sections, we elaborate the proposed dependable and privacy-preserving DFL-based identification model first. Then we explain the proposed model by outlining the complete workflow, and finally we thoroughly explain the details structure of the proposed model, followed by the privacy-preserving module.

3.2 Proposed DFL-based Identification Model

Figure 3.1 shows the proposed DFL-based identification model. In this case, the local devices and the central aggregation server are located at the medical service access point. There are a number of stages involved in the proposed DFL communication process between local devices and the central aggregation server.

During the system initialization and device selection phases, the central aggregator server chooses specific tasks and sets different learning parameters. These different learning parameters include the number of communication rounds, batch size, loss function, learning rates required to achieve an optimized single global model, etc. After the initial setup is complete, the central aggregation server builds a new model and distributes it to all

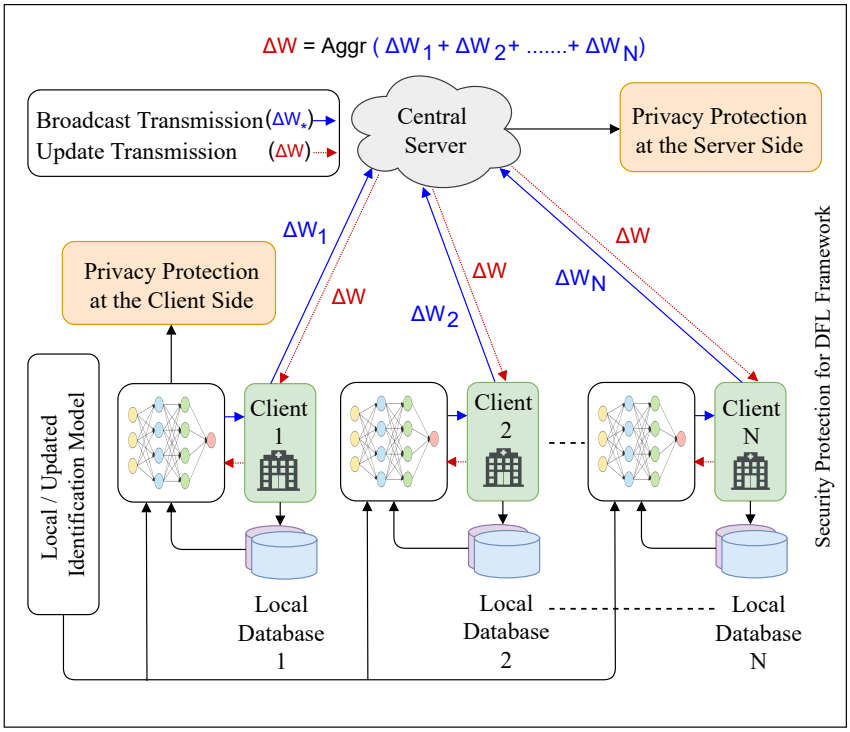


Figure 3.1: The overall architecture of the proposed DFL-based identification model.

local devices or clients in order to begin distributed training. Then, the local model is trained using the client's or device's own private data, and the update is calculated by minimizing the loss function. Finally, a new global model is created by compiling all model modifications from local devices and resolving the optimization issue on the server.

3.3 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 3.2 shows the overall workflow of the proposed DFL model with two clients and a key management authority.

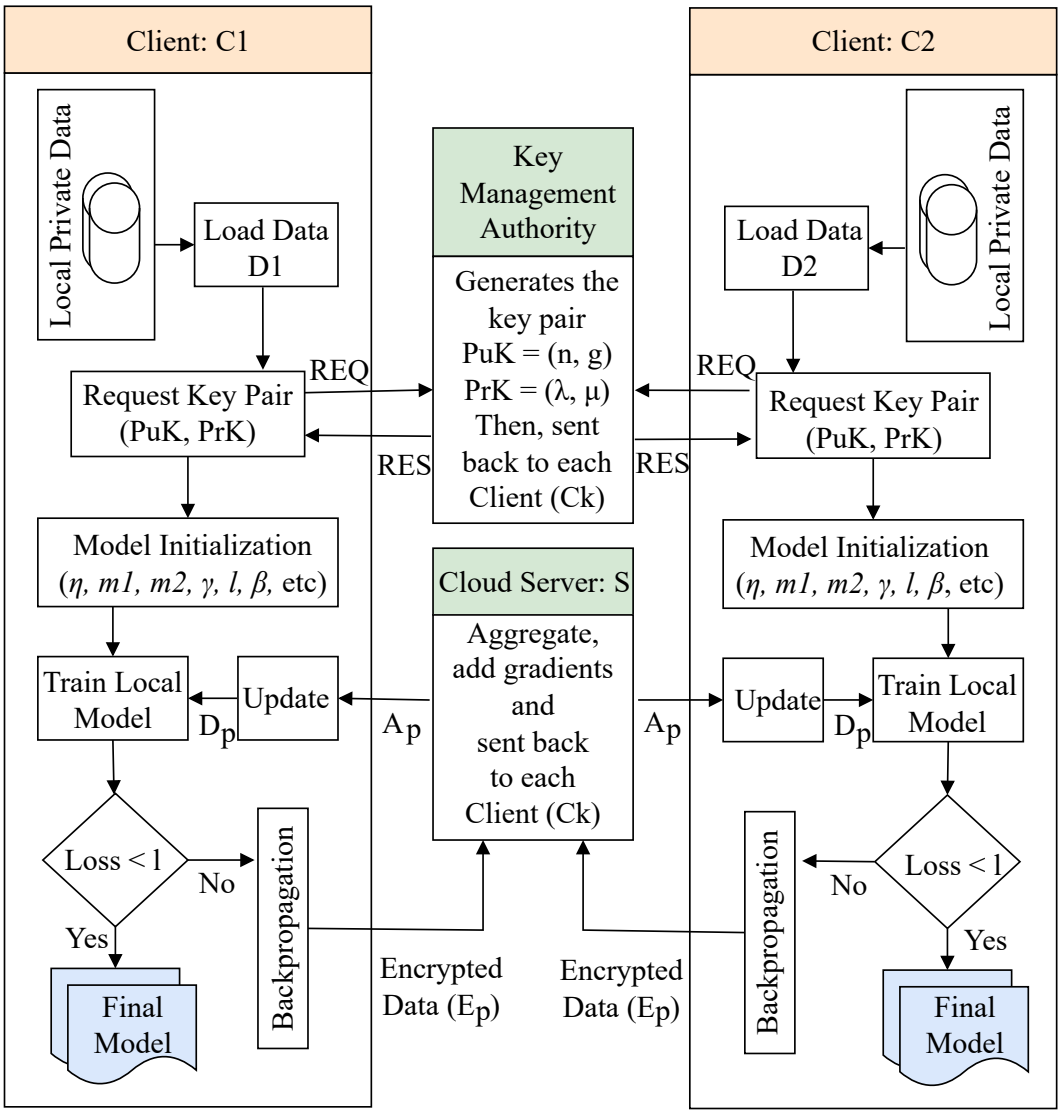


Figure 3.2: The overall workflow of the proposed DFL model with two clients and a key management authority.

3.3.1 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 cloud server S and each medical client C. Once a secure channel of communication has been established, S initializes the learning rate of the model η, the rate of exponential decay of moment estimator m_1, m_2 where $0 \leq m_1, m_2 < 1$, numerical stabilization γ, loss function of the model l, and batch size β. 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 $\alpha_k = N_k/(N_1 + N_2 + N_3 + ... + N_K)$;. Finally, set the index of the first communication round r to 1 between S and C.

3.3.2 Local Model Training by Medical Clients

Algorithm 1 shows the details procedure of the proposed model. 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 $\eta, m_1, m_2, \gamma, l, \beta, W^{r-1}, C, D_k$ from S. The training procedure of the local DL models continues until the loss function l 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 β and set the initial local model parameters by $W_k^r \leftarrow \bar{W}^{r-1}$. 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 $\bar{n}_1$, second moment estimate $\bar{n}_2$, and the bias-corrected learning rate η. Finally, update the local DL model parameter W_k^r and return the model parameters. Algorithm 2 summarizes the details of the training procedure of the local model. The training procedure of the local DL model is performed offline. So, there is no need to be concerned about the local model's computational complexity.

Algorithm 1: Privacy-Preserving DFL Model

Input : Security parameter: k, Medical clients set: C, Data resources of all medical 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 $\eta, m_1, m_2, \gamma, l, \beta$; 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 $\alpha_k = N_k/(N_1 + N_2 + ... + N_K)$; S set the index of first communication round r to 1;

Procedure:

for $r \leq R$ **do**

  (a) Medical clients (Upload):

  for $\forall k \in K$ **do**

    C_k calculates the r^{th} round model parameters W_k^r as per local DL model training with inputs: $\eta, m_1, m_2, \gamma, l, \beta, W^{r-1}, C, D_k$;

    for $\forall i \in \delta$ **do**

      $E_P(w_{k,i}^r) = P_E(w_{k,i}^r, PuK)$;

    C_k uploads $E_P(w_{k,i}^r)|i \in \delta$ to S;

  (b) Cloud server:

  for $\forall i \in \delta$ **do**

    $A_P = P_A(w_{1,i}^r, w_{2,i}^r, ..., w_{K,i}^r, \alpha_1, \alpha_2, ..., \alpha_K)$;

  S distributes $A_P|i \in \delta$ to all $C_k(k \in K)$;

  (c) Medical clients (Update):

  for $\forall k \in K$ **do**

    for $\forall i \in \delta$ **do**

      $D_P(\bar{w}_{k,i}^r) = P_D(A_P, PrK)$;

    C_k updates local model by $D_P(\bar{w}_{k,i}^r)|i \in \delta$;

  $r \leftarrow r + 1$;

return Efficient DFL model with parameters W^R.

Algorithm 2: Training of the Local DL Model

Input : Learning rate of the model η

Numerical stabilization γ

Loss function of the model l

Batch size β

Security parameter: k

Medical clients set: C

Data resources of all medical clients: $\{D_k|k \in K\}$

Moment estimator m_1, m_2, where $0 \leq m_1, m_2 < 1$

Output: Model parameters W_k^r

Initialization:

$\rightarrow$ Initialize momentum terms $n_1 = 0$ and $n_2 = 0$;

$\rightarrow$ Split D_k into batches with equal size β;

$\rightarrow$ Set initial local model parameters by $W_k^r \leftarrow \bar{W}^{r-1}$;

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: $\bar{n}_1 \leftarrow n_1/(1 - m_1^e)$;

    Compute bias-corrected 2^{nd} moment estimate: $\bar{n}_2 \leftarrow n_2/(1 - m_2^e)$;

    Compute bias-corrected learning rate: $\eta \leftarrow \eta.\sqrt{(1 - m_2)}/(1 - m_1)$;

    Update local model parameters: $W_k^r \leftarrow W_k^r - \eta.\bar{n}_1/(\sqrt{\bar{n}_2 + \gamma})$

  until *Loss function l converges*;

return Model parameters W_k^r.

3.3.3 Encryption of Model Parameters by Medical Clients

When the local DL model is trained and finally return 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 DL model are uploaded to the S by each C_k, where δ is the total number of parameters.

3.3.4 Aggregation of Model Parameters by Cloud Server

Cloud server S computes each C_k contribution ratio by calculating $\alpha_k = N_k/(N_1 + N_2 + N_3 + ... + N_K)$. Then, the contribution ratios α_k and encrypted parameters $E_P(w_{k,i}^r)|i \in \delta$ from all C_k are aggregated by S, where aggregated parameters $A_P = P_A(w_{1,i}^r, w_{2,i}^r, w_{3,i}^r, ..., w_{K,i}^r, \alpha_1, \alpha_2, \alpha_3, ..., \alpha_K)$. Finally, S sent back the aggregated parameters $A_P|i \in \delta$ to all $C_k(k \in K)$ by the secured communication channel.

3.3.5 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 DL model by the decrypted parameters $D_P(\bar{w}_{k,i}^r)|i \in \delta$. 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 ((an empirically determined threshold) rounds of interactions between S and C_k.

3.4 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 τ 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 $\mathbb{Z}_{n^2}$

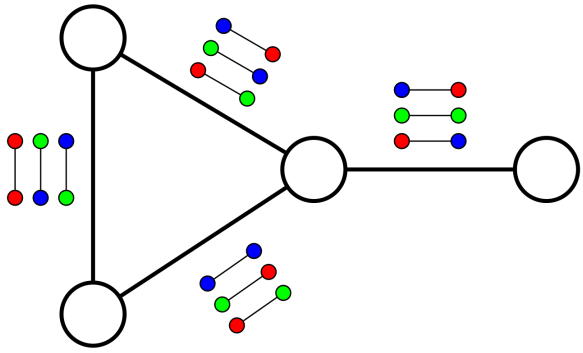


Figure 3.3: The internal computational complexity structure of the local model for each step.

is the set of integers $(0, 1, 2, 3, ..., n-1)$ with arithmetic being done modulo n^2. Figure 3.3 shows the internal computational complexity structure of the local model for each step.

As a result, the computational complexity of each C_k is approximately linearly proportional to δ ($C_k \underset{\sim}{\propto} \delta$) in a local DL model. Furthermore, S must perform K times of multiplication operations in $\mathbb{Z}^*_{n^2}$ when C_k aggregates all model parameters and contribution rations for each successful communication round.

3.5 Proposed DFL Model

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

3.5.1 Model Overall Architecture

The block diagram of the proposed model is shown in Figure 3.4, which contains two parts: the local part and the cloud server part. The local part can be further divided into two parts. The first one is the local model training part, and the last one is the local model validation part.

After preprocessing the raw DNA sequence data, we split the whole dataset into multiple datasets and applied it to the local model for training. The most important parameters for

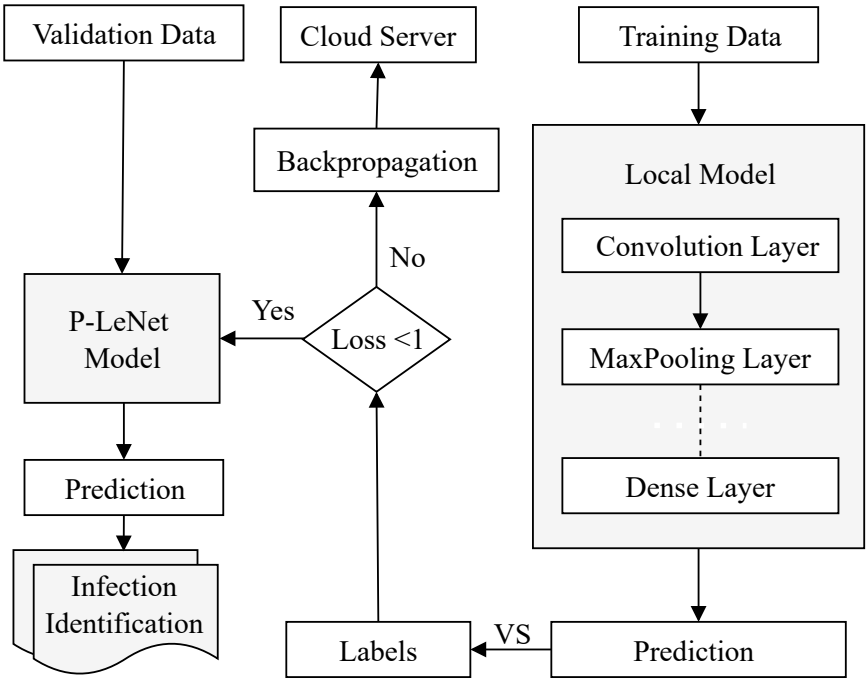


Figure 3.4: The block diagram of the proposed model with a local client device and a cloud server.

the selected model were determined through subsequent empirical experiments. We used a randomly selected training dataset to train the local model and a randomly selected validation dataset to validate the final model. If the local model loss is not minimal, the local model parameters have been back-propagated to the cloud server. Then, the cloud server collects and averages each of the local model parameters. After averaging the parameters, the cloud server distributes the updated parameters to all local client models. The procedure continues until the loss is reduced to a minimum. Thus, the final model has been selected based on its best performance metrics. We have also analyzed the dependability performance of the proposed model. Table 3.1 shows the hyper-parameters of the local model.

3.5.1.1 Internal Structure of the Local Model

The internal structure and the working mechanism of each layer of the client device's local model have been discussed in this section.

Table 3.1: The hyperparameters of the local model

Hyper-parameters	Value/Function
Number of Hidden Layers	2
Units in hidden layers	32, 64
Batch size	64
Epochs	200
Hidden layer activation function	ReLu
Output layer activation function	Softmax
Dropout	0.1
Optimizer	Adam
Learning Rate	0.001
Loss function	Categorical Crossentropy

3.5.1.2 Input Layer

The internal structure of the local model is shown in Figure 3.5. There are eight layers in the local model, including the input layer. The first layer is the input layer. The input of the network is the same shape for all clients, which brings the initial pre-processed encoded DNA sequence data into the model for further processing by subsequent layers.

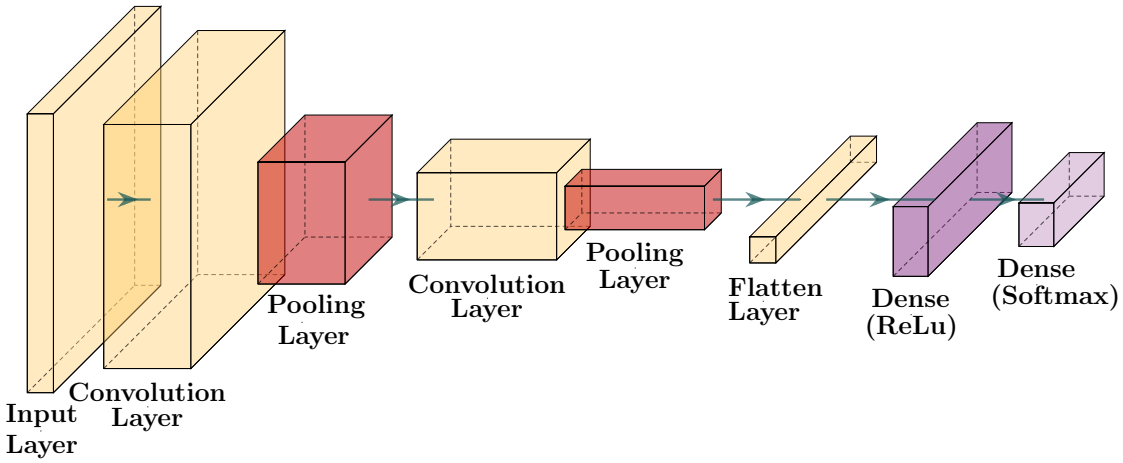


Figure 3.5: The internal structure of the local model including input and output layer.

3.5.1.3 Convolution Layer

The second layer of our model is the convolution layer, which is directly connected to the input layer. This layer is a trainable layer that converts the encoded DNA sequence into a vector of features [85]. Convolution is operated by windowing each convolution unit over the encoded DNA sequence [47]. Figure 3.6 shows the working procedure of the convolution layer of the local model.

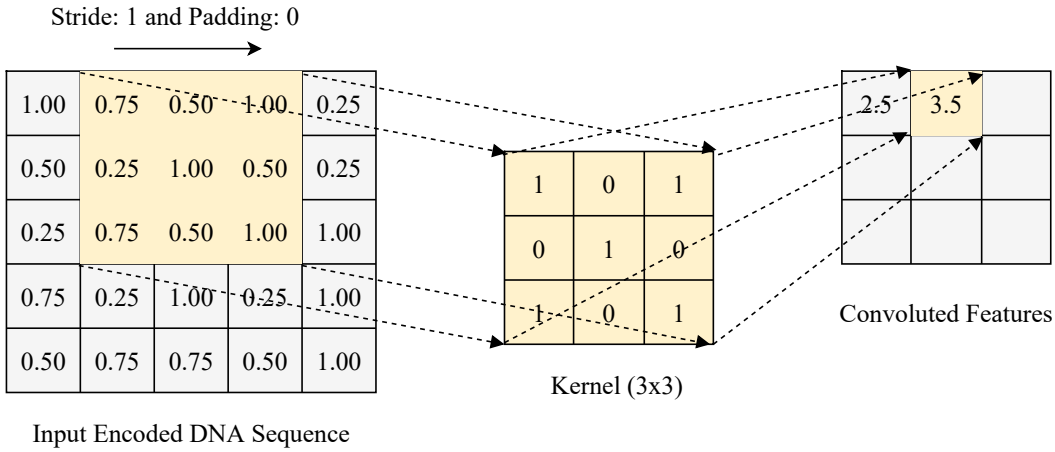


Figure 3.6: The working procedure of the convolution layer of the local model.

3.5.1.4 Max Pooling Layer

The third layer is the polling layer, which is a non-trainable layer to change the size of the feature map [85]. There are different pooling techniques, such as maximum pooling and average pooling. We used a total of two max pooling layers after the convolution layer, which selects the maximum activated value among neurons. Figure 3.7 shows the working procedure of the max pooling layer of the local model.

3.5.1.5 Dropout Layer

The dropout layer randomly removes some neurons or connections at training time between consecutive layers based on a predefined dropout ratio [23]. To avoid overfitting and train

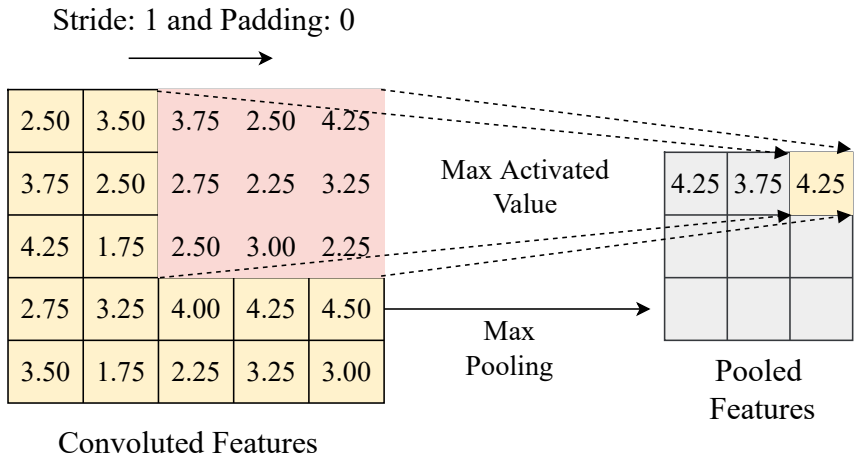


Figure 3.7: The working procedure of the max pooling layer of the local model.

robust features, we added a dropout layer with a (10/100) or 0.10 ratio after the second max pooling layer [86]. The mechanism of the dropout layer is shown in Figure 3.8.

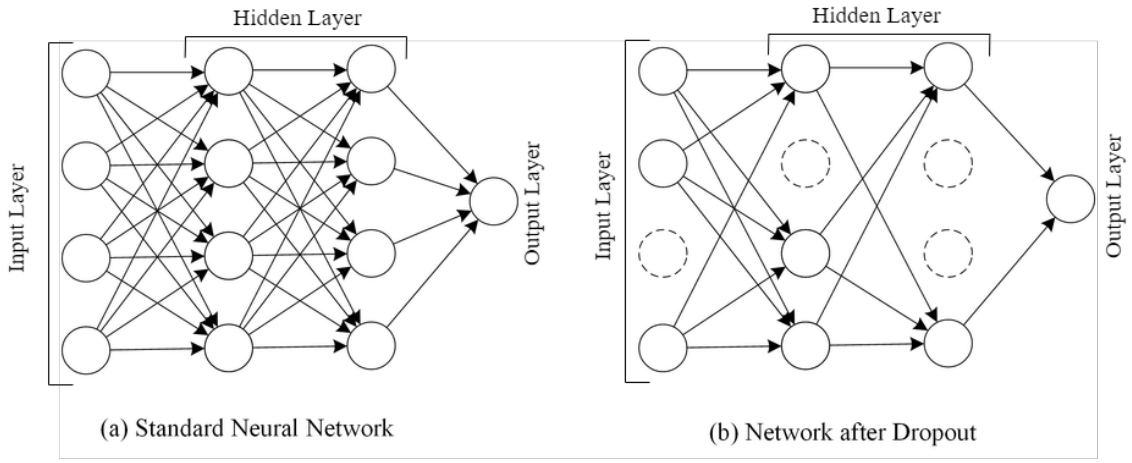


Figure 3.8: The working procedure of dropout layer of the local model. (a) Standard neural network and (b) Network after dropout.

3.5.1.6 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 [23]. Figure 3.9 shows the mechanism of the flattened layer.

3.5.1.7 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. To combine features from a sequence and to extract high-order features, we employ this hidden layer [7]. The ReLU activation function is used in this layer. Figure 3.9 also shows the working procedure of the ReLU activation function.

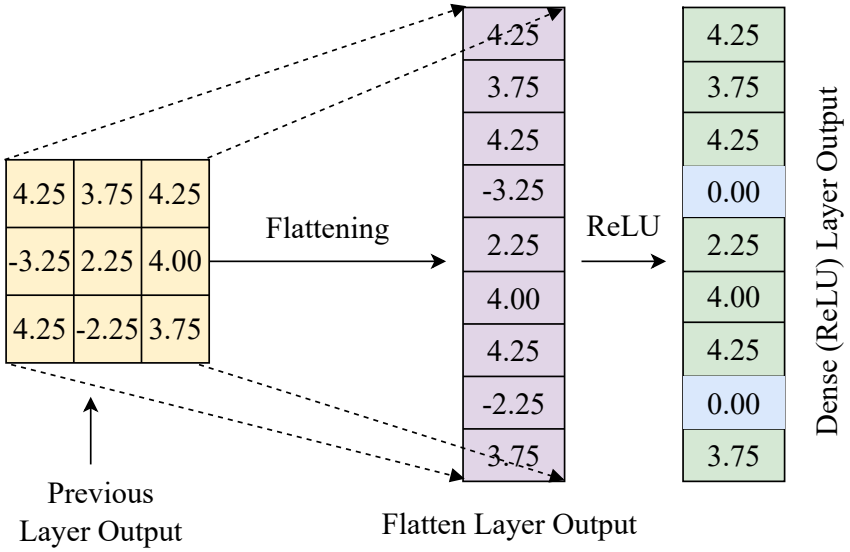


Figure 3.9: The working procedure of the flattened and dense layer of the local model.

3.5.1.8 Softmax Layer

The last layer of our model is the softmax layer, which is connected to a dense layer. This layer is used to calculate the probability of each class or label, whether it is a new infection or not [7]. The mechanism of this layer is shown in Figure 3.10.

3.6 Cryptosystem-Based Secure Communication Protocol

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

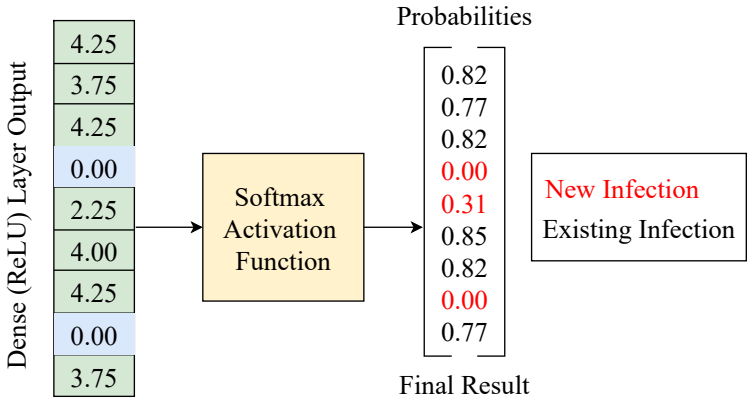


Figure 3.10: The working procedure of the softmax layer of the local model.

3.6.1 Key Generation

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 3.1.

$$gcd\left((p \times q), (p-1)(q-1)\right) = 1 \tag{3.1}$$

Then, calculate n and λ by the following equation 3.2 and 3.3.

$$n = (p \times q) \tag{3.2}$$

$$\lambda = lcm\ (p-1, q-1) \tag{3.3}$$

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 equation 3.4.

$$\mu = (L\ (g^{\lambda} \bmod n^2))^{-1} \bmod n \tag{3.4}$$

where function $L(\alpha)$ is defined as $L(\alpha) = (\alpha - 1)/n$.

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 and each C_i, where $i \in \{1, 2, 3, 4, ..., K\}$ to establish a secure communication channel between S and each C_k.

3.6.2 Parameter Encryption

Model parameters $w^r_{k,i}$ is to be encrypted where $0 \leq w^r_{k,i} < n$. Now, select a random number R where $0 < R < n$ or $R \in \mathbb{Z}^*_n$ and encrypt the model parameter using PuK by the following equation 3.5. Figure 3.11 shows the parameter encryption technique where all model parameters $w^r_{k,i}$ is to be encrypted.

$$E_P(w^r_{k,i}) = g^{(w^r_{k,i})} \cdot R^n \bmod n^2 \tag{3.5}$$

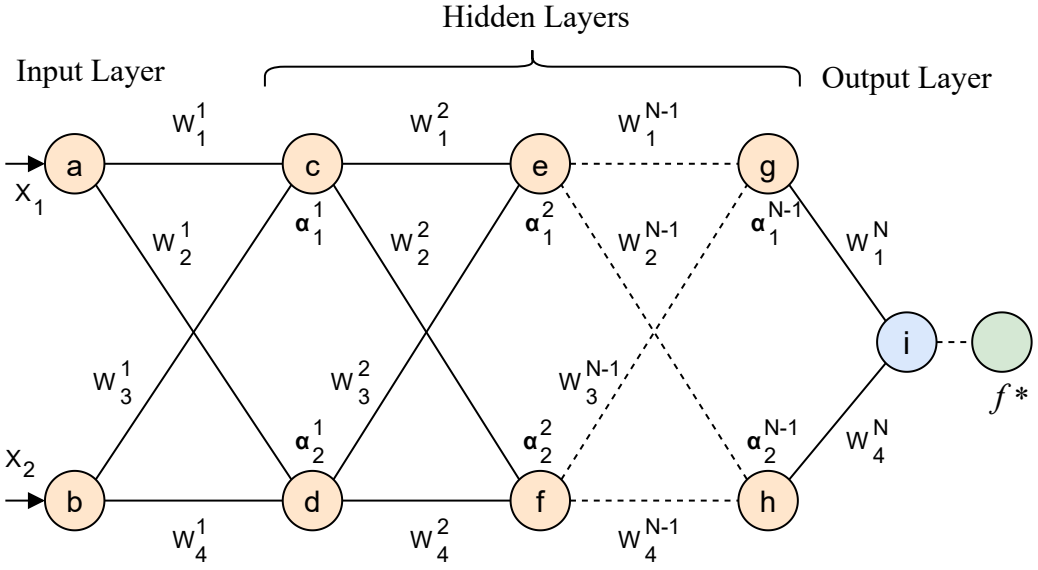


Figure 3.11: The parameter encryption technique where all model parameters $w^r_{k,i}$ is to be encrypted.

3.6.3 Parameter Aggregation

The contribution ratios $\alpha_k = \{\alpha_1, \alpha_2, ..., \alpha_k\}$ of each C_k are computed, and these α_k are multiplied by a factor of 10^3 to make them positive integers. Then, these amplified α'_k and encrypted model parameters $E_P(w^r_{k,i}) = \{E_P(w^r_{k,1}), E_P(w^r_{k,2}), ..., E_P(w^r_{k,K})\}$ from all C_k are aggregated by S using the following equation 3.6.

$$\begin{aligned} A_P &= \prod_{i=1}^{K} E_P\left(w^r_{k,i}\right)\alpha'_i \\ &= \left[E_P(w^r_{k,1})\alpha'_1\right] \cdot ... \cdot \left[E_P(w^r_{k,K})\alpha'_K\right] \\ &= \left[g^{(w^r_{k,1})\alpha'_1} R_1^{n\alpha'_1} \bmod n^2\right] \cdot ... \cdot \left[g^{(w^r_{k,K})\alpha'_K} R_K^{n\alpha'_K} \bmod n^2\right] \\ &= \left[g^{(w^r_{k,1})\alpha'_1} R_1^{n\alpha'_1}\right] \cdot ... \cdot \left[g^{(w^r_{k,K})\alpha'_K} R_K^{n\alpha'_K}\right] \bmod n^2 \\ &= g^{\sum_{i=1}^{K}(w^r_{k,i})\alpha'_i} \cdot \prod_{i=1}^{K} R_i^{n\alpha'_i} \bmod n^2 \end{aligned} \tag{3.6}$$

3.6.4 Parameter Decryption

When each C_k received the encrypted aggregated model parameter A_P, where $A_P \in \mathbb{Z}^*_{n^2}$ from the S. Then, each C_k decrypts the encrypted aggregated model parameter by each PrK by the following equation 3.7.

$$\begin{aligned} D_P(\bar{w}^r_{k,i}) &= L\left(A_P \bmod n^2\right) \cdot \mu \bmod n \\ &= L\left(A_P \bmod n^2\right) \cdot \left[L\left(g^\lambda \bmod n^2\right)\right]^{-1} \bmod n \\ &= L\left(A_P \bmod n^2\right) \cdot \left[\frac{1}{L\left(g^\lambda \bmod n^2\right)}\right] \bmod n \\ &= \frac{L\left(g^{\sum_{i=1}^{K}(w^r_{k,i})\alpha'_i} \cdot \prod_{i=1}^{K} R_i^{n\alpha'_i} \bmod n^2\right)}{L\left(g^\lambda \bmod n^2\right)} \bmod n \\ &= \sum_{i=1}^{K}(w^r_k) \cdot \alpha_i \bmod n \end{aligned} \tag{3.7}$$

Now, compute the average value and update the local model by the following equation 3.8. Here, 10^3 is a scalar used to transform the contribution ratios to positive integers.

$$D_P(\bar{w}_{k,i}^r) = \frac{D_P(\bar{w}_{k,i}^r)}{10^3} \tag{3.8}$$

3.6.5 An Illustrative Example

The Paillier cryptosystem is homomorphic, meaning that the encrypted values of two numbers can be added, subtracted, or multiplied together, and the result will be the encrypted value of the sum, difference, or product of the two numbers. This property makes the Paillier cryptosystem useful for a variety of applications, such as secure electronic voting and digital signatures.

Here is an illustrative example of how the Paillier cryptosystem can be used to encrypt and decrypt a message:

Step 1: Setup

Client1 wants to send an encrypted message to Client2 using the Paillier cryptosystem. Client1 generates his public and private keys:

Public Key: n = 1057 (product of two large prime numbers, p = 31 and q = 34) and g = 58 (randomly chosen integer modulo n)

Private Key: = lcm(p-1, q-1) = lcm(31, 34) = 527 (where lcm represents the least common multiple)

Client1 shares his public key (n, g) with Client2.

Step 2: Encryption

Client2 wants to receive an encrypted message from Client1. He writes his plaintext message, "Hello, Client2!". Now, he encrypts the message using Client1's public key.

To encrypt the message, Client2 first converts the plaintext message into a numerical representation. He assigns each character a unique number according to some agreed-upon mapping. For simplicity, let's say 'H' is 1, 'e' is 2, 'l' is 3, 'o' is 4, ',' is 5, ' ' is 6, 'C' is 7, 'l' is 8, 'i' is 9, 'e' is 10, 'n' is 11, 't' is 12, '2' is 13, and '!' is 14.

Now, client2 applies the encryption formula: $C = (g^m * r^n) \ mod \ n^2$, where C is the ciphertext, m is the plaintext message, r is a randomly chosen integer, and n is the public key component.

Client2 randomly selects r = 22. He substitutes the numerical representation of her plaintext message into the formula and performs the calculations for each character:

For 'H': - C1 = $(58^1 * 22^{1057}) \ mod \ 1057^2$

For 'e': - C2 = $(58^2 * 22^{1057}) \ mod \ 1057^2$

He repeats this process for each character in the message.

Finally, Client2 sends the encrypted message C1, C2, ..., Cn to client1.

Step 3: Decryption

Client1 receives the encrypted message C1, C2, ..., Cn from Client2. He applies the decryption process using his private key.

To decrypt the ciphertext, Client1 uses the decryption formula:

m = $(L(Cmodn^2)/L(gmodn^2))modn$, where m is the plaintext message, C is the ciphertext, is the private key component, and L(x) = (x - 1) / n.

Client1 calculates the decryption for each ciphertext component:

For C1: - m1 = $(L(C1^{527} \ mod \ 1057^2)/L(58^{527} \ mod \ 1057^2)) \ mod \ 1057$

For C2: - m2 = $(L(C2^{527} \ mod \ 1057^2)/L(58^{527} \ mod \ 1057^2)) \ mod \ 1057$

Client1 repeats this process for each ciphertext component.

Finally, Client1 combines the numerical representation of the decrypted message and converts it back into text according to the agreed-upon mapping. In this case, he obtains the message "Hello, Client2!".

Step 4: Communication

Client1 and Client2 can now communicate securely, with Client1 encrypting his messages using his public key, and Client2 decrypting them using Client1's private key.

3.6.6 Complexity Analysis

The complexity of the Paillier cryptosystem can be analyzed in terms of the following three operations:

Encryption:

The encryption operation takes a message and a public key as input and produces a ciphertext as output. The complexity of the encryption operation is $O(n^2)$.

Decryption:

The decryption operation takes a ciphertext and a private key as input and produces the original message as output. The complexity of the decryption operation is $O(n^3)$.

Key generation:

The key generation operation produces a public and private key pair. The complexity of the key generation operation is $O(n^3)$.

The overall complexity of the Paillier cryptosystem is $O(n^3)$. This means that the Paillier cryptosystem is relatively slow for large values of n. However, the Paillier cryptosystem is still a secure and efficient encryption algorithm for many applications.

3.7 Summary

In this chapter, we elaborate on the proposed dependable and privacy-preserving DFL model. First, we explained the overall workflow of the proposed DFL model with two clients and a key management authority. We have also briefly explained all the techniques, including 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. Then, we explained the overall computational complexity of the selected models. Furthermore, we explained the overall architecture of the proposed model, including the internal structure of the local and global models. Finally, we concluded the chapter by briefly describing the cryptosystem-based secure communication protocol.

This chapter sets the flow and context for coming chapters, which will build on the concepts and workflows from here on. The next chapter covers the initial steps of the environment setup, data collection, overview of the dataset, processing, and visualization procedures. Also, we explain the most important parts of the model-building process, such as model training, model-testing, evaluation, interpretation, and deployment.

Chapter 4

Model Implementation and Evaluation

The chapter 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.

4.1 Environmental Setup

We have performed all the analyses on the cloud with the following GPU: NVIDIA K80/T4, GPU memory: 16 GB, GPU memory clock: 3.40GHz, performance: 4.1 TFLOPS/8.1 TFLOPS, number of GPU cores: 7i, RAM: 32 GB (up-gradable to 35.75GB), solid-state drive: 512GB, max execution time: 24 hours, and max idle time: 90 min.

4.2 Implementation Procedure

Figure 4.1 shows the overall implementation process of the proposed model with *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 [87]. The main

idea behind the *PySyft* library is to extend the APIs of popular DL mechanisms such as *PyTorch*, *Kera*, and *TensorFlow* [58]. So, researchers can immediately build privacy-preserving private and secure applications without having to learn a new DL or DTL mechanism. Furthermore, *PyTorch* is another *Python* library that provides high-level features, including tensor computation (like *NumPy*) with strong GPU acceleration [74].

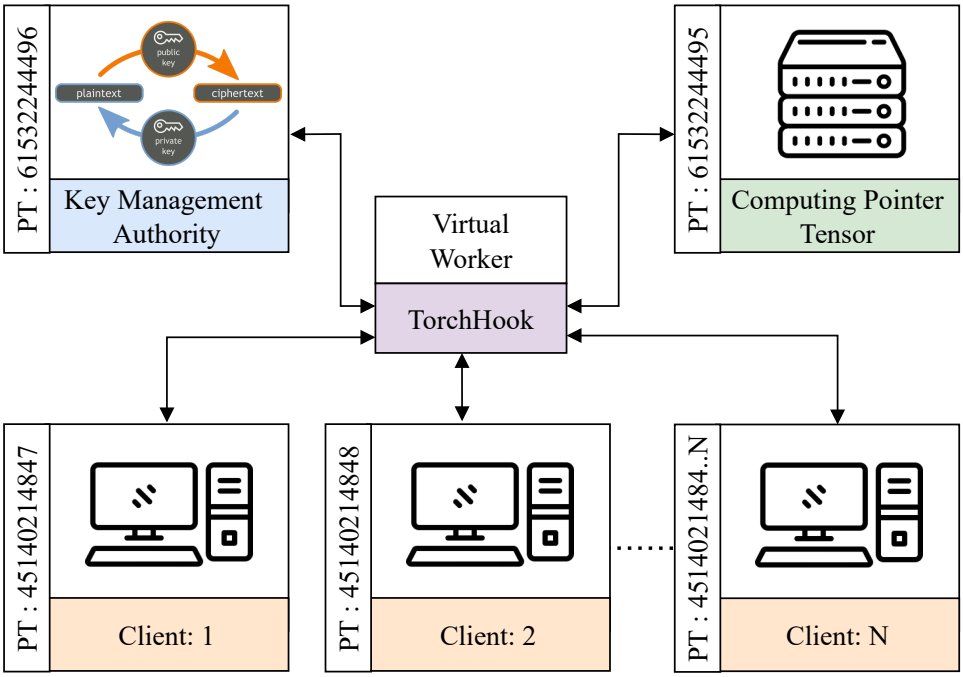


Figure 4.1: The overall implementation process of the proposed model with PySyft and PyTorch libraries.

First, we installed the *PySyft* and *PyTorch* libraries along with other necessary libraries to implement the proposed FL setting. Then, we created the virtual client devices by the *VirtualWorker* method with the help of the *TorchHook* mechanism that simulates each virtual client device. We have also initialized the central server called *"secure_worker"*. Each virtual client device 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 virtual client devices. After successful secure communication was established between client devices and the central server, we stored the data in a tensor and then divided the data into features and targets.

Furthermore, we calculated the number of samples per virtual client device and divided the

features and targets, and finally sent them to all virtual client devices. Next, we created the model and sent a copy of the model to all the client devices. 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 device individually. Each local client device 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 client devices. Then, each client device decrypted the model parameters, updated the local model, and retrained the local model using its private dataset. We performed rounds till we obtained the desired model. 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 devices without exposing each training data to others. We were also able to aggregate the updated model parameters from each client device by a trusted central server to prevent data privacy and security.

4.3 Dataset Description

The dataset has been downloaded from the genes database at the National Center for Biotechnology Information (NCBI) which contains 583 sequences [7]. Table 4.1 summarizes the dataset. 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.

Table 4.1: An overview of the dataset with virus genes, label, and the number of samples.

Virus Genes	Class Label	Number of Samples
SARS-CoV-2	0	96
MERS-CoV	1	236
HCoV-EMC	1	4
HCoV-OC43	1	138
HCoV-229E	1	22
HCoV-4408	1	2
HCoV-NL63	1	58
HCoV-HKU1	1	17
SARS-CoV	1	7
SARS-CoV P2	1	1
SARS-CoV HKU-39849	1	1
SARS-CoV GDH-BJH01	1	1
Total samples	-	583

4.3.1 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 is normally done by removing irrelevant features, removing duplicate samples, converting non-numerical features into numeric features, and dealing with missing values if any.

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 *unique_everseen()* method. Then, we used *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 [88]. Figure 4.2 shows the process of translating the DNA sequence into a sequence of words with window size 3 and slide stride 1.

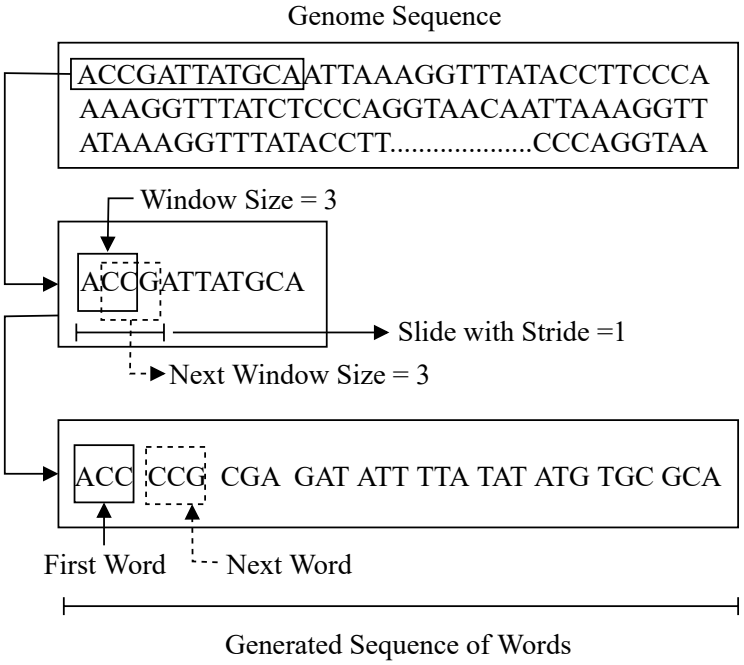


Figure 4.2: The process of translating the DNA sequence into a sequence of words with window size = 3 and slide stride = 1.

Figure 4.3 (a) 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 4.3 (b) 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.

4.4 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 [23, 88]. For the proposed DFL model, we first considered the

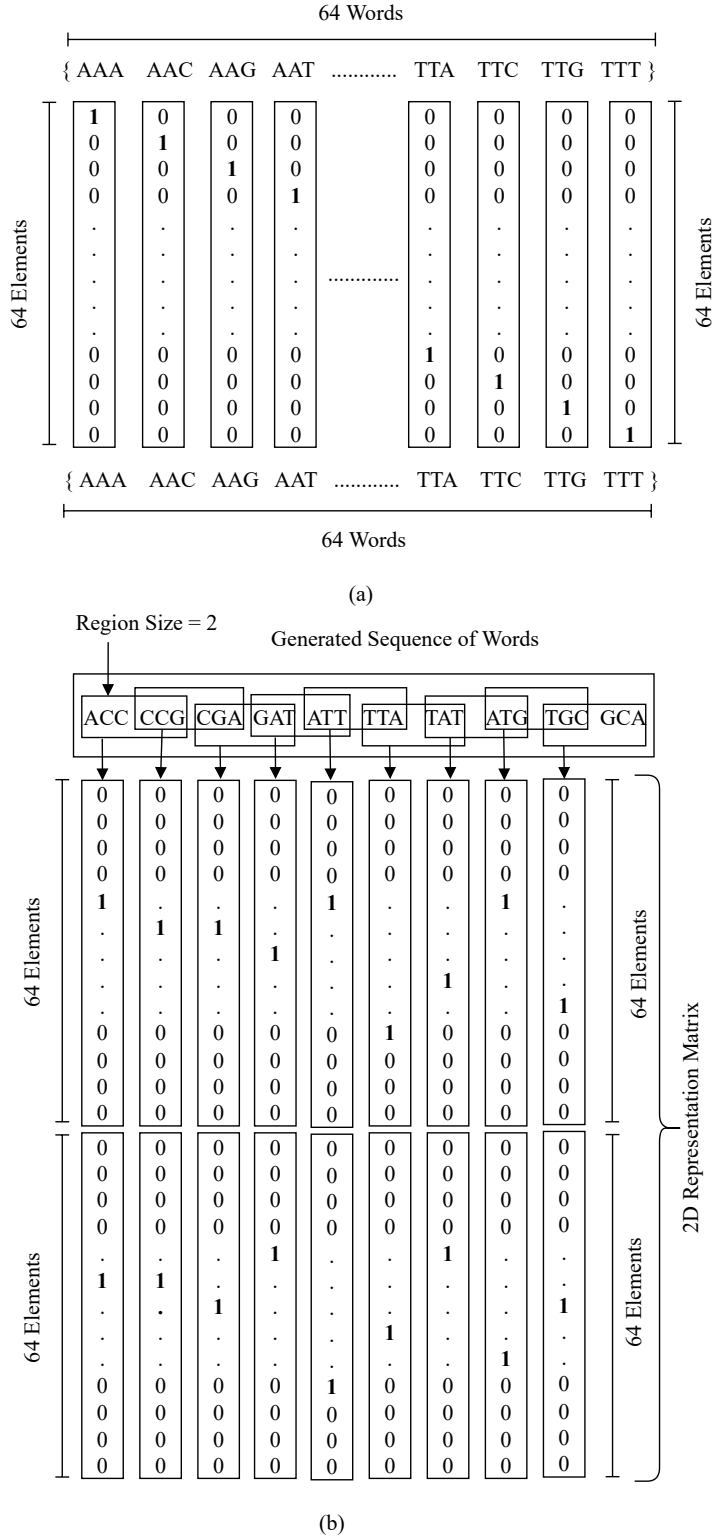


Figure 4.3: (a) Dictionary representation (b) One-hot 2D vector representation of generated DNA sequence words with region size = 2.

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. Figure 4.4 summarizes the steps involved in evaluating the performances of the selected DTL models and proposed DFL models.

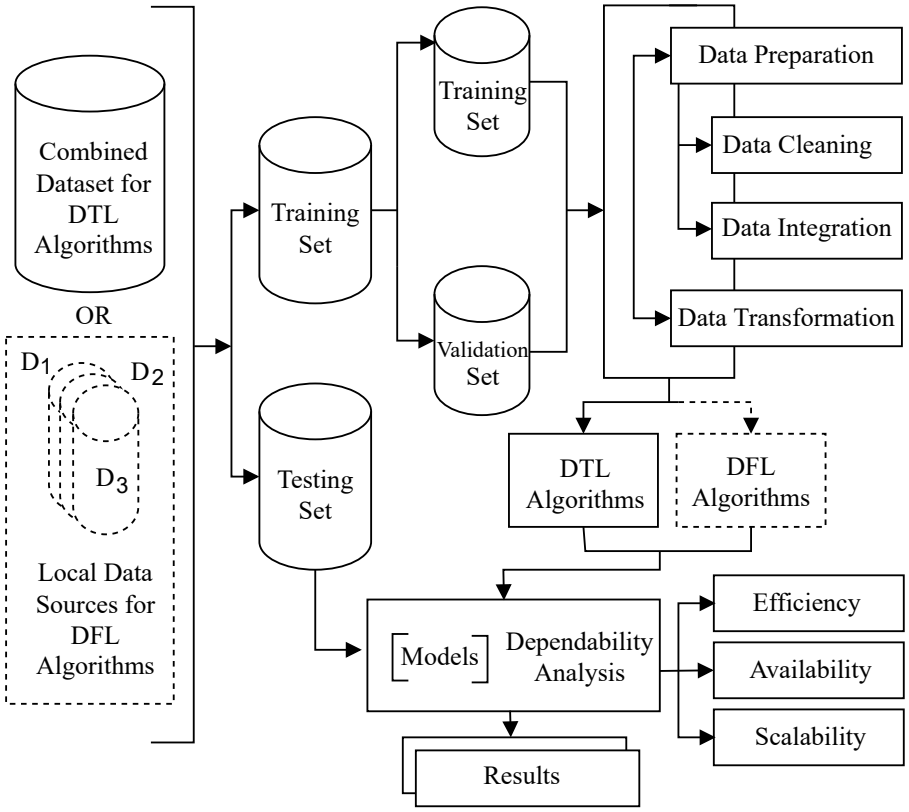


Figure 4.4: The overall evaluation process of the proposed DFL and the DTL models.

We have also considered DTL algorithms because some of their variants have been successfully applied to solve classification tasks related to DNA sequencing. Therefore, we have considered Convolutional Neural Network (CNN), Fully Convolutional Networks (FCN), Inception Network (IncepNet), LeCun Network (LeNet), and Residual Neural Network (ResNet) algorithms because of their optimal performance [89]. The overall performance of the selected DTL models has been evaluated with a wide range of hyperparameters.

For all the selected DTL models, batch size, loss function, and the optimizer are *64*, *categorical_crossentropy*, and *Adam*, respectively. For the CNN, ResNet, and LeNet models, the hidden layer activation function is the same. Furthermore, FCN, IncepNet, and ResNet models used the same number of hidden layers, but the units in the hidden layer were different. Particularly for the LeNet model, the number of hidden layers is 2, and the units in the hidden layer are 32, 64. We have evaluated the selected DTL models with a wide range of tested hyperparameters and obtained the optimal performance with these combinations. Moreover, we have used the *Adam* optimizer for all the models because it combines the best properties of the *AdaGrad* and *RMSProp* techniques to provide an optimization algorithm. The hyper-parameters of all the selected DTL algorithms are shown in Table 4.2.

Table 4.2: The hyper-parameters of the selected DTL models.

Parameters	CNN	FCN	IncepNet	ResNet	LeNet
Number of hidden Layers	4	3	3	3	2
Units in hidden layers	32,64,128,64	128,256,128	32,64,32	128,256,128	32,64
Batch size	64	64	64	64	64
Hidden layer activation function	relu	tanh	linear	relu	relu
Output activation function	sigmoid	softmax	sigmoid	softmax	softmax
Dropout	0.2	N/A	N/A	0.1	0.1
Optimizer	Adam	Adam	Adam	Adam	Adam

4.5 Summary

This chapter covered quite a lot of ground in terms of understanding, processing, and wrangling data. This chapter also includes the environment setup and model training pipeline. Furthermore, we explained the most important aspects of the model-building process, including deployment. Details of various types of model hyperparameters were also discussed in this chapter. In the next chapter, we will discuss the overall performance of the selected models with computational complexity and dependability analysis.

Chapter 5

Result Analysis

This chapter 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.

5.1 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 [46]. Equation 5.1 defines the mathematical representation of accuracy.

$$Accuracy = \frac{T_{pos} + T_{neg}}{T_{pos} + T_{neg} + F_{pos} + F_{neg}} \tag{5.1}$$

The precision shows how accurate the model is in identifying positive samples [46]. Equation 5.2 defines the mathematical representation of precision.

$$Precision = \frac{T_{pos}}{T_{pos} + F_{pos}} \tag{5.2}$$

The recall measures the model's ability to detect positive samples [46]. The higher the recall, the more positive samples detected. Recall can be defined by the equation 5.3.

$$Recall = \frac{T_{pos}}{T_{pos} + F_{neg}} \tag{5.3}$$

The F1-score computes the harmonic mean of precision and recall, respectively. The F1-score has a range of 0.0 to 1.0, with 1.0 denoting model perfection [46]. Equation 5.4 defines the mathematical representation of F1-score.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.4}$$

Also, the ROC AUC is an important metric since it shows how efficiently the proposed model can classify new infections and existing infections. Diagnostic accuracy, which is

needed to measure ROC AUC, depends on the sensitivity, or true positive rate (TPR), often called recall, and the specificity, or true negative rate (TNR) [7]. Equations 5.5 and 5.6 define the mathematical representation of sensitivity and specificity, respectively.

$$Sensitivity = \frac{T_{pos}}{T_{pos} + F_{neg}} \tag{5.5}$$

$$Specificity = \frac{T_{neg}}{T_{neg} + F_{pos}} \tag{5.6}$$

Additionally, we incorporated a variety of analysis scenarios with different parameters in this analysis (e.g., the number of hidden layers, the size of the hidden layers, the number of epochs, the activation functions for the hidden and output layers, loss functions, etc.) because these factors have a big impact on the performance metrics.

5.2 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 [23,46]. Therefore, we considered five DTL models (e.g., CNN, FCN, IncepNet, LeNet, and ResNet) because of their optimal performance. Table 5.1 shows the performance comparison metrics of the selected DTL models.

Table 5.1: DTL models' performance comparison metrics.

Algorithm	Accuracy	Precision	Recall	F1-Score	ROC AUC
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

In this analysis, we have considered a total number of 1000 epochs where the LeNet model shows the optimal performance compared to the others, with an accuracy score of 0.9907, precision score of 0.9903, recall score of 0.9907, F1-score of 0.9901, and ROC AUC score of 0.8598. Figure 5.1 shows the accuracy score of the selected DTL models for every single epoch for both the training and validation phases. On the other hand, the FCN model shows the lowest performance in both phases. In detail, according to Figure 5.1, the accuracy score of the FCN model starts with 0.9601 for the training phase and 0.9505 for the validation phase in epoch number 15. However, it increased to 0.9850 in epoch number 90 and 0.9885 in epoch number 550 for the training and validation phases, respectively.

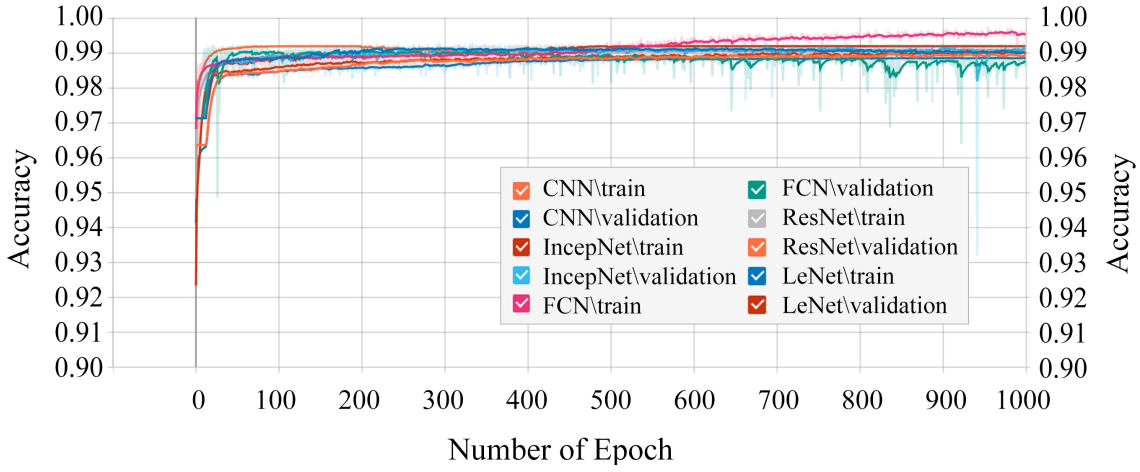


Figure 5.1: DTL models' training and validation accuracy.

Moreover, the CNN model achieves the second-best performance with an accuracy score of 0.9906. The IncepNet and ResNet models achieve an accuracy score of 0.9900, which is the third-best performance. However, the CNN, IncepNet, LeNet, and ResNet models have an overall precision score of almost 0.9903, but the FCN model has the lowest precision score of 0.9859. The lowest precision score of the FCN model indicates that most of the predicted labels are incorrect. Taking into account all aspects of performance indicators, we conclude that the LeNet model outperforms the other models. Figure 5.2 shows the trend of the accuracy score of the LeNet model for both phases, where accuracy increases rapidly at epoch number 10 and reaches a peak of close to 0.9907 at epoch number 530. However, it remains almost stable up to the epoch number 1000, as shown in Figure 5.2. The overall performance of the LeNet model indicates that most of the predicted labels correctly classify new infections and existing infections.

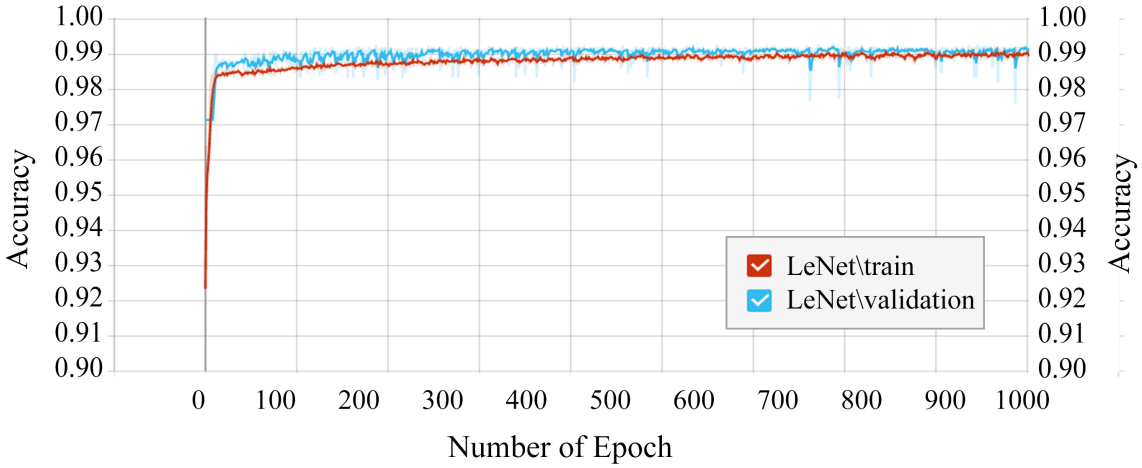


Figure 5.2: Training and validation accuracy of the LeNet model.

Furthermore, we analyzed the losses of the selected DTL models. Figure 5.3 shows each epoch's training and validation losses. The FCN model shows the highest losses in both phases, which indicates the model cannot provide a reliable classification. In contrast, the LeNet model shows the lowest loss in both phases. In detail, the loss of the LeNet model starts at 0.3113 for the training phase and 0.5208 for the validation phase, as shown in Figure 5.4. However, it decreases to 0.0707 in epoch number 130 and 0.0813 in epoch number 135 for the training and validation phases, respectively. Then, the loss score remains stable up to the end.

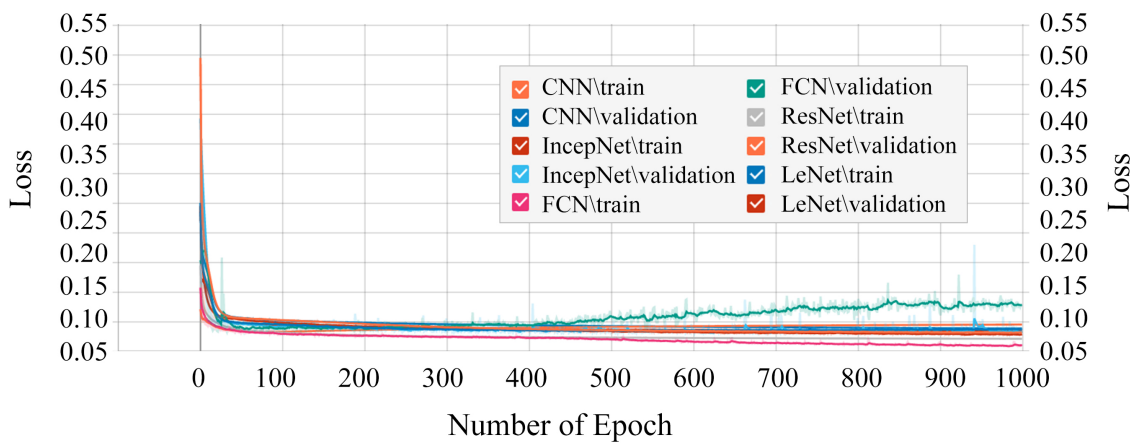


Figure 5.3: DTL models' training and validation loss.

The losses of the CNN, IncepNet, and ResNet models remain almost steady during both phases, which is shown in Figure 5.3. The loss of the CNN and IncepNet models is 0.3001 at the beginning, which declines gradually to 0.0801 at epoch number 500 and remains stable for both phases.

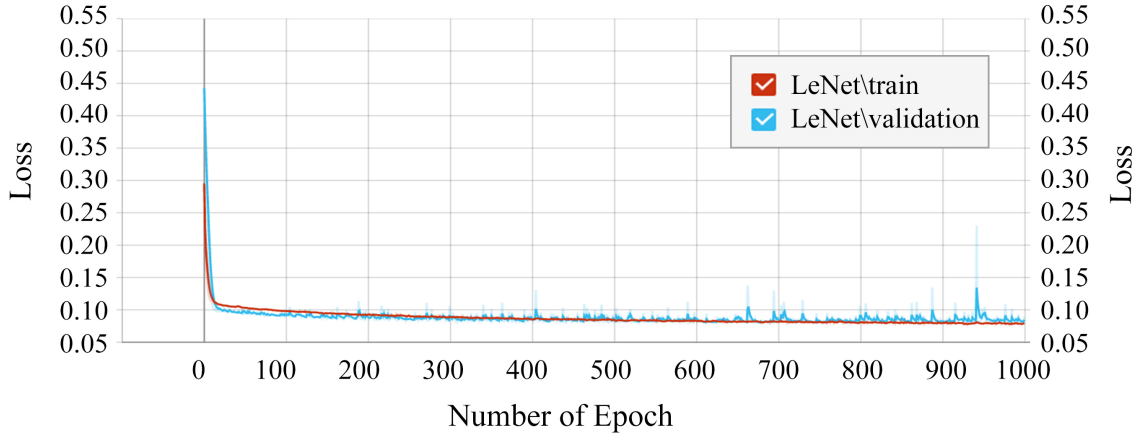


Figure 5.4: Training and validation loss of the LeNet model.

5.3 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 5.2 shows the quantitative performance metrics summary of the DFL models with different numbers of client devices.

Table 5.2: DFL models' performance comparison metrics.

DFL	Accuracy	Precision	Recall	F1 Score	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

First, we considered 4 client devices for this analysis. Figure 5.5 shows the classification

performance of the DFL model using 4 client devices. In more detail, the accuracy of the model starts at 0.8350 and increases to 0.9869 in round number 6. Then, the score remains stable up to round number 8, which indicates that the model can provide a reliable classification between new infections and existing infections. Also, the precision score is 0.5350 at the beginning and increases gradually to 0.9785 at round number 5, which remains stable up to the last round. Furthermore, the recall score of the model jumps rapidly in round number 4 and it reaches a peak of close to 0.9191 at epoch number 6. According to Figure 5.5, which remains almost stable up to round number 8. Moreover, the model shows the lowest F1 score at the beginning. In detail, the F1 score of the model starts with 0.2702 at round number 1, and it increases dramatically to 0.9189 in round number 3 and 0.9393 in round number 6. The score remained constant up to the last round. Finally, the ROC AUC of the model starts with a score of 0.4502 at the beginning. But this score rises gradually with the increase of the round number and reaches 0.9440 when the round number is 5, and then remains stable up to the end.

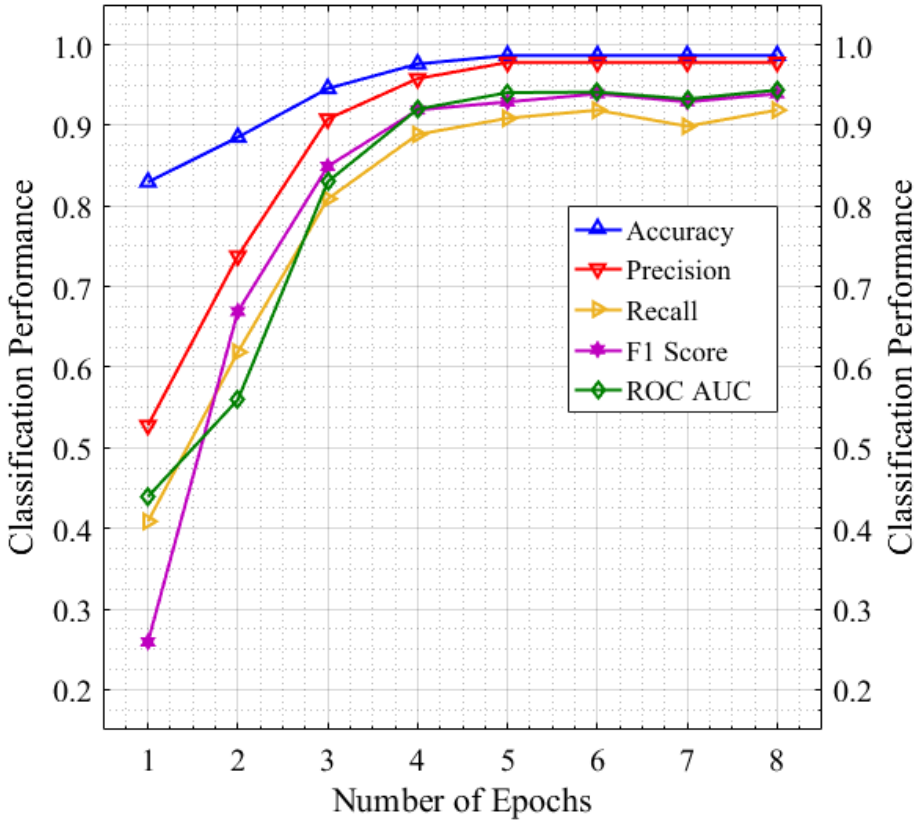


Figure 5.5: Classification performance of the DFL model using 4 client devices.

Next, we considered 6 client devices for analysis. Figure 5.6 shows the classification performance of the DFL model using 6 client devices. In more detail, the accuracy of the model starts at 0.8830 and increases to 0.9912 in round number 5. Then, the score remains stable up to round number 8, which indicates that the model can provide a reliable classification between new infections and existing infections. Also, the precision score is 0.5490 at the beginning and increases gradually to 0.9823 at round number 5, which remains stable up to the last round. Furthermore, the recall score of the model jumps rapidly in round number 5 and it reaches a peak of close to 0.9804 at epoch number 6. According to Figure 5.6, which remains almost stable up to round number 8.

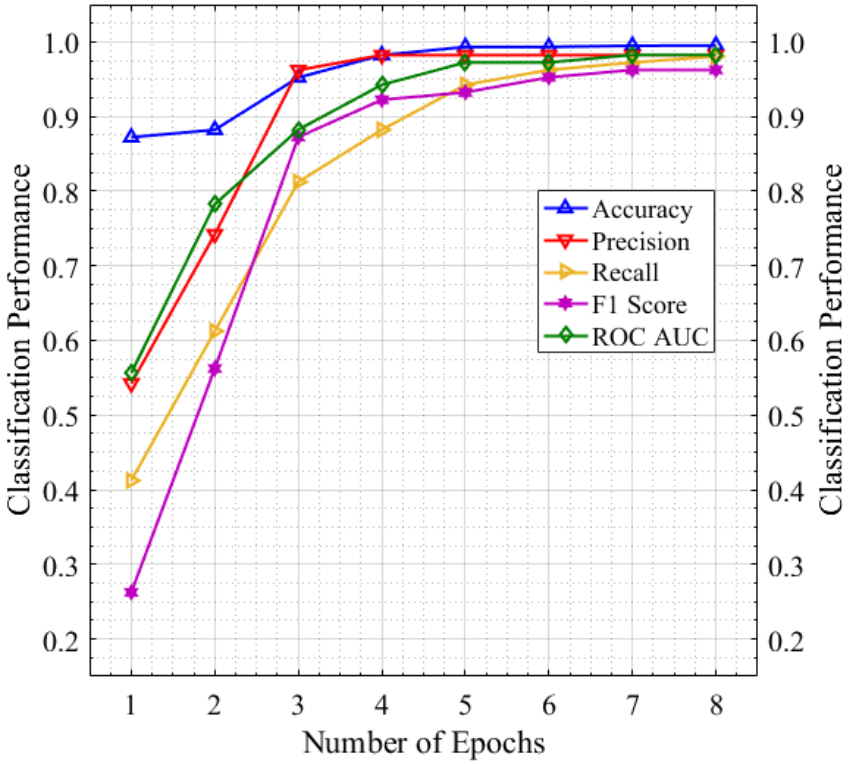


Figure 5.6: Classification performance of the DFL model using 6 client devices.

Moreover, the model shows the lowest F1 score at the beginning. In detail, the F1 score of the model starts with 0.2805 at round number 1, and it increases dramatically to 0.9288 in round number 4 and 0.9624 in round number 7. The score remained constant up to the last round. Finally, the ROC AUC of the model starts with a score of 0.5508 at the beginning. But this score rises gradually with the increase of the round number and reaches 0.9824 when the round number is 5, and then remains stable up to the end.

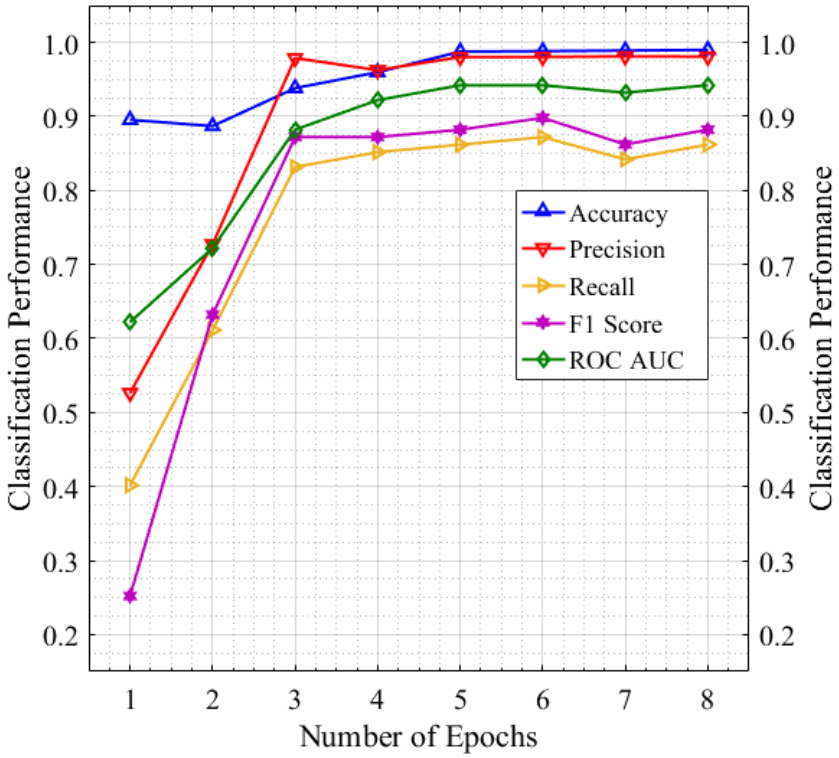


Figure 5.7: Classification performance of the DFL model using 10 client devices.

Moreover, we considered 10 client devices for analysis. Figure 5.7 shows the classification performance of the DFL model using 10 client devices. In more detail, the accuracy of the model starts at 0.9001 and increases to 0.9899 in round number 5. Then, the score remains stable up to round number 8, which indicates that the model can provide a reliable classification between new infections and existing infections. Also, the precision score is 0.5201 at the beginning and increases gradually to 0.9808 at round number 5, which remains stable up to the last round. Furthermore, the recall score of the model jumps rapidly in round number 5 and it reaches a peak of close to 0.8620 at epoch number 6. According to Figure 5.7, which remains almost stable up to round number 8. Moreover, the model shows the lowest F1 score at the beginning. In detail, the F1 score of the model starts with 0.2660 at round number 1, and it increases dramatically to 0.9288 in round number 4 and 0.8821 in round number 7. The score remained constant up to the last round. Finally, the ROC AUC of the model starts with a score of 0.6228 at the beginning. But this score rises gradually with the increase of the round number and reaches 0.9422 when the round number is 5, and then remains stable up to the end.

Furthermore, we considered 15 client devices for analysis. Figure 5.8 shows the classification performance of the DFL model using 15 client devices. In more detail, the accuracy of the model starts at 0.6830 and increases to 0.7931 in round number 6. Then, the score remains stable up to round number 8, which indicates that the model can not provide a reliable classification between new infections and existing infections.

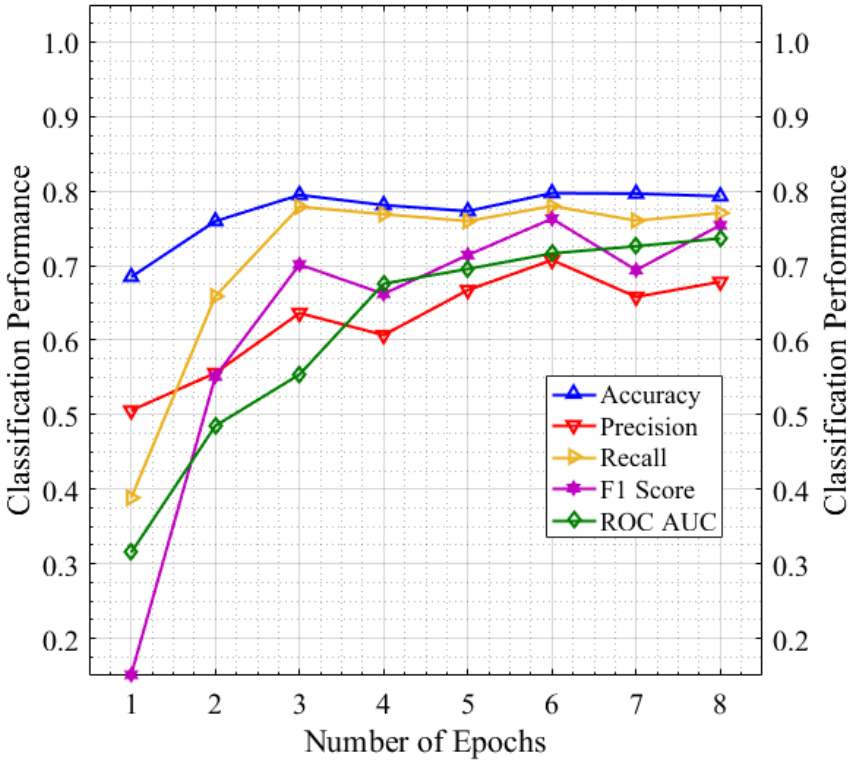


Figure 5.8: Classification performance of the DFL model using 15 client devices.

Also, the precision score is 0.5100 at the beginning and increases gradually to 0.6783 at round number 6, which is not stable up to the last round. Furthermore, the recall score of the model jumps rapidly in round number 5 and it reaches a peak of close to 0.7706 at epoch number 6. According to Figure 5.8, which remains almost stable up to round number 8. Moreover, the model shows the lowest F1 score at the beginning. In detail, the F1 score of the model starts with 0.1608 at round number 1, and it increases dramatically to 0.9288 in round number 4 and 0.7543 in round number 7. The score remained constant up to the last round. Finally, the ROC AUC of the model starts with a score of 0.3325 at the beginning. But this score rises gradually with the increase of the round number and reaches 0.7365 when the round number is 5, and then remains stable up to the end.

Lastly, we also considered 20 client devices for analysis. Figure 5.9 shows the classification performance of the DFL model using 20 client devices. Figure 5.10 and Figure 5.11 show the comparison of DFL model accuracy and precision with different numbers of clients. In more detail, the accuracy of the model starts at 0.4850 and increases to 0.6746 in round number 6. Then, the score decreases to 0.6646 to round number 8, which indicates that the model can not provide a reliable classification between new infections and existing infections. Also, the precision score is 0.1100 at the beginning and increases gradually to 0.6071 at round number 7, which is stable up to the last round. Furthermore, the recall score of the model jumps rapidly in round number 6 and it reaches a peak of close to 0.6280 at epoch number 6. According to Figure 5.9, which remains almost stable.

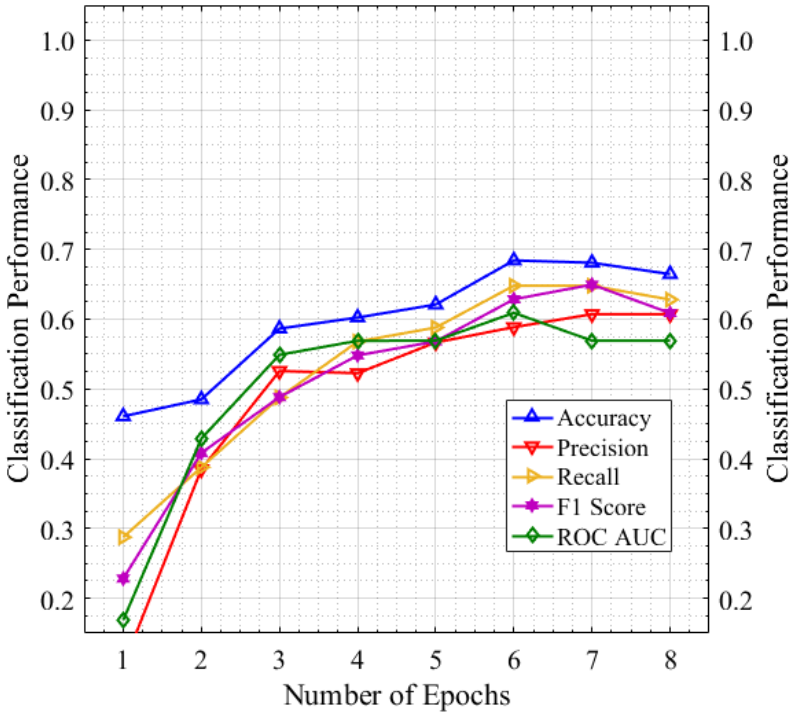


Figure 5.9: Classification performance of the DFL model using 20 client devices.

Moreover, the model shows the lowest F1 score at the beginning. In detail, the F1 score of the model starts with 0.2404 at round number 1, and it increases dramatically to 0.6244 in round number 4 and 0.6087 in round number 7. The score remained constant up to the last round. Finally, the ROC AUC of the model starts with a score of 0.1882 at the beginning. But this score rises gradually with the increase of the round number and reaches 0.5692 when the round number is 5, and then remains stable up to the end.

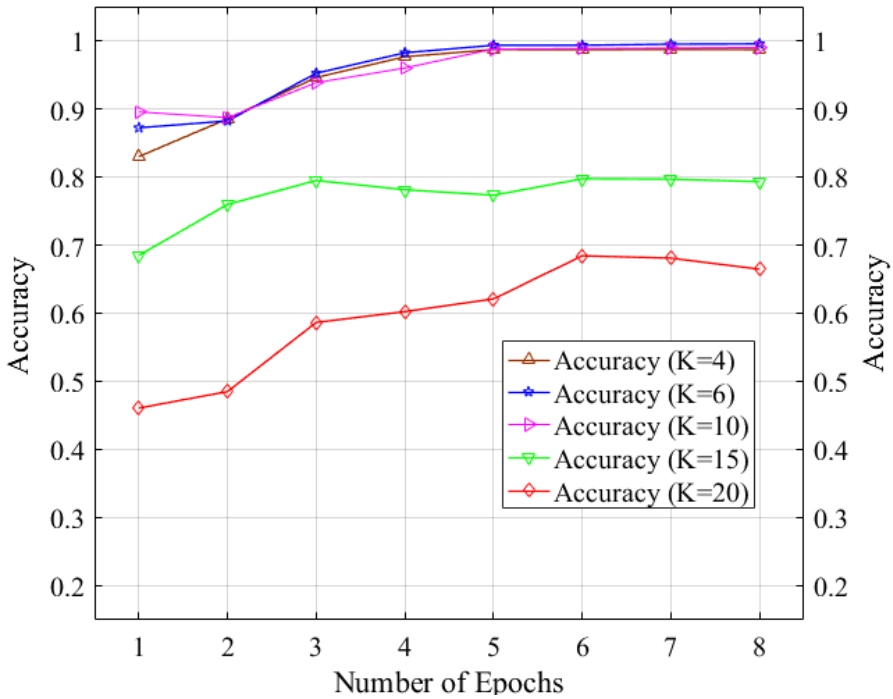


Figure 5.10: Comparison of DFL model accuracy with different numbers of clients.

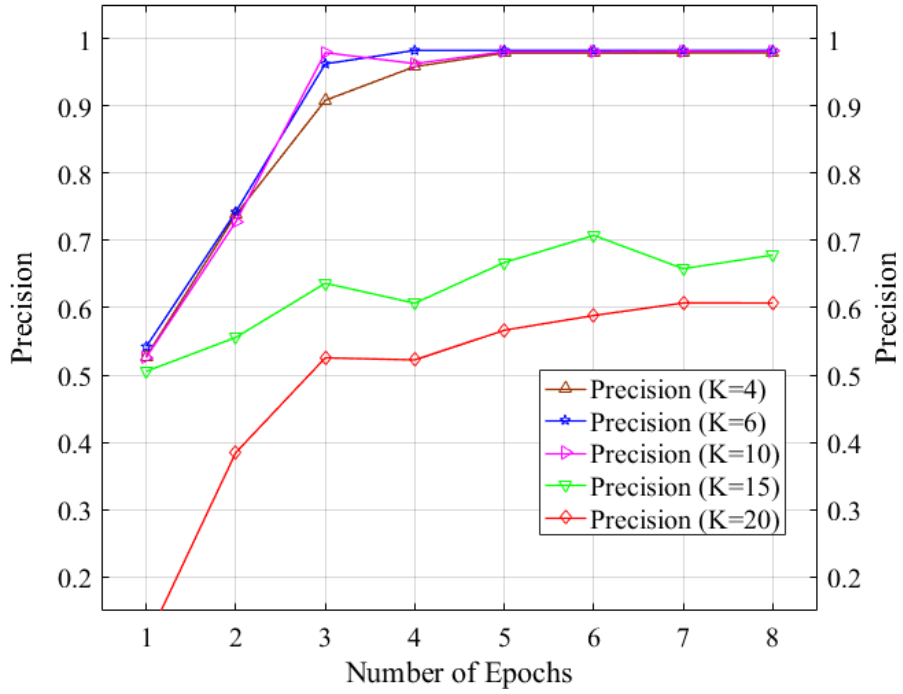


Figure 5.11: Comparison of DFL model precision with different numbers of clients.

5.4 Performance Comparison with State-of-the-Art Studies

We have considered different DTL and DFL models for their optimal performance and then applied them to the GS dataset. First, we applied different DTL models and selected the best model based on their performance. But, due to the sensitive nature of the patients data, the DTL technique of transferring data to the central machine or server may create serious privacy and security issues for the patient's data. Furthermore, we have applied the DFL models with different numbers of client devices and all the models perform significantly better in most cases by ensuring the security and privacy of patients' data. Moreover, the proposed model has adequate efficiency, scalability, low loss, and better classification accuracy than others. Figure 5.12 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). Finally, the proposed model can effectively identify and classify new infections and existing infections from genome sequences by ensuring the privacy and security of patients' data.

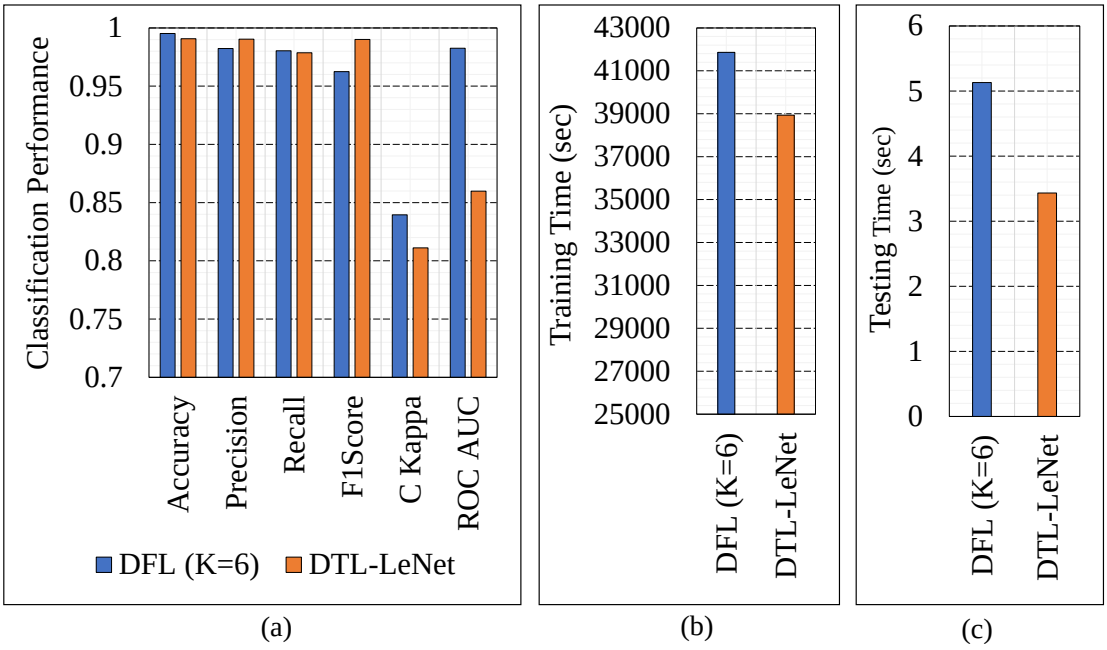


Figure 5.12: Comparison of proposed DFL (K=6) and DTL (LeNet) models in terms of (a) classification performance, (b) training time (s), and (c) testing time (s).

5.5 Computational Complexity Analysis

The computation complexity (CoX) has also been calculated for each DTL and DFL model. The total number of parameters, training time (sec), and testing time (sec) for each model are compared in Table 5.3. In terms of the DTL models, the CNN model has the minimum CoX of the others, requiring 33869.838 sec of training time and 2.718 sec of testing time for a total of 202,686 trainable parameters. In contrast, the FCN model has the maximum CoX and a total of 265,986 trainable parameters. Moreover, the IncepNet model has the second-maximum CoX, with a training time of 36657.213 seconds and a testing time of 5.721 seconds for 233,074 trainable parameters. Furthermore, the LeNet model has a training time of 38936.911 sec and a testing time of 3.490 sec, the second-longest testing time after the CNN model, and the LeNet model has more parameters than the CNN model. Finally, the ResNet model, which has a lower CoX than the FCN and IncepNet models, requires a testing time of 4.014 seconds and a training time of 44401.586 seconds for a total of 506,818 trainable parameters.

Table 5.3: Comparison of total parameters, training time, testing time of DTL and DFL models.

–	Model	Params	Training Time (Sec)	Testing Time (Sec)
DTL	CNN	202,686	33869.838	2.718
	FCN	265,986	46756.338	7.722
	IncepNet	233,074	36657.213	4.229
	LeNet	360,052	38936.911	3.490
	ResNet	506,818	44401.586	4.014
DFL	DFL (K=4, R=8)	350,985	38242.180	3.660
	DFL (K=6, R=8)	350,985	41928.245	5.105
	DFL (K=10, R=8)	350,985	42763.788	5.667
	DFL (K=15, R=8)	350,985	47921.078	6.140
	DFL (K=20, R=8)	350,985	49921.078	6.940

Next, we analyzed the CoX of the DFL models where each model has the same number of trainable parameters and the same number of communication rounds. However, the client devices vary, and we compared training and testing times with regard to the number of client devices. The DFL model with 4 client devices has the smallest CoX, with training

and testing times of 38242.180 and 3.660 seconds, respectively. The highest CoX, however, is seen in the DFL model with 20 client devices. In this analysis, we observed that the CoX rises in direct proportion to the number of client devices. Therefore, finding the best performance with a low CoX is our key priority. The proposed DFL model (K=6) requires a testing time of 5.105 sec and a training time of 41928.245 sec for a total of 350,985 trainable parameters with the best performance metrics. Considering all the performance indicators, we think the proposed DFL model (K=6) can be used for practical applications in this field with lower computational complexity.

5.6 Dependability Performance Analysis

In this section, we analyzed the dependability performance of our proposed model. Dependability performance analysis includes efficiency, availability, and scalability of the model [22, 23]. The proposed model outperforms several models with low CoX in terms of performance evaluations (e.g., accuracy, precision, recall, F1-score, ROC AUC, etc.), which ensures the efficiency of the proposed model.

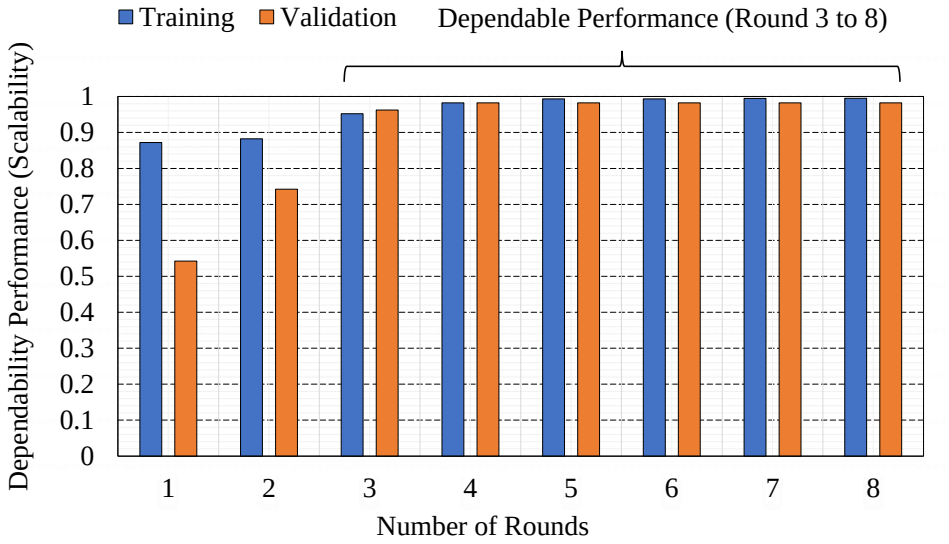


Figure 5.13: Dependability performance analysis of the proposed DFL model.

Additionally, we have taken various strategies for choosing important features before implementing the proposed model to correctly classify both new infections and existing infections without experiencing any type of failure or requiring a repair action, thus maintaining the availability of the proposed model. Figure 5.13 shows the scalability performance of the proposed model. Finally, we observed an increase in the scalability properties of the proposed model by dividing the dataset among 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.

5.7 Summary

We have explained the overall performance of the selected models in this chapter. First, we used several performance indicators to analyze the results. Then, we analyzed the performance metrics of deep transfer learning models. Furthermore, we have analyzed the performance metrics of the deep federated learning models with different numbers of clients. We also analyzed the computational complexity and dependability performance of the selected models.

In the next chapter, we will discuss the pitfalls and limitations that impacted the interpretation of the findings from our research. Finally, we will summarize the major contributions of this thesis and present a road map for future development.

Chapter 6

Conclusion

When there is an outbreak of a viral disease, it is important to know the nature of the virus so that it can be stopped, people who are sick with the virus can be treated, and vaccines can be made to stop the spread of the virus. Traditionally, alignment-based methods such as BLAST can be time-consuming, and face challenges when comparing large numbers of sequences that have significant differences in their composition. Also, about 35.2% of 173 samples did not test positive initially, resulting in false-negative findings. As a result, patients with negative results should repeat the test to avoid misdiagnosis. Current tools used to detect the virus, such as the molecular technique and RT-PCR, require support from newer and faster DL or DTL methods. Due to the sensitive nature of the patients data, the traditional DL or DTL methods of transferring data to the central machine or server may create serious privacy and security issues. Also, the methods did not consider dependability performance analysis. Thus, it is vital to develop a dependable model that is capable of identifying the new infection ensuring patient privacy and security.

6.1 Summary of the Research

In this work, we proposed a privacy-preserving DFL-based dependable identification model of new infections from genome sequences along with improved performance metrics in comparison to several other existing approaches. The proposed model has an overall

accuracy of 99.12% after IID distributes the dataset among six clients. In addition, the proposed model outperforms the existing models in terms of other performance metrics for the same configuration of the dataset. This demonstrates that the proposed model can effectively classify genome sequences, ensuring dependability, minimum computational complexity, and proper privacy and security of patients data. Moreover, the proposed model has proven its potential to protect other critical medical infrastructures where dependable identification models and secure data processing are the main challenges. More precisely, the proposed model provides a complete guideline for future researchers.

6.2 Limitations

In this section, we discussed the pitfalls and limitations of this research. The limitations of this study are those characteristics that impacted the interpretation of the findings from our research. The main pitfalls and limitations of this thesis are given below.

The first limitation involves bandwidth. As we have implemented all the models on the server (Google Colaboratory), bandwidth may cause time complexity [52, 69]. The second one is due to limited resources (RAM: 32 GB, HDD: 512 GB, Hardware Accelerator: TPU), so we can not implement more than 24 clients for our analysis, though we have a premium subscription. The third limitation is that the DFL model requires significantly more local device power and memory to train the model [58,69]. Also, if the model training is conducted while the device is in use, it reduces the devices performance [69].

Finally, we have implemented all the implementations in Python, but this language comes with its own share of pitfalls [90]. One of the most critical limitations it suffers from is in terms of execution speed [23, 90]. Being an interpreted language, it is slow when compared to compiled languages. This limitation can be restrictive in scenarios where extremely high-performance code is required. This is a major area of improvement for future implementations, and every subsequent Python version addresses it [90]. Although we have to admit it can never be as fast as a compiled language, we are convinced that it makes up for this deficiency by being super-efficient and effective in other fields.

6.3 Future Directions

In the future, we will concentrate on many DNA sequences and optimize the model hyper-parameters for improved performance. We will also try to implement this model based on decentralized devices or servers with proper privacy and security of patients' data. Deep federated learning is opening new doors every day. Its application to different domains and problems is showcasing its potential to solve problems previously unknown. Deep federated learning is an ever-evolving and very involved field to ensure the dependability of the model and the proper privacy and security of data.

Appendix A

Data Preparation

```
# -*- coding: utf-8 -*-
"""DNA Sequence Preprocessing.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1
    VBDNj6Vrz8vC7w209NlGOLSarPXmTBrE

## DNA Sequencing Pre-Processing

By: Sk. Tanzir Mehedi

1. Import necessary libraries
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from sklearn.feature_extraction.text import CountVectorizer

"""2. Load DNA sequence data"""

```

```
data = pd.read_table('sequencesLabels.txt')
data.head()

"""3. Load full dataset"""

data

"""4. Load dataset overview"""

Image("DataOverview.JPG")

"""5. Select window size and convert to lowaercase"""

def getKmers(sequences, size=3):
    return [sequences[x:x+size].lower() for x in range(len(sequences) - size + 1)]

"""6. Now convert data sequences into short overlapping k-mers of legth 3 using getKmers function."""

data['words'] = data.apply(lambda x: getKmers(x['sequences']), axis=1)
data = data.drop('sequences', axis=1)

data

"""### Now, coding sequence data is changed to lowercase, split up into all possible k-mer words of length 3 and ready for the next step.

7. Convert the lists of k-mers for each gene into string sentences of words that the count vectorizer can use. We can also make a y variable to hold the class labels.
"""

texts = list(data['words'])
for item in range(len(texts)):
    texts[item] = ' '.join(texts[item])
y_data = data.iloc[:, 0].values

y_data
```

```

print(texts[1])

"""8. Now apply the BAG of WORDS using CountVectorizer using NLP"""

cv = CountVectorizer(ngram_range=(2,2))
X = cv.fit_transform(texts)

cv

X

print(X.shape)

"""9. Check class distributation of the dataset"""

data['class'].value_counts().sort_index().plot.bar()

"""10. Dictionary representation of One-hot 2D vector representation of
    generated DNA sequence words with region size = 2"""

seqs = texts
CHARS = 'ACGT'
CHARS_COUNT = len(CHARS)

maxlen = max(map(len, seqs))
res = np.zeros((len(seqs), CHARS_COUNT * maxlen), dtype=np.uint8)

for si, seq in enumerate(seqs):
    seqlen = len(seq)
    arr = np.chararray((seqlen,), buffer=seq)
    for ii, char in enumerate(CHARS):
        res[si][ii*seqlen:(ii+1)*seqlen][arr == char] = 1

res

res.shape

"""11. Print full 2-D vector"""
```

```
96
97 np.set_printoptions(threshold=100000000)
98 print(res)
99
100 df = pd.DataFrame(res)
101 df
102
103 """12. Write the pre-processed dataset on our disk for further steps."""
104
105 file = open("sample.txt", "w+")
106 content = str(res)
107 file.write(content)
108 file.close()
109
110 """#TML Implementation """
111
112 from sklearn.naive_bayes import MultinomialNB
113 from sklearn.model_selection import train_test_split
114 from sklearn.metrics import accuracy_score, f1_score, precision_score,
    recall_score
115
116 """1. Splitting the dataset into the training set and test set"""
117
118 X_train, X_test, y_train, y_test = train_test_split(X, y_data, test_size =
    0.20, random_state=42)
119
120 print(X_train.shape)
121 print(X_test.shape)
122
123 """2. A multinomial naive Bayes classifier will be created.  I previously
    did some parameter tuning and found the ngram size of 2 (reflected in
    the Countvectorizer() instance) and a model alpha of 0.1 did the best (
    the alpha parameter was determined by grid search previously)"""
124
125 classifier = MultinomialNB(alpha=0.1)
126 classifier.fit(X_train, y_train)
127
128 """3. Define the classifier"""
129
```

```
y_pred = classifier.predict(X_test)

"""4. Check the prformance metrices """

print("Confusion matrix\n")
print(pd.crosstab(pd.Series(y_test, name='Actual'), pd.Series(y_pred, name=
    'Predicted')))
def get_metrics(y_test, y_predicted):
    accuracy = accuracy_score(y_test, y_predicted)
    precision = precision_score(y_test, y_predicted, average='weighted')
    recall = recall_score(y_test, y_predicted, average='weighted')
    f1 = f1_score(y_test, y_predicted, average='weighted')
    return accuracy, precision, recall, f1
accuracy, precision, recall, f1 = get_metrics(y_test, y_pred)
print("Accuracy = %.3f \nPrecision = %.3f \nRecall = %.3f \nF1-Score = %.3f
    " % (accuracy, precision, recall, f1))
```

Listing A.1: DNA sequence data preprocessing

Appendix B

DTL Models Implementation

```
# -*- coding: utf-8 -*-
"""DTL-LeNet.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1mTT_2-NeuXFSPJnvcvcJVu-i4omh2jQg

## Deep Transfer Learning with LeNet on DNA Sequence Dataset

By: Sk. Tanzir Mehedi

Importing libraries
"""

# Commented out IPython magic to ensure Python compatibility.
import time
import sklearn
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow.keras as keras
import matplotlib.pyplot as plt
```

```
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.callbacks import TensorBoard
import warnings
# %matplotlib inline
warnings.filterwarnings('ignore')

"""Importing the Dataset"""

dataset=pd.read_csv('preprocessedDNASequenceDatase.csv')

"""Exploratory Data Analysis"""

dataset.head()

properties = list(dataset.columns.values)
properties.remove('label')
X = dataset[properties]
y = dataset['label']

"""Split Dataset into Training Set and Test Set"""

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=0)

"""Check the nb classes"""

nb_classes = len(np.unique(np.concatenate((y_train, y_test), axis=0)))
nb_classes

"""Transform the labels from integers to one hot vectors"""

enc = sklearn.preprocessing.OneHotEncoder(categories='auto')
enc.fit(np.concatenate((y_train, y_test), axis=0).reshape(-1, 1))

y_train = enc.transform(y_train.values.reshape(-1, 1)).toarray()
y_test = enc.transform(y_test.values.reshape(-1, 1)).toarray()

```

```
"""Save orignal y because later we will use binary"""

y_true = np.argmax(y_test, axis=1)

"""If univariate then add a dimension to make it multivariate with one
    dimension"""

if len(X_train.shape) == 2:
        X_train = X_train.values.reshape((X_train.shape[0], X_train.shape
    [1], 1))
        X_test = X_test.values.reshape((X_test.shape[0], X_test.shape[1],
    1))
        input_shape = X_train.shape[1:]

"""Making the Model"""

input_layer = keras.layers.Input(input_shape)

conv_1 = keras.layers.Conv1D(filters=5,kernel_size=5,activation='relu',
    padding='same')(input_layer)
conv_1 = keras.layers.MaxPool1D(pool_size=2)(conv_1)

conv_2 = keras.layers.Conv1D(filters=20, kernel_size=5, activation='relu',
    padding='same')(conv_1)
conv_2 = keras.layers.MaxPool1D(pool_size=4)(conv_2)

# here did not mention the number of hidden units in the fully-connected
    layer
# so I took the lenet

flatten_layer = keras.layers.Flatten()(conv_2)
fully_connected_layer = keras.layers.Dense(500,activation='relu')(
    flatten_layer)

output_layer = keras.layers.Dense(nb_classes,activation='softmax')(
    fully_connected_layer)

model = keras.models.Model(inputs=input_layer,outputs=output_layer)

```

```
"""Compile the Model"""

model.compile(optimizer=keras.optimizers.Adam(lr=0.01,decay=0.005), loss='
    categorical_crossentropy', metrics=['accuracy'])
model.summary()

"""Result View with TensorBoard"""

NAME = "LeNet on DNA Sequence Dataset"
tensorboard = TensorBoard(log_dir="logs/{}".format(NAME), histogram_freq =
    1, profile_batch = 5)

"""Fitting the model"""

# x_val and y_val are only used to monitor the test loss and NOT for
    training
batch_size = 64
nb_epochs = 1000

mini_batch_size = int(min(X_train.shape[0] / 10, batch_size))

start_time = time.time()

history=model.fit(X_train, y_train, batch_size=mini_batch_size, epochs=
    nb_epochs, validation_data=(X_test, y_test),callbacks=[tensorboard])

duration = time.time() - start_time

"""Making Predictions"""

start_time = time.time()
y_pred = model.predict(X_test)
duration1 = time.time() - start_time

"""Convert the predicted from binary to integer"""

y_pred = np.argmax(y_pred, axis=1)

"""Evaluating the Algorithm"""
```

```
128
129 print(confusion_matrix(y_true,y_pred))
130 print(classification_report(y_true,y_pred))
131
132 # Model Accuracy: how often is the classifier correct?
133 print("Accuracy:",metrics.accuracy_score(y_true, y_pred))
134
135 # Model Precision: what percentage of positive tuples are labeled as such?
136 print("Precision:",metrics.precision_score(y_true, y_pred, average='
    weighted',labels=np.unique(y_pred)))
137
138 # Model Recall: what percentage of positive tuples are labelled as such?
139 print("Recall:",metrics.recall_score(y_true, y_pred,average='weighted',
    labels=np.unique(y_pred)))
140
141 #Calculate F1 Score
142 print("F1 Score:",metrics.f1_score(y_true, y_pred, average='weighted',
    labels=np.unique(y_pred)))
143
144 #Calculate Mean Absolute Error
145 print("Mean Absolute Error:",metrics.mean_absolute_error(y_true, y_pred))
146
147 # kappa
148 print("Cohens kappa:", metrics.cohen_kappa_score(y_true, y_pred))
149
150 # ROC AUC
151 print("ROC AUC:", metrics.roc_auc_score(y_true, y_pred))
152
153 #Train time
154 print('Train Time(s): ',duration)
155
156 #Test time
157 print('Test Time(s): ',duration1)
158
159 # list all data in history
160 print(history.history.keys())
161
162 # summarize history for loss
163 plt.plot(history.history['loss'])
```

```
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

keras.backend.clear_session()
```

Listing B.1: DTL with LeNet on DNA Sequence Dataset

Appendix C

DFL Models Implementation

```
# -*- coding: utf-8 -*-
"""DFL-LeNet.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1
    nmZ5HWhKam_t9QMCk1xY1kOXfzrrnxeO

## DFL with LeNet on DNA Sequence Dataset

By: Sk. Tanzir Mehedi
"""

import urllib.request

def download_url(url, save_as):
    response = urllib.request.urlopen(url)
    data = response.read()
    file = open(save_as, 'wb')
    file.write(data)
    file.close()
    response.close()

```

```
def read_binary_file(file):
    f = open(file,'rb')
    block = f.read()
    return block.decode('utf-16')

def split_text_in_lines(text):
    return text.split('\r\n')

def split_by_tabs(line):
    return line.split('\t')

names_link = 'https://raw.githubusercontent.com/tanzirmehedi/Deep-Federated-Learning/main/preprocessedDNASequenceDataset.names'
data_link = 'https://raw.githubusercontent.com/tanzirmehedi/Deep-Federated-Learning/main/preprocessedDNASequenceDataset.data'

diagnosis_names = 'preprocessedDNASequenceDataset.names'
diagnosis_data = 'preprocessedDNASequenceDataset.data'
download_url(names_link, diagnosis_names)
download_url(data_link, diagnosis_data)

import numpy as np

def parse_double(field):
    field = field.replace(',', '.')
    return float(field)

def parse_boolean(field):
    return 1. if field == 'yes' else 0.

def read_np_array(file = diagnosis_data):
    text = read_binary_file(file)
    lines = split_text_in_lines(text)
    rows = []
    for line in lines:
        if line == '': continue
        line = line.replace('\r\n', '')
        fields = split_by_tabs(line)
        row = []
```

```
        j = 0
        for field in fields:
            value = parse_double(field) if j == 0 else parse_boolean(field)
            row.append(value)
            j += 1
        rows.append(row)
    matrix = np.array(rows, dtype = np.float32)
    return matrix

matrix = read_np_array()
matrix

n_samples, n_dimensions = matrix.shape
print(n_samples)
print(n_dimensions)

def get_random_indexes(n):
    indexes = list(range(n))
    random_indexes = []
    for i in range(n):
        r = np.random.randint(len(indexes))
        random_indexes.append(indexes.pop(r))
    return random_indexes

def get_indexes_for_2_datasets(n, training = 80):
    indexes = get_random_indexes(n)
    train = int(training / 100. * n)
    return indexes[:train], indexes[train:]

matrix = read_np_array()
n_samples, n_dimensions = matrix.shape

train_indexes, test_indexes = get_indexes_for_2_datasets(n_samples)
train_data = matrix[train_indexes]
test_data = matrix[test_indexes]

def print_dataset(name, data):
    print('Dataset {}. Shape: {}'.format(name, data.shape))
    print(data)
```

```
100
101 print(len(train_data))
102 print(len(test_data))
103 print(train_indexes)
104 print(test_indexes)
105
106 print_dataset('Train', train_data)
107
108 print_dataset('Test', test_data)
109
110 import torch
111 from torch.autograd import Variable
112 import torch.nn as nn
113 import torch.nn.functional as F
114
115 input_size = 64
116 learning_rate = 0.01
117 num_iterations = 1000
118
119 class LogisticRegression(torch.nn.Module):
120
121     def __init__(self):
122         super(LogisticRegression, self).__init__()
123         self.linear = torch.nn.Linear(input_size, 1)
124
125     def forward(self, x):
126         return torch.sigmoid(self.linear(x))
127
128 # Commented out IPython magic to ensure Python compatibility.
129 def decide(y):
130     return 1. if y >= 0.5 else 0.
131
132 decide_vectorized = np.vectorize(decide)
133
134 to_percent = lambda x: '{:.2f}%'.format(x)
135
136 def compute_accuracy(model, input, output):
137     prediction = model(input).data.numpy()[:, 0]
138     n_samples = prediction.shape[0] + 0.
```

```
    prediction = decide_vectorized(prediction)
    equal = prediction == output.data.numpy()
    return 100. * equal.sum() / n_samples

def get_input_and_output(data):
    input = Variable(torch.tensor(data[:, :6], dtype = torch.float32))
    output = Variable(torch.tensor(data[:, 6], dtype = torch.float32))
    #output = Variable(torch.tensor(data[:, 6], dtype = torch.float32)[..., None])

    return input, output

input, output= get_input_and_output(train_data)
test_input, test_output = get_input_and_output(test_data)

import matplotlib.pyplot as plt
# %matplotlib inline

DFL_title = 'DFL with LeNet on DNA Sequence Dataset'


def plot_graphs(DFL_title, losses, accuracies):
    plt.plot(losses)
    plt.title(f"{DFL_title} - Training Loss")
    plt.xlabel("Iterations")
    plt.ylabel("Training Loss")
    plt.show()
    plt.plot(accuracies)
    plt.title(f"{DFL_title} - Training Accuracy")
    plt.xlabel("Iterations")
    plt.ylabel("Training Accuracy (Percent %)")
    plt.show()

def train_model(DFL_title, input, output, test_input, test_output):
    model = LogisticRegression()
    criterion = torch.nn.BCELoss(size_average=True)
    optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
    losses = []
    accuracies = []
```

```
    n_samples, _ = input.shape
    for iteration in range(num_iterations):
            optimizer.zero_grad()
            prediction = model(input)
            loss = criterion(prediction, output)
            loss.backward()
            optimizer.step()
            if iteration % 500 == 0:
                train_acc = compute_accuracy(model, input, output)
                train_loss = loss.item()
                losses.append(train_loss)
                accuracies.append(train_acc)
                print('iteration={}, loss={:.4f}, train_acc={}'.format(
    iteration, train_loss, to_percent(train_acc)))
    plot_graphs(DFL_title, losses, accuracies)
    test_acc = compute_accuracy(model, test_input, test_output)
    print('\nTesting Accuracy = {}'.format(to_percent(test_acc)))
    return model

model = train_model(DFL_title, input, output, test_input, test_output)

#pip install syft==0.2.9

import syft as sy
import torch as th
hook = sy.TorchHook(th)
from torch import nn, optim

local_client_devices = 6
client_devicess = []
for i in range(local_client_devices):
    local_client_devices_name = 'client_devices{}'.format(i)
    client_devices = sy.VirtualWorker(hook, id = local_client_devices_name)
    print(client_devices)
    print(str(client_devices._objects))
    client_devicess.append(client_devices)
secure_worker = sy.VirtualWorker(hook, id="secure_worker")
print(secure_worker)
print(secure_worker._objects)
```

```
215
216 def get_workers_names(workers):
217     return [worker.id for worker in workers]
218
219 def add_and_print_workers(worker, workers):
220     print('workers of {} = {}'.format(worker.id, get_workers_names(workers)
    ))
221     worker.add_workers(workers)
222
223 for i in range(local_client_devices):
224     workers = [client_devicess[i2] for i2 in range(local_client_devices) if
     i2 != i] + [secure_worker]
225     add_and_print_workers(client_devicess[i], workers)
226 add_and_print_workers(secure_worker, client_devicess)
227
228 n_samples = train_data.shape[0]
229 print(n_samples)
230 print(local_client_devices)
231 samples_per_client_devices = int((n_samples + 0.5) / local_client_devices)
232 print(samples_per_client_devices)
233
234 client_devices_features = []
235 client_devices_targets = []
236
237 train_data = th.tensor(train_data, dtype = torch.float32, requires_grad=
    True)
238 train_data
239
240 for i in range(local_client_devices):
241     train_data2 = train_data[i * samples_per_client_devices:(i + 1) *
    samples_per_client_devices].clone().detach().requires_grad_(True)
242     features = train_data2[:, :6].clone().detach().requires_grad_(True)
243     print(features)
244     targets = train_data2[:, 6][:, None].clone().detach()
245     print(targets)
246     client_devices_features.append(features.send(client_devicess[i]))
247     print(client_devices_features)
248     client_devices_targets.append(targets.send(client_devicess[i]))
249     print(client_devices_features)
```

```
250
251 print(model)
252
253 def plot_federated_graphs(DFL_title, losses, accuracies):
254     for i in range(local_client_devices):
255         plt.plot(losses[i], label=f'client_devices {i}')
256     legend = plt.legend(loc='upper right', shadow=True)
257     plt.title(f"{DFL_title} - Training Loss")
258     plt.xlabel("Iterations")
259     plt.ylabel("Training Loss")
260     plt.show()
261     for i in range(local_client_devices):
262         plt.plot(accuracies[i], label=f'client_devices {i}')
263     legend = plt.legend(loc='lower right', shadow=True)
264     plt.title(f"{DFL_title} - Training Accuracy")
265     plt.xlabel("Iterations")
266     plt.ylabel("Training Accuracy (Percent %)")
267     plt.show()
268
269 def compute_federated_accuracy(model, input, output):
270     prediction = model(input)
271     n_samples = prediction.shape[0]
272     s = 0.
273     for i in range(n_samples):
274         p = 1. if prediction[i] >= 0.5 else 0.
275         e = 1. if p == output[i] else 0.
276         s += e
277     return 100. * s / n_samples
278
279 iterations = 1000 #2000
280 worker_iterations = 8
281
282 def federated_learning(DFL_title, client_devices_features,
    client_devices_targets, test_input, test_output):
283     model = LogisticRegression()
284     criterion = torch.nn.BCELoss(size_average=True)
285     optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
286     losses = [[] for i in range(local_client_devices)]
287     accuracies = [[] for i in range(local_client_devices)]
```

```
    for iteration in range(iterations):
        models = [model.copy().send(client_devicess[i]) for i in range(
local_client_devices)]
        optimizers = [torch.optim.SGD(params = models[i].parameters(), lr =
 learning_rate) for i in range(local_client_devices)]
        for worker_iteration in range(worker_iterations):
            last_losses = []
            for i in range(local_client_devices):
                optimizers[i].zero_grad()
                prediction = models[i](client_devices_features[i])
                loss = criterion(prediction, client_devices_targets[i])
                loss.backward()
                optimizers[i].step()
                loss = loss.get().data.item()
                last_losses.append(loss)
        for i in range(local_client_devices):
            losses[i].append(last_losses[i])
            train_acc = compute_federated_accuracy(models[i],
client_devices_features[i], client_devices_targets[i])
            accuracies[i].append(train_acc)
            models[i].move(secure_worker)
        with th.no_grad():
            avg_weight = sum([models[i].linear.weight.data for i in range(
local_client_devices)]) / local_client_devices
            model.linear.weight.set_(avg_weight.get())
            avg_bias = sum([models[i].linear.bias.data for i in range(
local_client_devices)]) / local_client_devices
            model.linear.bias.set_(avg_bias.get())
        if iteration % 100 == 0:
            losses_str = ['{:.4f}'.format(losses[i][-1]) for i in range(
local_client_devices)]
            accuracies_str = [to_percent(accuracies[i][-1]) for i in range(
local_client_devices)]
            print('Iteration={}, losses={}, accuracies={}'.format(iteration
, losses_str, accuracies_str))
    plot_federated_graphs(DFL_title, losses, accuracies)
    test_acc = compute_accuracy(model, test_input, test_output)
    print('\nTesting Accuracy = {}'.format(to_percent(test_acc)))
    return model
```

```

model = federated_learning(DFL_title, client_devices_features,
    client_devices_targets, test_input, test_output)
```

Listing C.1: DFL models implementation on DNA sequence dataset

References

[1] F. Wu, S. Zhao, B. Yu, Y.-M. Chen, W. Wang, Z.-G. Song, Y. Hu, Z.-W. Tao, J.-H. Tian, Y.-Y. Pei, M.-L. Yuan, Y.-L. Zhang, F.-H. Dai, Y. Liu, Q.-M. Wang, J.-J. Zheng, L. Xu, E. C. Holmes, and Y.-Z. Zhang, "A new coronavirus associated with human respiratory disease in China," *Nature*, vol. 579, no. 7798, pp. 265–269, Mar. 2020. [Online]. Available: http://www.nature.com/articles/s41586-020-2008-3

[2] R. Lu, X. Zhao, J. Li, P. Niu, B. Yang, H. Wu, W. Wang, H. Song, B. Huang, N. Zhu, Y. Bi, X. Ma, F. Zhan, L. Wang, T. Hu, H. Zhou, Z. Hu, W. Zhou, L. Zhao, J. Chen, Y. Meng, J. Wang, Y. Lin, J. Yuan, Z. Xie, J. Ma, W. J. Liu, D. Wang, W. Xu, E. C. Holmes, G. F. Gao, G. Wu, W. Chen, W. Shi, and W. Tan, "Genomic characterisation and epidemiology of 2019 novel coronavirus: implications for virus origins and receptor binding," *The Lancet*, vol. 395, no. 10224, pp. 565–574, Feb. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0140673620302518

[3] D. A. Marston, L. M. McElhinney, R. J. Ellis, D. L. Horton, E. L. Wise, S. L. Leech, D. David, X. de Lamballerie, and A. R. Fooks, "Next generation sequencing of viral RNA genomes," *BMC Genomics*, vol. 14, no. 1, p. 444, Dec. 2013. [Online]. Available: https://bmcgenomics.biomedcentral.com/articles/10.1186/1471-2164-14-444

[4] J. S. Mackenzie and D. W. Smith, "COVID-19: a novel zoonotic disease caused by a coronavirus from China: what we know and what we don't," *Microbiology Australia*, vol. 41, no. 1, p. 45, 2020. [Online]. Available: http://microbiology.publish.csiro.au/?paper=MA20013

[5] C. Sohrabi, Z. Alsafi, N. O'Neill, M. Khan, A. Kerwan, A. Al-Jabir, C. Iosifidis, and R. Agha, "World Health Organization declares global emergency: A review of the 2019 novel coronavirus (COVID-19)," *International Journal of Surgery*, vol. 76, pp. 71–76, Apr. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1743919120301977

[6] D. Cucinotta and M. Vanelli, "WHO Declares COVID-19 a Pandemic," *Acta Bio Medica Atenei Parmensis*, vol. 91, no. 1, pp. 157–160, Mar. 2020. [Online]. Available: https://doi.org/10.23750/abm.v91i1.9397

[7] A. Whata and C. Chimedza, "Deep Learning for SARS COV-2 Genome Sequences," *IEEE Access*, vol. 9, pp. 59 597–59 611, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9405995/

[8] A. Lopez-Rincon, A. Tonda, L. Mendoza-Maldonado, D. G. J. C. Mulders, R. Molenkamp, C. A. Perez-Romero, E. Claassen, J. Garssen, and A. D. Kraneveld, "Classification and specific primer design for accurate detection of SARS-CoV-2 using deep learning," *Scientific Reports*, vol. 11, no. 1, p. 947, Dec. 2021. [Online]. Available: http://www.nature.com/articles/s41598-020-80363-5

[9] H. C. Metsky, C. A. Freije, T.-S. F. Kosoko-Thoroddsen, P. C. Sabeti, and C. Myhrvold, "CRISPR-based surveillance for COVID-19 using genomically-comprehensive machine learning design," Genomics, preprint, Mar. 2020. [Online]. Available: http://biorxiv.org/lookup/doi/10.1101/2020.02.26.967026

[10] G. Adams, "A beginners guide to RT-PCR, qPCR and RT-qPCR," *The Biochemist*, vol. 42, no. 3, pp. 48–53, Jun. 2020. [Online]. Available: https://portlandpress.com/biochemist/article/42/3/48/225280/A-beginner-s-guide-to-RT-PCR-qPCR-and-RT-qPCR

[11] S. A. Bustin and T. Nolan, "RT-qPCR Testing of SARS-CoV-2: A Primer," *International Journal of Molecular Sciences*, vol. 21, no. 8, p. 3004, Apr. 2020. [Online]. Available: https://www.mdpi.com/1422-0067/21/8/3004

[12] V. M. Corman, O. Landt, M. Kaiser, R. Molenkamp, A. Meijer, D. K. Chu, T. Bleicker, S. Brünink, J. Schneider, M. L. Schmidt, D. G. Mulders, B. L.

Haagmans, B. van der Veer, S. van den Brink, L. Wijsman, G. Goderski, J.-L. Romette, J. Ellis, M. Zambon, M. Peiris, H. Goossens, C. Reusken, M. P. Koopmans, and C. Drosten, "Detection of 2019 novel coronavirus (2019-nCoV) by real-time RT-PCR," *Eurosurveillance*, vol. 25, no. 3, Jan. 2020. [Online]. Available: https://www.eurosurveillance.org/content/10.2807/1560-7917.ES.2020.25.3.2000045

[13] C. Long, H. Xu, Q. Shen, X. Zhang, B. Fan, C. Wang, B. Zeng, Z. Li, X. Li, and H. Li, "Diagnosis of the Coronavirus disease (COVID-19): rRT-PCR or CT?" *European Journal of Radiology*, vol. 126, p. 108961, May 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0720048X20301509

[14] K. Padhan, C. Tanwar, A. Hussain, P. Y. Hui, M. Y. Lee, C. Y. Cheung, J. S. M. Peiris, and S. Jameel, "Severe acute respiratory syndrome coronavirus Orf3a protein interacts with caveolin," *Journal of General Virology*, vol. 88, no. 11, pp. 3067–3077, Nov. 2007. [Online]. Available: https://www.microbiologyresearch.org/content/journal/jgv/10.1099/vir.0.82856-0

[15] T. Ai, Z. Yang, H. Hou, C. Zhan, C. Chen, W. Lv, Q. Tao, Z. Sun, and L. Xia, "Correlation of Chest CT and RT-PCR Testing in Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases," *Radiology*, p. 23, 2020.

[16] Y. Yang, M. Yang, J. Yuan, F. Wang, Z. Wang, J. Li, M. Zhang, L. Xing, J. Wei, L. Peng, G. Wong, H. Zheng, W. Wu, C. Shen, M. Liao, K. Feng, J. Li, Q. Yang, J. Zhao, L. Liu, and Y. Liu, "Laboratory Diagnosis and Monitoring the Viral Shedding of SARS-CoV-2 Infection," *The Innovation*, vol. 1, no. 3, p. 100061, Nov. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666675820300643

[17] S. Woloshin, N. Patel, and A. S. Kesselheim, "False Negative Tests for SARS-CoV-2 Infection — Challenges and Implications," *New England Journal of Medicine*, vol. 383, no. 6, p. e38, Aug. 2020. [Online]. Available: http://www.nejm.org/doi/10.1056/NEJMp2015897

[18] J. Zhao, Q. Yuan, H. Wang, W. Liu, X. Liao, Y. Su, X. Wang, J. Yuan, T. Li, J. Li, S. Qian, C. Hong, F. Wang, Y. Liu, Z. Wang, Q. He, Z. Li, B. He, T. Zhang, Y. Fu, S. Ge, L. Liu, J. Zhang, N. Xia, and Z. Zhang, "Antibody

Responses to SARS-CoV-2 in Patients With Novel Coronavirus Disease 2019," *Clinical Infectious Diseases*, vol. 71, no. 16, pp. 2027–2034, Nov. 2020. [Online]. Available: https://academic.oup.com/cid/article/71/16/2027/5812996

[19] I. Arevalo-Rodriguez, D. Buitrago-Garcia, D. Simancas-Racines, P. Zambrano-Achig, R. Del Campo, A. Ciapponi, O. Sued, L. Martinez-García, A. W. Rutjes, N. Low, P. M. Bossuyt, J. A. Perez-Molina, and J. Zamora, "False-negative results of initial RT-PCR assays for COVID-19: A systematic review," *PLOS ONE*, vol. 15, no. 12, p. e0242958, Dec. 2020. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0242958

[20] B. Yuan, S. Ge, and W. Xing, "A Federated Learning Framework for Healthcare IoT devices," May 2020, number: arXiv:2005.05083 arXiv:2005.05083 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2005.05083

[21] H. Chen, H. Li, G. Xu, Y. Zhang, and X. Luo, "Achieving Privacy-preserving Federated Learning with Irrelevant Updates over E-Health Applications," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. Dublin, Ireland: IEEE, Jun. 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9149385/

[22] M. Z. A. Bhuiyan, S. yen Kuo, D. Lyons, and Z. Shao, "Dependability in cyber-physical systems and applications," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 1, 2018.

[23] S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed, and I. Rafiqul, "Dependable Intrusion Detection System for IoT: A Deep Transfer Learning-based Approach," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9749858/

[24] Q. Dou, T. Y. So, M. Jiang, Q. Liu, V. Vardhanabhuti, G. Kaissis, Z. Li, W. Si, H. H. C. Lee, K. Yu, Z. Feng, L. Dong, E. Burian, F. Jungmann, R. Braren, M. Makowski, B. Kainz, D. Rueckert, B. Glocker, S. C. H. Yu, and P. A. Heng, "Federated deep learning for detecting COVID-19 lung abnormalities in CT: a privacy-preserving multinational validation study,"

npj Digital Medicine, vol. 4, no. 1, p. 60, Dec. 2021. [Online]. Available: https://www.nature.com/articles/s41746-021-00431-6

[25] A. Qayyum, K. Ahmad, M. A. Ahsan, A. Al-Fuqaha, and J. Qadir, "Collaborative Federated Learning For Healthcare: Multi-Modal COVID-19 Diagnosis at the Edge," Jan. 2021, number: arXiv:2101.07511 arXiv:2101.07511 [cs]. [Online]. Available: http://arxiv.org/abs/2101.07511

[26] N. Neranjan Thilakarathne, G. Muneeswari, V. Parthasarathy, F. Alassery, H. Hamam, R. Kumar Mahendran, and M. Shafiq, "Federated Learning for Privacy-Preserved Medical Internet of Things," *Intelligent Automation & Soft Computing*, vol. 33, no. 1, pp. 157–172, 2022. [Online]. Available: https://www.techscience.com/iasc/v33n1/46163

[27] S. F. AltschuP, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic Local Alignment Search Tool," *Journal of Molecular Biology*, p. 8, 1990.

[28] R. Pearson, "[S] Rapid and Sensitive Sequence Comparison with FASTP and FASTA," *Methods Enzymol*, p. 36, 1990.

[29] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, Jul. 2009. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btp324

[30] P. C. Y. Woo, Y. Huang, S. K. P. Lau, and K.-Y. Yuen, "Coronavirus Genomics and Bioinformatics Analysis," *Viruses*, vol. 2, no. 8, pp. 1804–1820, Aug. 2010. [Online]. Available: http://www.mdpi.com/1999-4915/2/8/1804

[31] S. Vinga and J. Almeida, "Alignment-free sequence comparison–a review," *Bioinformatics*, vol. 19, no. 4, pp. 513–523, Mar. 2003. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btg005

[32] L. Pinello, G. Lo Bosco, and G.-C. Yuan, "Applications of alignment-free methods in epigenomics," *Briefings in Bioinformatics*, vol. 15, no. 3, pp. 419–430, May 2014. [Online]. Available: https://academic.oup.com/bib/article-lookup/doi/10.1093/bib/bbt078

[33] R. Datta, N. K. Podder, H. K. Rana, M. K. B. Islam, and M. A. Moni, "Bioinformatics approach to analyze gene expression profile and comorbidities of gastric cancer," in *2020 23rd International Conference on Computer and Information Technology (ICCIT)*. DHAKA, Bangladesh: IEEE, Dec. 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9392587/

[34] A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowski, "Alignment-free sequence comparison: benefits, applications, and tools," *Genome Biology*, vol. 18, no. 1, p. 186, Dec. 2017. [Online]. Available: http://genomebiology.biomedcentral.com/articles/10.1186/s13059-017-1319-7

[35] G. S. Randhawa, K. A. Hill, and L. Kari, "MLDSP-GUI: an alignment-free standalone tool with an interactive graphical user interface for DNA sequence comparison and analysis," *Bioinformatics*, vol. 36, no. 7, pp. 2258–2259, Apr. 2020. [Online]. Available: https://academic.oup.com/bioinformatics/article/36/7/2258/5675497

[36] G. S. Randhawa, M. P. M. Soltysiak, H. El Roz, C. P. E. de Souza, K. A. Hill, and L. Kari, "Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens: COVID-19 case study," *PLOS ONE*, vol. 15, no. 4, p. e0232391, Apr. 2020. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0232391

[37] H. Zeng, M. D. Edwards, G. Liu, and D. K. Gifford, "Convolutional neural network architectures for predicting DNAprotein binding," *Bioinformatics*, vol. 32, no. 12, pp. i121–i127, Jun. 2016. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btw255

[38] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani, and A. Telenti, "A primer on deep learning in genomics," *Nature Genetics*, vol. 51, no. 1, pp. 12–18, Jan. 2019. [Online]. Available: http://www.nature.com/articles/s41588-018-0295-5

[39] S. Seo, M. Oh, Y. Park, and S. Kim, "DeepFam: deep learning based alignment-free method for protein family modeling and prediction," *Bioinformatics*, vol. 34, no. 13, pp. i254–i262, Jul. 2018. [Online]. Available: https://academic.oup.com/bioinformatics/article/34/13/i254/5045722

[40] N. G. Nguyen, V. A. Tran, D. L. Ngo, D. Phan, F. R. Lumbanraja, M. R. Faisal, B. Abapihi, M. Kubo, and K. Satou, "DNA Sequence Classification by Convolutional Neural Network," *Journal of Biomedical Science and Engineering*, vol. 09, no. 05, pp. 280–286, 2016. [Online]. Available: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jbise.2016.95021

[41] H. Zhang, C.-L. Hung, M. Liu, X. Hu, and Y.-Y. Lin, "NCNet: Deep Learning Network Models for Predicting Function of Non-coding DNA," *Frontiers in Genetics*, vol. 10, p. 432, May 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fgene.2019.00432/full

[42] Y. Zhang, S. Qiao, S. Ji, and Y. Li, "DeepSite: bidirectional LSTM and CNN models for predicting DNAprotein binding," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 4, pp. 841–851, Apr. 2020. [Online]. Available: http://link.springer.com/10.1007/s13042-019-00990-x

[43] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learningbased sequence model," *Nature Methods*, vol. 12, no. 10, pp. 931–934, Oct. 2015. [Online]. Available: http://www.nature.com/articles/nmeth.3547

[44] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6296526/

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[46] S. T. Mehedi, A. Anwar, Z. Rahman, and K. Ahmed, "Deep Transfer Learning Based Intrusion Detection System for Electric Vehicular Networks," *Sensors*, vol. 21, no. 14, p. 4736, Jul. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/14/4736

[47] P. K. Roy, J. P. Singh, and S. Banerjee, "Deep learning to filter SMS Spam," *Future Generation Computer Systems*, vol. 102, pp. 524–533, Jan. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X19306879

[48] F. Ordóñez and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition," *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/1/115

[49] R. Johnson and T. Zhang, "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, 2015, pp. 103–112. [Online]. Available: http://aclweb.org/anthology/N15-1011

[50] D. Ravi, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep Learning for Health Informatics," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 4–21, Jan. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7801947/

[51] W. Schneble and G. Thamilarasu, "Attack Detection Using Federated Learning in Medical Cyber-Physical Systems," *William Schneble*, p. 9, 2019.

[52] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8843900/

[53] S. T. Mehedi, K. Ahmed, F. M. Bui, M. Rahaman, I. Hossain, T. M. Tonmoy, R. A. Limon, S. M. Ibrahim, and M. A. Moni, "MLBioIGE: integration and interplay of machine learning and bioinformatics approach to identify the genetic effect of SARS-COV-2 on idiopathic pulmonary fibrosis patients," *Biology Methods and Protocols*, vol. 7, no. 1, p. bpac013, Jan. 2022. [Online]. Available: https://academic.oup.com/biomethods/article/doi/10.1093/biomethods/bpac013/6595016

[54] J. Ren, K. Song, C. Deng, N. A. Ahlgren, J. A. Fuhrman, Y. Li, X. Xie, R. Poplin, and F. Sun, "Identifying viruses from metagenomic data using deep learning,"

Quantitative Biology, vol. 8, no. 1, pp. 64–77, Mar. 2020. [Online]. Available: https://journal.hep.com.cn/qb/EN/10.1007/s40484-019-0187-4

[55] R. Kumar, A. A. Khan, S. Zhang, J. Kumar, T. Yang, N. A. Golalirz, Zakria, I. Ali, S. Shafiq, and W. Wang, "Blockchain-Federated-Learning and Deep Learning Models for COVID-19 detection using CT Imaging," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 16 301–16 314, Jul. 2021, arXiv:2007.06537 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2007.06537

[56] Q. Wu, K. He, and X. Chen, "Personalized Federated Learning for Intelligent IoT Applications: A Cloud-Edge Based Framework," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 35–44, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9090366/

[57] H. R. Roth, K. Chang, P. Singh, N. Neumark, W. Li, V. Gupta, S. Gupta, L. Qu, A. Ihsani, B. C. Bizzo, Y. Wen, V. Buch, M. Shah, F. Kitamura, M. Mendonça, V. Lavor, A. Harouni, C. Compas, J. Tetreault, P. Dogra, Y. Cheng, S. Erdal, R. White, B. Hashemian, T. Schultz, M. Zhang, A. McCarthy, B. M. Yun, E. Sharaf, K. V. Hoebel, J. B. Patel, B. Chen, S. Ko, E. Leibovitz, E. D. Pisano, L. Coombs, D. Xu, K. J. Dreyer, I. Dayan, R. C. Naidu, M. Flores, D. Rubin, and J. Kalpathy-Cramer, "Federated Learning for Breast Density Classification: A Real-World Implementation," *Distributed And Collaborative Learning*, vol. 12444, pp. 181–191, 2020, arXiv:2009.01871 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2009.01871

[58] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9084352/

[59] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," *Neurocomputing*, 2018, arXiv:1806.00582 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1806.00582

[60] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," Feb. 2019, number: arXiv:1902.04885 arXiv:1902.04885 [cs]. [Online]. Available: http://arxiv.org/abs/1902.04885

[61] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated Learning for Healthcare Informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, Mar. 2021. [Online]. Available: http://link.springer.com/10.1007/s41666-020-00082-4

[62] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated Deep Learning for Intrusion Detection in Industrial CyberPhysical Systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9195012/

[63] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9153560/

[64] M. Billah, S. T. Mehedi, A. Anwar, Z. Rahman, and R. Islam, "A Systematic Literature Review on Blockchain Enabled Federated Learning Framework for Internet of Vehicles," Mar. 2022, number: arXiv:2203.05192 arXiv:2203.05192 [cs]. [Online]. Available: http://arxiv.org/abs/2203.05192

[65] S. K. Lo, Q. Lu, C. Wang, H.-Y. Paik, and L. Zhu, "A Systematic Literature Review on Federated Machine Learning: From A Software Engineering Perspective," *ACM Computing Surveys*, vol. 54, no. 5, pp. 1–39, Jun. 2022, arXiv:2007.11354 [cs]. [Online]. Available: http://arxiv.org/abs/2007.11354

[66] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305–311, Jun. 2020. [Online]. Available: http://www.nature.com/articles/s42256-020-0186-1

[67] F. Zerka, S. Barakat, S. Walsh, M. Bogowicz, R. T. H. Leijenaar, A. Jochems, B. Miraglio, D. Townend, and P. Lambin, "Systematic Review of Privacy-Preserving

Distributed Machine Learning From Federated Databases in Health Care," *JCO Clinical Cancer Informatics*, no. 4, pp. 184–200, Nov. 2020. [Online]. Available: https://ascopubs.org/doi/10.1200/CCI.19.00047

[68] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konený, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards Federated Learning at Scale: System Design," Mar. 2019, number: arXiv:1902.01046 arXiv:1902.01046 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1902.01046

[69] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving Data Privacy via Federated Learning: Challenges and Solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, May 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9055478/

[70] R. Rizzo, A. Fiannaca, M. La Rosa, and A. Urso, "A Deep Learning Approach to DNA Sequence Classification," in *Computational Intelligence Methods for Bioinformatics and Biostatistics*, C. Angelini, P. M. Rancoita, and S. Rovetta, Eds. Cham: Springer International Publishing, 2016, vol. 9874, pp. 129–140, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-44332-4_10

[71] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges," Jun. 2021, number: arXiv:2009.13012 arXiv:2009.13012 [cs]. [Online]. Available: http://arxiv.org/abs/2009.13012

[72] H. Fang and Q. Qian, "Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning," *Future Internet*, vol. 13, no. 4, p. 94, Apr. 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/4/94

[73] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated Electronic Health Records," *International Journal of Medical Informatics*, vol. 112, pp. 59–67, Apr. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S138650561830008X

[74] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng, "Privacy-preserving Federated Brain Tumour Segmentation," Oct. 2019, number: arXiv:1910.00962 arXiv:1910.00962 [cs]. [Online]. Available: http://arxiv.org/abs/1910.00962

[75] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions," Jun. 2021, number: arXiv:2106.09527 arXiv:2106.09527 [cs]. [Online]. Available: http://arxiv.org/abs/2106.09527

[76] S. Silva, B. Gutman, E. Romero, P. M. Thompson, A. Altmann, and M. Lorenzi, "Federated Learning in Distributed Medical Databases: Meta-Analysis of Large-Scale Subcortical Brain Data," Mar. 2019, number: arXiv:1810.08553 arXiv:1810.08553 [cs, q-bio, stat]. [Online]. Available: http://arxiv.org/abs/1810.08553

[77] M. Ammad-ud din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System," Jan. 2019, number: arXiv:1901.09888 arXiv:1901.09888 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1901.09888

[78] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated Learning for Wireless Communications: Motivation, Opportunities and Challenges," May 2020, number: arXiv:1908.06847 arXiv:1908.06847 [cs, eess, stat]. [Online]. Available: http://arxiv.org/abs/1908.06847

[79] Y. Chen, J. Wang, C. Yu, W. Gao, and X. Qin, "FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare," May 2021, number: arXiv:1907.09173 arXiv:1907.09173 [cs]. [Online]. Available: http://arxiv.org/abs/1907.09173

[80] D. Li and J. Wang, "FedMD: Heterogenous Federated Learning via Model Distillation," Oct. 2019, number: arXiv:1910.03581 arXiv:1910.03581 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1910.03581

[81] S. Lu, Y. Zhang, Y. Wang, and C. Mack, "Learn Electronic Health Records by Fully Decentralized Federated Learning," Dec. 2019, number: arXiv:1912.01792 arXiv:1912.01792 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1912.01792

[82] X. Li, Y. Gu, N. Dvornek, L. Staib, P. Ventola, and J. S. Duncan, "Multi-site fMRI Analysis Using Privacy-preserving Federated Learning and Domain Adaptation: ABIDE Results," Dec. 2020, number: arXiv:2001.05647 arXiv:2001.05647 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2001.05647

[83] O. Choudhury, Y. Park, T. Salonidis, A. Gkoulalas-Divanis, I. Sylla, and A. K. Das, "Predicting Adverse Drug Reactions on Distributed Health Data using Federated Learning," *AMIA Annu Symp Proc*, p. 10, 2020.

[84] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *npj Digital Medicine*, vol. 3, no. 1, p. 119, Dec. 2020. [Online]. Available: https://www.nature.com/articles/s41746-020-00323-1

[85] A. Tampuu, Z. Bzhalava, J. Dillner, and R. Vicente, "ViraMiner: Deep learning on raw DNA sequences for identifying viral genomes in human samples," *PLOS ONE*, vol. 14, no. 9, p. e0222271, Sep. 2019. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0222271

[86] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, p. 30, 2014.

[87] M. Abdul Salam, S. Taha, and M. Ramadan, "COVID-19 detection using federated machine learning," *PLOS ONE*, vol. 16, no. 6, p. e0252573, Jun. 2021. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0252573

[88] I. Guyon and T. B. Laboratories, "A scaling law for the validation-set training-set size ratio," *Computer Science*, p. 11, 1997.

[89] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J.

Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, Jan. 2008. [Online]. Available: http://link.springer.com/10.1007/s10115-007-0114-2

[90] D. Sarkar, R. Bali, and T. Sharma, *Practical Machine Learning with Python*. Berkeley, CA: Apress, 2018. [Online]. Available: http://link.springer.com/10.1007/978-1-4842-3207-1