

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Resource Allocation, Poisoning Attacks and Defense Strategies in Privacy-Preserving Federated Learning

Jingjing Zheng



Doctoral Program in Electrical and Computer Engineering

Supervisor: Dr. Kai Li

Co-Supervisor: Prof. Dr. Eduardo Tovar

Co-Supervisor: Prof. Dr. Luís Miguel Pinho de Almeida

October 9, 2024

Resource Allocation, Poisoning Attacks and Defense Strategies in Privacy-Preserving Federated Learning

Jingjing Zheng

Doctoral Program in Electrical and Computer Engineering

Approved by:

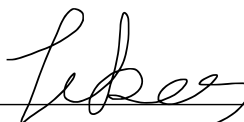
President: Prof. Dr. José Nuno Moura Marques Fidalgo

Referee: Prof. Dr. Mohsen Guizani

Referee: Prof. Dr. Susana Isabel Barreto de Miranda Sargento

Referee: Prof. Dr. Ana Cristina Costa Aguiar

Supervisor: Dr. Kai Li



October 9, 2024

Abstract

Mobile edge computing (MEC) has been considered as a promising technology to provide seamless integration of multiple application services. However, conventional enabling technologies for data analysis at mobile edge networks still require to upload personal data to the third-party, e.g., edge servers. In light of stringent requirements for data privacy legislations and low-latency of the emerging high-stake applications with smart devices, federated learning (FL) has been introduced naturally, which is carried out at edge clients in MEC for privacy-preserving training of data processing models.

FL collaboratively trains a shared and robust machine learning model under the orchestration of a server while distributing training data among a large number of clients in order to avoid data privacy leaking. However, a number of considerations make the FL in MEC non-trivial. In this dissertation, we focus on client scheduling of FL in MEC, poisoning attacks that an attacker carefully manipulates the training data or training model, and defense strategies of the FL poisoning attacks.

In the context of FL, selecting clients with larger datasets for training enhances the learning accuracy but consequently leads to substantial energy consumption. Conversely, opting for clients with smaller datasets reduces energy usage, yet this approach adversely affects the accuracy of the learning outcomes. We consider two scenarios, one is that the server has a-priori knowledge of clients, the other is not. Considering that the server knows the resource information of the edge client in prior, we proposed the suboptimal heuristic, *Federated Learning for Accuracy-energy Based Client Selection* (FedAECS) algorithm, which recursively assesses the potential clients in FL to balance the energy consumption and learning accuracy offline. Taking into account the practical scenario that the resource information of edge clients is unknown to the server in prior, and given the large state and action space, we proposed a new *FL-enabled twin-delayed deep deterministic policy gradient* (FL-DLT3) framework to achieve the optimal accuracy and energy balance in a continuous domain. Specially, long short-term memory (LSTM) is leveraged in FL-DLT3 to predict the time-varying data size, bandwidth, channel gain, and the remaining energy of the edge clients. FL-DLT3 is implemented in PyTorch. The effectiveness of FL-DLT3 is validated with the experimental data. Numerical results demonstrate that the proposed FL-DLT3 achieves fast convergence (less than 100 iterations) while the FL accuracy-to-energy consumption ratio is improved by 51.8% compared to the existing state-of-the-art benchmark.

While the two client scheduling methods we proposed both can achieve an efficient balance between the energy consumption of clients and learning accuracy of FL, we lack the consideration that some of selected clients may be potentially malicious. The malicious clients can poison local model updates by injecting poisoned instances into the training data (i.e., data poisoning attacks) or directly manipulating model updates (i.e., model poisoning attacks). As a result, the malicious client can undermine the FL model performance, such as significantly reduce the accuracy of the FL model or the convergence of FL training. Existing data poisoning attacks generally require the attacker to have the knowledge of (part of) the datasets used for FL training, so that it can

extract and manipulate the features of the datasets for effective attacks. However, without training data, it is challenging for the malicious local models to strike a balance between effectiveness and undetectability. To this end, we propose a novel, data-agnostic, model poisoning attack on FL, by designing a new *adversarial graph autoencoder* (GAE)-based framework. The attack requires no knowledge of FL training data and achieves both effectiveness and undetectability. By eavesdropping to the benign local models and the global model, the attacker extracts the graph structural correlations among the benign local models and the training data features substantiating the models. The attacker then adversarially regenerates the graph structural correlations while maximizing the FL training loss, and subsequently generates malicious local models using the adversarial graph structure and the training data features of the benign ones. A new algorithm is designed to iteratively train the malicious local models using GAE and sub-gradient descent. The convergence of FL under attack is rigorously proved, with a considerably large optimality gap. Experiments show that the accuracy of the FL local model under our proposed attack has a maximum drop of 0.33, 0.37, and 0.15 on the MNIST, Fashion-MNIST, and CIFAR-10 datasets respectively. Existing defense mechanisms fail to detect our proposed attack. The attack can give rise to an infection across all benign clients, making it a serious threat to FL.

Finally, to resist the GAE-based poisoning attacks, which can circumvent widely adopted Euclidean distance-based detection methods, we propose a new defense mechanism, namely, *GradCAM-AE*, against model poisoning attacks on FL. GradCAM-AE integrates Gradient-weighted Class Activation Mapping (GradCAM) and autoencoder (AE) to offer a substantially more powerful detection capability compared to existing Euclidean distance-based approaches. Particularly, GradCAM-AE generates a heat map for each uploaded local model update, transforming each local model update into a lower-dimensional, visualizable representation. An autoencoder further reprojects the GradCAM heat maps of all local module updates with improved distinguishability, thereby accentuating the hidden features of the heat maps and increasing the success rate of identifying anomalous heat maps and malicious local models. To mitigate potential misclassifications, we further design a voting algorithm, where model updates that consistently manifest as potentially malicious over several communication rounds are conclusively identified as malicious and removed from FL. A comprehensive evaluation of the proposed GradCAM-AE framework is conducted using the CIFAR-10 and GTSRB datasets under both Independent and Identically Distributed (IID) and Non-IID settings. The ResNet-18 and MobileNetV3-Large models are tested. The results substantiate that GradCAM-AE offers superior detection rates (Recall: 1.0, Precision: 0.943, Accuracy: 0.986, F1 score: 0.961, AUC: 0.992) and FL test accuracy (0.80) with ResNet-18 on Non-IID CIFAR-10 dataset compared to the state-of-the-art methods.

Resumo

A computação de borda móvel (Mobile Edge Computing - MEC) tem sido considerada uma tecnologia promissora para fornecer uma integração coesa de vários serviços de aplicação. No entanto, as tecnologias convencionais que permitem a análise de dados em redes de borda móvel ainda exigem o envio de dados pessoais para terceiros, como servidores de borda (edge servers). À luz dos rigorosos requisitos das leis de proteção de dados e da baixa latência das emergentes aplicações de alto risco com dispositivos inteligentes, surgiu naturalmente o aprendizado federado (FL), que é realizado em clientes de borda em MEC para treinamento de modelos de processamento de dados com privacidade.

O FL treina colaborativamente um modelo de aprendizado de máquina compartilhado e robusto sob a orquestração de um servidor enquanto distribui os dados de treinamento entre um grande número de clientes para evitar a divulgação de dados de privacidade. No entanto, uma série de considerações torna o FL em MEC não trivial. Nesta dissertação, focamos no escalonamento de clientes do FL em MEC, ataques de envenenamento em que o atacante manipula cuidadosamente os dados de treinamento ou o modelo de treinamento, e estratégias de defesa contra ataques de envenenamento do FL.

No contexto de FL, selecionar clientes com grandes conjuntos de dados para treinamento melhora a precisão do aprendizado, mas resulta em consumo de energia substancial. Inversamente, optar por clientes com conjuntos de dados menores reduz o consumo de energia, mas esse método afeta negativamente a precisão dos resultados do aprendizado. Consideramos dois cenários: um em que o servidor tem conhecimento prévio dos clientes e outro em que não. Considerando que o servidor conhece as informações de recursos do cliente de borda atempadamente, propomos o algoritmo subótimo Federated Learning for Accuracy-energy Based Client Selection (FedAECS), que avalia recursivamente os clientes potenciais no FL para equilibrar o consumo de energia e a precisão do aprendizado offline. Levando em conta o cenário prático em que as informações de recursos dos clientes de borda são desconhecidas para o servidor antecipadamente e considerando o grande espaço de estados e ações, propomos um novo framework FL habilitado por Twin-Delayed Deep Deterministic Policy Gradient (FL-DLT3) para alcançar o equilíbrio de precisão e energia ótimo em um domínio contínuo. Especificamente, a memória de longo alcance (LSTM) é usada no FLDLT3 para prever o tamanho dos dados, largura de banda, ganho do canal e energia restante dos clientes de borda variáveis com o tempo. O FL-DLT3 é implementado em PyTorch. A eficácia do FL-DLT3 é validada com dados experimentais. Os resultados numéricos demonstram que o FL-DLT3 proposto atinge convergência rápida (menos de 100 iterações), enquanto a relação precisão-consumo de energia do FL é melhorada em 51.8% em comparação com o benchmark de última geração existente.

Embora os dois métodos de escalonamento de clientes que propusemos possam alcançar um equilíbrio eficiente entre o consumo de energia dos clientes e a precisão do aprendizado do FL, não consideramos que alguns dos clientes selecionados possam ser potencialmente maliciosos. Os clientes maliciosos podem envenenar atualizações locais do modelo inserindo instâncias envene-

nadas nos dados de treinamento (ou seja, ataques de envenenamento de dados) ou manipulando diretamente as atualizações do modelo (ou seja, ataques de envenenamento de modelo). Como resultado, o cliente malicioso pode prejudicar o desempenho do modelo FL, como reduzir significativamente a precisão do modelo FL ou a convergência do treinamento do FL. Ataques de envenenamento de dados existentes geralmente exigem que o atacante tenha conhecimento (parcial) dos conjuntos de dados usados para treinamento do FL, para que possa extrair e manipular os recursos dos conjuntos de dados para ataques eficazes. No entanto, sem dados de treinamento, é difícil para os modelos locais maliciosos equilibrar a eficácia e a não detectabilidade. Para esse fim, propomos um novo ataque de envenenamento de modelo, independente de dados, projetando um novo framework baseado em adversarial graph autoencoder (GAE). O ataque não requer conhecimento dos dados de treinamento do FL e alcança eficácia e não detectabilidade. Ao analisar os modelos locais benignos e o modelo global, o atacante extrai as correlações estruturais de grafo entre os modelos locais benignos e os recursos dos dados de treinamento que sustentam os modelos. Em seguida, o atacante regenera adversariamente as correlações estruturais de grafo enquanto maximiza a perda de treinamento do FL e, subsequentemente, gera modelos locais maliciosos usando a estrutura de grafo adversarial e os recursos dos dados de treinamento dos modelos benignos. Um novo algoritmo é projetado para treinar iterativamente os modelos locais maliciosos usando GAE e descida de gradiente subgradiente. A convergência do FL sob ataque é rigorosamente demonstrada, com um déficit ótimo consideravelmente grande. Experimentos mostram que a precisão do modelo local FL sob nosso ataque proposto tem uma queda máxima de 0,33, 0,37 e 0,15 nos conjuntos de dados MNIST, Fashion-MNIST e CIFAR-10, respectivamente. Os mecanismos de defesa existentes não conseguem detectar nosso ataque proposto. O ataque pode dar origem a uma infecção em todos os clientes benignos, tornando-se uma séria ameaça ao FL.

Finalmente, para resistir a ataques de envenenamento baseados em GAE, que podem contornar amplamente os métodos de detecção baseados na distância euclidiana amplamente adotados, propomos um novo mecanismo de defesa, denominado GradCAM-AE, contra ataques de envenenamento de modelo no FL. GradCAM-AE integra GradCAM (Mapeamento de Atividade de Classe Ponderada por Gradiente) e autoencoder (AE) para oferecer uma capacidade de detecção significativamente mais potente em comparação com as abordagens existentes baseadas em distância euclidiana. Especificamente, o GradCAM-AE gera um mapa de calor para cada atualização local do modelo enviada, transformando cada atualização local do modelo em uma representação visualizável de menor dimensão. Um autoencoder remapeia adicionalmente os mapas de calor GradCAM de todas as atualizações de módulo local com melhor capacidade de distinção, acentuando assim os recursos ocultos dos mapas de calor e aumentando a taxa de sucesso na identificação de mapas de calor anômalos e modelos locais maliciosos. Para mitigar possíveis erros de classificação, projetamos ainda um algoritmo de votação, onde atualizações de modelo que se manifestam consistentemente como potencialmente maliciosas por várias rodadas de comunicação são identificadas conclusivamente como maliciosas e removidas do FL. Uma avaliação abrangente do framework GradCAM-AE proposto foi realizada usando os conjuntos de dados CIFAR-10 e GT-SRB sob configurações Independente e Identicamente Distribuída (IID) e Não IID. Os modelos ResNet-18 e MobileNetV3-Large foram testados. Os resultados comprovam que o GradCAM-AE oferece taxas de detecção superiores (Recall: 1,0, Precisão: 0,943, Precisão: 0,986, Pontuação F1: 0,961, AUC: 0,992) e precisão do teste FL (0,80) com ResNet-18 em CIFAR-Não-IID- 10 conjunto de dados em comparação com os métodos de última geração.

Acknowledgments

It is a long and arduous journey for me to reach here from a small, isolated and impecunious village in China. This dissertation would certainly not have come to its successful conclusion without the help, support and trust of supervisors, colleagues, friends and family.

First and foremost, I would like to sincerely thank my supervisor **Kai Li**, for his help, support, guidance, and dedication to making me an excellent researcher. It is exciting and thought-provoking to explore and discuss the new research fields with him, who is a resourceful, energetic and fruitful research scientist. I also thank my co-supervisor, **Eduardo Tovar**, the Director of CISTER, for creating an international environment and his unwavering support throughout my PhD journey. I am truly grateful to him for consistently encouraging me to participate in various activities. It was his frequent communication and coordination that made me fortunate enough to be a visiting scholar at Carnegie Mellon University in 2022. Moreover, his erudition and leadership have greatly influenced my academic career. I also extend my sincere appreciation to my co-supervisor, **Luis Alemida**, the director of Doctoral Program in Electrical and Computer Engineering and vice director of CISTER, who is a distinguished scientist. He tirelessly wrote recommendation letters for me so that I could obtain the scholarship. His kind and gracious ways to handle my every troubled situation, which helped me complete this journey.

I had the privilege of collaborating with a group of remarkable colleagues and researchers at different institutions, including but not limited to **Wei Ni**, **Mohsen Guizani**, **Naram Mhaisen**, and **Xin Yuan**. This dissertation would not exist without their extensive discussion, guidance and support. My special appreciation goes to **Wei Ni** for his expertise, taste, and great help for my last work of this dissertation.

Furthermore, I would like to express my sincere gratitude to **Geoffrey Nelissen** for first supervising me in the field of real-time systems during my first year of PhD study. Although he has started a new career at TU/e and I had to shift to a new research topic as I was unable to continue to work with him, I learnt from him how to pick the right research problems to work on and how to rigorously and elegantly share our research findings with the broader community, which I believe will benefit my entire career.

To **Anthony Rowe** and his team at Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, for their very warm welcome during my stay there, and for showing me an inspiring approach to doing research.

I would like to take this opportunity to thank my colleagues/friends at CISTER for their friendship and uncountable discussions, especially to **Harrison**, **Yilian**, **Javier**, **Jatin**, **Alam**, **Abdul Saeid**, **Ishfaq**, **Miguel**, **Patrick**, **Mubarak**, **José**, **Giann**, **Konstantinos**, **David**, **Yimin**, **Reydel**, **Shardul**, **Gowhar**, **Yousef**, and **Mohammad**. I would also like to thank **Sandra**, **Inês**, **Cristiana**, **Marwin** and **Benilde** for their indefatigable and invaluable administrative support.

Last but not least, I would like to express my forever deepest thanks to my parents, **Yongcai Zheng** and **Tianxiu Li**, my sister, **Ying Zheng**, for their unconditional love and support. I am so incredibly grateful for my wife, **Xiaowei He**, for her constant encouragement, love, and humor. I

am also grateful for my newborn daughter Abigail Jiahe Zheng. Without their unwavering support and sacrifices, I can not reach this far. The COVID pandemic stole over two years of in-person activities from all of us such that I could not meet and embrace them, I always hope to spend more time with them. This thesis is dedicated to them.

This dissertation was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDP/UIDB/04234/2020); also by the Operational Competitiveness Programme and Internationalization (COMPETE 2020) through the European Regional Development Fund (ERDF) and by national funds through the FCT, within project POCI-01-0145-FEDER-029074 (ARNET); also supported by the CISTER Research Unit (UIDP/UIDB/04234/2020) and by project ADANET (PTDC/EEI-COM/3362/2021), financed by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology).

Jingjing Zheng

“Train yourselves. Don’t wait to be fed knowledge out of a book. Get out and seek it. Make explorations. Do your own research work. Train your hands and your mind. Become curious. Invent your own problems and solve them. You can see things going on all about you. Inquire into them. Seek out answers to your own questions.”

Irving Langmuir

List of Publications

Journal Articles

- **J. Zheng**, K. Li, N. Mhaisen, W. Ni, E. Tovar and M. Guizani, "Exploring Deep-Reinforcement-Learning-Assisted Federated Learning for Online Resource Allocation in Privacy-Preserving EdgeIoT," in IEEE Internet of Things Journal, vol. 9, no. 21, pp. 21099-21110, 1 Nov.1, 2022, doi: 10.1109/JIOT.2022.3176739. [Chapter 3]
- K. Li, **J. Zheng**, X. Yuan, W. Ni, O. B. Akan and H. V. Poor, "Data-Agnostic Model Poisoning Against Federated Learning: A Graph Autoencoder Approach," in IEEE Transactions on Information Forensics and Security, vol. 19, pp. 3465-3480, 2024, doi: 10.1109/TIFS.2024.3362147. [Chapter 4]
- Li, Kai and **Zheng, Jingjing** and Ni, Wei and Huang, Hailong and Liò, Pietro and Dressler, Falko and Akan, Ozgur B, "Biasing Federated Learning With a New Adversarial Graph Attention Network," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2024.3499371.
- K. Li, X. Yuan, **J. Zheng**, W. Ni, F. Dressler and A. Jamalipour, "Leverage Variational Graph Representation for Model Poisoning on Federated Learning," in IEEE Transactions on Neural Networks and Learning Systems, doi: 10.1109/TNNLS.2024.3394252.
- **J. Zheng**, K. Li, X. Yuan, W. Ni and E. Tovar, "GradCAM-AE: A New Shield Defense Against Poisoning Attacks on Federated Learning" ACM Transactions on Privacy and Security (Under review) [Chapter 5]

Conference Papers

- **J. Zheng**, K. Li, N. Mhaisen, W. Ni, E. Tovar and M. Guizani, "Federated Learning for Online Resource Allocation in Mobile Edge Computing: A Deep Reinforcement Learning Approach," 2023 IEEE Wireless Communications and Networking Conference (WCNC), Glasgow, United Kingdom, 2023, pp. 1-6, doi: 10.1109/WCNC55385.2023.10118940. [Chapter 3]
- **J. Zheng**, K. Li, E. Tovar and M. Guizani, "Federated Learning for Energy-balanced Client Selection in Mobile Edge Computing," 2021 International Wireless Communications and Mobile Computing (IWCMC), Harbin City, China, 2021, pp. 1942-1947, doi: 10.1109/IWCMC51323.2021.9498853. [Chapter 3]
- K. Li, X. Yuan, **J. Zheng**, W. Ni and M. Guizani, "Exploring Adversarial Graph Autoencoders to Manipulate Federated Learning in The Internet of Things," 2023

International Wireless Communications and Mobile Computing (IWCMC), Marrakesh, Morocco, 2023, pp. 898-903, doi: 10.1109/IWCMC58020.2023.10183217.

- **J. Zheng**, K. Li, X. Yuan, W. Ni and E. Tovar, "Detecting Poisoning Attacks on Federated Learning Using Gradient-Weighted Class Activation Mapping," Companion Proceedings of the ACM on Web Conference 2024. [Chapter 5]
- **J. Zheng**, K. Li, X. Yuan, W. Ni, E. Tovar and J. Crowcroft, "Exploring Visual Explanations for Defending Federated Learning against Poisoning Attacks," ACM MobiCom 2024 (Accepted)
- Li, Kai and **Zheng, Jingjing** and Ni, Wei and Huang, Hailong and Liò, Pietro and Dressler, Falko and Akan, Ozgur B, "E-GATE: Explainable Graph-based Fairness Attacks on Federated Learning-enabled EdgeIoT," IEEE ICC 2025 (Under review)

Other Publications

- K. Li, W. Ni, **J. Zheng**, E. Tovar and M. Guizani, "Confidentiality and Timeliness of Data Dissemination in Platoon-based Vehicular Cyber-Physical Systems," in IEEE Network, vol. 35, no. 4, pp. 248-254, July/August 2021, doi: 10.1109/MNET.011.2000538.
- Kai Li, Ning Lu, **Jingjing Zheng**, Pei Zhang, Wei Ni, and Eduardo Tovar. 2021. BloothAir: A Secure Aerial Relay System Using Bluetooth Connected Autonomous Drones. ACM Trans. Cyber-Phys. Syst. 5, 3, Article 27 (July 2021), 22 pages. <https://doi.org/10.1145/3448254>
- K. Li, N. Lu, **J. Zheng**, P. Zhang, W. Ni and E. Tovar, "A Practical Secret Key Management for Multihop Drone Relay Systems based on Bluetooth Low Energy," 2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), Rome, Italy, 2021, pp. 1-2, doi: 10.1109/SECON52354.2021.9491599.

Contents

1	Introduction	1
1.1	Federated Learning (FL) in Mobile Edge Computing (MEC)	2
1.2	Poisoning Attacks on FL	4
1.3	Motivations and Challenges of FL in MEC	5
1.4	Overview of Contributions	7
1.4.1	Heuristic-based and DRL-based Resource Allocation for FL	7
1.4.2	Data-Agnostic Model Poisoning against FL: A Graph Autoencoder Approach	8
1.4.3	GradCAM-AE: A New Shield Defense Against Poisoning Attacks on FL	9
1.5	Dissertation Structure	10
2	Background and Related Work	11
2.1	Background	11
2.1.1	Deep Reinforcement Learning-based Resource Allocation for FL	11
2.1.2	Graph Autoencoder-based Poisoning Attacks against FL	13
2.1.3	Gradient-weighted Class Activation Mapping-based Defense against Poisoning Attacks on FL	14
2.2	Related Work	15
2.2.1	Resource Allocation for FL in MEC	15
2.2.2	Poisoning Attacks against FL	17
2.2.3	Defense Models against Poisoning Attacks on FL	18
2.3	Chapter Summary and Concluding Remarks	20
3	Resource Allocation of FL in MEC	21
3.1	System Model	22
3.1.1	FL Protocol with MEC	22
3.1.2	Energy Consumption Model	24
3.1.3	Training Time of FL	25
3.1.4	FL Accuracy	25
3.1.5	Evenness	25
3.2	Client Scheduling Based on Heuristic Algorithm	26
3.2.1	Problem Formulation	26
3.2.2	FedAECS	27
3.2.3	Numerical Results	29
3.3	DRL-based Resource Allocation with FL in MEC	31
3.3.1	Problem Formulation	31
3.3.2	DRL-based Clients Selection and Transmit Power Allocation	34
3.3.3	Numerical Experiments	39

3.4	Chapter Summary and Concluding Remarks	45
4	Data-Agnostic Model Poisoning against FL: A Graph Autoencoder Approach	47
4.1	System Model	47
4.1.1	Threat Model	47
4.1.2	Defense Model for Attacker Detection	50
4.2	Proposed Data-agnostic Model Poisoning Attack against FL	50
4.2.1	GAE Model for Data-Agnostic Model Poisoning	50
4.2.2	Training Algorithm of the Proposed GAE Model	54
4.2.3	Convergence Analysis of FL under Attack	54
4.3	Performance Evaluation	56
4.3.1	Implementation with PyTorch	57
4.3.2	Performance Analysis	57
4.4	Chapter Summary and Concluding Remarks	67
5	GradCAM-AE: A New Shield Defense Against Poisoning Attacks on FL	69
5.1	FL for Image Classification with Deep Neural Networks	69
5.2	Proposed GradCAM-AE against Model Poisoning Attacks	71
5.2.1	GradCAM-AE Architecture	71
5.2.2	Autoencoder for Malicious GradCAM Heat Map Identification	72
5.3	Performance Evaluation	75
5.3.1	Experiment Setup	75
5.3.2	Performance under IID Datasets	77
5.3.3	Performance under Non-IID Datasets	79
5.4	Chapter Summary and Concluding Remarks	82
6	Conclusion and Future Work	84
6.1	Conclusion	84
6.2	Future Work	85
6.2.1	Resource Allocation for FL	85
6.2.2	Data-Agnostic Poisoning Attacks against on FL	86
6.2.3	Defense Strategies for FL	86
A	Appendix	106
A.1	PROOF OF THEOREM 1	106

List of Figures

1.1	Mobile edge computing architecture.	2
1.2	MEC for smart cities.	3
1.3	FL-enabled MEC systems. A local model is trained with the local dataset at the client. The local model is uploaded by the client and aggregated by an edge server, where the global model is trained and returned to the clients.	4
1.4	Example of poisoning attacks. During the FL training process, an attacker can execute poisoning attacks by tampering the client's local data (data poisoning attacks) or directly crafting the local model parameters (model poisoning attacks).	5
2.1	Reinforcement learning, agent and environment.	12
2.2	A architecture of GAE.	13
2.3	(a) An original image, and (b) - (f) the GradCAM heat maps of VGG-16, ResNet-50, EfficientNet-B0 , MobileNetV3-Large and RegNetY-800MF.	14
3.1	The whole training process of FL, in each round, the time consumption of client k includes global model download, local model update and upload.	22
3.2	The objective function value $\Phi_i(\beta_i)$ varying with communication rounds.	30
3.3	The objective function value $\Phi_t(\beta_t)$ varying with the number of clients.	31
3.4	The relationship among objective function value $\Phi_t(\beta_t)$, the number of clients, and length of the communication round T_{max}	32
3.5	The relationship among objective function value $\Phi_t(\beta_t)$, length of the epoch T_{max} , and model accuracy of lower bound value ϵ_0	32
3.6	The relationship among runtime, length of the epoch T_{max} , and the number of clients.	33
3.7	An illustration of the proposed FL-DLT3 framework, where a policy network, a policy target network, two critic networks and two critic target networks are trained. Each of the networks consists of a feedforward branch and recurrent branch. A training round of FL is performed by the selected clients with the allocated transmit power. Then, the edge server gets the reward R_α and the state update $S_{\alpha'}^o$ of the clients. The transition $\{(A_{\alpha^-}, S_{\alpha'}^o, A_\alpha, R_\alpha, S_{\alpha'}^o)\}$ is stored into the replay buffer $\mathcal{B}$, and a mini-batch of transitions is randomly sampled from $\mathcal{B}$ to train the policy and critic networks on the edge server.	36
3.8	Comparison of the AE gains, where $T = 1000$ and $K = 40$	41
3.9	Comparison of the AE gain obtained by using FL-DLT3, FL-DLT3 without LSTM, FedAECS, FedCs and FedAvg with different number of clients	42
3.10	Compare the FL accuracy and energy consumption of FL-DLT3 and FL-DLT3 without LSTM as the number of devices changes	42
3.11	The AE gain achieved by the proposed FL-DLT3 given $K = 20, 40$, or 80	43

3.12	AE gain varying with data size following normal distribution while keeping the same variance (0.2 MB)	44
3.13	AE gain varying with bandwidth following normal distribution while keeping the same variance (4 KHz)	44
3.14	Compare the FL accuracy and energy consumption of FL-DLT3 and FL-DLT3 without LSTM as the average channel size of clients changes	45
3.15	Runtime with t -th round	46
4.1	The proposed data-agnostic, model poisoning attack, where the attacker eavesdrops the global model and the local models uploaded by the benign clients. Next, the attacker generates a malicious local model to contaminate the global model and the benign local models.	48
4.2	The proposed GAE model for generating data-agnostic, malicious local models, where the attacker overhears $\mathbf{W}_g^t$ and $\mathbf{W}_k^t, \forall k$ and applies the GCN-based encoder to create Z^M . The output of the encoder, i.e., the feature representations, is input to the decoder for feature reconstruction.	51
4.3	Given 100 FL communication rounds and five benign clients, we compare the local model testing accuracy under the GAE-based attack and the existing MP attack on the MNIST dataset.	59
4.4	Given 100 FL communication rounds and five benign clients, we compare the local model testing accuracy under the GAE-based attack and the existing MP attack on the Fashion-MNIST dataset.	60
4.5	Given 100 FL communication rounds and five benign clients, we compare the local model testing accuracy under the GAE-based attack and the existing MP attack on the CIFAR-10 dataset.	61
4.6	The global model accuracy with no attack.	62
4.7	The global model accuracy under the GAE-based attack.	62
4.8	Taking FL with five clients as an example, the Euclidean distances between the local models and the global model are presented, where client 5 is the attacker and launches the new GAE-based attack or the MP attack with MNIST dataset. . . .	63
4.9	Taking FL with five clients as an example, the Euclidean distances between the local models and the global model are presented, where client 5 is the attacker and launches the new GAE-based attack or the MP attack with Fashion-MNIST dataset. . . .	64
4.10	Taking FL with five clients as an example, the Euclidean distances between the local models and the global model are presented, where client 5 is the attacker and launches the new GAE-based attack or the MP attack with CIFAR-10 dataset. . .	65
4.11	When the number of clients, i.e., K , increases from 5 to 25, the y-axis shows the average accuracy under the GAE-based attack based on the MNIST datasets. The number of attackers is $K' = 2$ by default. Otherwise, K' increases with K , while keeping the ratio $K' : K = 2 : 5$	66
4.12	When the number of clients, i.e., K , increases from 5 to 25, the y-axis shows the average accuracy under the GAE-based attack based on the Fashion-MNIST datasets. The number of attackers is $K' = 2$ by default. Otherwise, K' increases with K , while keeping the ratio $K' : K = 2 : 5$	66
4.13	When the number of clients, i.e., K , increases from 5 to 25, the y-axis shows the average accuracy under the GAE-based attack based on the CIFAR-10 datasets. The number of attackers is $K' = 2$ by default. Otherwise, K' increases with K , while keeping the ratio $K' : K = 2 : 5$	67

5.1	FL with poisoning attack, where the attacker eavesdrops on the benign local models uploaded by the clients. The attacker fabricates a malicious local model update based on its eavesdropped local model updates. The Euclidean distances between the malicious and benign local models fall within a preset threshold distance. . . .	70
5.2	GradCAM-assisted defense against poisoning attacks on FL. The server arbitrarily selects an image (e.g., an image with the label “bird”) from the global model testing dataset to create GradCAM heat maps for every uploaded local model update. These GradCAM heat maps flow into an autoencoder for malicious model detection.	72
5.3	Autoencoder-based abnormal GradCAM heat map identification. The server flattens and concatenates GradCAM heat maps as input to the encoder neural network, which compresses the GradCAM heat maps from a high dimension to a low dimension Z . The decoder neural network takes Z as its input to reconstruct the original input GradCAM heat maps.	73
5.4	Test accuracy vs. communication rounds for ResNet-18 on IID CIFAR-10 and GTSRB.	78
5.5	Test accuracy vs. communication rounds for MobileNetV3-Large on IID CIFAR-10 and GTSRB.	79
5.6	Test accuracy vs. communication rounds for ResNet-18 on Non-IID CIFAR-10 and GTSRB.	80
5.7	Test accuracy vs. communication rounds for MobileNetV3-Large on Non-IID CIFAR-10 and GTSRB.	80
5.8	Comparison between generated heat maps by GradCAM and reconstructed heat maps by the autoencoder.	81
5.9	Reconstruction errors for ResNet18 on Non-IID CIFAR-10.	81
5.10	Scalability of GradCAM-AE vs. GradCAM-Krum	82
5.11	Detection rates of ResNet-18 and MobileNetV3-Large on Non-IID CIFAR-10 with Att = 0.4	82

List of Tables

3.1	Comparison results with Cplex.	30
3.2	Simulation parameters	40
5.1	Detection rates of ResNet-18 on IID and Non-IID CIFAR-10.	77
5.2	Detection rates of ResNet-18 on IID and Non-IID GTSRB.	78
5.3	Detection rates of MobileNetV3-Large on IID and Non-IID CIFAR-10.	78
5.4	Detection rates of MobileNetV3-Large on IID and Non-IID GTSRB.	78

Abbreviations

FL	Federated Learning
MEC	Mobile Edge Computing
IoT	Internet of Thing
FedAvg	Federated Averaging
ML	Machine Learning
FedAECS	Federated Learning for Accuracy-energy Based Client Selection
MKP	Multidimensional Knapsack Problem
DDPG	Deep Deterministic Policy Gradient
TD3	Twin-delayed Deep Deterministic Policy Gradient
FL-DLT3	FL-enabled Twin-delayed Deep Deterministic Policy Gradient
LSTM	Long Short-term Memory
GAE	Graph Autoencoder
GFT	graph Fourier transform
MP	Model Poisoning
NN	Neural Network
SVM	Support Vector Machine
TLS	Transport Layer Security
GradCAM	Gradient-weighted Class Activation Mapping
AE	Autoencoder
MDP	Markov Decision Process
MAB	Multi-armed Bandit
GANs	Generative Adversarial Networks
IID	Independent Identically Distributed
Non-IID	Non-Independent Identically Distributed
FAA-DL	Federated Anomaly Analytics enhanced Distributed Learning
DNN	Deep Neural Network

Chapter 1

Introduction

Nowadays, the ever-growing smart Internet of Things (IoT) clients, such as wearable devices, laptops, smart cars, and smart homes generate a wealth of data outside the server each day. In the traditional cloud computing approach, data collected by clients¹ is uploaded to server for machine learning (ML)-based task process (e.g., image classification), which is no longer sustainable. The reason is that transferring the big data to the server not only consumes a huge amount of bandwidth, energy and time, but also poses a great risk of data privacy leakage (Li et al., 2018). Mobile Edge Computing (MEC) as a potential solution in which the storage and computing capabilities of clients and edge servers are leveraged on to bring artificial intelligence (AI) processing closer to where data is produced. It can significantly reduces dependence on centralized cloud servers by processing data on edge clients, sensors, and edge nodes. This distributed computing model can provide faster, more efficient and more secure computing services than cloud computing. Figure 1.1 shows traditional MEC architecture. The MEC system consists of a edge server and terminal devices/clients, where edge clients can include IoT sensors, computers, smartphones, and other devices.

There are numerous examples of MEC use cases among industries that make MEC significantly important and benefit the industries a lot. For example, with MEC, autonomous vehicles can make faster and more accurate decisions based on real-time data, greatly reducing the latency that is typically associated with cloud computing, which allows for more precise navigation, better obstacle detection, and improved traffic flow management. MEC can enable autonomous vehicles to communicate with each other and infrastructure, enhancing safety and reducing accidents. Moreover, MEC offers a significant reduction in bandwidth usage and data storage requirements, allowing autonomous vehicles to operate with great efficiency and cost-effectiveness. As a result, the combination of autonomous vehicles and MEC promises a future of safer, more accessible, and sustainable transportation (Letaief et al., 2022; Li et al., 2021a). In healthcare, MEC enables real-time processing of health data collected from wearable devices and sensors. This allows for continuous monitoring of patients' vital signs and health metrics, which helps in analyzing

¹We use the term 'client' throughout the dissertation to describe entities in the network, such as devices, nodes, sensors, IoTs or organizations.

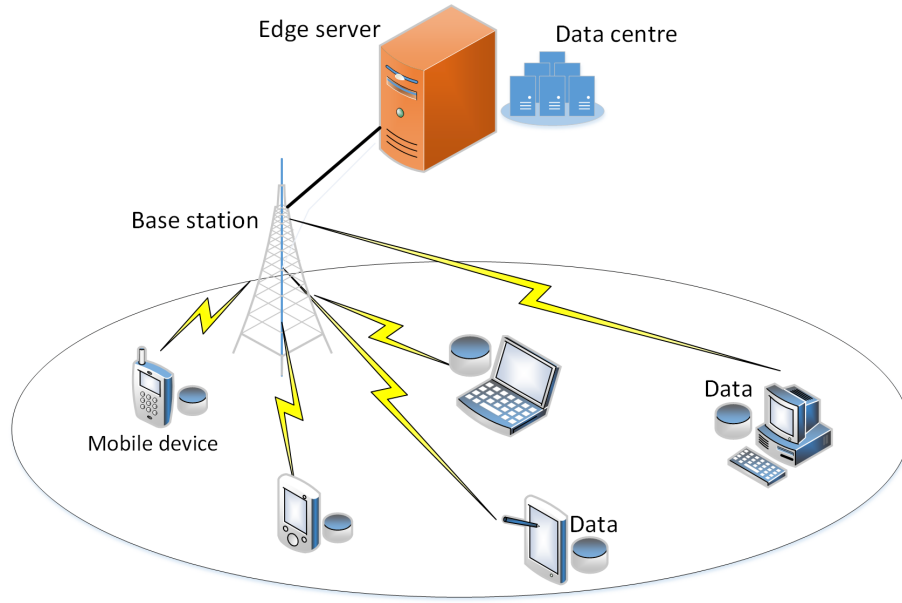


Figure 1.1: Mobile edge computing architecture.

data locally, sending relevant information to healthcare providers, and triggering alerts in case of anomalies (Hartmann et al., 2022). In smart cities, as shown in Figure 1.2, MEC enables real-time data processing for applications such as traffic management, video surveillance, environmental monitoring, and smart lighting, which contributes to the efficiency and responsiveness of urban services (Khan et al., 2020b; Jaber et al., 2023).

Compared with cloud computing, MEC has more obvious advantages of low cost, low latency and high privacy. However, computation offloading and data processing at edge servers still involve the transmission of potentially sensitive personal data. It is increasingly attractive to guarantee the data remains on personal clients and push network computation to the edge. Federated learning (FL), as a decentralized ML approach, enables multiple clients to cooperatively train a common ML model under the orchestration of a server without sharing their local data, which has greatly alleviated the issues of centralized ML in MEC.

The remainder of this chapter is organized as follows. First, we introduce FL in MEC and poisoning attacks on FL in Section 1.1 and Section 1.2, respectively. Then, we clarify the research motivations and challenges of FL for this dissertation in Section 1.3. A summary of the main contributions is given in Section 1.4. Finally, Section 1.5 describes the structure of the remainder of this dissertation.

1.1 Federated Learning (FL) in Mobile Edge Computing (MEC)

With data sources mainly located outside the cloud server today, an emergent architecture, MEC, has developed as a solution in which the computing and storage capabilities (Mao et al., 2017) of clients and an edge server are leveraged on to bring model training closer to where data is generated

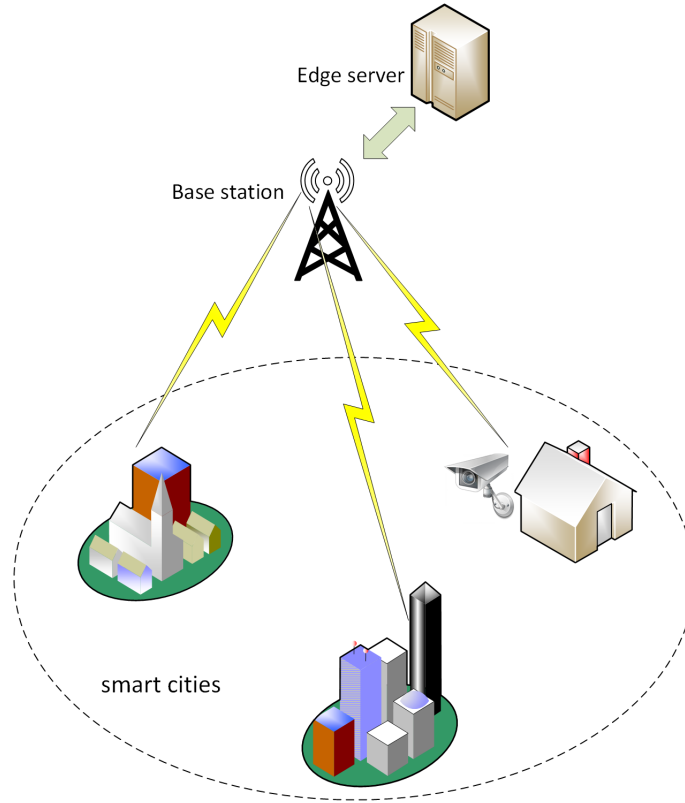


Figure 1.2: MEC for smart cities.

(Hua et al., 2023). In MEC, an edge server provides data processing services for computation-intensive and latency-critical applications of edge clients, such as augmented reality (Suresh et al., 2023), self-driving cars (Li et al., 2020; Zhang et al., 2022c), unmanned aerial vehicles (Chen et al., 2022; Li et al., 2022; Kurunathan et al., 2023), and healthcare (Peng et al., 2023) etc. In other words, the privacy-persevering MEC-based IoTs or clients can offload their local computation-intensive tasks to computationally powerful edge servers (Mach and Becvar, 2017; Wang et al., 2020a; Zheng et al., 2022). Nevertheless, offloading the source data of the clients to the edge server incurs significant communication costs and poses a high risk of sensitive personal data leakage (Lim et al., 2020). To overcome these challenges, a decentralized ML approach called federated learning (FL) is introduced in (McMahan et al., 2017). In FL, clients use their local data to cooperatively train a joint ML model. On a server, only locally computed updates are received and aggregated for an enhanced global model benefiting from the distributed learning. The source data consistently remains at the clients, which can increase data privacy.

Figure 1.3 depicts a typical MEC architecture where clients deployed sense and process patient physiological information, e.g., pulse rate, body temperature, and blood pressure (Xiao et al., 2018). Specifically, clients use image classification models to classify pictures or images. The data are offloaded to the server in the cloud for health analysis and reporting. However, in the FL-enabled MEC, FL can help multiple clients to cooperatively train an effective global image classification model, without the need of sharing the images and pictures used for the training.

Particularly, A local model (e.g., the weight vector (McMahan et al., 2017; Xiong et al., 2023; Karimireddy et al., 2020) or gradient (Geiping et al., 2020; Yin et al., 2021; Sun et al., 2021; Wang et al., 2022b, 2023; Li et al., 2023c; Hatamizadeh et al., 2023)) is trained on the sensing data of a client to sense and process private information. Next, the edge server aggregates the local models of all clients to create a comprehensive and effective global model without collecting the private data of the clients. A global model is obtained at the edge server, e.g., by applying federated averaging (FedAvg) (McMahan et al., 2017; Lee et al., 2023; Xiong et al., 2023) to the local models. The edge server broadcasts the global model back to all the clients. The above process called one communication round. According to the global model, each of the clients renews the training of its local model. By iteratively training the local model at the client and updating the global model at the edge server, the learning accuracy of FL on the data classification and event prediction can be progressively improved (Abdulrahman et al., 2021b) after multiple communication rounds.

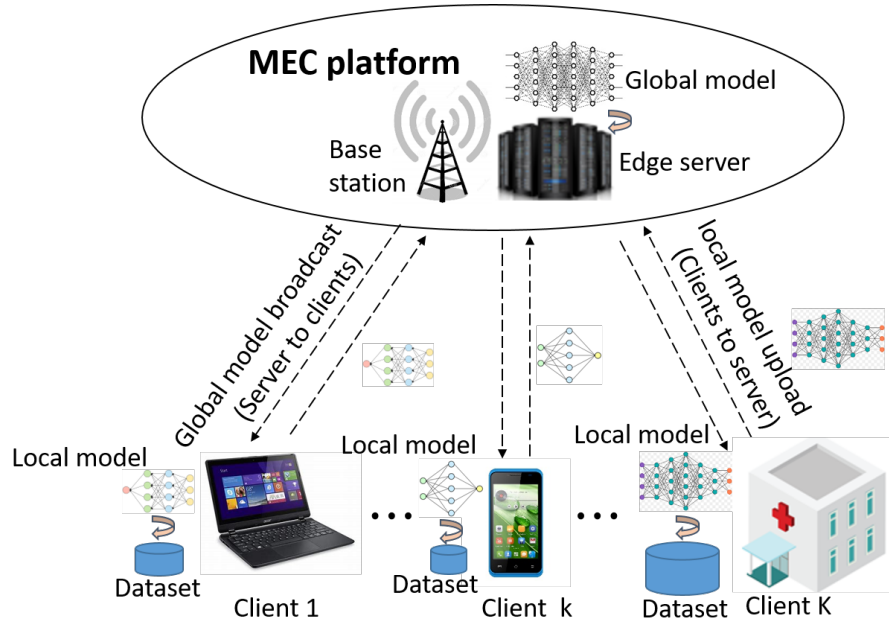


Figure 1.3: FL-enabled MEC systems. A local model is trained with the local dataset at the client. The local model is uploaded by the client and aggregated by an edge server, where the global model is trained and returned to the clients.

1.2 Poisoning Attacks on FL

Despite the fact that FL can help prevent an attacker from accessing the private data of clients, an attacker (in most cases, a malicious client) can potentially launch cyber-epidemic attacks to manipulate FL and propagate the attacks into other benign clients (Jere et al., 2021; Rahman et al., 2020; Mothukuri et al., 2021; Kairouz et al., 2021; Ghimire and Rawat, 2022; Zhang et al., 2023b) resulting in a failure of FL training. Based on the source of the poisoned local model updates,

poisoning attacks can be classified into data poisoning attack (Bagdasaryan et al., 2020; Gupta et al., 2023; Sun et al., 2022) and model poisoning attack (Bhagoji et al., 2019; Fang et al., 2020; Yang et al., 2023; Cao et al., 2023). The attacker on the left in Figure 1.4 shows data poisoning attacks by modifying local data such as replacing labels or specific features of the data. Unlike data poisoning attacks, the attacker on the right in Figure 1.4 attempts to directly manipulate the local model that it sends to the server for aggregation. At a high level, both poisoning attacks attempt to modify the behavior of the target model in some undesirable way to falsify the FL learning process during global aggregation. Consequently, the global model will be updated incorrectly, and the whole learning system becomes corrupted (Cao and Gong, 2022; Lyu et al., 2022). Note that any data poisoning attack can be transformed into a model poisoning attack, i.e., we can obtain the local model update by training local poisoning dataset and regarded it as the poisoned local model update.

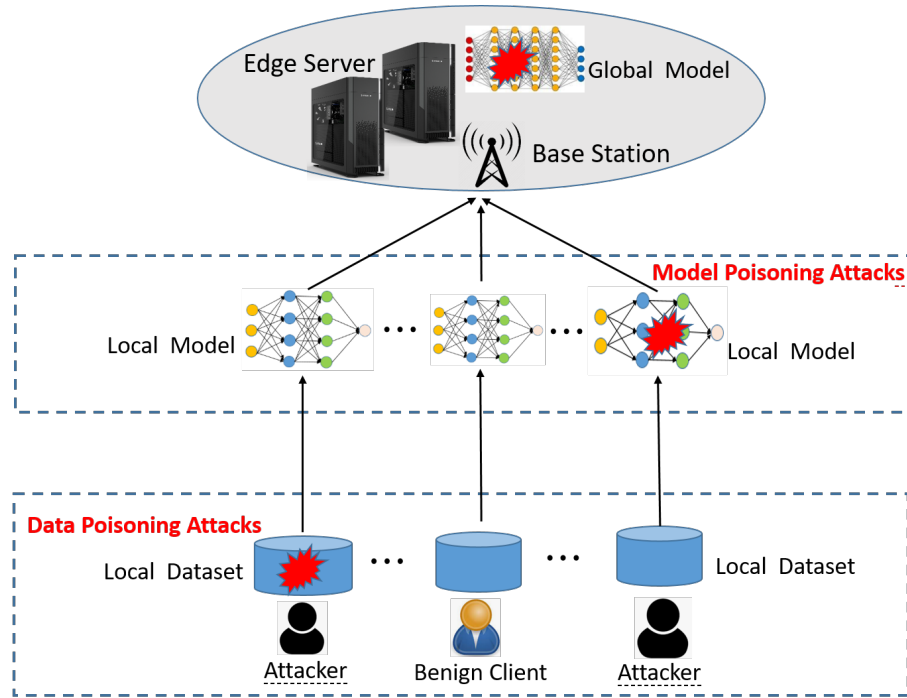


Figure 1.4: Example of poisoning attacks. During the FL training process, an attacker can execute poisoning attacks by tampering the client's local data (data poisoning attacks) or directly crafting the local model parameters (model poisoning attacks).

1.3 Motivations and Challenges of FL in MEC

While FL has been successfully applied in many fields, such as next-word prediction for mobile keyboard (Hard et al., 2018), to develop predictive models for diagnosis in health AI (Brisimi et al., 2018; Li et al., 2023b), to detect hazardous zones in a specific search space by monitoring the air quality index (Chhikara et al., 2021; Pandya et al., 2023), as a nascent field, FL leaves

a lot to be desired in MEC. The motivation behind our choice to investigate the topics of client scheduling, model poisoning attacks, as well as defense strategies with FL is given below.

Improper client scheduling in FL can lead to various undesirable consequences, affecting the performance, fairness of the FL model, and energy consumption of the clients. For instance, if only clients with a small amount of data are scheduled, although energy consumption is reduced because short-time training data is required, the performance of the FL model is unsatisfactory due to insufficient training data; If only clients with large amounts of data are scheduled, although the performance of the model is improved due to sufficient training data, it also increases the energy consumption of clients due to the need for long-term training data. Moreover, If clients are disproportionately selected based on certain demographics or characteristics, the FL model may become biased towards those groups.

Challenge 1: Resource allocation for FL in MEC. Client scheduling is a critical bottleneck in federated networks, coupled with heterogeneous systems in practice, where the clients have different data resources, energy states, computational and storage capacities, wireless channel conditions, and different willingness to participate in model training etc. Additionally, the network size and systems-related constraints on each client typically result in only a small fraction of the clients can be selected in each communication round, e.g., hundreds of active devices in a million-device network (Bonawitz et al., 2019). Due to time-varying data quantity and channels, the energy consumption of the edge clients (Zheng et al., 2021; Dinh et al., 2021) on data training and transmission can be greatly different from each other. On the one hand, scheduling the edge clients with a large dataset improves the learning accuracy of FL, analyzing large datasets rises energy consumption at the edge clients. On the other hand, the energy consumption can be reduced by selecting the clients with small data sets for FL, resulting in a falling learning accuracy. Moreover, scheduling the edge clients with poor channel condition (Zheng et al., 2023a) to transmit requires a high transmit power at the client due to packet retransmissions. Therefore, client selection for balancing the energy consumption and the learning accuracy of FL is crucial in MEC.

Poisoning attacks may lead to unsatisfactory and serious consequences. For instance, some early-stage cancer patients may be misdiagnosed as healthy individuals by the poisoned global model, leading to missing the best chance of being cured; the traffic stop signs would be incorrectly recognized as speed limit signs by the ML model trained on the poisoned sign with attacker-defined trigger, thereby yielding an autonomous car to continue moving at an intersection; in a FL setting for fraud detection in financial transactions, attackers injects synthetic fraudulent transactions into their local dataset. The global fraud detection model may become less effective in distinguishing genuine transactions from fraudulent ones, potentially resulting in financial losses. Therefore, it is exigent to investigate the evolving poisoning attacks in FL and design defenses to mitigate the risks associated with them.

Challenge 2: Poisoning Attacks against FL. By launching model poisoning attacks (Fang et al., 2020; Shejwalkar and Houmansadr, 2021; Xie et al., 2020) or data poisoning attacks (Tolpegin et al., 2020; Gupta et al., 2023), an attacker could manipulate either the hyperparameters of the local model or the training dataset of benign clients to compromise learning accuracy. Exist-

ing model poisoning attacks generally require the attacker to have the knowledge of (part of) the datasets used for FL training (Tian et al., 2022), so that it can extract and manipulate the features of the datasets for effective attacks (Zhang et al., 2022a). This knowledge also helps minimize the detectability of malicious local models. Much less constrained and potentially more threatening model poisoning attacks on FL would result if they could be based solely on the benign local models overheard by an attacker and the global models broadcast by the aggregator; i.e., when the attacker has no access to the training data. However, without training data, it is challenging for the malicious local models to strike a balance between effectiveness and undetectability (Shejwalkar et al., 2022).

Challenge 3: FL Defense Strategies. As we discussed in **Challenge 2**, the distributed architecture of FL makes it particularly susceptible to poisoning attacks. Clients compromised by adversarial actors can alter local model update parameters, subsequently contaminating the global FL models (Cao and Gong, 2022; Fang et al., 2020). Existing countermeasures, leveraging Euclidean distance-based metrics to discern deviations between malicious and benign model updates, have demonstrated efficacy against such attacks (Blanchard et al., 2017; Zhang et al., 2022a). However, sophisticated adversaries can craft malicious model updates such that the Euclidean distances to benign counterparts remain below a designated threshold, thereby eluding detection by defenses reliant on this metric. In other words, the sophisticated adversaries can craft malicious model updates very similar to benign ones, breaking existing defense rules. This makes it difficult to develop more FL defense strategies from server side to shield evolving and intractable poisoning attacks.

1.4 Overview of Contributions

In this section, we summarize the main contributions of this dissertation in relation to the research challenges outlined in Section 1.3. The section is organized around three pillars, namely: (1) resource allocation for FL in MEC; (2) poisoning attacks on FL; and finally, (3) FL defense strategies against poisoning attacks.

1.4.1 Heuristic-based and DRL-based Resource Allocation for FL

This contribution addresses **Challenge 1**. We propose a new FL-based client selection optimization to balance the trade-off between energy consumption of the edge clients and the learning accuracy of FL. *Considering the scenario of the resource information of edge clients is known to the server in prior*, we first demonstrate that the optimization problem is NP-complete. An energy-accuracy balancing heuristic algorithm, Federated Learning for Accuracy-energy based Client Selection (FedAECS) is proposed to approximate the optimal solutions in polynomial time. In particular, FedAECS prioritizes the edge clients according to the learning time, data size, and channel quality. Furthermore, FedAECS recursively traverses the priority list and selects the edge clients, which minimizes the ratio of energy consumption and learning accuracy of FL **offline** in

each communication round. *Taking into account the practical scenario that the resource information of edge clients is unknown to the server in prior*, we propose to jointly optimize the selection of clients and their transmit powers in an FL-empowered MEC system, thereby balancing the learning accuracy of FL and energy consumption of the clients. The optimization is **online**, adapting to the time-varying arrivals of training data and the energy budget of the clients, and the changing channel conditions between the clients and the base station (i.e., the model aggregator). A new FL-enabled twin-delayed deep deterministic policy gradient (FL-DLT3) framework is proposed to learn the network state dynamics while maximizing the ratio of the learning accuracy of FL to the energy consumption of the clients. Considering the continuous transmit powers of the clients, FL-DLT3 optimizes the edge server's selection of clients for each communication round of FL and the transmit powers of the clients, based on twin delayed deep deterministic policy gradient (TD3). A new long short-term memory (LSTM) layer is further designed in coupling with the proposed FL-DLT3 to predict the time-varying network states, e.g., data size, bandwidth, channel gain, and the remaining energy of the clients. The LSTM layer estimates the unobserved states at every training iteration of the TD3.

The above contributions are presented in the following publications:

- **J. Zheng**, K. Li, E. Tovar and M. Guizani, "Federated Learning for Energy-balanced Client Selection in Mobile Edge Computing," 2021 International Wireless Communications and Mobile Computing (IWCMC), Harbin City, China, 2021, pp. 1942-1947, doi: 10.1109/IWCMC51323.2021.9498853.
- **J. Zheng**, K. Li, N. Mhaisen, W. Ni, E. Tovar and M. Guizani, "Exploring Deep-Reinforcement-Learning-Assisted Federated Learning for Online Resource Allocation in Privacy-Preserving EdgeIoT," in IEEE Internet of Things Journal, vol. 9, no. 21, pp. 21099-21110, 1 Nov.1, 2022, doi: 10.1109/JIOT.2022.3176739.
- **J. Zheng**, K. Li, N. Mhaisen, W. Ni, E. Tovar and M. Guizani, "Federated Learning for Online Resource Allocation in Mobile Edge Computing: A Deep Reinforcement Learning Approach," 2023 IEEE Wireless Communications and Networking Conference (WCNC), Glasgow, United Kingdom, 2023, pp. 1-6, doi: 10.1109/WCNC55385.2023.10118940.

1.4.2 Data-Agnostic Model Poisoning against FL: A Graph Autoencoder Approach

This contribution addresses **Challenge 2**. We propose a novel, data-agnostic, model poisoning attack on FL, by designing a new adversarial graph autoencoder (GAE)-based framework. The attack requires no knowledge of FL training data and achieves both effectiveness and undetectability. By eavesdropping the benign local models and the global model, the attacker extracts the graph structural correlations among the benign local models and the training data features substantiating the models. The attacker then adversarially regenerates the graph structural correlations while maximizing the FL training loss, and subsequently generates malicious local models using the adversarial graph structure and the training data features of the benign ones. A new algorithm is

designed to iteratively train the malicious local models using GAE and sub-gradient descent. The convergence of FL under attack is rigorously proved, with a considerably large optimality gap.

The above contributions are presented in the following publications:

- K. Li, X. Yuan, **J. Zheng**, W. Ni and M. Guizani, "Exploring Adversarial Graph Autoencoders to Manipulate Federated Learning in The Internet of Things," 2023 International Wireless Communications and Mobile Computing (IWCMC), Marrakesh, Morocco, 2023, pp. 898-903, doi: 10.1109/IWCMC58020.2023.10183217.
- K. Li, **J. Zheng**, X. Yuan, W. Ni, O. B. Akan and H. V. Poor, "Data-Agnostic Model Poisoning Against Federated Learning: A Graph Autoencoder Approach," in IEEE Transactions on Information Forensics and Security, vol. 19, pp. 3465-3480, 2024, doi: 10.1109/TIFS.2024.3362147.
- K. Li, X. Yuan, **J. Zheng**, W. Ni, F. Dressler and A. Jamalipour, "Leverage Variational Graph Representation For Model Poisoning on Federated Learning" IEEE Transactions on Neural Networks and Learning Systems (Under review)

1.4.3 GradCAM-AE: A New Shield Defense Against Poisoning Attacks on FL

This contribution addresses **Challenge 3**. We propose a new defense mechanism, namely, GradCAM-AE, against GAE-based model poisoning attacks on FL, which integrates Gradient-weighted Class Activation Mapping (GradCAM) and autoencoder (AE) to offer a substantially more powerful detection capability compared to existing Euclidean distance-based approaches. Particularly, GradCAM is adopted to produce heat maps for each uploaded local model update, hence transforming each local model into a lower-dimensional, visualizable representation. This provides a conduit for pinpointing malicious model updates by singling out anomalous GradCAM heat maps. An autoencoder is used to reproject the GradCAM heat maps to accentuate the hidden features of the heat maps, and improve the distinguishability of the heat maps and the success rate of identifying anomalous heat maps and malicious local models. To mitigate potential misclassifications, we also design a voting algorithm, where model updates that consistently manifest as potentially malicious over several communication rounds are conclusively identified as malicious and removed from FL.

The above contributions are presented in the following publications:

- **J. Zheng**, K. Li, X. Yuan, W. Ni and E. Tovar, "Detecting Poisoning Attacks on Federated Learning Using Gradient-Weighted Class Activation Mapping," THE WEB CONFERENCE 2024 (Accepted)
- **J. Zheng**, K. Li, X. Yuan, W. Ni and E. Tovar, "GradCAM-AE: A New Shield Defense Against Poisoning Attacks on Federated Learning," IEEE Transactions on Dependable and Secure Computing (Under review)

1.5 Dissertation Structure

The remainder of this dissertation is organized as follows.

Chapter 2 discusses the background and related work for the better understanding of this dissertation. First, we provide an overview of DRL, GAE and GradCAM. The reasons behind why these technologies can be applied to our research work are presented. Then, we introduced state-of-the-art works of resource allocation for FL in MEC, poisoning attacks against FL, and defense models against poisoning attacks on FL, respectively. We further highlight the novelty of our work in every subsection. Finally, the integrated summary of these two parts is conclude.

Chapter 3 details the FL and system models. It also presents the two proposed solutions of resource allocation for FL in MEC, referred to as FedAECS and FL-DLT3, with offline and online, respectively. The chapter shows a substantial amount of numerical results of the two proposed solutions and benchmarks.

Chapter 4 introduces the proposed a novel, data-agnostic, model poisoning attack on FL, by designing a new adversarial GAE-based framework. The chapter implements the GAE-based attack against the FL on SVM models using PyTorch. Performances were evaluated based on the MNIST, fashionMNIST, and CIFAR-10 datasets. In addition, the chapter conducts a rigorous performance analysis, which proves the convergence of the global model under attack, but to an inferior optimality gap.

Chapter 5 elaborates the new defense mechanism, namely, GradCAM-AE, against model poisoning attacks on FL. This chapter conducts a comprehensive assessment of the proposed GradCAM-AE framework using two public datasets, CIFAR-10 and GTSRB, under both IID and Non-IID settings. The assessment encompasses two prominent deep learning models, i.e., ResNet-18 and MobileNetV3-Large. Numerous validation results demonstrate that GradCAM-AE offers superior detection rates and FL test accuracy compared to the state-of-the-art methods.

Chapter 6 contains a summary of the dissertation. Finally, this chapter discusses the limitations of this dissertation and provides some directions for future work.

Chapter 2

Background and Related Work

This chapter covers general information on key technical concepts and related work which are essential for a better understanding of this dissertation. We start by providing an overview of DRL, GAE and GradCAM in Section 2.1. The state-of-the-art work and the novelty of our work are given in Section 2.2. We conclude with an integrated summary of these two parts in Section 2.3.

2.1 Background

In this section, we will briefly introduce the key technical concepts DRL involved in Chapter 3, GAE involved in Chapter 4, and GradCAM involved in Chapter 5, respectively. The reasons why these technologies can be artfully integrated in our work are also stated.

2.1.1 Deep Reinforcement Learning-based Resource Allocation for FL

Deep reinforcement learning (DRL) is a subfield of ML that combines reinforcement learning algorithms with deep neural networks. Reinforcement learning involves computational agents to make sequences of decisions by trial and error in an environment to optimize an objective (e.g. maximizing the game score). This problem is often modeled mathematically as a Markov decision process (MDP), where an agent is in state S_t at every timestep t , takes action A_t , receives a scalar reward R_t and transitions to the next state S_{t+1} according to environment dynamics, as shown in Figure 2.1. However, in many practical sequential decision-making problems, the state space S_t of the MDP are high-dimensional (e.g., images from a camera) and cannot be addressed by traditional RL algorithms. DRL incorporates deep learning to solve such MDPs, often representing the policy functions as a neural network and developing specialized algorithms that perform well in high-dimensional state space. DRL has been used for a diverse set of applications such as UAVs (Li et al., 2019a, 2021b), robotics (Zhao et al., 2020), video games (Torrado et al., 2018; Lample and Chaplot, 2017), finance (Avramelou et al., 2024) etc.

DRL can be applied to FL resource allocation for the following reasons: (a) The joint online optimization of client scheduling and transit power allocation for clients to balance the learning

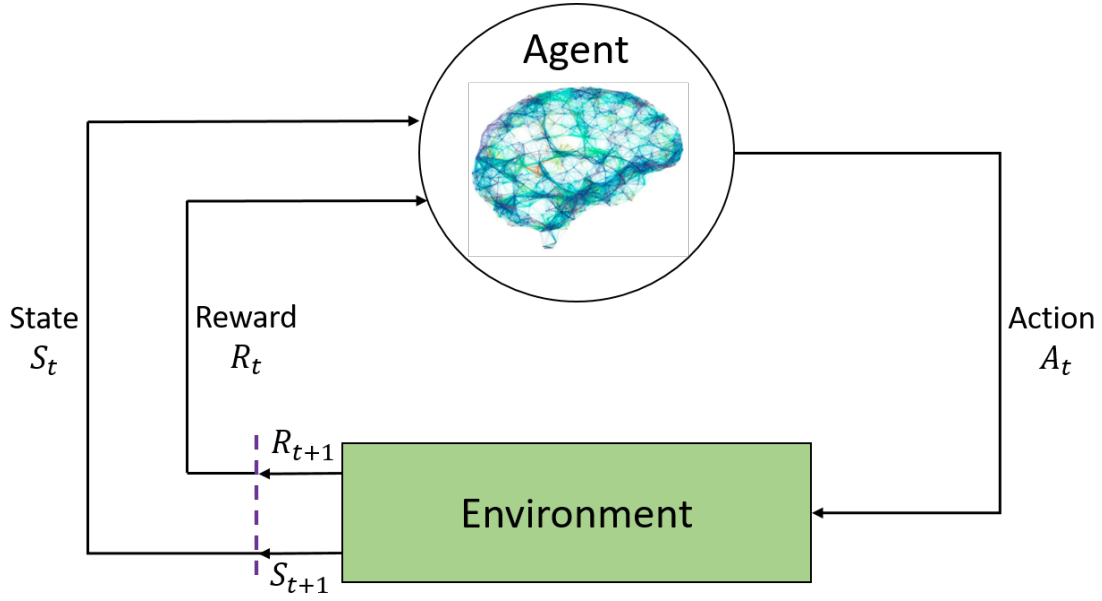


Figure 2.1: Reinforcement learning, agent and environment.

accuracy of FL and energy consumption of the clients in MEC, which can be regarded as a sequential decision-making problem with single objective and commonly modeled as MDP. (b) FL server may face challenges in implementing mathematical models of the complex environment. (c) The network state consists of the source data size of the clients, channel conditions between the edge server and the clients, bandwidth and remaining energy of the selected clients. All the entries of a network state are continuous. The action space includes the discrete selection of clients and the continuous transmit power allocation. Given the continuous network states and actions, a large number of clients lead to a large state and action space, and complex network state transitions. (d) The edge server often lack up-to-date knowledge about the status of clients, including data size, battery, energy, and channel conditions.

In RL problems with continuous action spaces, existing RL algorithms such as deep deterministic policy gradient (DDPG) (Lillicrap et al., 2016) suffer from the issues of overestimated value estimates and suboptimal policies as a result of function approximation errors. To address these issues, the authors (Fujimoto et al., 2018) developed twin delayed deep deterministic policy gradient (TD3), which greatly improves both the learning speed and performance of DDPG in a number of challenging tasks in the continuous control setting. The framework of TD3 is illustrated in Section 3.3.2.2, it consists of a policy network, a policy target network, two critic networks and two critic target networks. In the context of resource allocation in privacy-preserving FL, TD3 plays a crucial role in optimizing various operational aspects, including client scheduling and transmit power allocation of clients. The state information captured by the TD3 can encompass the client's current data size, bandwidth, channel gain, and remaining energy of the clients. A training round of FL is performed by the selected clients with the allocated transmit power. Then, the edge server gets the AE gain (i.e., the ratio of the FL accuracy to the energy consumption of the selected clients, and the penalty resulting from the data unevenness among the selected clients) and the state update the

clients. Experience replay is utilized in TD3 to train the policy and critic networks on the edge server. Particularly, to find out the hidden state in the environment, e.g., the state of a client that has not been selected previously, and accelerate the convergence of the FL, we embed long short term memory (LSTM) into each policy and critic network in the original TD3. The LSTM unit is composed of a cell activation vector, an input gate, an output gate and a forget gate, the details are presented in Section 3.3.2.4. The improved version of TD3 framework in FL with resource allocation is coined as federated learning enabled twin-delayed deep deterministic policy gradient (FL-DLT3), in which formulation is delineated in Section 3.3.2.

2.1.2 Graph Autoencoder-based Poisoning Attacks against FL

Graph Autoencoder (GAE) (Kipf and Welling, 2016) consists of a neural network encoder and a neural network decoder. The encoder works as an inference layer that captures the topological structure and node content of a graph, and the decoder acts as a generative layer that reconstructs the graph from the learned latent representation. Figure 2.2 depicts the structure of the GAE model. The adjacency and feature matrix are given as input to the encoder and a generated adjacency matrix is given as output from the decoder that processes a generated graph.

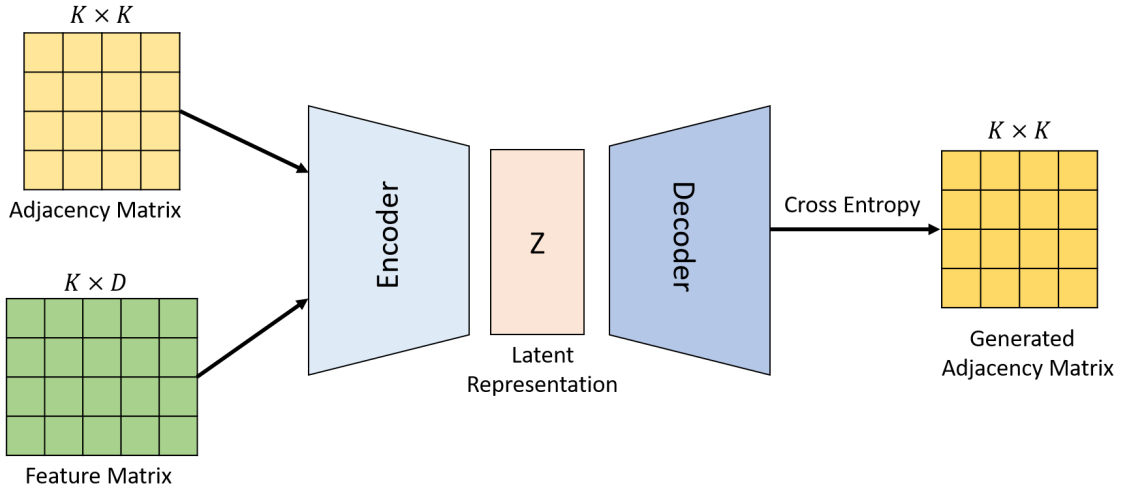


Figure 2.2: A architecture of GAE.

The reasons behind GAE can be used in FL model poisoning attacks are as follows: each local model and its parameters can be regarded as a graph node and node features, respectively. These local benign models can be eavesdropped by attackers. The attacker utilizes eavesdropped local models to build the input graph that is composed of an adjacency matrix and a feature matrix, which are processed by GAE to generate a new graph that is highly similar and related to the original input graph. Attacker can decouple the correlation between models and then focus on the data features that confirm the local model, thereby selecting a vector from the feature data as a malicious model and uploading it to the server. Existing Euclidean distance and cosine distance based defense mechanisms are commonly used metrics to assess the similarities between two

vectors, which are particularly useful in scenarios where large deviations in the magnitude of updates are indicative of malicious model updates. However, the Euclidean distance may not effectively capture scenarios where vectors are close in magnitude but differ in direction. Thus, the malicious model generated by GAE based on non-Euclidean distance space makes the protection mechanism based on Euclidean distance readily fail.

2.1.3 Gradient-weighted Class Activation Mapping-based Defense against Poisoning Attacks on FL

Gradient-weighted class activation mapping (GradCAM) is a technique for producing visual explanations for convolutional neural network (CNN) or deep neural network (DNN). GradCAM uses the gradient information flowing back into the last convolutional layer of the CNN or DNN to assign importance values to each neuron for a particular decision of interest (Selvaraju et al., 2017). It helps to understand which parts of the image are most important for classification. Figure 2.3 shows the GradCAM heat maps generated with different pretrained DNN architectures, i.e., VGG-16 (Simonyan and Zisserman, 2015), ResNet-50 (He et al., 2016), EfficientNet-B0 (Tan and Le, 2019), MobileNetV3-Large (Howard et al., 2019), RegNetY-800MF (Radosavovic et al., 2020). The specific generation process of GradCAM heat maps is described in Section 5.2.1. It is worth noting that although the GradCAM heat maps generated by different DNN structures are distinct, the heat maps generated by benign local models under the federated learning framework are similar because they use the same DNN structure.

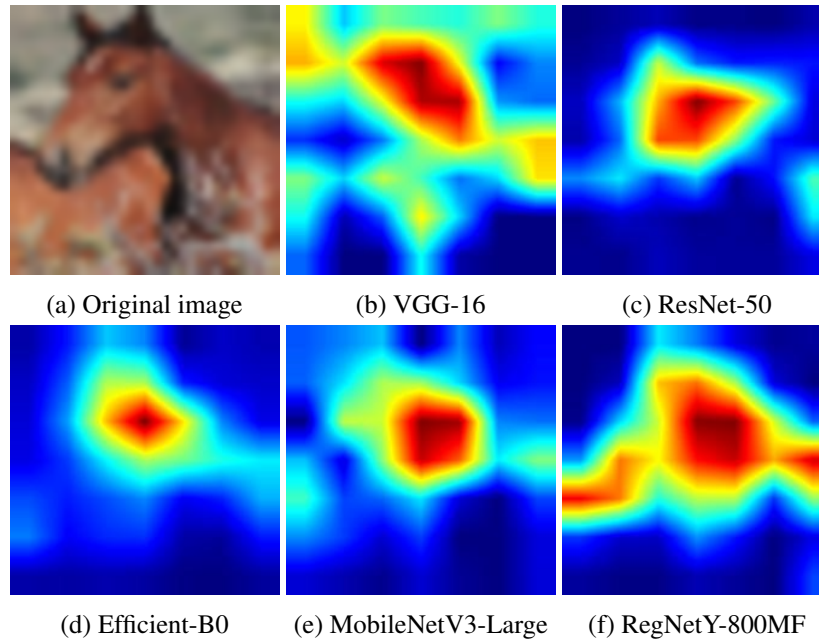


Figure 2.3: (a) An original image, and (b) - (f) the GradCAM heat maps of VGG-16, ResNet-50, EfficientNet-B0, MobileNetV3-Large and RegNetY-800MF.

The principle of GradCAM being utilized to detect malicious local model for FL is as follows: the GradCAM heat maps generated by replacing the pre-trained DNN model parameters with

highly correlated benign local models are highly similar, while the GradCAM heat maps generated by malicious local models are significantly different from those of benign clients. Moreover, the local models of DNN exhibit characteristics of high-dimensional feature spaces, leading to the curse of dimensionality and data sparsity. This sparsity poses challenges for existing Euclidean distance-based defense methods (Awan et al., 2021; Sun et al., 2023) in establishing a suitable margin for separating benign from malicious local model updates. GradCAM can generate heat maps for each uploaded local model, transform them into lower-dimensional, visualizable representations, which subtly addresses the challenge of curse of dimensionality in Euclidean distance-based defense.

2.2 Related Work

The relevant state-of-the-art works FL in MEC can be classified into three categories: i) resource allocation for FL in MEC, ii) poisoning attacks on FL, and iii) the defense strategies against poisoning attacks on FL.

2.2.1 Resource Allocation for FL in MEC

This section presents the literature on resource allocation for FL in MEC.

Assuming that the clients have the same data size, computational resources and wireless channel conditions, FedAvg (McMahan et al., 2017) randomly selects clients to participate FL training and synchronously aggregates local model updates. This process is repeated until a desirable training accuracy is achieved. In (Nishio and Yonetani, 2019), different clients own different computing capabilities and wireless channel conditions, resulting in different local model training time and upload time. The authors develop an FL protocol called FedCS, which allows the server to aggregate as many local model updates as possible to improve image classification accuracy.

To improve the training accuracy of FL systems in the context of wireless channels and energy arrivals of clients, Chu et al. (Chu et al., 2022) modeled the transmission power allocation and client selection of the FL training as a constrained Markov decision process (MDP). Due to a high complexity, stochastic learning methods and Lagrange multipliers are used to simplify the model and to obtain an efficient policy for all clients. In (Anh et al., 2019), the clients with limited battery energy, CPU computations, and bandwidths are considered in a mobile crowd network. A deep Q-learning-based resource allocation for data, energy, and CPU cycles is developed to reduce the energy consumption of FL-based clients and training time of the FL. Given limited computation and communication resources at the clients, (Wang et al., 2019b) analyzes the convergence bound of distributed gradient descent. A control algorithm is developed to determine the tradeoff between local model update and global parameter aggregation. Some clients have limited communication and computing resources and fail to complete training tasks, which leads to many discarded learning rounds affecting the model accuracy (Abdulrahman et al., 2021a). The authors

study the multicriteria-based approach for client selection in FL that reducing the number of communication rounds to reach the intended accuracy and increasing the number of selected clients in each round.

Yoshida et al. (Yoshida et al., 2020) studied a trial-and-error-based client selection based on multi-armed bandit (MAB), where computation tasks, traffic, and link qualities are unknown. The MAB-based client selection balances the client selection according to the selection frequency and clients' resources. In (Xu et al., 2021), the client scheduling in MEC is formulated without the clients' channel and computing information. To reduce the training latency, the client scheduling is formulated as an MAB problem, where ϵ -greedy is used to reduce the learning accuracy. Xia et al. (Xia et al., 2020) developed an MAB-based client scheduling framework to reduce the training latency of FL. Given the known independent and identically distributed (IID) local data at the clients, an FL-based client scheduling algorithm is designed.

Zhu et al. (Zhu et al., 2022) aimed to reduce the average age of data sources by controlling the clients, scheduling the data, and allocating the bandwidth. An actor-critic learning framework is developed, where the clients learn the scheduling strategies based on their local observations. To reduce the training time and energy consumption of clients, Zhan et al. (Zhan et al., 2020a) designed an experience-driven algorithm based on proximal policy optimization and produce sub-optimal results. To improve the network throughput, Kwon et al. (Kwon et al., 2020) presented a cell association and base station allocation method. Multi-agent deep deterministic policy gradient is used to handle unexpected events and unreliable channels in underwater wireless networks. In (Yin et al., 2020), the clients are selected to improve the FL accuracy and reduce the training time and energy consumption. To balance the training accuracy and delay of FL and energy consumption, twin delayed deep deterministic policy gradient algorithm (TD3) is employed to capture the interplay between function approximation error in both policy and value updates, and produce the clients scheduling policy, the CPU frequency allocated for training, and the transmit power allocation.

Moreover, a stateful FL heuristic is studied to schedule clients to improve target accuracy of the FL (Mohammed et al., 2021). The authors in (Cho et al., 2022) analyze the convergence rate of biased client selection. Based on the analysis results that biasing client selection towards clients with higher local loss achieves faster error convergence, an energy-efficient client selection framework is developed to flexibly span the trade-off between convergence speed and solution bias.

In contrast, we focus on a new client selection optimization, which balances the tradeoff between the energy consumption of the clients and the learning accuracy of FL in MEC. The optimization model takes advantage of the a-priori knowledge of the network state information, e.g., data size, bandwidth, and channel gain. Due to the NP-hardness of the optimization, we proposed an energy efficiency-FL accuracy balancing heuristic algorithm (FedAECS) to approximate the optimal client selection policy offline (Zheng et al., 2021). Furthermore, we consider a practical scenario without the prior information about data size, bandwidth, channel gain, and the remaining energy of the clients. Due to a large state and action space, we propose a new FL-DLT3 to balance

FL accuracy and energy consumption of the clients online, where LSTM is leveraged to predict the hidden state of the clients as the input state of TD3 (Zheng et al., 2022). In addition, we also compare the performance of the proposed FL-DLT3 with the FedAECS.

2.2.2 Poisoning Attacks against FL

This section reviews the literature on adversarial attacks against FL, including model poisoning, data poisoning, backdoor and inference attacks.

In order to corrupt the FL, the attacker can launch either a data poisoning or a model poisoning attack. In the data poisoning attack, the attacker injects fake data with manipulated features and flips labels into the benign clients. In the model poisoning attack, the attacker submits malicious local models to the server. Both attacks aim to corrupt the FL by introducing false information.

In (Chen et al., 2021b), an adversarial attack mitigation scheme based on clustering was studied. The scheme aimed to protect FL by using unsupervised weight training to split and merge weight clusters at the server to filter out malicious local model updates that were uploaded by the clients without identity verification. In (Tolpegin et al., 2020), malicious local models were derived from mislabeled data to manipulate the global model. The study found that this attack could result in a significant drop in classification accuracy, and that it was difficult to detect due to its negative impact on the target client and minimal impact on other benign clients.

In (Gao et al., 2023), an inference model was formulated to take local models as input and output the categories of data. A malicious local model based on a differential selection strategy was used to select two adjacent categories. To approximate the benign local model, a category inference attack was studied, in which the attacker learns the data features underlying benign local models. In (Wang et al., 2020b), the authors presented a backdoor attack against FL in MEC, which targeted the tail of the input data distribution at the local clients. The attack used projected gradient descent to maintain the distance between the malicious local model and the global model, to misclassify the targeted samples and bypass defense mechanisms.

In (Zhang et al., 2021), generative adversarial networks (GANs) were utilized to construct data poisoning attacks against FL. The attacker trained the GAN to replicate the local data of the benign clients. Since the attacker had no information about the local data, the GAN-based data poisoning updated the global model to re-select the potential targeted clients. In (Zhang et al., 2019), a GAN-based FL poisoning attack was studied, where the attacker posed as one of the benign clients and trained the GAN to mimic the dataset of the benign clients. The malicious data generated by the attacker were trained to compromise the global model. In (Wang et al., 2019a), a malicious server deployed a GAN-based reconstruction attack against FL to tamper with the private data of the clients. The malicious server discriminated the clients' identities and data representatives to supervise the training of GANs and generate malicious data for each specific client. In (Song et al., 2020), the authors focused on a client-level privacy leakage attack launched by a malicious server. A GAN-based framework was presented to discriminate the data category and client's identity and recover the private data of the client. The attack could associate the data features from different clients to re-identify the local models.

The vulnerability of FL to label inference attacks is presented in (Fu et al., 2022), where a malicious client can infer the private labels of other benign clients. With the observed aggregate model updates, three label inference attacks have been developed to infer private labels with the benign clients, including direct, passive, and active label inference attacks. In (Gong et al., 2022), a coordinated backdoor attack on FL is designed using model-dependent triggers, where an attacker can inject a backdoor trigger into a target model and then train the models in FL to perform coordinated attacks. The trigger uses the model dependency in FL to activate the backdoor when the target client uses the compromised model. A distributed attacking algorithm is also provided to enable the attackers to select their respective backdoor models for a high attack success rate while maintaining a low impact on the overall training accuracy of FL. The authors of (Nuding and Mayer, 2022) focus on data poisoning attacks in both sequential and parallel FL settings. These attacks could weaken the performance of trained models by injecting malicious data into the training datasets used in FL. Sequential FL involves clients training successively, using the output model from the previous client. In contrast, parallel FL involves each client simultaneously training a local model before sending model updates to the server for aggregation. An attacker can later trigger malicious behavior during the prediction phase by modifying specific training inputs using a specific pattern. In (Shejwalkar and Houmansadr, 2021), a model poisoning attack is introduced, which accounts for the characteristics of FL, such as variability in the training data and randomness of the training process. The model poisoning attack uses a transfer learning strategy to improve the attack efficiency.

Unfortunately, the existing data poisoning or model poisoning attacks lack the description of the implicit relationship between different local model updates, which can be exploited by recent graph-based FL privacy protection (Mei et al., 2019; Caldarola et al., 2021). Moreover, the output features of the attacks can be over-smoothed by convolutional layers at the aggregator, making the differences between the malicious local model and the benign ones distinguishable. In contrast, the proposed GAE-based attack is a new attacking method for model poisoning, which is independent of the data. The GAE-based attack manipulates the correlations among multiple data features in selected benign local models while preserving the genuine data features that support those models, thus keeping the malicious local models undetectable.

2.2.3 Defense Models against Poisoning Attacks on FL

This section reviews the literature on defense models against poisoning attacks on FL.

Euclidean distance-based defense. Several extant solutions, predicated upon Euclidean distance, have been devised to detect the poisoning attack within FL contexts. Within the framework, Krum (Blanchard et al., 2017; Zhang et al., 2022b) or Multi-Krum (Blanchard et al., 2017; Zhang et al., 2022b) computes a score for each local model update, the score is the sum of its Euclidean distance from its neighbors. Multi-Krum considers those with high scores as malicious model updates, which are excluded. In stark contrast, Trimmed-mean methodology (Yin et al., 2018) is a coordinate-wise aggregation rule that aggregates each coordinate of the model update separately.

For each coordinate, first sort the values of the corresponding coordinates in the users' model updates. The largest and the smallest k values are removed, then the trimmed mean calculates the average of the remaining values as the corresponding coordinate of the aggregated model update. To defense against poisoning attacks with a large number of malicious clients, FLDetector (Zhang et al., 2022a) is developed to predict a client's model update in each communication round based on historical model updates. The received model update from the client and the predicted model update are inconsistent in multiple communication rounds are flagged as malicious. A defense strategy is developed against poisoning attacks on FL (Cao et al., 2022), where clients are categorized into distinct groups. A global model is trained for each client group according to an existing FL aggregation rule. Based on the global models of all the groups, a majority vote mechanism is used to identify whether a test input is poisoned by the attacker.

Machine learning-based defense. To detect the malicious local clients while generating an accurate model, a statistical mechanism called Auror is introduced (Shen et al., 2016). AUROR is based on the observation that the most important model features from the majority of honest clients will exhibit a similar distribution, while those from malicious clients will exhibit an anomalous distribution. AUROR then uses k -means to cluster uploaded local model updates over training rounds and discards the malicious model updates, i.e., contributions from small clusters that exceed a threshold distance are considered malicious. The authors of (Shi et al., 2022) presented the Federated Anomaly Analytics enhanced Distributed Learning (FAA-DL) framework, which is a lightweight, unsupervised anomaly detection method based on support vector machine (SVM). The main idea of FAA-DL is to utilize an appropriate kernel function and soft margins to estimate a nonlinear decision boundary and separate the benign and malicious local model updates. Detecting network attacks is crucial for preventing and alleviating the consequences of network attacks. To detect and filter malicious local model updates before global model aggregation, the authors of (Liu et al., 2023) developed the FL framework, which is an integrated isolation forest algorithm. The authors argue that the leaf nodes representing the malicious model are closer to the root. The malicious local model updates can be detected by employing deep reinforcement learning methods to adjust the detection threshold dynamically. To resist GAN attacks, the authors of (Chen et al., 2021a) developed a system that isolates local model update parameters of the network from all clients, derailing any chance that the attacker can set up GAN to carry out attacks.

The existing defense mechanisms for malicious local model updates, particularly those predicated on Euclidean distances, notably encounter challenges pertaining to the "curse of dimensionality" within the context of deep neural networks (DNNs), especially considering that local model update parameters encompass millions or potentially billions of parameters (encompassing both weights and biases). In the milieu of high-dimensional spaces, Euclidean distances may inadvertently overestimate the distances between model updates, thereby diminishing the efficacy of detecting malicious model updates. Predominantly, machine-learning-based detection mechanisms significantly hinge upon the meticulous tuning of hyperparameters and the astute setting of thresholds. Recent experimental findings (Han et al., 2022; Zhao et al., 2021) contend that this approach potentially verges on ineffective anomaly detection.

In contrast, the proposed GradCAM-AE pioneers the exploration of GradCAM-based malicious local model detection in FL, developing a methodology that eschews traditional Euclidean distance measurement. The GradCAM-AE defense strategy against poisoning attacks in FL is capable of converting the local model parameters ranging in the millions—into a low-dimensional GradCAM heat map. Additionally, an autoencoder is designed with GradCAM to enhance the separability of the heat maps, thereby increasing the success rate in identifying anomalous heat maps generated by malicious model updates.

2.3 Chapter Summary and Concluding Remarks

In this chapter, we have clarified the scope of this thesis in terms of DRL, GAE and GRadCAM fundamentals which are most relevant for our research. Particularity, the technologies of DRL, GAE, and GradCAM are leveraged in Chapter 3, Chapter 4 and Chapter 5, respectively.

We have presented the literature on resource allocation with FL in MEC, introduced the background of adversarial attacks against FL, discussed the literature on defense models against poisoning attacks on FL. We have also revealed the novelty of our research work.

Note that all related important concepts and technologies have been introduced, we will present the contributions of this dissertation in the following chapters.

Chapter 3

Resource Allocation of FL in MEC

In this chapter, we focus on addressing the **challenge 1** – resource allocation of FL in MEC, as we discussed in Section 1.3. we first propose a client selection optimization to minimize the ratio of energy consumption and learning accuracy of FL in MEC. The optimization problem is formulated by nonlinear integer programming, with consideration of training time and required learning accuracy of FL. We further propose a joint online optimization of client scheduling and transit power allocation for clients to balance the learning accuracy of FL and energy consumption of the clients in MEC. In practice, the instantaneous information of data size, transmit power, and link qualities between the edge server and the clients is unlikely to be known. The optimization is formulated as a partially observable Markov decision process (POMDP), where the network state consists of the source data size of the clients, channel conditions between the edge server and the clients, bandwidth, and the remaining energy of the selected clients. All the entries of a network state are continuous. The action space includes the discrete selection of clients and the continuous transmit power allocation. Given the continuous network states and actions, a large number of clients lead to a large state and action space, and complex network state transitions. This results in difficulty in the joint optimization of client selection and transmits power allocation. Due to a large and continuous state and action space in the formulated POMDP, a new deep reinforcement learning (DRL)-based client selection and transmit power allocation algorithm is proposed to maximize the ratio of the learning accuracy of FL and the energy consumption of the clients. The rest of this chapter is organized as follows. In Section 3.1, we present the common system model in which FL, energy consumption model, training time of FL, FL accuracy and evenness is studied. Section 3.2 presents the client scheduling based on heuristic algorithm, where the problem formulation, the proposed FedAECS and numerical results are given. Section 3.3 dedicates to DRL-based resource allocation with FL in MEC, in which the problem formulation, the proposed FL-DLT3 framework and the evaluations of FL-DLT3 are presented. Finally, Section 3.4 concludes this chapter.

3.1 System Model

We consider a MEC system that consists of one edge server and K number of clients, where $k \in [1, K]$, as shown in Fig. 1.3.

3.1.1 FL Protocol with MEC

Figure 3.1 depicts the FL protocol, where the global model at the server and the local models at the clients are trained in T rounds. Each FL round is composed of resource request, client selection, global model aggregation, global model download, local model update and upload. The server initializes the hyperparameters of the global model, e.g., learning rate, batch size, and the weights of the global model.

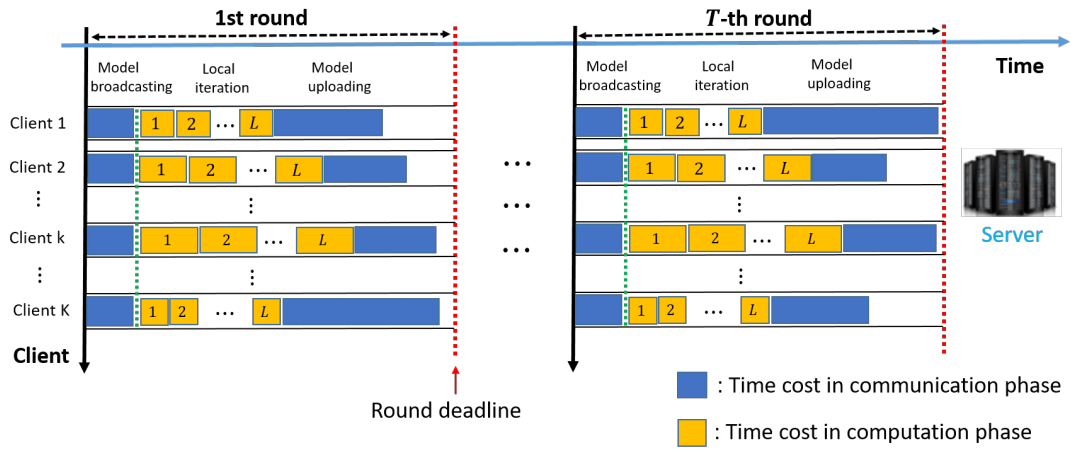


Figure 3.1: The whole training process of FL, in each round, the time consumption of client k includes global model download, local model update and upload.

- **Resource Request:** The clients send to the server the information needed, i.e., data size, for the device selection.
- **Client Selection:** The server selects the clients to upload the local model for the training of the global model, where the details will be presented in the next section.
- **Global Model Aggregation:** The server aggregates the local models of the selected clients to produce the global model.
- **Global Model Download:** All the clients download the the global model from the server.
- **Local Model Update and Upload:** The selected clients individually train their local models according to the FL parameters of the global model.

Let x_{ki} and y_{ki} denote the input (e.g., pixels of an image) and the output (e.g., labels of the image) of the FL model (Qayyum et al., 2022), respectively. The dataset of client k is denoted as

$\mathcal{D}_{t,k} = \{\mathbf{x}_{ki}, y_{ki}\}_{i=1}^{D_{t,k}}$, where $D_{t,k}$ is the size of the dataset of client k in the t -th round and data sample i in client k . Let $f(\mathbf{W}, \mathbf{x}_{ki}, y_{ki})$ denote the loss function of FL, which captures approximation errors over the input $\mathbf{x}_{ki}$ and the output y_{ki} . $\mathbf{W}$ is the weight parameter of the loss function of the neural network being trained according to the FL procedure. Given $D_{t,k}$, the loss function at client k can be specified as

$$F_{t,k}(\mathbf{W}) = \frac{1}{D_{t,k}} \sum_{i=1}^{D_{t,k}} f(\mathbf{W}, \mathbf{x}_{ki}, y_{ki}), \quad (3.1)$$

where $f(\mathbf{W}, \mathbf{x}_{ki}, y_{ki})$ can be specified according to the FL structure. For example, $f(\mathbf{W}, \mathbf{x}_{ki}, y_{ki}) = \frac{1}{2}(\mathbf{x}_{ki}^T \mathbf{W} - y_{ki})^2$ is used to model linear regression, or $f_s(\mathbf{W}, \mathbf{x}_{ki}, y_{ki}) = -y_{ki} \times \log\left(\frac{1}{1+\exp(-\mathbf{W}^T \mathbf{x}_{ki})}\right) - (1 - y_{ki}) \log\left(1 - \frac{1}{1+\exp(-\mathbf{W}^T \mathbf{x}_{ki})}\right)$ accounts for cross entropy function (Shalev-Shwartz and Ben-David, 2014). Since the training of FL aims to minimize the weighted global loss function, we have

$$\min_{\mathbf{W}} F_t(\mathbf{w}) = \sum_{k=1}^K \frac{D_{t,k}}{D_t} F_{t,k}(\mathbf{W}) = \frac{1}{D_t} \sum_{k=1}^K \sum_{i=1}^{D_{t,k}} f(\mathbf{W}, \mathbf{x}_{ki}, y_{ki}), \quad (3.2)$$

where $D_t = \sum_{k=1}^K D_{t,k}$, it represents the total amount of data in the t -th round. The goal of the training process is to find the optimal parameter $\mathbf{W}^*$ such that the global loss function $F(\mathbf{W})$ is minimized, i.e.,

$$\mathbf{W}^* = \arg \min_{\mathbf{W}} F_t(\mathbf{W}). \quad (3.3)$$

To solve problem (3.3) while preserving the data privacy for each client, a canonical gradient-descent technique is widely used in state-of-the-art FL systems (McMahan et al., 2017; Yue et al., 2023; Lycklama et al., 2023). Specially, in the above step of Local Model Update and Upload, each client trains its local model based on its local dataset $\mathcal{D}_k$ for L local training epochs, starting with the global model $\mathbf{W}^t$, i.e.,

$$\mathbf{W}_{k,\ell+1}^t = \mathbf{W}_{k,\ell}^t - \eta_t \nabla F_k(\mathbf{W}_{k,\ell}^t, \mathcal{D}_{t,K}), \ell \in \{1, \dots, L\}, \quad (3.4)$$

where η_t denotes the learning rate in the t -th communication round, and $\nabla F_k(\mathbf{W}_{k,\ell}^t, \mathcal{D}_k)$ defines the local gradient estimate over $\mathcal{D}_k$ at local training epoch ℓ . After L local training epochs, each client k uploads its updated local model $\mathbf{W}_k^t = \mathbf{W}_{k,L}^t$ to the server.

In the above step of Global Model Aggregation, the server aggregates the K clients' updated local models $\{\mathbf{W}_k^t\}_{k=1}^K$ to obtain the updated global model, given by

$$\mathbf{W}^t = \sum_{k \in \mathcal{K}} \frac{D_k}{D} \mathbf{W}_k^t, \quad (3.5)$$

where $\mathbf{W}^t$ is sent back to all K clients.

3.1.2 Energy Consumption Model

The energy consumption of the clients accounts for the local training of the dataset and the transmissions of the local model. We assume that training a data sample at the client requires c_k CPU cycles per bit. Given the data size of $D_{t,k}$, the number of CPU cycles for the local model training is $c_k D_{t,k}$. We denote f_k as the computation capacity of client k , which is measured in CPU cycles per second. According to (Dinh et al., 2021), the computation time of training the local model at client k in the each t -th round, we have

$$\tau_{t,k}^{train} = \frac{c_k D_{t,k} L}{f_k}, \quad (3.6)$$

where $L_{t,k}$ is the number of local iterations of FL training.

According to (Mao et al., 2016; Yang et al., 2021; Zheng et al., 2023b), the energy consumption on $c_k D_{t,k}$ CPU cycles at client k is

$$E_{t,k1}^{cmp} = \zeta_k c_k D_{t,k} f_k^2, \quad (3.7)$$

where ζ_k is the effective capacitance coefficient of computing chipset for client k . To compute the local model, client k needs to compute $c_k D_{t,k}$ CPU cycles. Thus, the total computation energy at client k in the t -th round can be given as

$$E_{t,k}^{cmp} = L_{t,k} E_{t,k1}^{cmp} = L_{t,k} \zeta_k c_k D_{t,k} f_k^2. \quad (3.8)$$

Furthermore, the achievable uplink transmit rate of client k is given by

$$r_{t,k}^{up} = b_{t,k} \log_2 \left(1 + \frac{P_{t,k} G_{t,k}}{N_0 b_{t,k}} \right), \quad (3.9)$$

where $b_{t,k}$ is the bandwidth allocated to client k by the server in the t -th round, $P_{t,k}$ is the power consumption of data transmit for client k , $G_{t,k}$ is the uplink channel gain between client k and the server, and N_0 is the power spectral density of the Gaussian noise.

The achievable downlink transmit rate of client k is

$$r_{t,k}^{down} = b_{t,k} \log_2 \left(1 + \frac{P_t^s H_{t,k}}{N_0 b_{t,k}} \right), \quad (3.10)$$

where P_t^s denotes the transmit power of the server, and $H_{t,k}$ is the downlink wireless channel gain from the server to client k .

The downloading time of the global model at client k in the t -th round is

$$\tau_{t,k}^{down} = \frac{\mathfrak{S}_d}{r_{t,k}^{down}}, \quad (3.11)$$

where $\mathfrak{S}_d$ denotes the size of the global model. The transmission time of the local model at the t -th round can be given by

$$\tau_{t,k}^{up} = \frac{\mathfrak{S}_u}{r_{t,k}^{up}}, \quad (3.12)$$

where $\mathfrak{S}_u$ is the size of the local model. By substituting (3.9) to (3.12), the energy consumption of a client on the local model transmission is

$$E_{t,k}^{up} = P_{t,k} \tau_{t,k}^{up} = \frac{P_{t,k} \mathfrak{S}_u}{b_{t,k} \log_2(1 + \frac{P_{t,k} G_{t,k}}{N_0 b_{t,k}})}. \quad (3.13)$$

The total energy consumption $E_{t,k}^c$ of the selected client k in the t -th round is

$$E_{t,k}^c = E_{t,k}^{cmp} + E_{t,k}^{up}. \quad (3.14)$$

The remaining battery energy of the selected client k in the t -th round is

$$E_{t,k} = E_{t-1,k} - E_{t-1,k}^c + \Delta E_{t,k}, \quad (3.15)$$

where $\Delta E_{t,k}$ is the amount of harvested energy (Li et al., 2019b).

3.1.3 Training Time of FL

We denote the total completion time $\tau_{t,k}$ of client k , which is:

$$\tau_{t,k} = U_{t,k}(\tau_{t,k}^{train} + \tau_{t,k}^{up} + \tau_{t,k}^{down}) \quad (3.16)$$

where $U_{t,k}$ is the number of global iterations of client k in the t -th round increases with the local accuracy and guarantees the FL algorithm converge (Yang et al., 2021). Note that the training time at the server is neglected since the server typically supports more powerful CPUs than the clients.

3.1.4 FL Accuracy

Let $\beta_{t,k}$ be a binary indicator. If client k is selected by the server in the t -th round, $\beta_{t,k} = 1$; otherwise, $\beta_{t,k} = 0$. We define the accuracy of FL as the the fraction of predictions FL model got right. According to (Zhan et al., 2020b; Lim et al., 2021; Khan et al., 2020a), the accuracy of FL, denoted by $\Gamma(\beta_{t,k})$, can be simplified as,

$$\Gamma(\beta_{t,k}) = \log(1 + \sum_{k=1}^K \mu_k \beta_{t,k} D_{t,k}) \quad \forall t \in \mathcal{T}, \quad (3.17)$$

where $\mu_k > 0$ is a system parameter (Lim et al., 2021).

3.1.5 Evenness

To improve the evenness of data size of the selected clients, similar to (Lyu et al., 2019), we define the expectation of the difference between the total amount of data for all clients and the amount of

data for the selected clients at the t -th training round, and normalize the expectation, we have

$$\Delta_t = \frac{\mathbb{E} \left[v \sum_{k=1}^K D_{t,k} - \beta_{t,k} D_{t,k} \right]}{\sum_{k=1}^K D_{t,k}} \quad \forall t \in \mathcal{T}, \quad (3.18)$$

where $v \in (0, 1]$ is a constant value, the arrival data $D_{t,k}$ of client k follows uniform distribution or normal distribution. The objective function can be determined as $\frac{\Gamma(\beta_{t,k})}{\sum_{k=1}^K \beta_{t,k} E_{t,k}^c} - \Delta_t$, which aims to balance the learning accuracy of FL and energy consumption of the client and improve the evenness of data size of the selected clients.

3.2 Client Scheduling Based on Heuristic Algorithm

In case of the resource information of clients is known to the server in prior, client scheduling based on heuristic algorithm is investigated.

3.2.1 Problem Formulation

In this section, we formulate the optimization of balancing overall energy consumption of the clients and learning accuracy of FL:

$$\min_{\beta_{t,k}} \sum_{t=1}^T \frac{\sum_{k=1}^K \beta_{t,k} E_{t,k}^c}{\Gamma(\beta_{t,k})} \quad (3.19)$$

$$\Gamma(\beta_{t,k}) \geq \varepsilon_0, \quad (\varepsilon_0 \in (0, 1]). \quad (3.20)$$

$$\sum_{k=1}^K \beta_{t,k} b_{t,k} \leq B, \quad (t \in [1, T], k \in [1, K]). \quad (3.21)$$

$$1 \leq \sum_{k=1}^K \beta_{t,k} \leq K, \quad (t \in [1, T], k \in [1, K]). \quad (3.22)$$

$$\beta_{t,k} \tau_{t,k} \leq T_{max}, \quad (t \in [1, T], k \in [1, K]). \quad (3.23)$$

$$P_k^{min} \leq P_{t,k} \leq P_k^{max}, \quad (t \in [1, T], k \in [1, K]). \quad (3.24)$$

$$\beta_{t,k} \in \{0, 1\}, \quad (t \in [1, T], k \in [1, K]). \quad (3.25)$$

Specifically,

- Constraint $(\Gamma(\beta_{t,k}) \geq \varepsilon_0)$ specifies the minimum requirement of the FL accuracy in t -th round, where $\varepsilon_0 \in (0, 1]$ defines the lower bound threshold.

- Constraint ($\sum_{k=1}^K \beta_{t,k} b_{t,k} \leq B$) guarantees that the total bandwidth of the selected clients is smaller than the bandwidth capacity B .
- Constraint ($1 \leq \sum_{k=1}^K \beta_{t,k} \leq K$) describes that at least one client is selected for FL.
- Constraint ($\beta_{t,k} \tau_{t,k} \leq T_{max}$) ensures that the download, computation and transmit delay of the selected device has to be less than the duration of the round t_d . As shown in Figure 3.1, the each t -th round contains the downloading time of the global model, and local computation and transmission time of the local model. Note that the client selection time and training time of the global model at the edge server can be neglected since the edge server supports more powerful CPUs than the client.
- Constraint ($P_k^{min} \leq P_{t,k} \leq P_k^{max}$) indicates the upper and lower bounds of clients' transmit power.

3.2.2 FedAECS

The proposed optimization problem in (3.19) — (3.25) is a typical 0-1 Multidimensional Knapsack Problem (MKP) (Puchinger et al., 2010). Specifically, the items to be put in knapsacks are the clients with energy consumption $E_{t,k}$, data size $D_{t,k}$, and bandwidth $b_{t,k}$. The capacity of the knapsack is equal to total bandwidth, the variable $\beta_{t,k}$ is a binary indicator of item (client) k , $\beta_{t,k}$ is set to 1 indicates that item k is selected. Otherwise, $\beta_{t,k}$ is set to 0. The total weight of the knapsack has a lower bound which is equal to the minimum requirement of accuracy constraint (3.20). In addition, it is note worthy that every item to be put into the backpack must obey the time constraint (3.23). Our goal is to select which clients for the training of FL to minimize the ratio of the energy consumption and learning the accuracy of the FL. Therefore, the proposed optimization is NP-complete.

For each communication round t , let $\Phi_t(\boldsymbol{\beta}_t) = \frac{\sum_{k=1}^K \beta_{t,k} E_{t,k}}{\Gamma(\boldsymbol{\beta}_{t,k})}$, where $\boldsymbol{\beta}_t$ is a vector of clients selection. To balance the energy consumption and learning accuracy, a heuristic, named FL for accuracy-energy based client selection (FedAECS) algorithm is proposed, which is presented in Algorithm 1. Specifically, the clients that fulfill time constraint (3.23) and accuracy constraint (3.20) are selected and sorted into a list η_t non-descendingly according to the ratio of the energy consumption and the FL accuracy; The unselected clients are saved into a vector $\boldsymbol{\beta}_t'$; see Steps 3 – 12 in Algorithm 1. FedAECS algorithm returns once a client can satisfy the accuracy constraint (3.20) of the optimization; see Steps 14 – 17 in Algorithm 1. Otherwise, FedAECS continues to traverses the list η_t until the client $\beta_{t,m}$ satisfies the accuracy constraint (3.20), see Steps 19 – 40 in Algorithm 1. The corresponding objective function value $\Phi_t(\boldsymbol{\beta}_t^p)$ will be determined; meanwhile, FedAECS further explore the possible combinations of the client $\beta_{t,m}$ to find the optimal combination $\hat{\boldsymbol{\beta}}_t$, the corresponding objective function value $\Phi_t(\hat{\boldsymbol{\beta}}_t)$ will be calculated, finally, the optimal objective function value $\Phi_t(\boldsymbol{\beta}_t^*)$ will be chosen from $\Phi_t(\boldsymbol{\beta}_t^p)$ and $\Phi_t(\hat{\boldsymbol{\beta}}_t)$.

Algorithm 1: FedAECS—Federated Learning for Accuracy-energy Based Client Selection Algorithm

```

1 Input:  $f_k, D_{t,k}, P_{t,k}, b_{t,k}$ ;
2 Initialization;
3 for ( $k \in [1, K]$ ) do
4   if ( (3.21) and (3.23) hold) then
5     Calculate  $E_{t,k}^c$  according to (3.14);
6     Calculate  $\Gamma(\beta_{t,k})$  according to (3.17);
7     Calculate  $\{\frac{E_{t,k}^c}{\Gamma(\beta_{t,k})}\}$  and store it into  $\eta_t$  one by one;
8   else
9     Store client  $k$  into  $\beta'_t$ ;
10  end
11 end
12  $\eta_t$  are sorted in non-descending order and  $m = 1$ ;
13 while ( $m \leq |\eta_t|$ ) do
14   if ((3.20) holds for client  $m$ ) then
15      $\beta_t^* = (\beta_{t,m} = 1, \bigcup_{j \in |\eta_t| \setminus m} \beta_{t,j} = 0) \cup \beta'_t$ ;
16      $\Phi_t(\beta_t^*) = \eta_{t,m}$ ;
17     Break;
18   else
19     if ( $m \leq |\eta_t| - 1$ ) then
20        $m = m + 1$ ;
21       while ( $m \geq 2$  and  $m \leq |\eta_t|$ ) do
22         if (Client  $m$  satisfies (3.20)) then
23            $\beta_t^p = (\beta_{t,m} = 1, \bigcup_{j \in |\eta_t| \setminus m} \beta_{t,j} = 0)$ ;
24            $\Phi_t(\beta_t^p) = \eta_{t,m}$ ;
25           if ( $(\beta_{t,1}, \dots, \beta_{t,m-1}, \beta_{t,m} \equiv 1)$  satisfy (3.20) and (3.21)) then
26              $\hat{\beta}_t = \arg \min \Phi_t(\beta_{t,1}, \dots, \beta_{t,m-1}, \beta_{t,m} \equiv 1)$ ;
27           else
28             Break;
29           end
30            $\Phi_t(\beta_t^*) = \min\{\Phi_t(\beta_t^p), \Phi_t(\hat{\beta}_t)\}$ ;
31           if  $\Phi_t(\beta_t^*) = \Phi_t(\beta_t^p)$  then
32              $\beta_t^* = \beta_t^p \cup \beta'_t$ ;
33           else
34              $\beta_t^* = \hat{\beta}_t \cup \beta'_t$ ;
35           end
36         else
37            $m = m + 1$ ;
38         end
39       end
40     end
41   end
42   Output:  $\beta_t^*, \Phi_t(\beta_t^*)$ ;
43 end

```

3.2.3 Numerical Results

In this section, we evaluate the proposed FedAECS algorithm. We compare the performance with the optimal solution as well as the existing solutions in the literature. We also study the impact of the number of clients and T_{max} on the runtime of FedAECS.

3.2.3.1 Simulation Configuration

For our simulations, we set $I = 1000$. The number of clients $K = \{20, 40, 60, 80, 100\}$, correspondingly, the total bandwidth $B = \{1, 3, 5, 7, 9\}$ MHz. For each client, $D_{t,k}$, b_k , f_k , $P_{t,k}$, c_k are uniformly distributed in $[2, 10]$ MB, $[50, 150]$ KHz, $[2, 4]$ Hz, $[4, 10]$ dBm, $[1, 3]$ cycles/bit respectively. A transmit data size $S_k = 100$ kbits. The number of local iterations $L_{t,k} = 10$ and and global iterations $U_{t,k} = 4$. In addition, the noise power spectral density is $N_0 = -174$ dBm/Hz. The effective switched capacitance in local computation is $\zeta_k = 10^{-28}$ and the the system parameter $\mu = 1.7 \times 10^{-8}$. The proposed FedAECS is implemented in MATLAB R2015a, running on 3.0 GHz Intel core processor with 24 GB of memory.

We compare FedAECS with the following four client selection schemes, i.e., FedAvg (McMahan et al., 2017), FedCS (Nishio and Yonetani, 2019), Dataset-based and Energy-based policies.

FedAvg: the clients are randomly selected participating in FL training, while satisfying the bandwidth constraint.

FedCS: the FL server selects the clients as long as the bandwidth capacity constraint holds.

Energy-based client selection: the clients are prioritized by energy consumption, FL server selects a few number of clients while satisfying the bandwidth constraint.

Data-based client selection: the clients are prioritized by data size, FL server selects a few number of clients while satisfying the bandwidth constraint.

3.2.3.2 Simulation Results and Analysis

Comparing to optimal client selections. Table 3.1 summarizes running time, the ratio value of $\Phi_i(\beta_i)$ and Error(= $\frac{|\Phi_i \text{ of FedAECS} - \Phi_i \text{ of CPLEX}|}{\Phi_i \text{ of CPLEX}} \times 100\%$). We assess the ratio value of FedAECS when the number of clients are increased from 2 to 10. Meanwhile, we conducted in ILOG CPLEX 12.10, and obtained optimal results, which are compared with the results that obtained by FedAECS. It is found that the optimal solution have the maximum difference which is 58.699% when $K = 10$. On average, FedAECS guarantees exactly the client scheduling schemes as optimal schedules. Moreover, FedAECS is much more efficient than CPLEX on runtime.

Figure 3.2 demonstrates the advantage of FedAECS with respect to the objective function value $\Phi_i(\beta_i)$. Considering the case of twenty clients and one server. For wireless communication model, the total bandwidth $B = 1$ MHz, in addition, the threshold value of delay $T_{max} = 5$ s. As an illustration, it is evident that $\Phi_i(\beta_i)$ of FedAECS client selection scheme keeps it to a minimum at all times with the change of communication round. In particular, $\Phi_i(\beta_i)$ of FedCS-based client selection is about 5-folds as compared with FedAECS. The principal reason is that FedCS-based method can obtain more clients, whilst it incurs a substantial of energy consumption.

Table 3.1: Comparison results with Cplex.

# Clients	Cplex(Φ_i)	Runtime	FedAECS(Φ_i)	Runtime	Error
2	5.2786 J	19 s	5.3242 J	0.061026 s	0.864%
3	12.156 J	27 s	12.1782 J	0.034008 s	0.183%
4	14.478 J	44 s	14.5272 J	0.067833 s	0.340%
5	6.3921 J	21 s	8.8479 J	0.095634 s	38.419%
6	6.0202 J	27 s	6.0669 J	0.097590 s	0.776%
7	4.4662 J	19 s	4.4879 J	0.082551 s	0.486%
8	5.8882 J	21 s	5.9214 J	0.063147 s	0.564%
9	4.8528 J	21 s	4.9042 J	0.039516 s	1.059%
10	3.1643 J	33 s	5.0217 J	0.071169 s	58.699%

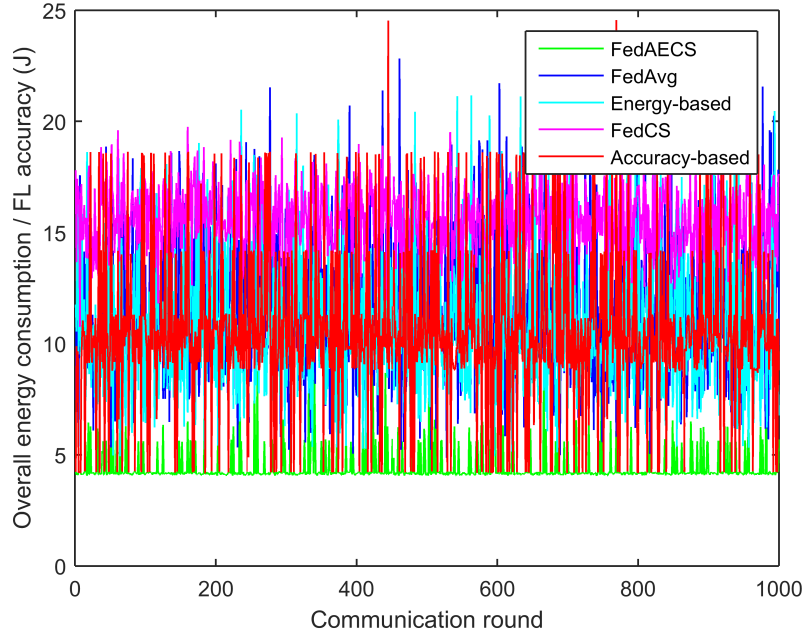
Figure 3.2: The objective function value $\Phi_i(\beta_i)$ varying with communication rounds.

Figure 3.3 shows that the objective function value $\Phi_i(\beta_i)$ varies with the total number of clients $I = \{20, 40, 60, 80, 100\}$, in response, total bandwidth is also increasing with the number of clients, that is $B = \{1, 3, 5, 7, 9\}$ MHz. We calculate the mean value and standard deviation of ratio for 1000 epochs. Obviously, $\Phi_i(\beta_i)$ of FedAECS always keep the minimum, while $\Phi_i(\beta_i)$ of FedCS increasing with the more number of participating clients and more energy consumption. Moreover, with the participating clients increasing, the increasing of energy consumption is more significant than the increasing of FL model accuracy. In contrast, the ratio values of the others client selection methods not significantly affected by the number of clients.

Figure 3.4 shows that the relationship among objective function value $\Phi_i(\beta_i)$, the number of clients and length of the epoch T_{max} . With the increasing of the total number of clients and

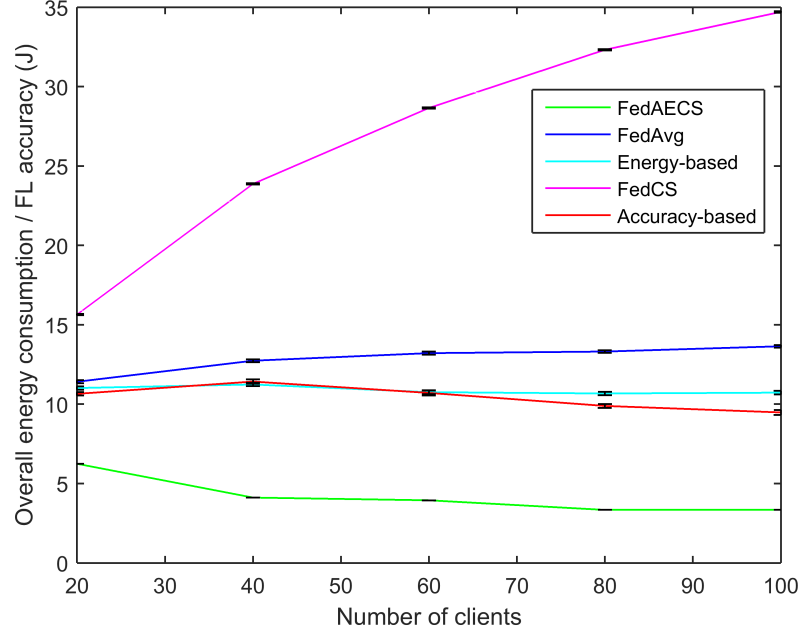


Figure 3.3: The objective function value $\Phi_t(\beta_t)$ varying with the number of clients.

T_{max} , the more potential clients will be selected to participate in the process of model training. However, When T_{max} increases to 6 seconds, $\Phi_t(\beta_t)$ will not change significantly with the increase of the number of users. This is mainly because the number of clients available no longer changes significantly.

Figure 3.5 shows that the relationship among objective function value $\Phi_t(\beta_t)$, T_{max} and lower bound of accuracy ϵ_0 . It can be clearly seen that $\Phi_t(\beta_t)$ decreasing with the increasing of T_{max} and the decreasing of ϵ_0 . Note that when $T_{max} = 1$ s and $\epsilon_0 = 0.07$ or $\epsilon_0 = 0.09$ the ratio values of $\Phi_t(\beta_t)$ are non-exist because the constraints are too strict to obtain available clients.

Figure 3.6 illustrates that the runtime changes with length of the epoch T_{max} and the number of clients changes. The runtime is first decline and then moderate rise with the increase of T_{max} when the number of clients equal to 60, 80, 100. In particular, when T_{max} is greater than 3 seconds, for higher T_{max} , a longer run-time is required. On the other hand, the number of clients equal to 20 and 40, the runtime increases correspondingly from $T_{max} = 1$ s to $T_{max} = 2$ s in that more available candidate clients, and afterwards follows the same law as the number of clients equal to 60, 80, 100.

3.3 DRL-based Resource Allocation with FL in MEC

3.3.1 Problem Formulation

In this section, we further study the client selection and transmit power allocation to maximize the ratio of the learning accuracy of FL to the energy consumption of clients. We formulate the

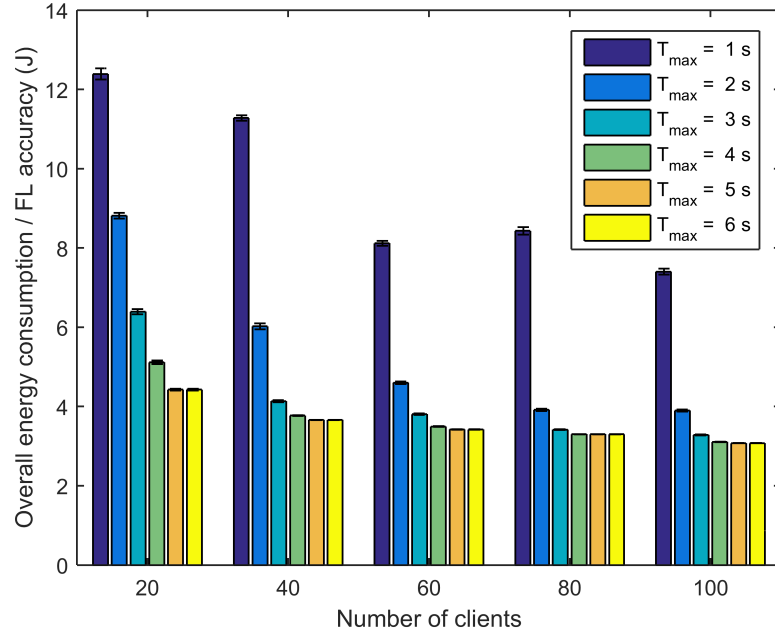


Figure 3.4: The relationship among objective function value $\Phi_t(\beta_t)$, the number of clients, and length of the communication round T_{max} .

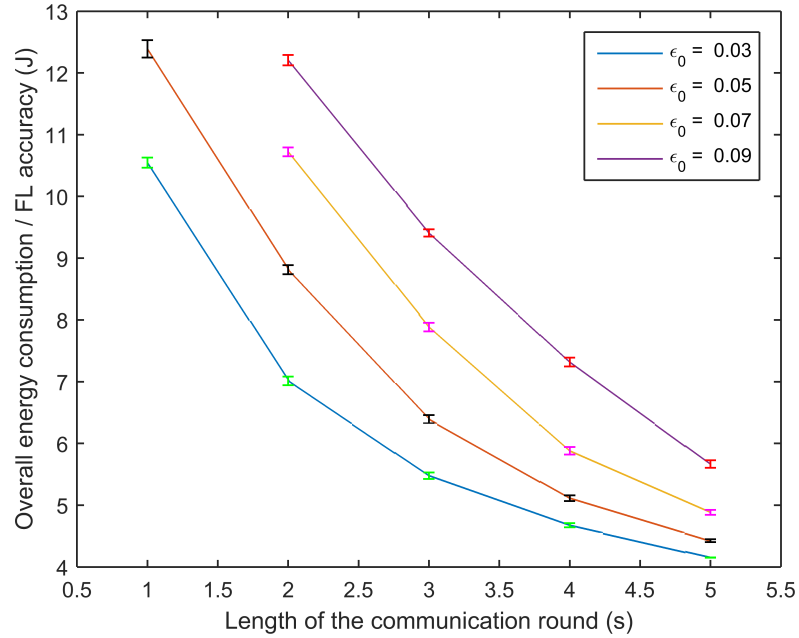


Figure 3.5: The relationship among objective function value $\Phi_t(\beta_t)$, length of the epoch T_{max} , and model accuracy of lower bound value ϵ_0 .

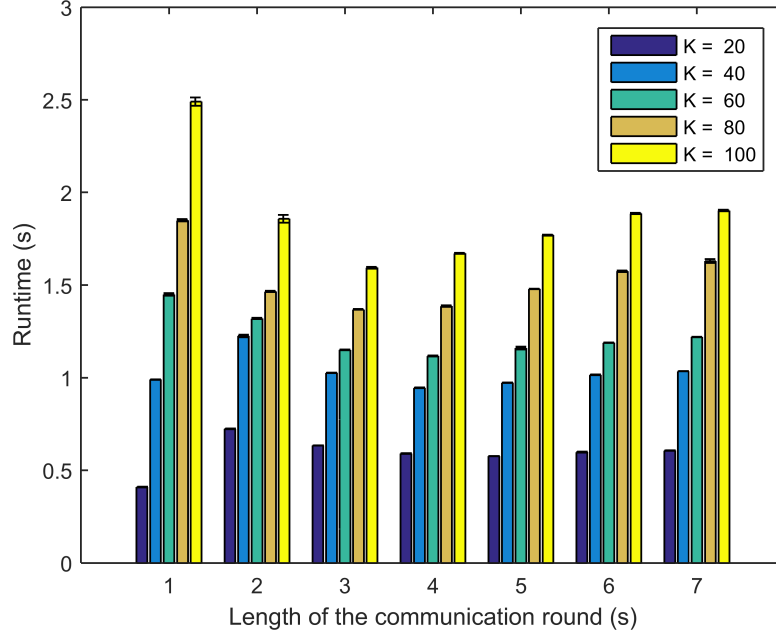


Figure 3.6: The relationship among runtime, length of the epoch T_{max} , and the number of clients.

optimization as **P1**.

$$\begin{aligned} \mathbf{P1}: \quad & \max_{\beta_{t,k}, P_{t,k}} \sum_{t=1}^T \left[\frac{\Gamma(\beta_{t,k})}{\sum_{k=1}^K \beta_{t,k} E_{t,k}^c} - \Delta_t \right] \\ \text{s.t.} : \quad & \beta_{t,k} E_{t,k}^c \leq E_{t,k}, \quad (t \in [1, T], k \in [1, K]). \end{aligned} \quad (3.26)$$

$$\Gamma(\beta_{t,k}) \geq \varepsilon_0, \quad (\varepsilon_0 \in (0, 1]). \quad (3.27)$$

$$\sum_{k=1}^K \beta_{t,k} b_{t,k} \leq B, \quad (t \in [1, T], k \in [1, K]). \quad (3.28)$$

$$1 \leq \sum_{k=1}^K \beta_{t,k} \leq K, \quad (t \in [1, T], k \in [1, K]). \quad (3.29)$$

$$\beta_{t,k} \tau_{t,k} \leq T_{max}, \quad (t \in [1, T], k \in [1, K]). \quad (3.30)$$

$$P_k^{min} \leq P_{t,k} \leq P_k^{max}, \quad (t \in [1, T], k \in [1, K]). \quad (3.31)$$

$$\beta_{t,k} \in \{0, 1\}, \quad (t \in [1, T], k \in [1, K]). \quad (3.32)$$

Constraint (3.26) guarantees that the selected client has sufficient energy to complete the local

model training in t -th round. Constraints (3.27)–(3.32) have the same meanings with (3.20)–(3.25), respectively.

Problem **P1** involves nonlinear problems with continuous and integer variables (Sahinidis, 2019). The typical 0-1 Multidimensional Knapsack Problem (MKP) (Puchinger et al., 2010) is a special case of problem **P1**. Assume that transmit power is a constant. In this case, the only variable is the selected client. The items to be put in the knapsack are the clients with energy consumption $E_{t,k}$, data size $D_{t,k}$, and bandwidth $b_{t,k}$. The capacity of the knapsack is equal to the total bandwidth, the optimization variable $\beta_{t,k}$ is a binary indicator of item (client) k selection. $\beta_{t,k}$ is set to 1 to indicate that item k is selected. Otherwise, $\beta_{t,k}$ is set to 0. The total reliability of the knapsack has lower bounds which are equal to the minimum requirement of accuracy constraint (3.27). Note that every item to be put into the knapsack must obey the time constraint (3.30). Coupled with the energy consumption of clients on the transmission is a non-linear function of the transmit power, which has a continuous variable. Therefore, the proposed optimization is NP-hard. It is also mentioning that problem **P1** involves a long time horizon with random and unpredictable data and energy arrivals. This leads to an intractable large state space of the problem require online optimization of selection and allocation decisions.

3.3.2 DRL-based Clients Selection and Transmit Power Allocation

3.3.2.1 POMDP Formulation for FL-based client Selection and Resource Allocation

The considered resource allocation can be formulated as a POMDP that is a generalization of a Markov decision process (MDP) with only partially observable states (Kaelbling et al., 1998). A POMDP can be represented by a 6-tuple (S, A, P, R, O, Ω) , where S is the state space, A is the action space, P is the transition probability, R is the reward function, O is the observation space and Ω is the observation model. The edge server cannot observe the underlying state. Instead, an observation $S_{\alpha'}^o \in O$ is received after a state transition to the next state $S_{\alpha'}$ with the probability $\Omega(S_{\alpha'}^o | S_{\alpha'})$.

State and Action Space: According to problem **P1**, the state space of the POMDP consists of data size, uplink channel gains, downlink channel gains, bandwidth and remaining energy of the clients. The network state S_{α} is defined as

$$S_{\alpha} = \{(D_{\alpha,k}, G_{\alpha,k}, H_{\alpha,k}, b_{\alpha,k}, E_{\alpha,k}), k = 1, \dots, K\}. \quad (3.33)$$

The action of the POMDP is the selection of the clients for FL, denoted by $\beta_{\alpha,k}$, and the transmit powers of the selected clients, $P_{\alpha,k}$, as given by

$$\mathcal{A} \in \{(\beta_{\alpha,k}, P_{\alpha,k}), k = 1, \dots, K\}, \quad (3.34)$$

where $\beta_{\alpha,k} \in \{0, 1\}$ and $P_{\alpha,k} \in [P_k^{\min}, P_k^{\max}]$.

Observation Space: At each state S_{α} , the edge server can observe partially the network state from the selected clients, where the state observation S_{α}^o can be packed in its uploaded local model.

Particularly, the state observation $S_\alpha^o \in S_\alpha$ is given by

$$S_\alpha^o = \{(D_{\alpha,k}, G_{\alpha,k}, H_{\alpha,k}, b_{\alpha,k}, E_{\alpha,k})_o, k = 1, \dots, K\}. \quad (3.35)$$

Note that the state of unselected clients cannot be observed by the edge server.

Reward: Let $R(S_{\alpha'}^o | S_\alpha^o, A_\alpha)$ denote the immediate reward received when the action $A_\alpha \in \mathcal{A}$ is taken at state S_α . The reward is defined to consist of the AE gain that is the ratio of the FL accuracy to the energy consumption of the selected clients, and the penalty resulting from the data unevenness among the selected clients, i.e.,

$$R(S_{\alpha'}^o | S_\alpha^o, A_\alpha) = \frac{\Gamma(\beta_{\alpha,k})}{\sum_{k=1}^K \beta_{\alpha,k} E_{\alpha,k}^c} - \Delta_\alpha, \quad (3.36)$$

where $R(S_{\alpha'}^o | S_\alpha^o, A_\alpha)$ indicates that the state observation transits to subsequent $S_{\alpha'}^o$ from the current state observation S_α^o . For illustration convenience, we use R_α to denote the reward in the following sections.

To evaluate the action selected by a policy π_θ with parameters θ , where π_θ is a mapping from state observations to actions, and the set of all policies is defined as Π . We aim to maximize the expected total reward denoted as action-value function $Q_{\pi_\theta}(S_\alpha^o, A_\alpha)$,

$$Q_{\pi_\theta}(S_\alpha^o, A_\alpha) = \max_{\pi \in \Pi} \mathbb{E}_{S_\alpha^o}^{\pi_\theta} \left\{ \sum_{n=0}^{\infty} \gamma^n R_\alpha \right\}, \quad (3.37)$$

where $\gamma \in [0, 1]$ is a discount factor for future state observations. $\mathbb{E}_{S_\alpha^o}^{\pi_\theta} \{\cdot\}$ takes the expectation with respect to policy π_θ and state observation S_α^o . According to the Bellman equation (?), the optimal action-value function (3.37) of a state-action pair (S_α^o, A_α) and the value of the subsequent state-action pair $(S_{\alpha'}^o, A_{\alpha'})$ can be further rewritten as

$$Q_{\pi_\theta}(S_\alpha^o, A_\alpha) = \max_{\pi_\theta \in \Pi} \mathbb{E}_{S_\alpha^o}^{\pi_\theta} \left\{ R_\alpha + \gamma Q_{\pi_\theta}(S_{\alpha'}^o, A_{\alpha'}) \right\}. \quad (3.38)$$

The optimal action, namely, A_α^* , which satisfies (3.38), can be given by

$$A_\alpha^* = \arg \max_{\pi_\theta \in \Pi} \mathbb{E}_{S_\alpha^o}^{\pi_\theta} \left\{ R_\alpha + \gamma Q_{\pi_\theta}(S_{\alpha'}^o, A_{\alpha'}) \right\}, \quad (3.39)$$

where A_α^* provides the maximized AE gain.

Given a practical scenario where the edge server has no prior knowledge on the transition probabilities, we propose a FL-DLT3 framework that joint client selection and transmit power allocation algorithm that utilizes TD3, one of the deep reinforcement learning techniques, to maximize the AE gain.

The state and action space increases dramatically with the number of devices, and the action space has both continuous and discrete actions. Moreover, the network state, in practice, is not always observable at the edge server because the data size and remaining energy information of

clients are always kept locally before the server takes the devices selection. In view of these challenges, it is difficult to find the exact solution of **P1**. To this end, we develop FL-DLT3 framework to find the near optimal solution of problem **P1**.

In general, FL-DLT3 consists of the TD3-based deep reinforcement learning and the LSTM-based state characterization layer, as depicted in Figure 3.7. FL-DLT3 leverages the actor-critic neural network structure to develop the TD3-based deep reinforcement learning (Fujimoto et al., 2018). The TD3 at the edge server is trained to optimize client selection and transmit power allocation in a continuous action space, where the edge server has no priori information on the state transition probabilities. The LSTM is used to predict the hidden state of the clients as input state of TD3. The AE gain is maximized over the large continuous state and action spaces. Specially, the global model is trained iteratively to improve the learning accuracy of the local model. With the growth of FL iterations, FL-DLT3 optimally selects the clients to maximize the accuracy of FL while ensuring the energy consumption requirement.

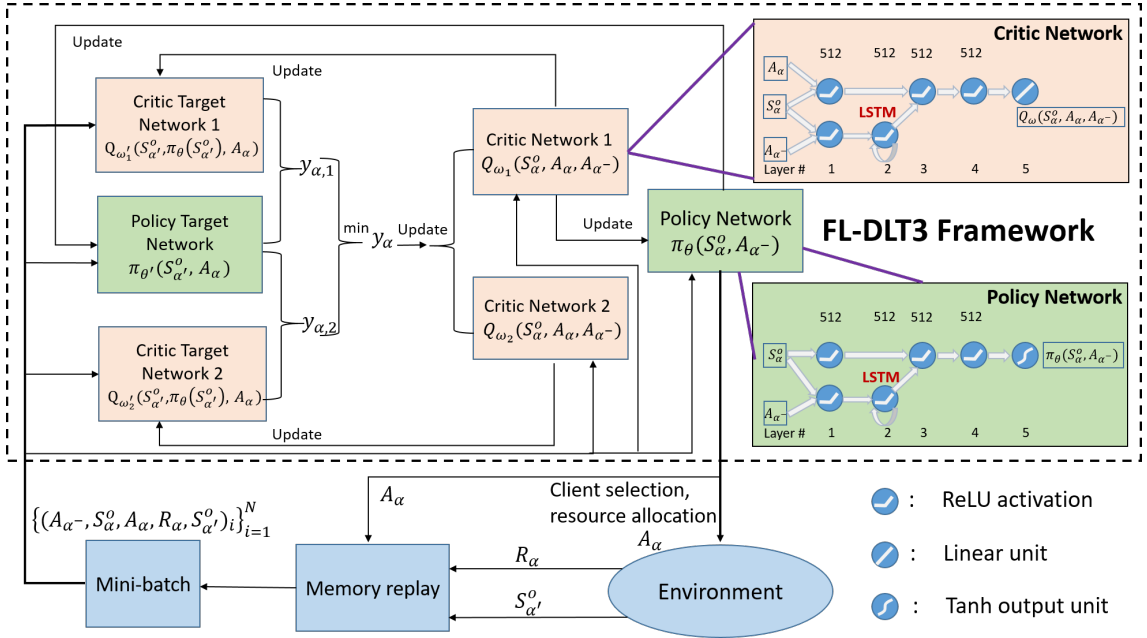


Figure 3.7: An illustration of the proposed FL-DLT3 framework, where a policy network, a policy target network, two critic networks and two critic target networks are trained. Each of the networks consists of a feedforward branch and recurrent branch. A training round of FL is performed by the selected clients with the allocated transmit power. Then, the edge server gets the reward R_α and the state update S_α^o of the clients. The transition $\{(A_\alpha^-, S_\alpha^o, A_\alpha, R_\alpha, S_\alpha^o)\}$ is stored into the replay buffer $\mathcal{B}$, and a mini-batch of transitions is randomly sampled from $\mathcal{B}$ to train the policy and critic networks on the edge server.

3.3.2.2 TD3 on the Server

TD3 utilizes the deterministic policy gradient algorithm (DPG) (Silver et al., 2014) to optimally update the current policy by deterministically mapping network states to a specific action of the edge server. The critic is used to approximate the action-value function, which is related to the

update of the actor, also known as policy. Moreover, the edge server stores the transition about previous actions of the edge server, current observation, actions of the edge server, AE gain, next observations, i.e., $(A_{\alpha-}, S_{\alpha}^o, A_{\alpha}, R_{\alpha}, S_{\alpha'}^o)$, into the replay buffer $\mathcal{B}$ at each training step. A mini-batch of N transitions are randomly sampled from $\mathcal{B}$ to train the actor-critic networks. The action will be selected according to the policy network with exploration noise after the first M time steps. Since the client selection action output from the policy network may not be binary, we classify them as binary by setting thresholds. Based on the TD3 framework, the policy and critic networks can be trained to approximate the optimal policy of the formulated POMDP problem.

For the continuous client selection and transmit power allocation, the objective is to find the optimal policy π_{θ} , which maximizes the expected AE gain $J(\theta) = \mathbb{E}_{S_0^o, A_0}^{\pi_{\theta}} [R(S_0^o, A_0)]$, the parameterized policies π_{θ} can be updated by taking the gradient of the expected return with respect to θ , i.e.,

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{A_{\alpha}} Q_{\pi_{\theta}}(S_{\alpha}^o, A_{\alpha}) |_{A_{\alpha}=\pi_{\theta}(S_{\alpha}^o)} \nabla_{\theta} \pi_{\theta}(S_{\alpha}^o)], \quad (3.40)$$

where the optimal action-value function can be approximated by the critic neural network (Mnih et al., 2015) $Q_{\omega}(S_{\alpha}^o, A_{\alpha})$ with parameters ω , which obtains the AE gain. To update $Q_{\omega}(S_{\alpha}^o, A_{\alpha})$, the critic neural network minimizes the approximation loss between the current target value and $Q_{\omega}(S_{\alpha}^o, A_{\alpha})$ by adjusting the parameter ω ,

$$\min_{\omega} \mathbb{E} \left[\left(R_{\alpha} + \gamma Q_{\omega'}(S_{\alpha'}^o, \pi_{\theta}(S_{\alpha'}^o)) - Q_{\omega}(S_{\alpha}^o, A_{\alpha}) \right)^2 \right], \quad (3.41)$$

where ω' stands for the periodically updated parameters of the critic target network, $\pi_{\theta}(S_{\alpha'}^o)$ is the policy target network takes action in the next state observation $S_{\alpha'}^o$.

However, the update of the critic neural network with the above value function may result in overestimation, which can produce sub-optimal policies of the policy network. To deal with the overestimation bias problem, TD3 uses two approximately independent critic networks $\{Q_{\omega_1}, Q_{\omega_2}\}$ to estimate the value function, i.e.,

$$\begin{aligned} y_{\alpha,1} &= R_{\alpha} + \gamma Q_{\omega_1'}(S_{\alpha'}^o, \pi_{\theta}(S_{\alpha'}^o)), \\ y_{\alpha,2} &= R_{\alpha} + \gamma Q_{\omega_2'}(S_{\alpha'}^o, \pi_{\theta}(S_{\alpha'}^o)), \end{aligned} \quad (3.42)$$

where the minimum of the two estimators is utilized in the update of the value function, i.e., $y_{\alpha} = \min\{y_{\alpha,1}, y_{\alpha,2}\}$.

Moreover, FL-DLT3 follows TD3 to update the policy and targets less frequently than the Q functions. Delayed policy updates consists of only updating the policy and critic target network every d intervals. And we also add noise to the target action, which makes the policy network less likely to exploit actions with high Q-value estimates. The proposed FL-DLT3 client selection and transmit power allocation policy is shown in Algorithm 2. Since the network system has K clients, each state of the clients is made up of the source of data size, uplink channel gain, downlink channel gain, bandwidth, and the remaining energy of the client. It takes T rounds of iterations to before the algorithm terminates. Therefore, the complexity of FL-DLT3 is $\mathcal{O}(T[7K(n_{pa} + n_{pc1} +$

$n_{pc2}) + (l_{pa} - 1)n_{pa}^2 + (l_{pc1} - 1)n_{pc1}^2 + (l_{pc2} - 1)n_{pc2}^2 + 12 \times (7K \times N_{lstm} + (7K)^2 + 7K)]$, where n_{pa} , n_{pc1} and n_{pc2} are the number of neurons in the hidden layer of policy network, critic network 1 and critic network 2, respectively; l_{pa} , l_{pc1} and l_{pc2} are the number of the hidden layers of policy network, critic network 1 and critic network 2, respectively; N_{lstm} is the hidden size of LSTM.

3.3.2.3 Network Architecture

Each network follows two-branch structure as in (Peng et al., 2018), which consists of a feedforward branch and recurrent branch (as shown in Figure 3.7). The feedforward branch and recurrent branch both use five fully-connected layers neural network of 512 hidden nodes. The recurrent branch consists of an embedding layer of 512 fully connected units followed by 512 LSTM units. For each hidden layer (apart from the LSTM), ReLU activations are used between the two hidden layers for both the policy and critic. The final tanh unit and linear unit following the output of the policy and critic, respectively.

3.3.2.4 LSTM Layer

The unknown network observation transitions resulting from time-varying data size, channel gains, bandwidth, and energy harvesting, which increase learning uncertainties and reduce learning accuracy. The edge server running the FL-DLT3 cannot observe the complete states of all the clients. It can only make the observation of a client when the device is selected and uploads its state information to the edge server. The learning efficiency and accuracy of the TD3-based FL-DLT3 can be compromised by the imperfect knowledge of the states of the clients. Motivated by this fact, we develop a state characterization layer to predict the states of the clients which are not observable, and feed the predicted states into the every neural network of FL-DLT3.

For every policy (target) network, two critic (target) networks and memory replay, we use LSTM to predict their respective hidden states. The LSTM-based neural network is used to find out the hidden state in the environment, e.g., the state of a device that has not been selected previously. Moreover, the LSTM based state characterization layer helps to accelerate the convergence of the FL. At time state i , the hidden states h_i^{hid} is calculated by the following composite function

$$h_i^{hid} = o_i \tanh(C_i), \quad (3.43)$$

$$o_i = \sigma(W_0 \cdot [C_i, h_{i-1}^{hid}, A_i] + e_0), \quad (3.44)$$

$$C_i = F_i C_{i-1} + p_i \tanh(W_c \cdot [h_{i-1}^{hid}, A_i] + e_c), \quad (3.45)$$

$$F_i = \sigma(W_f \cdot [h_{i-1}^{hid}, C_{i-1}, A_i] + e_f), \quad (3.46)$$

$$p_i = \sigma(W_p \cdot [h_{i-1}^{hid}, C_{i-1}, A_i] + e_p), \quad (3.47)$$

where o_i , C_i , F_i , and p_i denote the output gate, cell activation vectors, forget gate, and input gate of the LSTM layer, respectively. σ , and $\tanh$ refer to logistic sigmoid function and the hyperbolic

tangent function, respectively. $\{W_0, W_c, W_f, W_p\}$ are the weight matrix, and $\{e_0, e_c, e_f, e_p\}$ are the bias matrix.

Algorithm 2: The developed FL-DLT3 client selection and resource allocation policy

```

1 Initialize critic networks  $\{Q_{\omega_1}(S_\alpha^o, A_\alpha, A_{\alpha^-}), Q_{\omega_2}(S_\alpha^o, A_\alpha, A_{\alpha^-})\}$  and policy network
    $\pi_\theta(S_\alpha^o, A_{\alpha^-})$  with random parameters  $\omega_1, \omega_2, \theta$ .
2 Initialize target networks  $\omega'_1 \leftarrow \omega_1, \omega'_2 \leftarrow \omega_2, \theta' \leftarrow \theta$ .
3 Initialize environment  $S_\alpha^o$ , previous actions  $A_{\alpha^-} \leftarrow 0$ , replay buffer  $\mathcal{B}$ .
4 for  $\alpha = 1, \dots, T$  do
5   if  $\alpha \leq N$  then
6     Explore  $N$  steps with random policy, obtain  $R_\alpha$  and  $S_{\alpha'}^o$ , and storage the transition
        $(A_{\alpha^-}, S_\alpha^o, A_\alpha, R_\alpha, S_{\alpha'}^o)$  of the  $N$  steps into  $\mathcal{B}$ .
7   else
8     Select action with exploration noise  $A_\alpha \sim \pi_\theta(S_\alpha^o, A_{\alpha^-}) + \rho, \rho \sim \mathcal{N}(0, \sigma)$ .
9     Server allocates the selected clients  $P_{\alpha,k}$ .
10    Server performs model aggregation to obtain updated global model and distributes
       it to all selected devices in the next round.
11    Server observes new observation  $S_{\alpha'}^o$ , calculates  $R_\alpha$ , and stores the experience
        $(A_{\alpha^-}, S_\alpha^o, A_\alpha, R_\alpha, S_{\alpha'}^o)$  into  $\mathcal{B}$ .
12    Sample mini-batch of  $N$  experiences  $\{(A_{\alpha^-}, S_\alpha^o, A_\alpha, R_\alpha, S_{\alpha'}^o)_i\}_{i=1}^N$  from  $\mathcal{B}$ .
13    Obtain target action  $\hat{A}_{\alpha'} \leftarrow \pi_{\theta'}(S_{\alpha'}^o, A_\alpha) + \hat{\rho}, \hat{\rho} \sim \text{Clip}(\mathcal{N}(0, \hat{\sigma}), -c, c)$ .
14     $\hat{y} \leftarrow R_\alpha + \gamma \min_{m=1,2} Q_{\omega_m}(S_{\alpha'}^o, \hat{A}_{\alpha'}, A_\alpha)$ .
15    Update critics  $\omega_m \leftarrow \arg \min_{\omega_m} \mathbb{E}(\hat{y} - Q_{\omega_m}(S_\alpha^o, A_{\alpha^-}))^2$ .
16    if  $\alpha \bmod d = 0$  then
17      Update  $\theta$  by the deterministic policy gradient using equation (3.40).
18      Update target networks:
19       $\omega_{m'} \leftarrow \phi \omega_m + (1 - \phi) \omega_{m'}$ .
20       $\theta' \leftarrow \phi \theta + (1 - \phi) \theta'$ .
21    end
22  end
23 end
```

3.3.3 Numerical Experiments

In this section, we present the implementation of the proposed FL-DLT3 on PyTorch, which is an open source machine learning library based on the Torch library. We compare FL-DLT3 with benchmarks in terms of network size, data size, and communication bandwidth. Table 3.2 specifies the configuration of simulation parameters.

3.3.3.1 Implementation and Training of FL-DLT3

We implement the proposed FL-DLT3 with Python 3.9. PyTorch is set up on a Linux workstation with 64-bit Ubuntu 18.04. FL-DLT3 trains the resource allocation with FL on 2 Nvidia's GPUs,

Table 3.2: Simulation parameters

Parameters	Values
Number of rounds (T)	1000
Number of local iterations (L)	4
For training one data sample CPU cycles per bit (c_k)	20 cycles / bit
Computation capacity of client k (f_k)	[2, 4] GHZ
Transmit power of client k ($P_{t,k}$)	[0.1, 60] W
Transmit power of the edge server (P_t^s)	[100, 1000] W
Uplink channel gain ($G_{t,k}$)	$[10^{-3}, 10^{-1}]$ dB
Downlink channel gain ($H_{t,k}$)	$[10^{-1}, 10]$ dB
Power spectral density of the Gaussian noise (N_0)	1.0×10^{-8}
Parameter size of the global model (S_d)	1×10^4 bit
Upload data size of client ($\mathfrak{S}_u$)	5×10^4 bit
Binary indicator of device selection ($\beta_{t,k}$)	{0, 1}
System parameter (μ_k)	4.2×10^{-9}
Effective capacitance coefficient (ϵ_k)	1.2×10^{-28}
The amount of harvested energy ($\Delta E_{t,k}$)	[50, 200] J
The constant value (v)	1.0
Critic network learning rate	3×10^{-4}
Policy network learning rate	3×10^{-4}
Neural network weight coefficient (ϕ)	5×10^{-3}
Interval of policy target network update (d)	10
Discount factor (γ)	0.99
Batch size (N)	45
Replay buffer size ($ \mathcal{B} $)	5×10^5
Exploration noise (σ)	0.5
Clipped normal noise (c)	0.5

one is GeForce GTX 1060 with 3 GB memory, the other is GeForce RTX 2060 with 6 GB memory.

The experience replay memory can store 5×10^5 training samples in terms of *previous actions of the edge server, current observation, actions of the edge server, AE gain, next observations*. The previous actions and current observation are combined to feed into policy network and critic networks to infer the hidden state. Both state space and action space are updated by using Adam optimizer, where the learning rate is 3×10^{-4} . At one training step, the policy and critic networks are trained with a mini-batch of 45 transitions, sampled from the experience replay memory.

The policy target network is implemented by adding $p \sim \mathcal{N}(0, 0.5)$ to the actions chosen by the policy target network, clipped to (0, 60). The delayed policy updates the policy and critic target networks every d intervals, where $d = 10$.

3.3.3.2 AE Gain Performance

For performance validation, we compare FL-DLT3 with existing state-of-the-art FL-based device scheduling approaches, i.e., FedAECS (Zheng et al., 2021), FedCS (Nishio and Yonetani, 2019) and FedAvg (McMahan et al., 2017).

Figure 3.8 shows the AE gains, where the t -th round is from 1 to 1000 and $K = 40$. The data size $D_{t,k}$ and bandwidth $b_{t,k}$ of the K devices vary in $[2, 10]$ MB and $[10, 50]$ KHz, respectively. In general, the proposed FL-DLT3 achieves the highest AE gain, as compared to the existing FedAECS, FedCS, and FedAvg, improved by 51.8%, 82.4% and 85.0% respectively. The reason is that FL-DLT3 leverages experience replay and predict the states of the clients which are not observable, however, FedAECS, FedCS, and FedAvg are unable to predict the states of the clients.

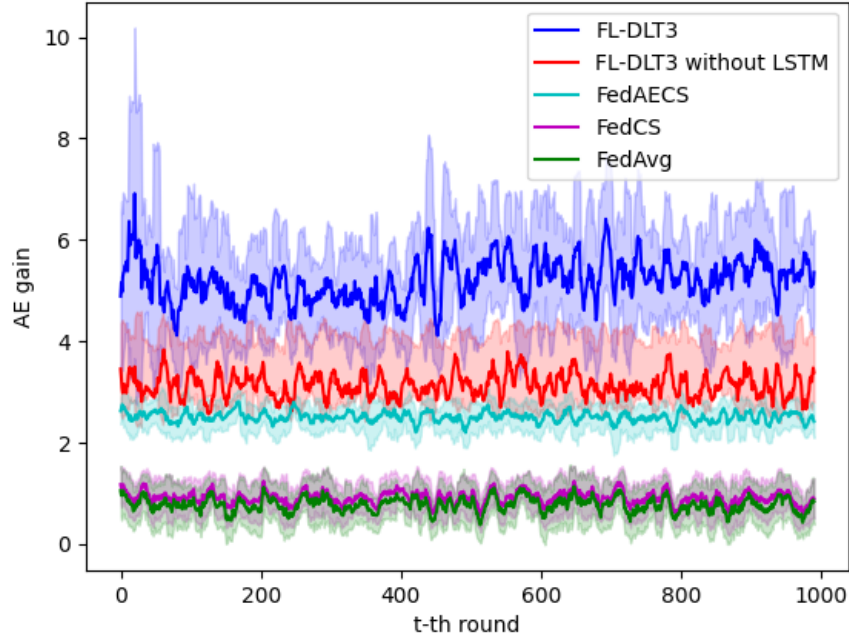


Figure 3.8: Comparison of the AE gains, where $T = 1000$ and $K = 40$.

FL-DLT3 outperforms FL-DLT3 without the LSTM layer with a gain of 39.8%. This is because the LSTM layer efficiently predicts the unknown network observation transitions, which enriches the training environment for FL and TD3.

Figure 3.9 studies the AE gain of the proposed FL-DLT3, where K increases from 10 to 80. In general, the proposed FL-DLT3 achieves the highest AE gain 11.4557, as compared to the existing FedAECS (2.7212), FedCS (1.3952), and FedAvg (1.3498) given $K = 10$.

In Figure 3.10, it can be observed that the energy consumption of FL-DLT3 dominates the performance when K increase from 10 to 50. When $K > 60$, the FL accuracy dominates the performance. This confirms that the AE gain's fluctuation in Figure 3.9 when K increase from 10 to 50. The reasons is that FL-DLT3 can select more clients, hence the FL accuracy and energy consumption increase monotonically with the number of devices.

Figure 3.11 shows the AE gain given 1000 rounds. With an increase of K , the AE gain achieved by FL-DLT3 decreases from 5.3411 to 3.7066. This also validates the performance in Figure 3.10.

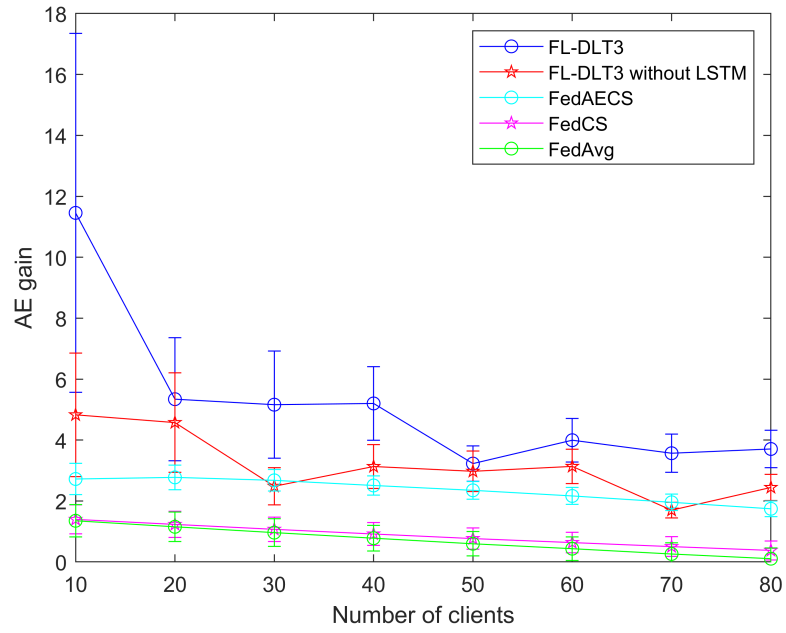


Figure 3.9: Comparison of the AE gain obtained by using FL-DLT3, FL-DLT3 without LSTM, FedAECS, FedCs and FedAvg with different number of clients

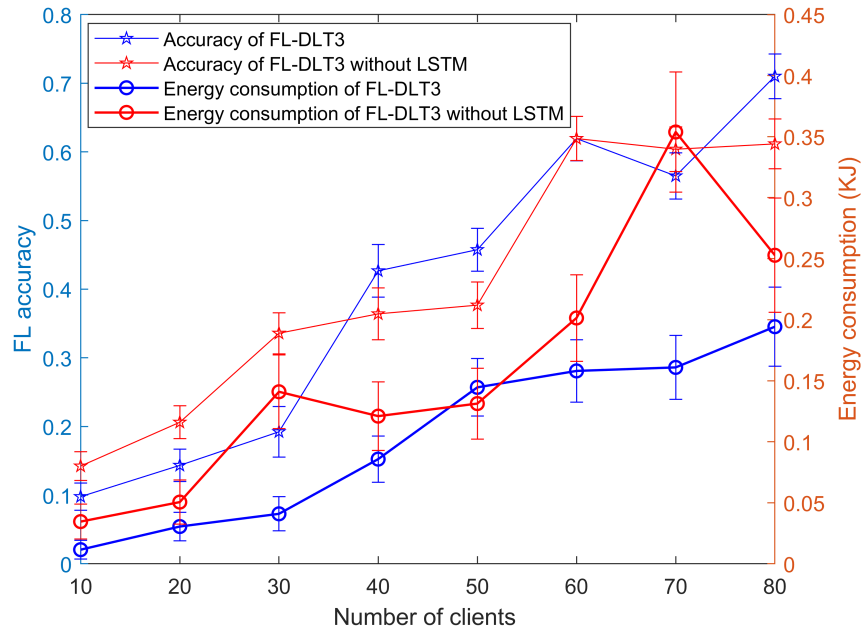


Figure 3.10: Compare the FL accuracy and energy consumption of FL-DLT3 and FL-DLT3 without LSTM as the number of devices changes

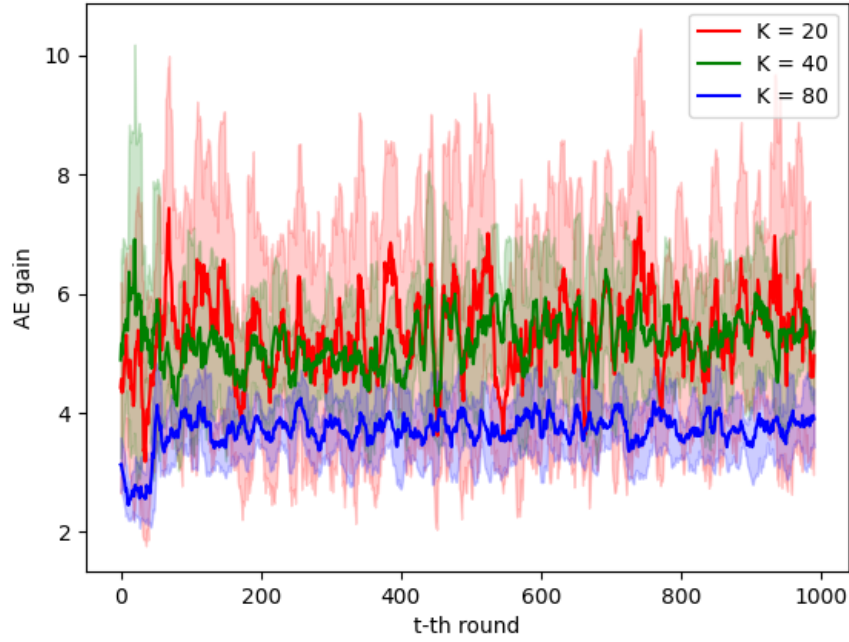


Figure 3.11: The AE gain achieved by the proposed FL-DLT3 given $K = 20, 40$, or 80 .

Figure 3.12 depicts the AE gain of FL-DLT3 with regards to the average data size. In general, the AE gain increases with the growth of the data size. This is because the FL accuracy rises in a high rate while the energy consumption of the clients on the training of FL-DLT3 slightly increases. Moreover, the AE gain obtained by FL-DLT3 is about twice that obtained by FedAECS, thanks to the transmit power allocation in FL-DLT3.

Figure 3.13 describes the AE gain of FL-DLT3 with regards to the average bandwidth. Overall, the AE gain raises with an increase of the average bandwidth since the energy consumption of the clients on the local models transmission is reduced, in other words, a better channel quality leads to smaller packet retransmission and energy consumption. In addition, FL-DLT3 achieves the highest AE gain given different bandwidths. This is achieved by the LSTM layer is integrated into the FL-DLT3 to predict the time-varying bandwidth.

Figure 3.14 shows the FL accuracy and energy consumption of FL-DLT3 and FL-DLT3 without LSTM in regard to the average bandwidth size of clients changes. Generally, the energy consumption of FL-DLT3 decreases with an increase of the bandwidth since the retransmission of the local models is reduced.

Figure 3.15 shows the runtime of FL-DLT3, where K is set to 10, 40, 60, or 80. In the first 45 rounds, the runtime of FL-DLT3 is about 0.018 s. This is because the experience replay buffer is initialized in which the training of FL-DLT3 is not conducted yet. Once the learning experience is sufficient and FL-DLT3 carries out the training, the runtime raises to 0.74 s given $K = 10$. When K increases to 80, the training of FL-DLT3 takes 1.3286 seconds. The reason is that an increase of

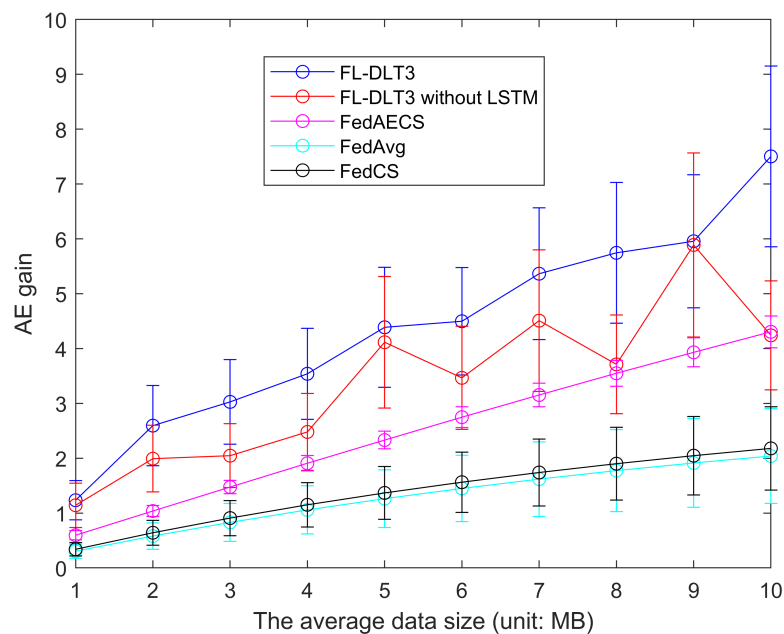


Figure 3.12: AE gain varying with data size following normal distribution while keeping the same variance (0.2 MB)

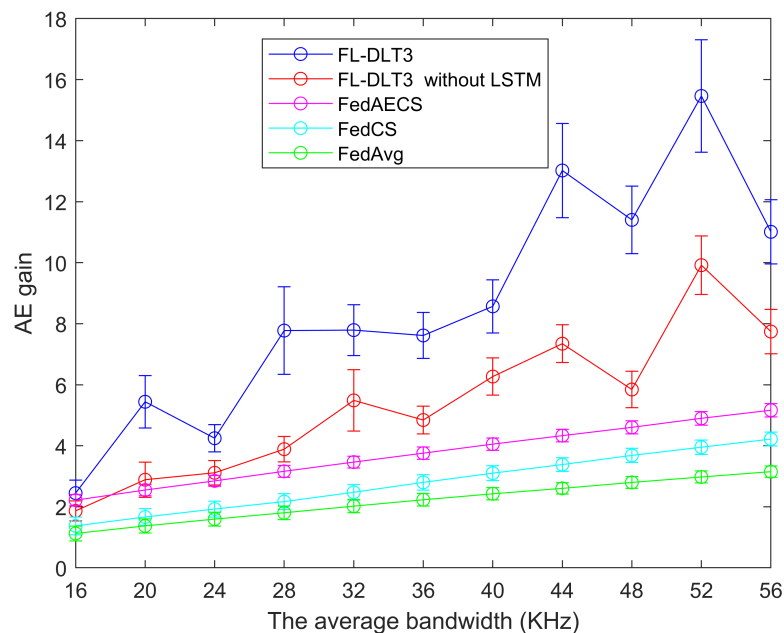


Figure 3.13: AE gain varying with bandwidth following normal distribution while keeping the same variance (4 KHz)

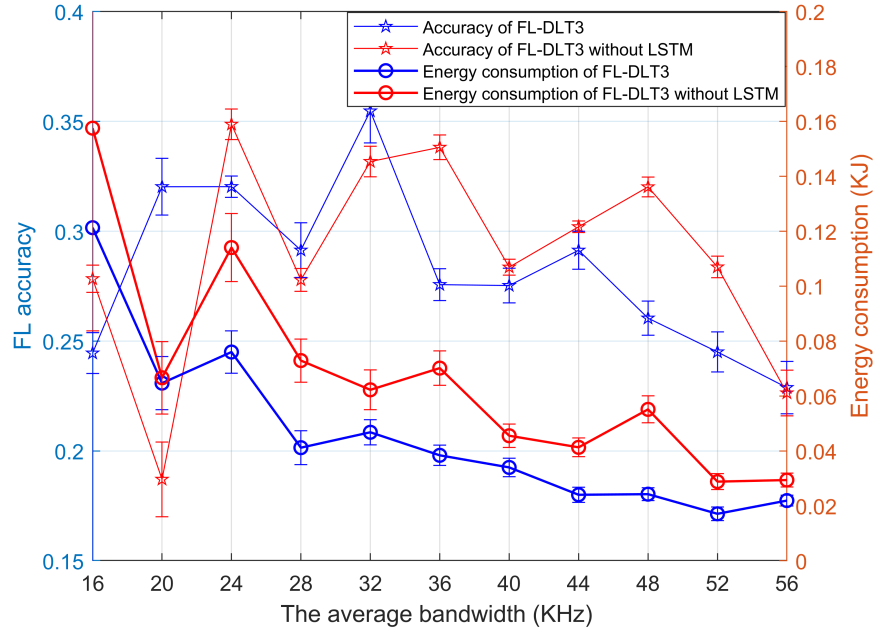


Figure 3.14: Compare the FL accuracy and energy consumption of FL-DLT3 and FL-DLT3 without LSTM as the average channel size of clients changes

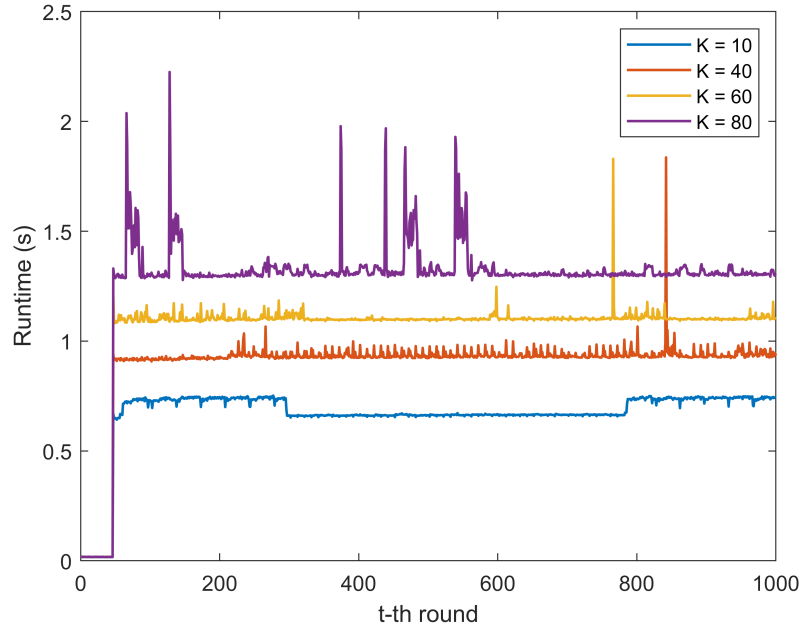
the clients results in a large state and action space, hence the learning time of FL-DLT3 increases. In addition, the runtime of FL-DLT3 randomly fluctuates. This is due to random interruptions from other program executions that are concurrently operated on the server.

3.4 Chapter Summary and Concluding Remarks

In this chapter, the FL-based client selection optimization is formulated to balance the total energy consumption of clients and learning accuracy of the FL. Based on whether the resource information of clients is known to the edge server or not, we investigated two different schemes, namely, heuristic-based algorithm to approximate the optimal client selection in polynomial time and FL-DLT3 framework to balance FL accuracy and energy consumption in a continuous domain.

To solve the problem in polynomial time, we propose the suboptimal heuristic, FedAECS algorithm, which recursively assesses the potential clients in FL to balance the energy consumption and learning accuracy. The numerical results show that the proposed FedAECS outperforms the existing solutions in the literature. Moreover, we solved the optimization problem by JILOG CPLEX 12.10, and obtained optimal results, which are compared with the results that obtained by FedAECS. On average, FedAECS not only guarantees exactly the client scheduling schemes as optimal schedules but also is much more efficient than CPLEX on runtime.

We further propose FL-DLT3, which is a new deep reinforcement learning based resource allocation with FL for client. Given the large state and action space, FL-DLT3 learns the network state dynamics while maximizing the ratio of the learning accuracy of FL to the energy consumption

Figure 3.15: Runtime with t -th round

of the clients in a continuous domain. Considering the continuous transmit powers of the clients, FL-DLT3 optimizes the edge server's selection of clients for each round of FL and the transmit powers of the clients, based on TD3. To improve the learning accuracy, a new state characterization LSTM layer is designed in coupling with the proposed FL-DLT3 to predict the time-varying network states, e.g., data size, bandwidth, channel gain, and remaining energy of the clients. The LSTM layer estimates the unobserved states at every training iteration of the TD3. Moreover, the LSTM based state characterization layer helps to accelerate the convergence of the FL. To the best of our knowledge, this is the first time that LSTM is employed in coupling with TD3 for the resource allocation of FL-enabled client. FL-DLT3 is implemented in PyTorch. The effectiveness of FL-DLT3 is validated with the experimental data.

Chapter 4

Data-Agnostic Model Poisoning against FL: A Graph Autoencoder Approach

Despite the fact that the two client scheduling methods we proposed in Chapter 3 both can achieve an efficient trade-off between the energy consumption of clients and learning accuracy of FL, some malicious clients can also be readily selected by the server to participate in the FL training process. The malicious clients potentially launch model poisoning or data poisoning attacks to manipulate FL and propagate the attacks into benign clients, as we described in Section 1.2, which results in a failure of FL training. In this chapter, we propose a novel, data-agnostic, model poisoning attack on FL, by designing a new GAE-based framework. The attack requires no knowledge of FL training data and achieves both effectiveness and undetectability. Extensive experiments indicate that the FL accuracy drops gradually under the proposed attack, and the existing poisoning defense mechanisms can hardly detect the attack. The rest of this paper is organized as follows. Section 4.1 discusses FL with benign clients and server, as well as the eavesdropping model. The proposed GAE-based epidemic attack is delineated in Section 4.2. Performance analysis is conducted in Section 4.3. Section 4.4 concludes this chapter.

4.1 System Model

In this section, we present the threat model, where malevolent clients can act as attackers. An attacker creates and uploads malicious local model updates to progressively contaminate the global model of the FL. we further describe an attacker detection model that the server can adopt to discern malicious local models by measuring the Euclidean distances between the models.

4.1.1 Threat Model

We consider a new data-agnostic model poisoning attack, where malicious local models are generated solely based on the benign local models overheard and the correlation features of the benign local and global models. This attack could be particularly severe in FL systems under wireless settings, due to the broadcast nature of radio. As shown in Figure 4.1, an attacker within the vicinity

of benign clients and equipped with radio transceivers can passively eavesdrop on the transmitted local models of some (if not all) of the benign clients, extracting their features and generating its own malicious local model. Although cryptography can prevent eavesdropping attacks to some extent, existing techniques, such as those developed in (Aviram et al., 2016; Hebrok et al., 2023; Díaz-Sánchez et al., 2019), have demonstrated the possibility of deciphering encrypted information with limited initial data.

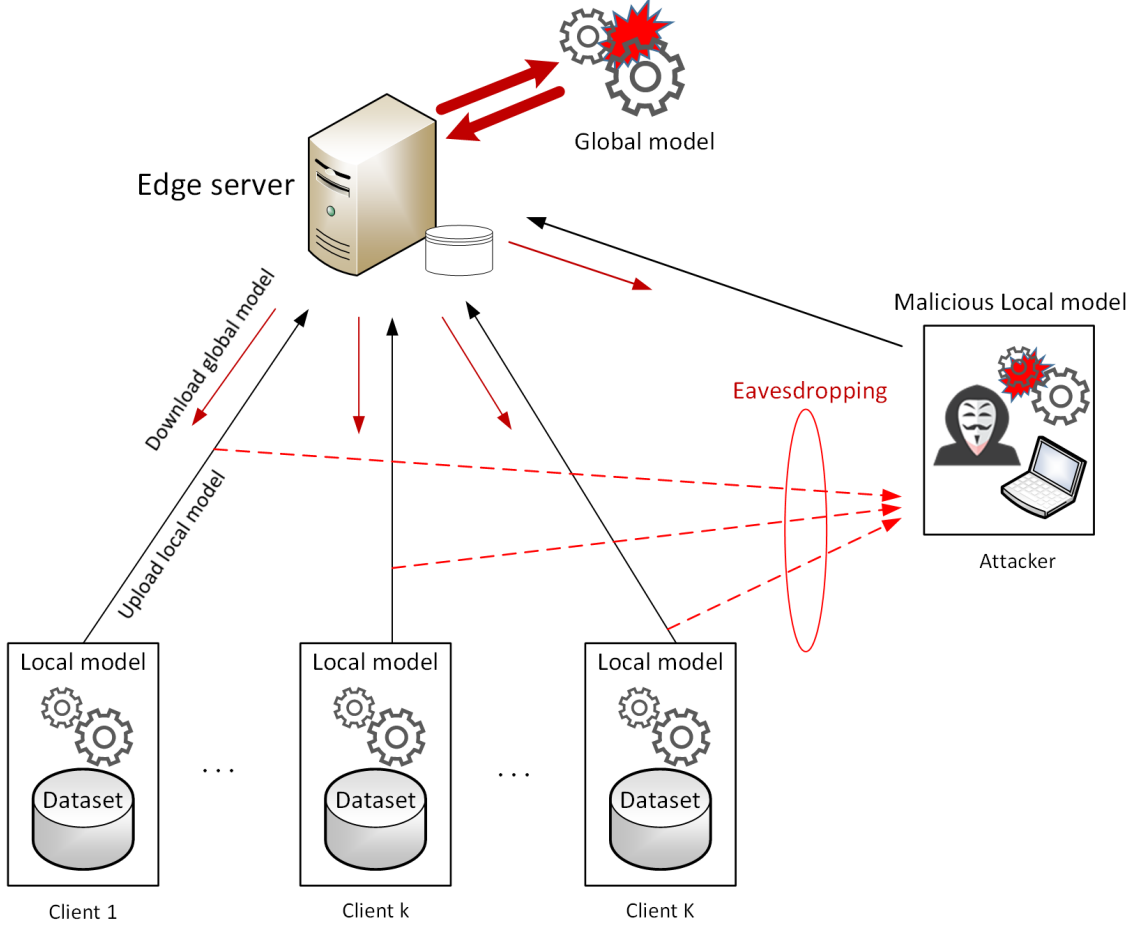


Figure 4.1: The proposed data-agnostic, model poisoning attack, where the attacker eavesdrops the global model and the local models uploaded by the benign clients. Next, the attacker generates a malicious local model to contaminate the global model and the benign local models.

The attacker creates and uploads a malicious local model, denoted by $\mathbf{W}_{k'}^t$, to contaminate the global model $\mathbf{W}^t$, and subsequently the local models of the benign clients, i.e., $\mathbf{W}_k^t, \forall k \in [1, K]$, where t indicates the t -th communication round. $\mathbf{W}_{k'}^t$ is adversarially created based on the benign local model parameters overheard by the attacker in the t -th communication round. During the FL process, the attackers collected by the set of $\mathcal{M} \triangleq \{1', \dots, K'\}$ ($K' < K$) maintain the same FL structure as benign clients. Inspired by projected gradient descent attacks (Madry et al., 2018; Zhuang et al., 2023), an attacker may find the perturbation on the eavesdropped local model updates that maximize the loss of the global model while keeping the size of the perturbation

smaller than a specified threshold.

Unaware of the ill-intentioned attacker, the server aggregates the local models of all clients, including both the benign and malicious local models, and unintentionally creates a contaminated global model, denoted by $\mathbf{W}_g^t$, at the t -th communication round. The total size of the local training data reported to the server is $S = \sum_{k \in \mathcal{K}} D_k + \sum_{k' \in \mathcal{M}} D_{k'}$, where $D_{k'}$ is the claimed data size by attacker k' . The contaminated global model is given by

$$\mathbf{W}_g^t = \sum_{k \in \mathcal{K}} \frac{D_k}{S} \mathbf{W}_k^t + \sum_{k' \in \mathcal{M}} \frac{D_{k'}}{S} \mathbf{W}_{k'}^t. \quad (4.1)$$

The server broadcasts $\mathbf{W}_g^t$ to all clients.

To this end, the FL training process in essence trains the global model based on the local datasets of all clients, including the nonexistent dataset claimed by the attacker, by minimizing the following global loss function:

$$\min_{\mathbf{W}_g^t} F(\mathbf{W}_g^t) = \sum_{k \in \mathcal{K}} \frac{D_k}{S} F_k(\mathbf{W}_g^t) + \sum_{k' \in \mathcal{M}} \frac{D_{k'}}{S} F_{k'}(\mathbf{W}_g^t), \quad (4.2)$$

where $F_k(\cdot)$ and $F_{k'}(\cdot)$ are the local loss function of the k -th client and the claimed local loss function of the k' -th attacker respectively, which are conform to (3.1). Note that without loss of generality, we assume that the size of the client data remains fixed throughout the FL process, that is, $D_{t,k} \equiv D_k \forall t$.

To impair the FL training process, the attacker aims to maximize $F(\mathbf{W}_g^t)$, i.e., equivalent to minimizing the training accuracy of FL, while keeping $\mathbf{W}_{k'}^t$ undetectable by the server that typically constantly assesses the similarities among all local models and rules out those substantially different from the rest, e.g., the state-of-the-art Euclidean distance-based (Blanchard et al., 2017; Yin et al., 2018) or similarity-based detection methods (Cao et al., 2021). As a result, the attacked global model diverges in a direction opposite to the one intended in the absence of the attack.

At the t -th communication round, the attacker formulates a data-agnostic, model poisoning attack problem:

$$\max_{\mathbf{W}_{k'}^t} F(\mathbf{W}_g^t) \quad (4.3a)$$

$$\text{s.t } \mathcal{D}(\mathbf{W}_{k'}^t, \mathbf{W}_g^t) \leq r, \forall k' \in \mathcal{M}, \quad (4.3b)$$

where $\mathcal{D}(\mathbf{W}_{k'}^t, \mathbf{W}_g^t)$ evaluates the Euclidean distance between $\mathbf{W}_{k'}^t$ and $\mathbf{W}_g^t$, and r is a pre-specified threshold that ensures the generated malicious local model is close to the global model in the Euclidean space to escape the scrutiny of the server. The optimal malicious local model updates are

$$(\mathbf{W}_{k'}^t)^* = \arg\max_{\mathbf{W}_{k'}^t} F(\mathbf{W}_g^t). \quad (4.4)$$

Note that the attack model considered is a white-box attack.

4.1.2 Defense Model for Attacker Detection

In response to the prevalent threat of model poisoning in FL, an attacker detection model residing on the server can be applied, which leverages the Euclidean distance metric to discern malicious local models, for instance, (Zhang et al., 2022a) and (Li et al., 2023d). By measuring the straight-line distance between each incoming local model and the aggregated global model, this model aims to identify anomalous deviations indicative of malicious intent. The underlying rationale is that genuine local models from benign devices are expected to cluster within a certain proximity in the model space, while malicious local models, designed to sabotage the global model's integrity, would exhibit more pronounced deviations. By setting a distance threshold, local models that exceed this threshold can be flagged or discarded, effectively isolating and mitigating the impact of malicious local models on the global model's integrity. This server side defense mechanism underscores the potential of geometric measures, like Euclidean distance, as powerful tools in safeguarding FL systems from adversarial attacks.

4.2 Proposed Data-agnostic Model Poisoning Attack against FL

In this section, we elaborate on the proposed data-agnostic model poisoning attack, where adversarial GAE is designed to extract the feature correlation among the local models of the benign clients and reconstructs an adversarial adjacency matrix. With the adjacency matrix, the attacker trains the GAE to generate malicious local models without being detected by the server.

4.2.1 GAE Model for Data-Agnostic Model Poisoning

The arbitrary features of $\mathbf{W}_{k'}^t$ and those of the benign local models may have a low feature correlation, which can be potentially detected by the server. To address this, we develop a new GAE model for the novel, data-agnostic, model poisoning attack.

The optimization problem in (4.3) can be transformed using the Lagrangian method (Tran et al., 2021). Let λ denote the dual variable. The Lagrange function is given by

$$L(\mathbf{W}_{k'}^t, \lambda) = F(\mathbf{W}_g^t) + \lambda(r - \mathfrak{D}(\mathbf{W}_{k'}^t, \mathbf{W}_g^t)) \quad (4.5)$$

The Lagrange dual function is

$$H(\lambda) = \max_{\mathbf{W}_{k'}^t} L(\mathbf{W}_{k'}^t, \lambda) \quad (4.6)$$

The dual problem of the problem in (4.3) is given by

$$\min_{\lambda'} H(\lambda) \quad (4.7)$$

At the t -th communication round, given $\lambda^t = \lambda$, the primary variable $\mathbf{W}_{k'}^t$ of the data-agnostic model poisoning attack can be optimized by solving

$$(\mathbf{W}_{k'}^t)^* = \arg \max_{\mathbf{W}_{k'}^t} \{F(\mathbf{W}_g^t) - \lambda \mathcal{D}(\mathbf{W}_{k'}^t, \mathbf{W}_g^t)\} \quad (4.8)$$

With obtained $(\mathbf{W}_{k'}^t)^*$, the sub-gradient descent method can be taken to update λ^t by solving the dual problem (4.7). Specifically, λ^t is updated by (Boyd and Vandenberghe, 2004)

$$\lambda^{t+1} = \left[\lambda^t - \eta (\mathcal{D}(\mathbf{W}_{k'}^t, \mathbf{W}_g^t) - r) \right]^+ \quad (4.9)$$

where η is the step size, and $[x]^+ = \max(0; x)$. At initialization, λ^t is non-negative, i.e., $\lambda^t \geq 0$, to ensure (4.9) converges.

We propose to solve (4.8) by developing a new GAE model, followed by the sub-gradient descent to update (4.9). These two steps are performed in an alternating manner, as illustrated in Figure 4.2. Specifically, we propose to decompose the local model parameters of the benign devices into a graph capturing the correlations (or similarity) between the benign local models, and the underlying spectral-domain data features that the local models capture. Then, we regenerate manipulatively the graph with the GAE and subsequently compose malicious local models with the regenerated graph and the original, genuine data features. The rationale of this design is provided as follows.

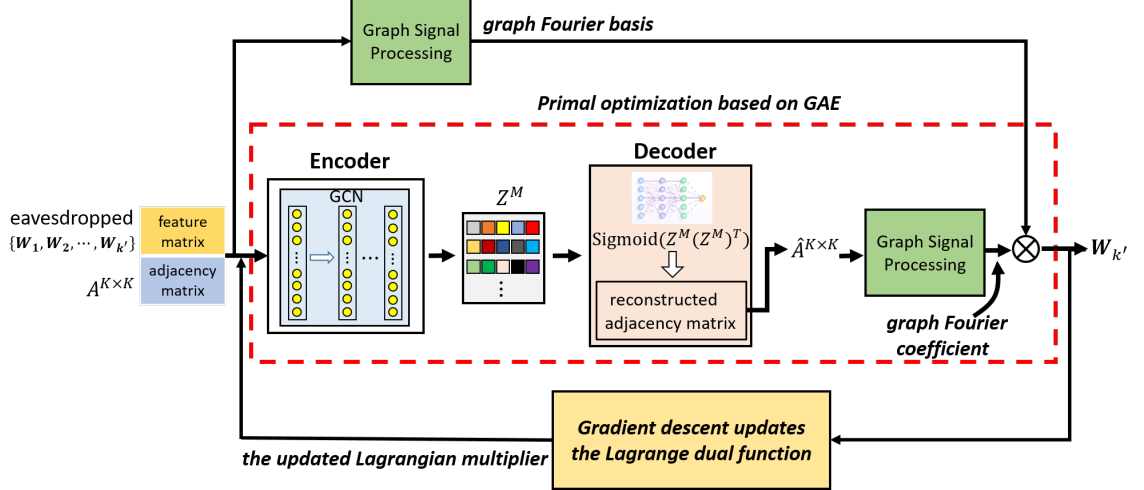


Figure 4.2: The proposed GAE model for generating data-agnostic, malicious local models, where the attacker overhears $\mathbf{W}_g^t$ and $\mathbf{W}_{k'}^t, \forall k$ and applies the GCN-based encoder to create Z^M . The output of the encoder, i.e., the feature representations, is input to the decoder for feature reconstruction.

- By regenerating the graph with the GAE, we retain and manipulate the correlations between the local models, and also deter the convergence of the global model, i.e., by maximizing (4.8). The decoder of the GAE reproduces the correlations while satisfying constraint (4.3b). This suppresses structural dissimilarity between the malicious and benign local models.

- By using the genuine underlying spectral-domain data features, the malicious local models are substantiated by the genuine data features. Hence, they are less likely to be detected by the server

1) *GAE for Malicious Model Generation*: The attacker aims to construct $\mathbf{W}_{k'}^t$ without knowing any data of the benign clients. As illustrated in Fig. 2, a graph, denoted by $G(V, E, \mathbf{F})$, is used to formulate the benign local models in FL, where V , E , and $\mathbf{F}$ represent vertexes, edges, and the feature matrix of the graph, respectively.

Let $\mathbf{A} \in \mathbb{R}^{K \times K}$ denote the adjacency matrix that describes the correlation among the local models of the clients. At the t -th communication round of the FL, the (k, j) -th element of $\mathbf{A}$, denoted by $\bar{w}_{k,j}$, measures the similarity between $\mathbf{W}_k^t$ and $\mathbf{W}_j^t$ (Zhu et al., 2020), as given by

$$\bar{w}_{k,j} = \frac{\mathbf{W}_k^t \cdot \mathbf{W}_j^t}{\|\mathbf{W}_k^t\| \cdot \|\mathbf{W}_j^t\|} \quad (4.10)$$

According to $\mathbf{A}$, the topological structure of the graph $\mathcal{G}$ can be constructed. Moreover, $\mathcal{F} = [\mathbf{W}_1^t, \dots, \mathbf{W}_k^t, \mathbf{W}_{k'}^t]$ is a vector that collects all local models of both benign and malicious clients, $\mathbf{W}_{k'}^t \in \mathbb{R}^{1 \times D}$.

The GAE consists of an encoder and a decoder, where the encoder encodes the graph data with the features and the decoder takes the encoder's output as the input to reconstruct $G(V, E, \mathbf{F})$ (Wang et al., 2020c).

• **Encoder**: The encoder in the proposed GAE is responsible for mapping $G(V, E, \mathbf{F})$ to a lower-dimensional representation. We build the encoder based on an M -layer graph convolutional network (GCN) architecture, which learns a representation that captures the underlying features of $G(V, E, \mathbf{F})$. The encoded representation is then used as input to the decoder, to reconstruct the original graph from the lower-dimensional representation to obtain the malicious local model $(\mathbf{W}_{k'}^t)^*$ in (4.8).

The encoder takes the matrices $\mathcal{F}$ and $\mathbf{A}$ as its input to its M -layer GCN. The output at the M -th layer is

$$\mathbf{Z}^M = f_G(\mathbf{Z}^{M-1}, \mathbf{A} | \mathbf{w}^M) \quad (4.11)$$

where $f_G(\cdot, \cdot | \cdot)$ is a spectral convolution function and $\mathbf{w}^M$ defines the weight matrix at the M -th layer of the GCN.

With the identity matrix $\mathbf{I} \in \mathbb{R}^{K \times K}$, we define $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\bar{\mathbf{A}}_{kk} = \sum_j \tilde{\mathbf{A}}_{kj}$. To generate a feature representation of the graph, the encoder can be written as

$$f_G(\mathbf{Z}^{M-1}, \mathbf{A} | \mathbf{w}^M) = \Phi^M(\bar{\mathbf{A}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \bar{\mathbf{A}}^{-\frac{1}{2}} \mathbf{Z}^{M-1} \mathbf{w}^M) \quad (4.12)$$

where Φ^M represents a nonlinear activation function, e.g., $\tanh(\cdot)$ or $\text{ReLU}(\cdot)$; and $\bar{\mathbf{A}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \bar{\mathbf{A}}^{-\frac{1}{2}}$ is the symmetrically formulated adjacency matrix (Zhu et al., 2020).

• **Decoder**: The decoder is responsible for taking the lower-dimensional representation generated by the encoder, i.e., $\mathbf{Z}$ in (4.11), and mapping it back to the original $G(V, E, \mathbf{F})$. This can be viewed as the inverse operation of the encoder. The decoder aims to generate the original graph

from its reduced representation. The output of the decoder is compared with the original input graph to evaluate a loss. The encoder and decoder are trained together to minimize the loss.

A reconstructed adjacency matrix is generated at the decoder, which is defined as

$$\hat{\mathbf{A}} = \text{sigmoid}(\mathbf{Z}^M (\mathbf{Z}^M)^T) \quad (4.13)$$

where the sigmoid function is defined as $\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$. The larger the inner product $(\mathbf{Z}^M (\mathbf{Z}^M)^T)$, the more likely the vertexes j and j' are connected in the graph (Pan et al., 2019).

The output of the decoder is the reconstructed adjacency matrix $\hat{\mathbf{A}}$. A reconstruction loss function that measures the difference between $\mathbf{A}$ and $\hat{\mathbf{A}}$ can be formulated as (Qiu et al., 2022).

$$\phi_{loss} = \mathbb{E}_{f_G(\mathbf{Z}^{M-1}, \mathbf{A} | \mathbf{w}^M)} \left[\log p(\hat{\mathbf{A}} | \mathbf{Z}^M) \right], \quad (4.14)$$

where $p(\hat{\mathbf{A}} | \mathbf{Z}^M)$ at the decoder indicates the correlation among the embedding vertexes, and is given by

$$p(\hat{\mathbf{A}} | \mathbf{Z}^M) = \prod_{k=1}^K \prod_{j=1}^K p(\hat{\mathbf{A}}_{jj'} | \mathbf{Z}_j^M, \mathbf{Z}_{j'}^M), \quad (4.15)$$

where

$$p(\hat{\mathbf{A}}_{kj} = 1 | \mathbf{Z}_k^M, \mathbf{Z}_j^M) = \text{sigmoid}(\mathbf{Z}_k^M (\mathbf{Z}_j^M)^T). \quad (4.16)$$

• **Malicious Model Generation:** A graph signal processing module is designed to decompose the correlation features of the benign local models, and the data features substantiating the local models, as described earlier. A Laplacian matrix (Molitierno, 2016) is built based on the adjacency matrix of the benign models, i.e., $\mathbf{A}$, as given by

$$\mathcal{L} = \text{diag}(\mathbf{A}) - \mathbf{A}. \quad (4.17)$$

By applying singular value decomposition (SVD) (Lange and Lange, 2010) to $\mathcal{L}$, i.e., $\mathcal{L} = B \Sigma B^T$, we can obtain a complex unitary matrix $B \in \mathbb{R}^{J \times J}$, also known as graph Fourier transform (GFT) basis, that is used to transform graph data, e.g., $\mathbf{F}$, to its spectral-domain representation. Σ is a diagonal matrix with the eigenvalues of $\mathcal{L}$ along its main diagonal.

As a result, the attacker can obtain a matrix S that contains the spectral-domain data features of all benign local models, by removing the correlations among the models and subsequently focusing on the data features substantiating the local models. S is given by

$$S = B^{-1} \mathcal{F}. \quad (4.18)$$

Likewise, the attacker can use the graph signal processing module to produce a Laplacian matrix based on the output of the GAE, as given by

$$\hat{\mathcal{L}} = \text{diag}(\hat{\mathbf{A}}) - \hat{\mathbf{A}}. \quad (4.19)$$

The corresponding GFT basis, denoted by $\widehat{B}$, can be obtained by applying SVD to $\widehat{\mathcal{L}}$. With reference to (4.18), the malicious local model that follows $\mathbf{A}$ in the GAE can be determined by

$$\widehat{\mathcal{F}} = \widehat{B}\mathbf{S}, \quad (4.20)$$

where $\widehat{\mathcal{F}} \in \mathbb{R}^{J \times D}$. The vector $\mathbf{W}_{k'}^t$ in $\widehat{\mathcal{F}}$ is selected as the malicious local model and uploaded by the attacker to the aggregator for global model aggregation in the t -th communication round.

Since the attacker aims to generate the malicious local models to disorient FL, the proposed GAE is constructed and trained to maximize $L(\mathbf{W}_{k'}^t, \lambda^t) - \phi_{loss}$. As a consequence, the malicious local model $\mathbf{W}_{k'}^t$ progressively and increasingly contaminates the FL training process with the increase in global model aggregations, i.e., $t = 1, 2, \dots$.

4.2.2 Training Algorithm of the Proposed GAE Model

Algorithm 3 summarizes the training process of the proposed GAE-based, data-agnostic, model poisoning attack model, which operates along with the FL training of the benign clients and the server. Specifically, in every FL communication round, i.e., the t -th round, the server broadcasts $\mathbf{W}_g^t$. The benign clients apply the $LocalUpdate(k, \mathbf{W}^t)$ function to train their local models $\mathbf{W}_k^t, \forall k = 1, \dots, K$; see Steps 12 – 15 in Algorithm 3.

On the other hand, the attacker overhears the local model $\mathbf{W}_k^t, \forall k$ from the benign clients at the t -th FL communication round, and recall the global model $\mathbf{W}_g^{t-1}$ overheard from the server at the $(t-1)$ -th round. The GAE is trained to maximize the data-agnostic model poisoning attack problem in (4.3) with V and F . Specifically, the problem in (4.3) is transformed into a primal and a dual problem using the Lagrangian method. Given the dual variable λ^t , $\mathbf{W}_{k'}^t$ is optimized using the GAE; see Steps 16 – 23 in Algorithm 3. With the obtained $(\mathbf{W}_{k'}^t)^*$, the subgradient descent method is taken to update λ^t by (4.9); see Step 10. At the output of the GAE, the attacker achieves the optimal malicious local model, i.e., $\mathbf{W}_{k'}^t$. Next, $\mathbf{W}_{k'}^t$ is uploaded to the server for the next communication round of the FL training. As $\mathbf{W}_{k'}^t$ is highly correlated with $\mathbf{W}_k^t$ from the benign clients, the server is unable to identify the attacker.

4.2.3 Convergence Analysis of FL under Attack

To analyze the convergence of our method, we derive the convergence upper bound for the FL under the proposed, data-agnostic, model poisoning attack. The following assumptions are made before the analysis, as typically considered in the literature (Wei et al., 2020; Truex et al., 2020; Zhang et al., 2023a; Feng et al., 2023).

Assumption 1: $\forall k' \in \mathcal{M}$,

- 1) The gradient of $F_k(\mathbf{W}_k)$ is L -Lipschitz continuous (O'Searcoid, 2006), that is, $\|\nabla F_k(\mathbf{W}_k^{t+1}) - \nabla F_k(\mathbf{W}_k^t)\| \leq L\|\mathbf{W}_k^{t+1} - \mathbf{W}_k^t\|$ with L being a constant depending on the loss function so that the gradient of the global loss function is also L -Lipschitz continuous;

Algorithm 3: The proposed GAE-based, data-agnostic model poisoning attack against FL

input : $\mathcal{K}$ - the set of benign clients; $\mathcal{M}$ - the set of attackers; T_{FL} - the total number of communication round; L - the number of local epochs; $\mathcal{D}_k$ - local data size of client k ; r - Euclidean distance. η_t - the learning rate of clients; $\mathbf{W}^0$ - the initialized global model;

output: The final global model weight $\mathbf{W}^{T_{FL}}$.

```

1 for  $t = 1, \dots, T_{FL}$  do
2   Server broadcasts  $\mathbf{W}_g^t$  to all clients;
3   for  $k \in \mathcal{K}$  do
4      $\mathbf{W}_k^t \leftarrow \text{LocalUpdate}(k, \mathbf{W}^t)$ ;
5   Server collects  $\{\mathbf{W}_1^t, \dots, \mathbf{W}_k^t, \dots, \mathbf{W}_K^t\}$ ;
6   for  $k' \in \mathcal{M}$  do
7     Attacker overhears  $\{\mathbf{W}_1^t, \dots, \mathbf{W}_k^t, \dots, \mathbf{W}_K^t\}$ ;
8      $\mathbf{W}_{k'}^t \leftarrow \text{GAE}(\mathbf{W}_k^t, \forall k, \mathbf{F}, \lambda^t)$ ;
9     The attacker uploads the malicious local model  $\mathbf{W}_{k'}^t$  to server;
10  Update  $\lambda^t$ , according to (4.9);
11  The server aggregates all the local models to obtain the global model under attack
     $\mathbf{W}_g^{t+1}$  by (4.1), and broadcasts  $\mathbf{W}_g^{t+1}$ .
12  $\text{LocalUpdate}(k, \mathbf{W}^t)$ ;
13 for  $\ell = 1, \dots, L$  do
14   for  $b_k \in \mathcal{D}_k$  do
15     (3.4).
16  $\text{GAE}(\mathbf{W}_k^t, \forall k, \mathbf{F}, \lambda^t)$ ;
17 for  $k' \in \mathcal{M}$  do
18    $\mathbf{A} = \{\bar{w}_{k,j}\} \in \mathbb{R}^{K \times K} \leftarrow$  according to (4.10);
19   Input  $\mathbf{A}$  and  $\mathbf{F}$  into the GAE;
20    $\hat{\mathbf{A}} \leftarrow$  train the GAE to maximize the reconstruction loss  $L(\mathbf{W}_{k'}^t, \lambda^t) - \phi_{loss}$ ;
21    $S \leftarrow$  based on (4.17) and (4.18);
22    $\hat{\mathbf{F}} \leftarrow$  based on (4.19) and (4.20);
23    $\mathbf{W}_{k'}^t \leftarrow$  based on  $\hat{\mathbf{F}}$ .

```

- 2) $F_k(\mathbf{W}_k)$ is L_c -Lipschitz continuous; in other words, $|F_k(\mathbf{W}_k) - F_k(\mathbf{W}'_k)| \leq L_c \|\mathbf{W}_k - \mathbf{W}'_k\|$, $\forall \mathbf{W}_k, \mathbf{W}'_k$;
- 3) The learning rate is $\eta \leq \frac{1}{L}$;
- 4) At client k , the expected squared norm of the stochastic gradients is uniformly bounded by $\mathbb{E}\|\nabla F_k(\mathbf{W}_k^t)\|^2 \leq \kappa \|\nabla F_k(\mathbf{W}_g^t)\|^2, \forall k, \kappa \geq 0$;
- 5) With $\rho \geq 0$, $F_k(\mathbf{W})$ fulfills the Polyak-Lojasiewicz requirement (Karimi et al., 2016), indicating that $F(\mathbf{W}) - F(\mathbf{W}^*) \leq \frac{1}{2\rho} \|\nabla F(\mathbf{W})\|^2$, where $\mathbf{W}^* = \arg \min_{\mathbf{W}} F(\mathbf{W})$;
- 6) $F(\mathbf{W}_g^0) - F(\mathbf{W}^*) = \theta$, where θ is a constant.

Under Assumption 1, we develop the following theorem that provides the convergence bound of the gap between $\mathbf{W}_g^t, \forall t$ and $\mathbf{W}^*$.

Theorem 1: At the t -th communication round, the convergence upper bound of the attacked FL is obtained as

$$F(\mathbf{W}_g^t) - F(\mathbf{W}^*) \leq \theta \zeta^t + \frac{1 - \zeta^t}{1 - \zeta} \cdot \frac{\rho \eta D D_{k'}}{(D - D_{k'})^2} F^{max} \quad (4.21)$$

where $\zeta = 1 - \frac{\rho \eta D^2}{(D - D_{k'})^2}$, and F^{max} is a maximum value of $F(\mathbf{W}_{k'}^t)$ due to the constraint in (4.3a) and (4.3b), i.e., $F(\mathbf{W}_{k'}^t) \leq F^{max}$.

Proof: See Appendix A.1.

As stated in Theorem 1, despite the attack launched by the attacker, the global model of FL can still converge, but to an inferior global model. Specifically, as $t \rightarrow +\infty$, the optimality gap would stabilize at $\frac{2D_{k'}L_{cr}}{D - D_{k'}}$, which cannot be further reduced by training.

4.3 Performance Evaluation

In this section, we present the implementation of the proposed GAE-based, data-agnostic, model poisoning attack in PyTorch. We evaluate the testing accuracy of the local and global models of FL under attack, using the MNIST (Deng, 2012), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 (Cukierski, 2013) datasets. We also report the detection rate of the attack, where the detection is based on the Euclidean distances of the malicious local models and the benign local models to the global models, as typically done in the latest literature, e.g., (Cao et al., 2019; Nguyen et al., 2022).

Moreover, we compare the proposed attack with the existent data-agnostic model poisoning (MP) attack that produces malicious local models by mimicking other benign clients' training samples to degrade the learning accuracy. As discussed earlier in Section 4.2, our GAE-based attack represents a novel type of attack, which only depends on the benign local models overheard and the global models, has no access to any of the training data, and attempts to compromise FL training processes. Few existing techniques can produce malicious models in such a way, i.e., based solely on the overheard benign models, as most existing techniques would require the knowledge of (part of) the dataset used in the FL training processes, e.g., for a different purpose,

such as inserting a backdoor (Nguyen et al., 2022) or injecting malicious traffic into the benign training dataset (Nguyen et al., 2020). The MP attack mechanism considered for comparisons with our proposed new attack has been implemented in several existing studies, e.g., (Cao and Gong, 2022) and (Zhang et al., 2021), in which the attacker manipulates the training process by injecting a fake device and sending fake local models to the server.

4.3.1 Implementation with PyTorch

We set the number of benign clients to $K = 5; 10; 15; 20; 25$. The number of iterations per communication round to $T_L = 10$. The maximum number of communication rounds is $T_{FL} = 10$ unless otherwise specified. We implement the proposed GAE-based attack against the FL on an SVM model using PyTorch 1.12.1, Python 3.9.12 on a Linux workstation with an Intel(R) Core(TM) i7-9700K CPU@3.60GHz (8 cores) and 16 GB of DDR4 memory@2400 MHz.

The experiments are conducted on three datasets:

- The standard MNIST dataset comprises 60,000 training examples and 10,000 testing examples, which are grayscale images of handwritten digits from 1 to 10;
- The Fashion-MNIST dataset, which contains Zalando’s article images (i.e., 28×28 grayscale images) in ten classes, including 60,000 examples for training and 10,000 examples for testing;
- The CIFAR-10 dataset, which contains 60,000 images with the size of 32×32 in ten classes (6,000 per class), 50,000 for training and 10,000 for testing.

At each client, we use a standard quadratic optimization algorithm to train the SVM models on the three datasets. The loss function used for training the SVM models is $F_k(\mathbf{W}_k^t) = \frac{1}{2} \|\mathbf{W}_k^t\|^2 + \frac{1}{D_k} \sum_{i=1}^{D_k} \max\{0, 1 - y_k^i(\beta_k + (\mathbf{W}_k^t)^T x_k^i)\}$ where β_k is a parameter that can be obtained based on $\mathbf{W}_k^t$.

4.3.2 Performance Analysis

1) FL Accuracy under Attack: Figures 4.3a, 4.4a and 4.5a plot the accuracy of the local models under the proposed GAE-based, data-agnostic, model poisoning attack on the MNIST, Fashion-MNIST, and CIFAR-10 datasets, where there are five benign clients (i.e., $K = 5$) and 100 communication rounds for the FL. We compare the proposed attack with a model poisoning (MP) attack (Nguyen et al., 2020), in which the attacker manipulates the training process by injecting a fake client and sending fake local models to the server. For comparison purposes, Figure 4.6 plots the accuracy of the benign local models without any attacks. In this scenario, the accuracy of the client can be improved efficiently by FL and rapidly converge to 0.96.

In Figures 4.3a and 4.3b, we show that when using the MNIST dataset, the accuracy of all five clients under the proposed GAE-based attack gradually decreases and fluctuates dramatically. The performance of clients 1 and 2 drops from 0.75 to 0.55 and from 0.92 to 0.59, respectively. The accuracy of the model drops from 0.91 to 0.80 when exposed to the MP attack in which the

performance of the five clients follows a similar pattern. This is because the new GAE-based attack reconstructs the adversarial adjacency matrix according to the individual features of the clients. As a result, the attacker falsifies the local models to maximize the FL loss; see (4.8).

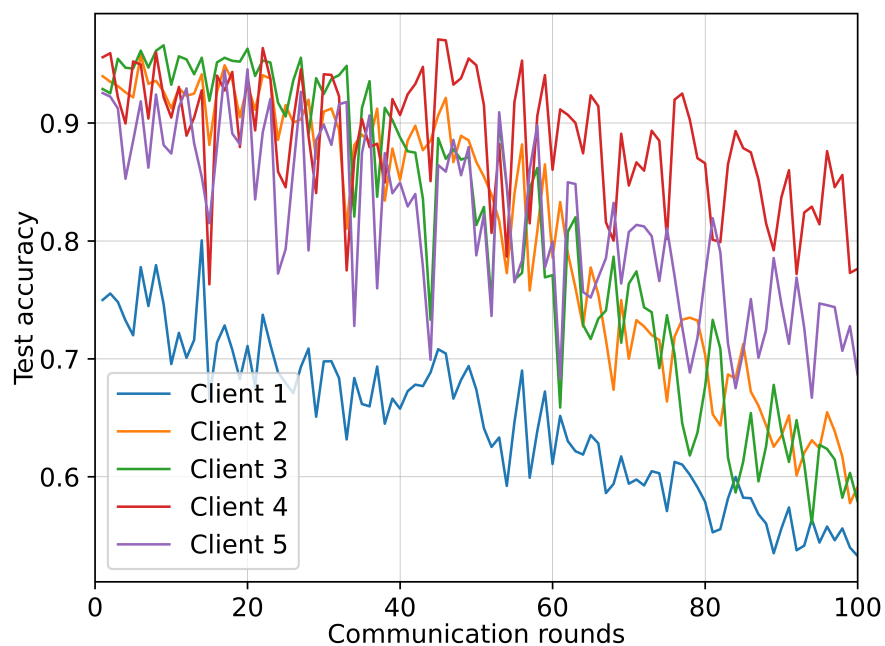
As shown in Figures 4.4a and 4.4b, the accuracy of client 3 and client 5 drops significantly by 0.37 and 0.24, respectively, when using the Fashion-MNIST dataset and the proposed GAE-based attack. However, while the accuracy of clients 1 and 2 may slightly increase, their convergence rates are greatly slowed in comparison to Figure 4.6. Additionally, the accuracy under the MP attack varies between 0.50 and 0.80, with a minimal decrease in accuracy observed. In Figure 4.5a, it can be seen that the proposed GAE-based attack with the CIFAR-10 dataset greatly hinders the performance of FL, as the accuracy of all four clients falls below 0.50. In contrast, the accuracy of all five clients under the MP attack is above 0.50, as shown in Figure 4.5b. Furthermore, it can be observed that the accuracy with the CIFAR-10 dataset is generally lower than the performance with the MNIST and Fashion-MNIST datasets. This is because the CIFAR-10 dataset contains a more diverse set of images, which makes it more challenging to differentiate and label, leading to lower overall accuracy.

Figure 4.7 illustrates the accuracy of the global model at the model aggregator. It can be observed that the proposed GAE-based attack hinders the training convergence when compared to the performance without the attack. As a result of the infection of the local FL model, the accuracy with the MNIST, Fashion-MNIST, or CIFAR-10 dataset fluctuates around 0.82, 0.81, or 0.50, respectively.

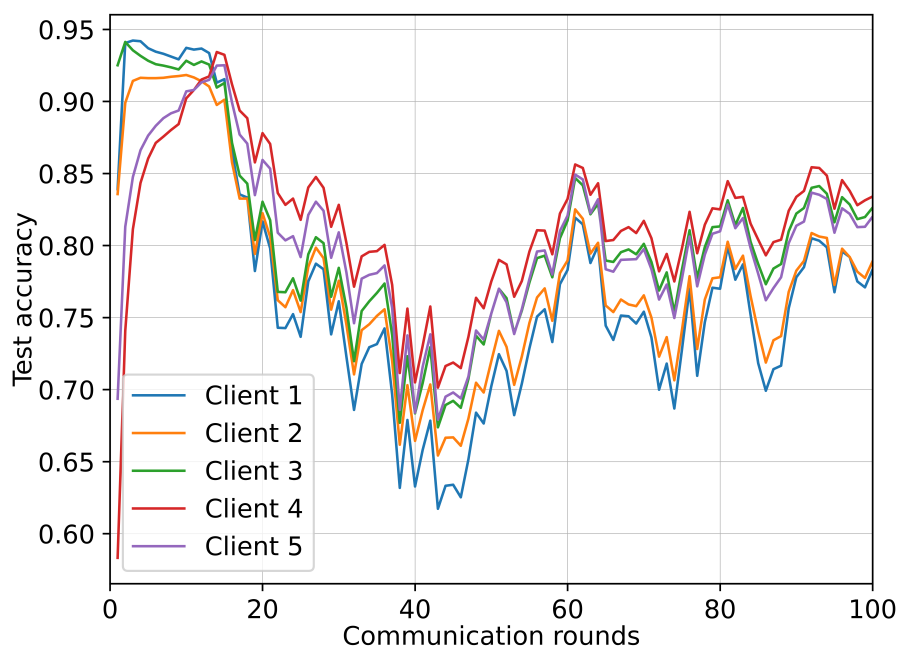
2) Detection of the Attack: Existing model poisoning attacks in FL aim to maximize the training loss of the FL model. One way to detect these malicious attacks is by comparing the Euclidean distance between the malicious local model and the global model to the distance between the benign local model and the global model. If the distance between the malicious local model and the global model is larger, it can be considered as an indication of a malicious attack and the server can detect it accordingly.

To evaluate the invisibility of the proposed GAE-based, data-agnostic, model poisoning attack Figures 4.8, 4.9 and 4.10 present the Euclidean distance between the local models and the global model on MNIST, Fashion-MNIST and CIFAR-10, with device 1 being the attacker. It can be observed that, in general, the local models with the MNIST dataset have the smallest distance compared to the other two datasets. This is expected as the handwritten digits in MNIST are relatively simple to recognize or falsify.

As shown in Figures 4.8a, 4.9a, and 4.10a, the Euclidean distances of the malicious local model (i.e., of client 5) generated by the GAE-based attack are below that of the benign local models. This makes it difficult for the aggregator to identify the attacker and defend against the attack. In contrast, the MP attack results in a significantly larger Euclidean distance between the malicious local model and the global model, making it easier to detect, as shown in Figures 4.8b, 4.9b, and 4.10b. This highlights the key advantage of the proposed GAE-based attack, which is designed to generate malicious local models based on the feature correlation between the benign local and global models, making the differences between the malicious local model and the benign

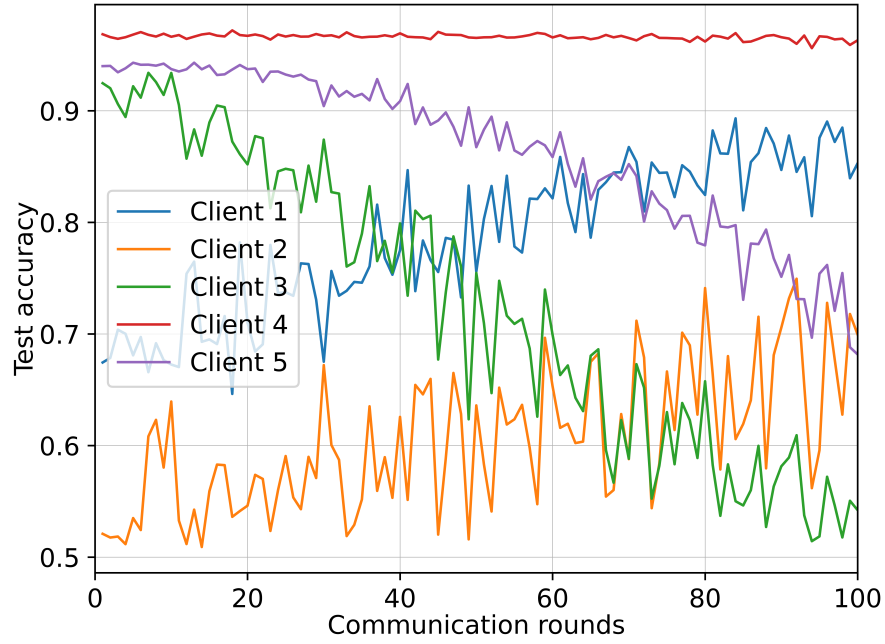


(a) The GAE-based attack with MNIST

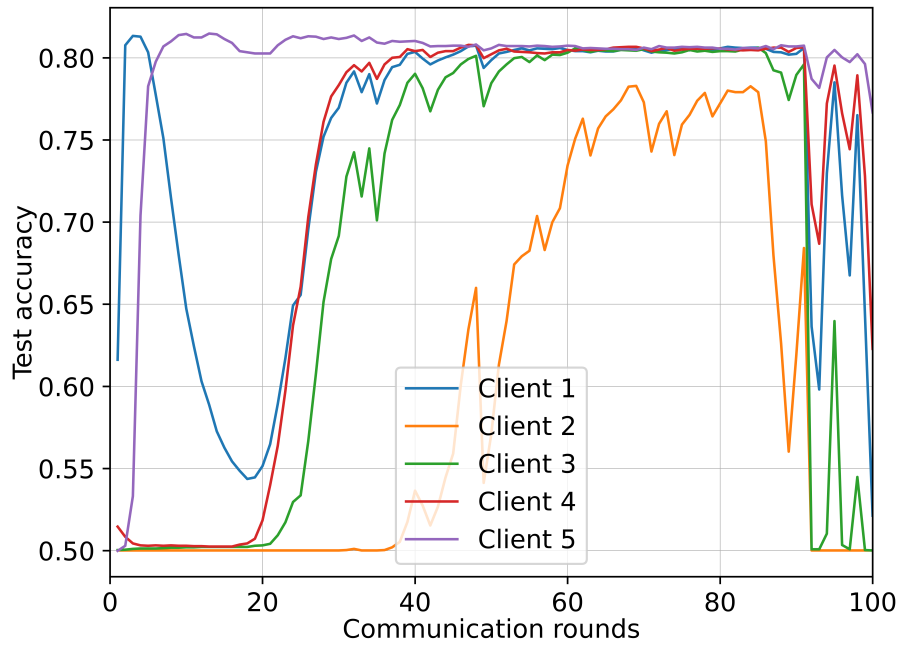


(b) The MP attack with MNIST

Figure 4.3: Given 100 FL communication rounds and five benign clients, we compare the local model testing accuracy under the GAE-based attack and the existing MP attack on the **MNIST** dataset.

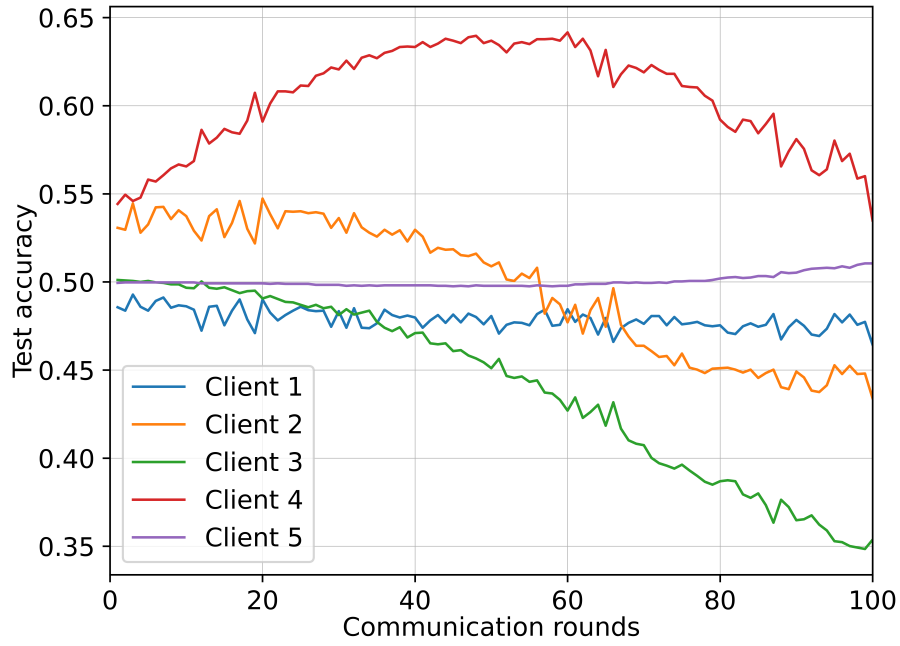


(a) The GAE-based attack with Fashion-MNIST

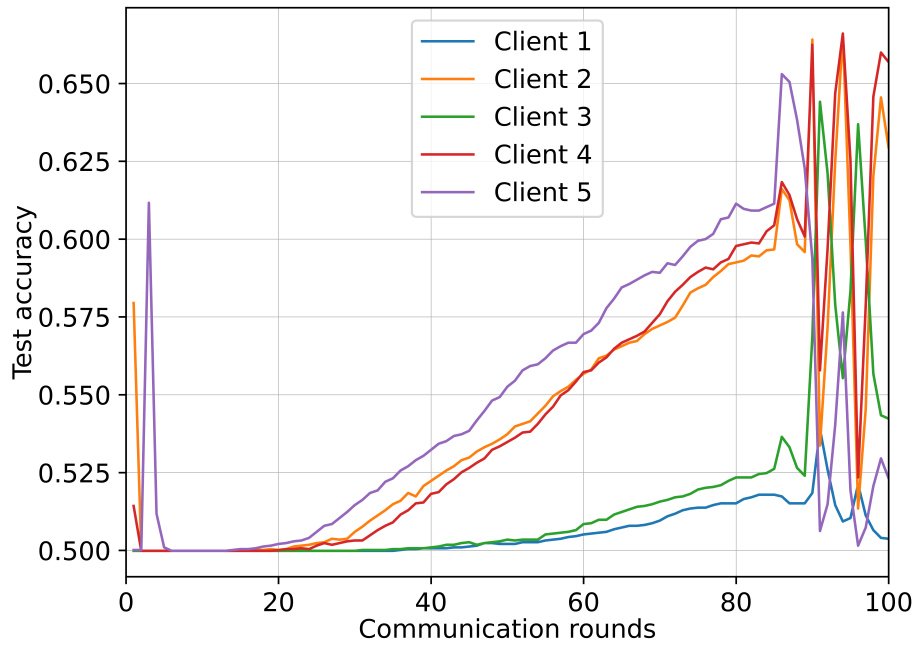


(b) The MP attack with Fashion-MNIST

Figure 4.4: Given 100 FL communication rounds and five benign clients, we compare the local model testing accuracy under the GAE-based attack and the existing MP attack on the **Fashion-MNIST** dataset.



(a) The GAE-based attack with CIFAR-10



(b) The MP attack with CIFAR-10

Figure 4.5: Given 100 FL communication rounds and five benign clients, we compare the local model testing accuracy under the GAE-based attack and the existing MP attack on the **CIFAR-10** dataset.

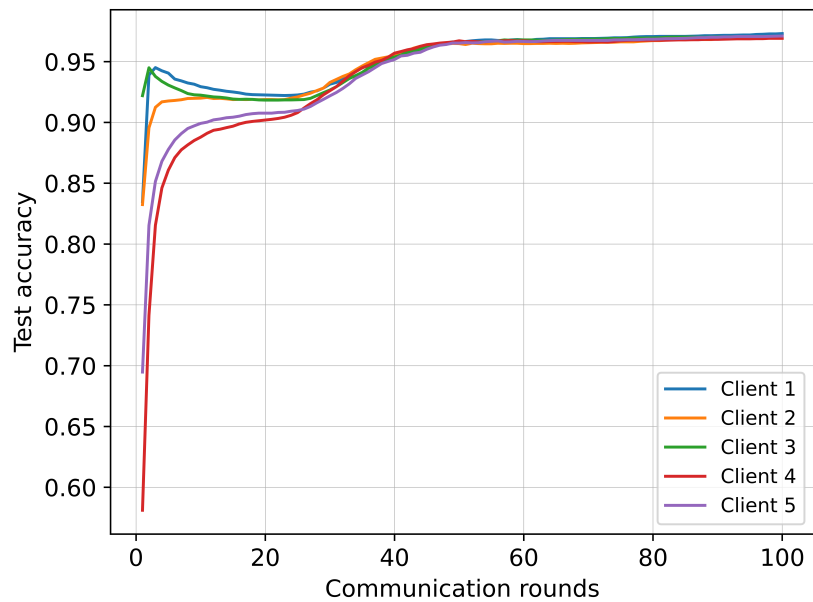


Figure 4.6: The global model accuracy with no attack.

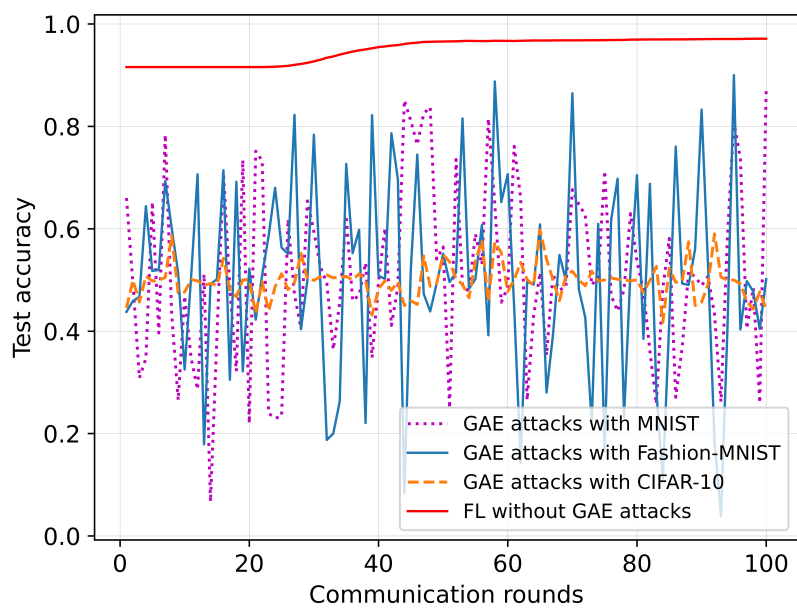
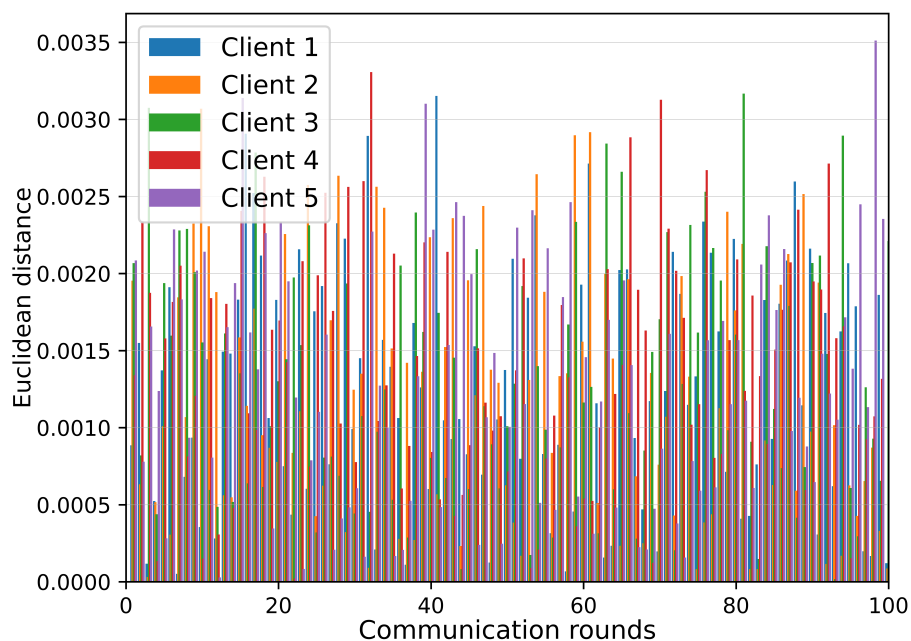
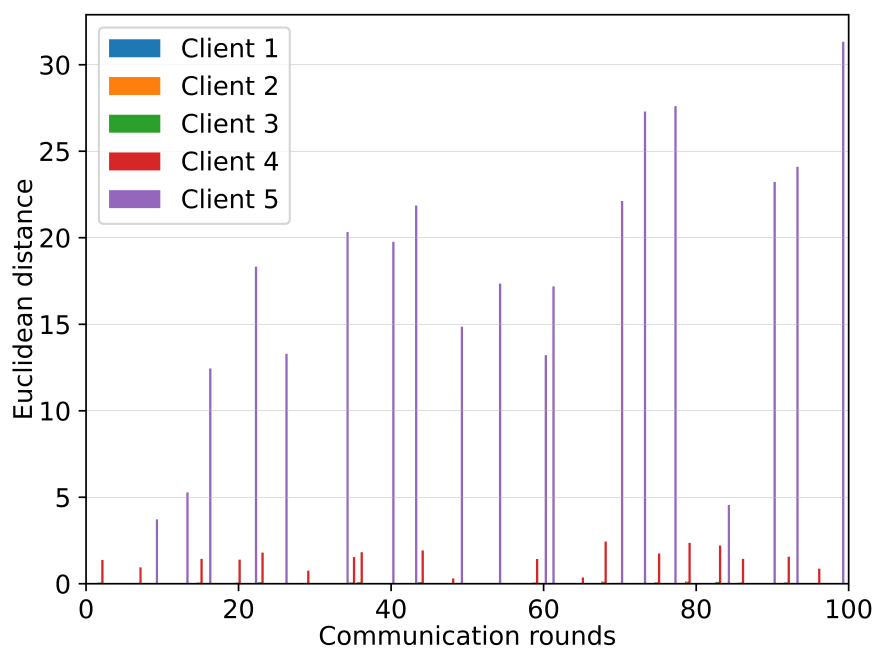


Figure 4.7: The global model accuracy under the GAE-based attack.

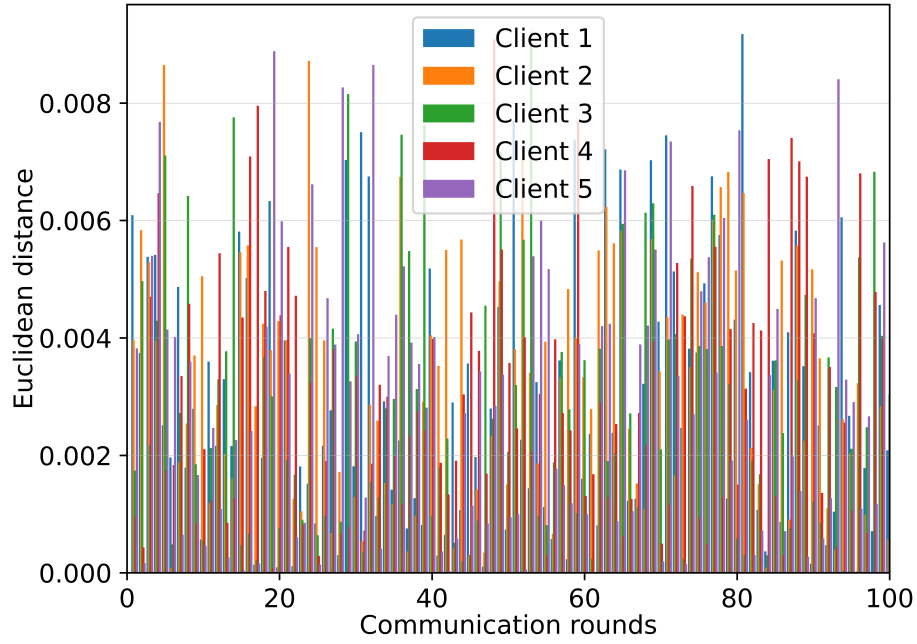


(a) The GAE-based attack with MNIST.

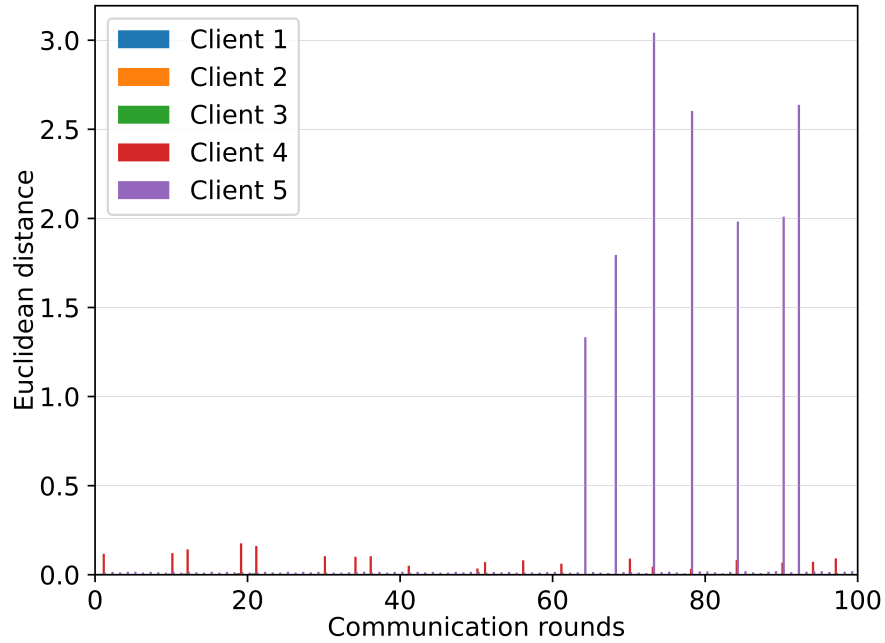


(b) The MP attack with MNIST.

Figure 4.8: Taking FL with five clients as an example, the Euclidean distances between the local models and the global model are presented, where client 5 is the attacker and launches the new GAE-based attack or the MP attack with **MNIST** dataset.

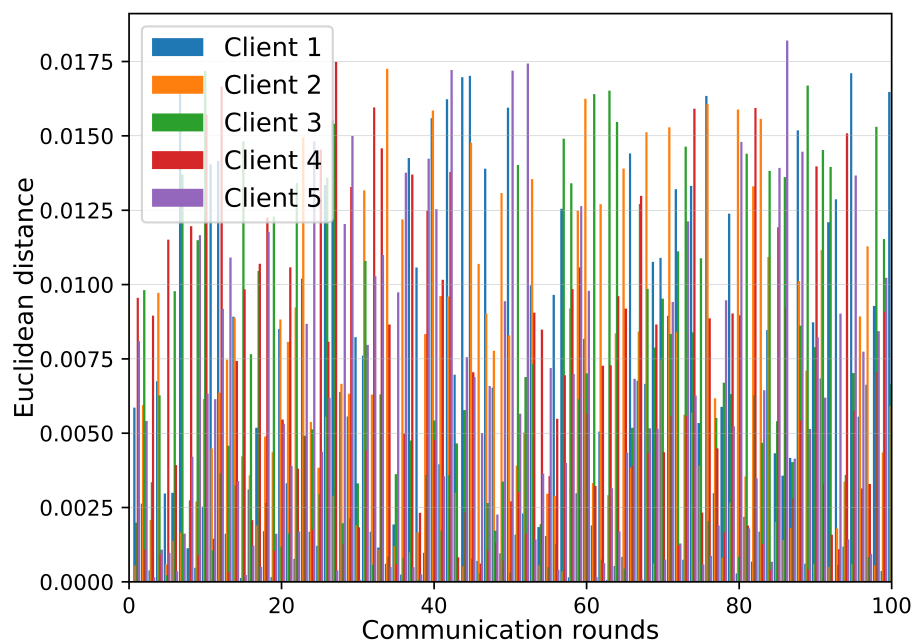


(a) The GAE-based attack with Fashion-MNIST.

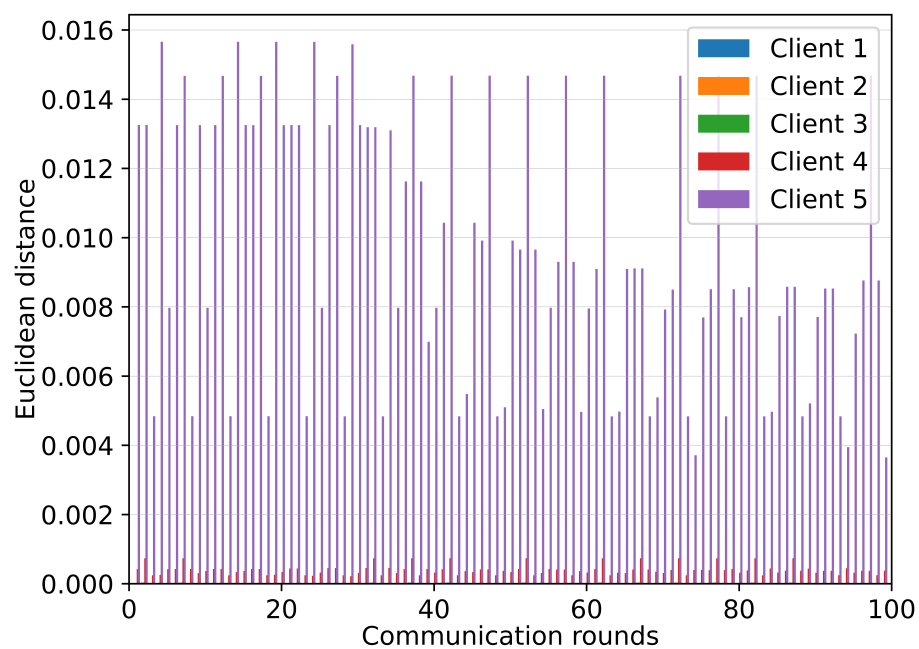


(b) The MP attack with Fashion-MNIST.

Figure 4.9: Taking FL with five clients as an example, the Euclidean distances between the local models and the global model are presented, where client 5 is the attacker and launches the new GAE-based attack or the MP attack with **Fashion-MNIST** dataset.



(a) The GAE-based attack with CIFAR-10.



(b) The MP attack with CIFAR-10.

Figure 4.10: Taking FL with five clients as an example, the Euclidean distances between the local models and the global model are presented, where client 5 is the attacker and launches the new GAE-based attack or the MP attack with **CIFAR-10** dataset.

local models indistinguishable.

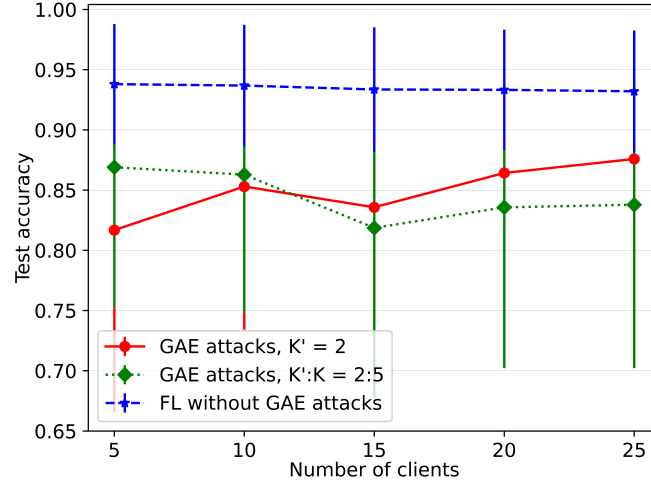


Figure 4.11: When the number of clients, i.e., K , increases from 5 to 25, the y-axis shows the average accuracy under the GAE-based attack based on the **MNIST** datasets. The number of attackers is $K' = 2$ by default. Otherwise, K' increases with K , while keeping the ratio $K' : K = 2 : 5$.

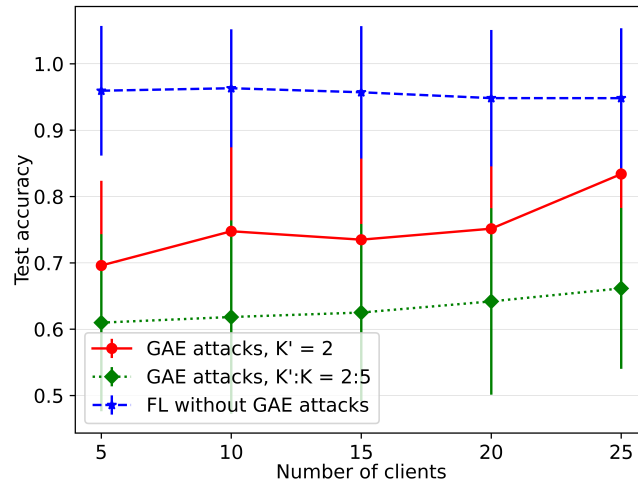


Figure 4.12: When the number of clients, i.e., K , increases from 5 to 25, the y-axis shows the average accuracy under the GAE-based attack based on the **Fashion-MNIST** datasets. The number of attackers is $K' = 2$ by default. Otherwise, K' increases with K , while keeping the ratio $K' : K = 2 : 5$.

3) Impact of Benign Local Model Number: Figures 4.11, 4.12 and 4.13 show the average accuracy of the local models based on the MNIST, Fashion-MNIST, and CIFAR-10 datasets, respectively. The total number of benign clients, i.e., K , increases from 5 to 25. The number of

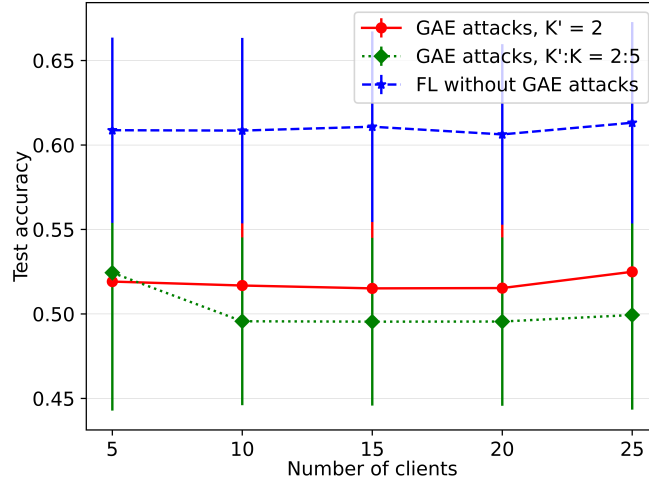


Figure 4.13: When the number of clients, i.e., K , increases from 5 to 25, the y-axis shows the average accuracy under the GAE-based attack based on the **CIFAR-10** datasets. The number of attackers is $K' = 2$ by default. Otherwise, K' increases with K , while keeping the ratio $K' : K = 2 : 5$.

attackers is set to $K' = 2$ by default. Otherwise, K' increases with K , while keeping the ratio of K' to K to be $K' : K = 2 : 5$. It is observed that the new GAE-based attack reduces the average accuracy below the performance of FL without the attack. As the number of clients increases, the average accuracy with MNIST, Fashion-MNIST, and CIFAR-10 drops about 0.2, 0.3, and 0.12, respectively, when $K = 15$. This suggests that the proposed GAE-based attack can effectively infect FL, regardless of the network size. It is also observed that the average accuracy with the three datasets gradually increases with K . This confirms that increasing the number of benign clients improves FL training through increased data diversity. This enhances the robustness of FL training, by partially counteracting the infection caused by the attack. In particular, the average accuracy of FL improves more notably when $K' = 2$, as proportionally more benign local models are involved in the training.

4.4 Chapter Summary and Concluding Remarks

In this chapter, we have investigated a new, data-agnostic, model poisoning attack to FL, where the proposed adversarial GAE gives rise to an infection of benign clients and the FL training accuracy gradually drops. The adversarial GAE allows the attacker to extract the common underlying data features of the benign local models as well as their correlations to generate the malicious model with which the FL training loss is maximized. Since the malicious and benign local models are indistinguishable, it is difficult to identify the GAE-based attack at the server. We implemented the GAE-based attack against the FL on SVM models using PyTorch. Performances were evaluated based on the MNIST, Fashion-MNIST, and CIFAR-10 datasets.

Regarding the prerequisites for eavesdropping local model updates, the attacker does not need to break transport layer security (TLS) or other secure channels to decode the benign local models overheard. In fact, GAE is adept at capturing complex relationships and structures inherent in graph-structured data. It can effectively encode graph information into a lower-dimensional latent space, while preserving the essential topological features of a graph. Using GAE, the attacker extracts the graph structure capturing the correlations between the benign local models (that could be transmitted over TLS protocol), and decouples the graph structure from underlying data features substantiating the local models. The attacker first regenerates manipulatively the graph structure to retain the structural features of the local models and maximize the FL training loss using the GAE, and then generates malicious local models using the regenerated graph structure to the data features of the benign local models. As a result, the malicious local models can effectively compromise the global model, while remaining compatible with the benign models and hence reasonably undetectable.

Last but not least, to ensure a fair comparison, we also implement SVMs in this chapter. Nevertheless, the choice of model architecture, neural networks (NNs) or SVMs, for training at benign clients, does not alter the underlying principle of the proposed data-agnostic model poisoning attack. This is because the proposed attack hinges on the creation of malevolent local models that, upon integration, deteriorate the performance of the global model by augmenting the FL training loss. It incorporates a new adversarial GAE designed to fabricate these deleterious local models by leveraging benign local models overheard. The adversarial GAE is adept at extracting and utilizing the correction features inherent in both the benign local models (which could be based on NN or SVM architectures) and the global model. We will implement NNs in the next chapter, our proposed corresponding defense strategies will also be given.

Chapter 5

GradCAM-AE: A New Shield Defense Against Poisoning Attacks on FL

To defend the GAE-based poisoning attacks we proposed in Chapter 4, which can circumvent widely adopted Euclidean distance-based detection methods and significantly degrade the FL accuracy, we propose a new defense mechanism, namely, GradCAM-AE, against model poisoning attacks on FL in this chapter. GradCAM-AE integrates Gradient-weighted Class Activation Mapping (GradCAM) and autoencoder (AE) to offer a substantially more powerful detection capability compared to existing Euclidean distance-based approaches. Specifically, GradCAM is applied at the server to create GradCAM heat maps for every uploaded model update. An autoencoder is applied to reconstruct the heat maps, while magnifying the discernible features of the heat maps. The reconstruction errors of the GradCAM heat maps are measured, and a threshold is created based on the statistics of the reconstruction errors. A reconstructed GradCAM heat map with a reconstruction error surpassing the threshold is categorized as atypical, and the corresponding model update as malicious. The rest of this chapter is organized as follows. Section 5.1 presents FL for image classification with deep neural networks. In Section 5.2, we detail the proposed GradCAM-AE against model poisoning attacks. Performance evaluation is presented in Section 5.3. This chapter is concluded in Section 5.4.

5.1 FL for Image Classification with Deep Neural Networks

In this section, we consider image classification as an example to depict the training process of FL. The system consists of a server and K benign clients where $\mathcal{K} \triangleq \{1, \dots, K\}$, as shown in Figure 5.1. Each benign client trains a Deep Neural Network (DNN) that is particularly well-suited for the image classification, which also enriches the work in Chapter 4 on FL models using SVMs instead of NNs. The DNN analyzes an input image (e.g., deer), extracts features (e.g., edges, corners, and textures) through several convolutional layers, feeds the feature maps into fully connected (FC) layers, and assigns one of the labels (such as “bird”, “dog”, “horse” or “deer”). The DNN model is learned based on training data and represented by the parameter vector $\mathbf{W}$ (weights and biases).

The client possesses D_k training data pairs $\{(\mathbf{x}_s, y_s)\}_{s=1}^{D_k}$, consisting of a feature vector $\mathbf{x}_s$ (e.g., the pixels of an image) as the input of the model and a corresponding scalar value y_s (e.g., the real label of the image) that is the desired output of the model (Wang et al., 2019b). The dataset of the client is denoted by $\mathcal{D}_k$, and we have

$$\mathcal{D}_k = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_s, y_s), \dots, (\mathbf{x}_{D_k}, y_{D_k})\}, \quad (5.1)$$

where D_k denotes the cardinality of $\mathcal{D}_k$, i.e., $D_k \triangleq |\mathcal{D}_k|$. Note that the local data size remains fixed with every communication round, which is different from the time-varying local data size described in Chapter 3. The FL training process repeats until convergence is achieved or the maximum number of communication rounds T_{FL} is met, which is conform to Section 3.1.1. Without loss of generality, we assume all K clients are involved in FL.

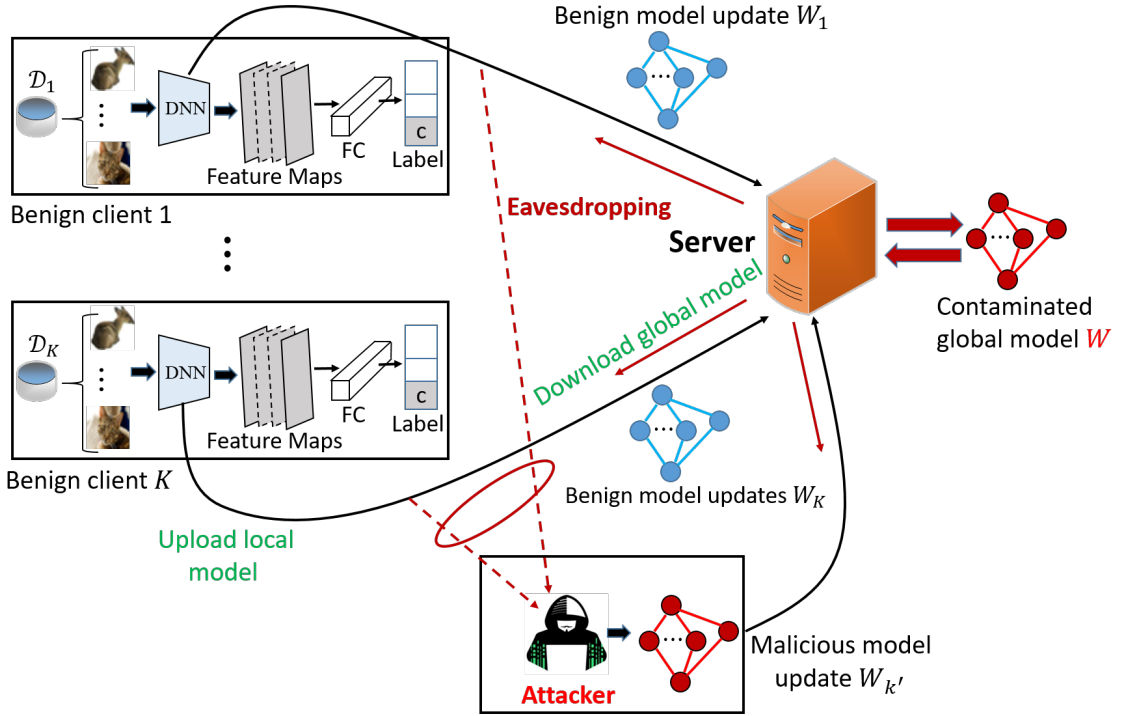


Figure 5.1: FL with poisoning attack, where the attacker eavesdrops on the benign local models uploaded by the clients. The attacker fabricates a malicious local model update based on its eavesdropped local model updates. The Euclidean distances between the malicious and benign local models fall within a preset threshold distance.

Threat Model. Malicious clients may attempt to compromise the global and local models by injecting poisoned data into their local models or intentionally tampering with the local model parameters during training. We consider that the model poisoning attacks are launched by the attacker whose goal is to impair or degrade the accuracy and convergence of FL. The detailed threat model we have described in Section 4.1.1, where an GAE is designed to generate malicious local models solely based on the benign local models overheard and capturing the correlation

features of the benign local and global models.

5.2 Proposed GradCAM-AE against Model Poisoning Attacks

5.2.1 GradCAM-AE Architecture

On the client side, each of the K benign clients utilizes the DNN tailored specifically for image classification tasks, as shown in Figure 5.2. The DNN model extracts relevant features from an input image (e.g., a bird) and subsequently maps them to the corresponding classes. The architecture of a typical DNN comprises multiple layers, each with a specific function. (a) The first layer is a convolutional layer, which applies a set of filters to the input image, thereby extracting intrinsic features, such as edges, corners, and textures. The output of the convolutional layer is a set of feature maps, each representing different aspects of the image. (b) The following layer is a pooling layer, which reduces the dimensionality of the feature maps while retaining essential information, such as the salient features of the bird’s beak or feather. Various pooling methods can be employed, including max pooling or average pooling, all with the objective of downsampling the feature maps while retaining their salient features. Some DNN architectures may differ from traditional pooling operations. For example, models like SqueezeNet (Iandola et al., 2016), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), and MobileNet (Howard et al., 2019) employ alternative strategies. (c) Subsequent to several convolution and pooling layers, the extracted features are fed into one or multiple fully connected layers, where the final classification is performed.

Upon receiving the local model updates from the devices, the server aggregates the local models, where the benign local models can be mingled with malicious local models. We aim to design a defense countermeasure that can pinpoint and filter malicious devices. GradCAM (Selvaraju et al., 2017) visualization is distinguished by its high resolution and high-class discriminative ability compared to other methods, e.g., Class Activation Mapping (Zhou et al., 2016). GradCAM is leveraged in our design to detect malicious local model updates. Specifically, the server randomly picks an image from the global model testing dataset that incorporates all categories of the devices dataset as input and passes through the convolutional layers with weight and bias parameters that are replaced by each model update $\mathbf{W}_l, l \in \mathcal{H} \cup \mathcal{M}$ (l is the index of local model updates, including benign and malicious), obtains the feature maps¹ A_l that has n channels. The feature maps A_l are fed into a fully connected layer for final classification.

To obtain the class discriminative localization map of $L_{l, \text{GradCAM}}^{(c)} \in \mathbb{R}^{B \times H}$ with width B and height H for any class c , GradCAM first computes the gradient of the score for class c , $Y^{(c)}$ (before softmax) with respect to each feature map A_l^m of a convolutional layer, i.e., $\frac{\partial Y^{(c)}}{\partial A_l^m}$. $m \in [1, n]$ is the index of channels. The neuron importance weights $\alpha_{l,m}^{(c)}$ can be obtained through global average

¹In this work, we focus on extracting the feature maps after the final convolution layer of deep learning model in that can capture high-level features and hold information regarding the important regions of the input image.

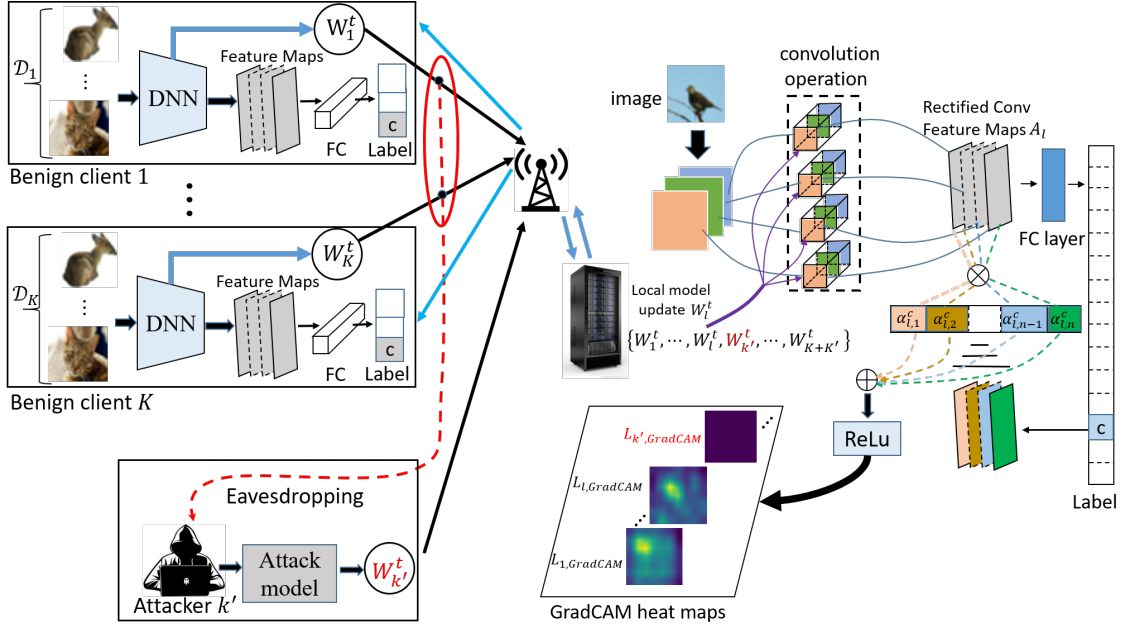


Figure 5.2: GradCAM-assisted defense against poisoning attacks on FL. The server arbitrarily selects an image (e.g., an image with the label “bird”) from the global model testing dataset to create GradCAM heat maps for every uploaded local model update. These GradCAM heat maps flow into an autoencoder for malicious model detection.

pooling:

$$\alpha_{l,m}^{(c)} = \frac{1}{B \times H} \sum_{i=1}^B \sum_{j=1}^H \frac{\partial Y^{(c)}}{\partial A_l^m(i, j)}, \forall l \in \mathcal{K} \cup \mathcal{M}, \quad (5.2)$$

where $A_l^m(i, j)$ is the activation at location (i, j) of the feature map A_l^m . We apply a rectified linear unit (ReLU) to the weighted linear combination of the forward activation. $L_{l, GradCAM}^{(c)}$ is given by

$$L_{l, GradCAM}^{(c)} = ReLU\left(\sum_{m=1}^n \alpha_{l,m}^{(c)} A_l^m\right). \quad (5.3)$$

The server uses the image and local model updates to obtain a single-channel GradCAM heat map that is input into the autoencoder for identification. For conciseness, we suppress the indicator of classes c and rewrite $L_{l, GradCAM}^{(c)}$ as $L_{l, GradCAM}$ in what follows.

5.2.2 Autoencoder for Malicious GradCAM Heat Map Identification

Considering all GradCAM heat maps are unlabeled, an autoencoder is an efficient tool in unsupervised learning to discover non-linear features across anomaly detection systems (Zhou and Paffenroth, 2017; Hojjati and Armanfard, 2023; Batzner et al., 2024). We use an autoencoder to pinpoint abnormal GradCAM heat maps corresponding to malicious local model updates uploaded by the attackers.

As shown in Figure 5.3, a canonical autoencoder consists primarily of three components: an encoder, a code (or a latent space) and a decoder. Each GradCAM heat map $L_{l, GradCAM}$ with size

of $H \times B$ is flattened into a vector with size of $1 \times H \times B$, which is further concatenated with the vectors of other GradCAM heat maps to form $\mathbf{L}_{\text{GradCAM}}$ as the input to the encoder. During the autoencoder training, the encoder $e_\theta(\mathbf{L}_{\text{GradCAM}})$ with parameter θ compresses the GradCAM heat maps from a high-dimensional space to a low-dimensional space $z = e_\theta(\mathbf{L}_{\text{GradCAM}})$, also called the code or the latent space. The code learns the underlying features or representation of the GradCAM heat maps, which are input to the decoder $d_\phi(z)$ with parameter ϕ . The decoder further reconstructs the input GradCAM heat maps from the code, i.e., $d_\phi(z) = \mathbf{L}'_{\text{GradCAM}} = d_\phi(e_\theta(\mathbf{L}_{\text{GradCAM}}))$. After the training of (θ, ϕ) , the reconstructed GradCAM heat maps are reshaped into the same size as the original GradCAM heat maps.

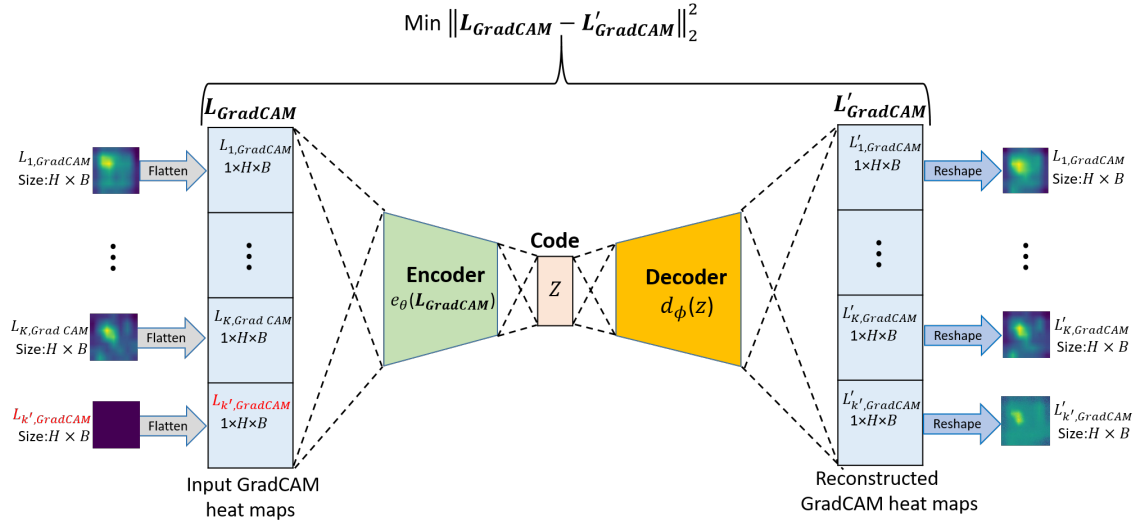


Figure 5.3: Autoencoder-based abnormal GradCAM heat map identification. The server flattens and concatenates GradCAM heat maps as input to the encoder neural network, which compresses the GradCAM heat maps from a high dimension to a low dimension Z . The decoder neural network takes Z as its input to reconstruct the original input GradCAM heat maps.

Loss Function. To minimize the difference between the original input GradCAM heat maps and reconstructed GradCAM heat maps, the autoencoder loss function is defined as the mean squared error (MSE) between the encoder input GradCAM heat maps $\mathbf{L}_{\text{GradCAM}}$, and the decoder reconstructed GradCAM heat maps $\mathbf{L}'_{\text{GradCAM}}$, i.e.,

$$\begin{aligned}
 L(\theta, \phi) &= \min_{\theta, \phi} \frac{1}{|K| + |K'|} \|\mathbf{L}_{\text{GradCAM}} - \mathbf{L}'_{\text{GradCAM}}\|_2^2 \\
 &= \min_{\theta, \phi} \frac{1}{K + K'} \sum_{l=1}^{K+K'} \|L_{l, \text{GradCAM}} - d_\phi(e_\theta(L_{l, \text{GradCAM}}))\|_2^2.
 \end{aligned} \tag{5.4}$$

Once the autoencoder completes training, the server computes the reconstruction errors between each reconstructed GradCAM heat map and its corresponding input GradCAM heat map and obtains the mean reconstruction error, i.e., $\forall l \in \mathcal{K} \cup \mathcal{M}$,

$$R_l = \frac{\sum_{i=1}^B \sum_{j=1}^H |L_{l, \text{GradCAM}}(i, j) - L'_{l, \text{GradCAM}}(i, j)|}{H \times B}. \quad (5.5)$$

The average reconstruction error of all GradCAM heat maps is

$$\bar{R} = \frac{1}{R_l} \sum_{l=1}^{K+K'} R_l. \quad (5.6)$$

A threshold δ is defined as

$$\delta = \bar{R} + \alpha \times \sqrt{\frac{\sum_{l=1}^{K+K'} (R_l - \bar{R})^2}{K + K'}}, \quad (5.7)$$

where α is an empirically configured efficient. Here, δ is used to distinguish between the benign and malicious GradCAM heat maps. If the mean reconstruction error of each GradCAM heat map is greater than the threshold δ , the corresponding input of the GradCAM heat map is considered potentially abnormal. Otherwise, it is a potential normal GradCAM heat map because the AE learns to capture variations in normal GradCAM heat maps during training. The AE can encounter difficulties in handling anomalies that do not conform to the learned patterns.

Note that the AE is optimized to minimize the reconstruction errors between the input GradCAM heat maps and the reconstructed GradCAM heat maps during training. In other words, the AE learns to reconstruct normal GradCAM heat maps. It is reasonable to expect that the GradCAM heat maps that can be reconstructed with low reconstruction errors are considered normal, while those with high reconstruction errors are potentially abnormal.

Also note that δ is time-varying with communication rounds as the GradCAM heat maps change over the communication rounds. The detection results are stored in a buffer:

$$O_l^t = \begin{cases} 1, & \text{if } R_l \leq \delta; \\ 0, & \text{if } R_l > \delta, \end{cases} \quad (5.8)$$

where “1” and “0” indicate a benign and a malicious local model update, respectively.

A voting mechanism is further designed to further reduce the possibility of misclassifying a benign local model update. The key idea is that the server makes final decisions every fixed ξ communication rounds to determine which devices are malicious. Specifically, the local model updates ε out of ξ communication rounds are detected as potentially malicious. They are determined malicious and removed from global aggregations. Finally, (4.1) is rewritten as

$$\mathbf{W}_g^{t+1} = \begin{cases} \sum_{l \in \mathcal{K} \cup \mathcal{M}} O_l^t \times \frac{D_l}{S} \times \mathbf{W}_l^t, & \text{if } t \text{ modulo } \xi \neq 0; \\ \sum_{l \in \mathcal{K} \cup \mathcal{M}} Y_l^t \times \frac{D_l}{S} \times \mathbf{W}_l^t, & \text{if } t \text{ modulo } \xi = 0, \end{cases} \quad (5.9)$$

where Y_l^t is given by

$$Y_l^t = \begin{cases} 1, & \text{if } \sum_{t=N\xi-\xi+1}^{N\xi} O_l^t > \varepsilon, N = 1, \dots, \lfloor \frac{T_{FL}}{\xi} \rfloor; \\ 0, & \text{if } \sum_{t=N\xi-\xi+1}^{N\xi} O_l^t \leq \varepsilon, N = 1, \dots, \lfloor \frac{T_{FL}}{\xi} \rfloor. \end{cases} \quad (5.10)$$

The updated global model $\mathbf{W}_g^{t+1}$ is sent back to all clients. The detailed pseudo-code is given in Algorithm 4.

5.3 Performance Evaluation

5.3.1 Experiment Setup

Datasets. Two datasets, i.e., CIFAR-10 (Krizhevsky et al., 2009) and GTSRB (Stallkamp et al., 2012), are used to evaluate the performance of our proposed GradCAM-AE.

- **CIFAR-10:** It consists of 60000 32 low-resolution color images in 10 different classes, with 6000 images per class. There are 50,000 training images allocated to all benign clients for local model training, and 10,000 test images are allocated to the server for the global model testing until the end of each communication round.
- **GTSRB:** It contains 43 different classes of German traffic signs. The images range in size from 15 pixels to 250 pixels. There are 39209 training images, and 12630 test images are assigned to all local clients for training and the server for testing. Notice that we resize the images of different sizes to a fixed resolution of 128 pixels for training and testing.

Benchmarks. We consider the following state-of-the-art defense schemes, i.e., Multi-Krum (Blanchard et al., 2017), AUROR (Shen et al., 2016), Trimmed-mean (Yin et al., 2018), and FAA-DL (Shi et al., 2022). In addition, we study a GradCAM-Krum anomaly detection scheme in which Krum is applied to identify the malicious GradCAM heat maps that are generated at the server.

Experimental setting. The total communication rounds $T_{FL} = 100$. For each communication round, 18 benign clients train the local models for 20 epochs using the Adam optimizer, with a batch size of 64, a learning rate of 0.0001, and a weight decay of 0. The autoencoder trains the GradCAM heat maps for 200 epochs using the Adam optimizer, with a hidden-layer size of 128, a learning rate of 0.001, and a weight decay of 0.00001. We conduct the experiments on a single Zotac GeForce RTX 4090 with 24GB GDDR6X memory.

As far as the detection performance of defense methods is concerned, Recall (the proportion of actual malicious clients that were correctly identified by the defense model out of all malicious clients), Precision (the proportion of correctly predicted malicious clients in all predicted malicious clients by the defense model), FPR (false positive rate, the proportion of actual benign clients that were incorrectly classified as malicious by FedCAMAE out of all benign clients), ACC (Accuracy, the proportion of correctly classified clients in the total number of clients), and F1

Algorithm 4: The proposed GradCAM-AE for detecting malicious local model updates on FL

input : $\mathcal{K}$ - the set of benign clients; $\mathcal{M}$ - the set of attackers; T_{FL} - the total number of communication round; L - the number of local epochs; $\mathcal{D}_k$ - local data size of client k ; η_t - the learning rate of clients; $\mathbf{W}^0$ - the initialized global model; $\mathbf{M}$ - Buffer; ξ - Buffer size; ε - voting threshold; I_0 - An image picked from testing dataset.

output: The final global model weight $\mathbf{W}^{T_{FL}}$.

```

1 for  $t = 1, \dots, T_{FL}$  do
2   Server broadcasts  $\mathbf{W}^t$  to all clients;
3    $\mathbf{M} \leftarrow \mathbf{0}, \mathbf{M} \in \mathbf{R}^{\xi \times (K+K')}$  ;
4   for  $k \in \mathcal{K}$  do
5      $\mathbf{W}_k^t \leftarrow \text{LocalUpdate}(k, \mathbf{W}^t)$  ;
6   Solve (4.3a) with constraint (4.3);
7    $(\mathbf{W}_{1'}^t)^*, \dots, (\mathbf{W}_{K'}^t)^* \leftarrow (4.4)$ .
8   if  $t \bmod \xi \neq 0$  then
9      $A_l \leftarrow \mathbf{W}_l^{t+1}$  and  $I_0$  perform convolution operation ;
10     $L_{l, \text{GradCAM}} \leftarrow$  according to (5.2) and (5.3) ;
11     $\mathbf{L}_{\text{GradCAM}} \leftarrow$  concatenate  $(L_{l, \text{GradCAM}})$  ;
12     $\mathbf{L}'_{\text{GradCAM}} \leftarrow \text{Autoencoder}(\mathbf{L}_{\text{GradCAM}})$  ;
13     $O_l^t \leftarrow (5.8)$  ;
14    Store  $O_l^t$  into  $\mathbf{M}$  ;
15  else
16     $A_l \leftarrow \mathbf{W}_l^{t+1}$  and  $I_0$  perform convolution operation ;
17     $L_{l, \text{GradCAM}} \leftarrow$  according to (5.2) and (5.3) ;
18     $\mathbf{L}_{\text{GradCAM}} \leftarrow$  concatenate  $(L_{l, \text{GradCAM}})$  ;
19     $\mathbf{L}'_{\text{GradCAM}} \leftarrow \text{Autoencoder}(\mathbf{L}_{\text{GradCAM}})$  ;
20     $O_l^t \leftarrow (5.8)$  ;
21    Store  $O_l^t$  into  $\mathbf{M}$  ;
22     $\text{Stat} \leftarrow \text{sum}(\mathbf{M}, \text{axis} = 0)$  ;
23    for  $\text{stat}(l) \in \text{Stat}$  do
24      if  $\text{stat}(l) \leq \varepsilon$  then
25         $Y_l^t = 0$ ;
26      else
27         $Y_l^t = 1$ ;
28    Server updates the global model according to (5.9) & (5.10).
29   $\text{LocalUpdate}(k, \mathbf{W}^t)$  ;
30 for  $\ell = 1, \dots, L$  do
31   for  $b_k \in \mathcal{D}_k$  do
32     (3.4).

```

Table 5.1: Detection rates of ResNet-18 on IID and Non-IID CIFAR-10.

ResNet-18	IID CIFAR-10					Non-IID CIFAR-10				
Methods	Recall	Precision	ACC	F1 score	AUC	Recall	Precision	ACC	F1 score	AUC
Multi-Krum	0.045	0.09	0.859	0.06	0.497	0.085	0.085	0.817	0.085	0.492
AUROR	0.015	0.006	0.736	0.009	0.416	0.035	0.017	0.75	0.023	0.432
Trimmed-mean	0.08	0.08	0.816	0.08	0.489	0.09	0.09	0.818	0.09	0.494
FAA-DL	0.61	0.098	0.412	0.168	0.50	0.55	0.109	0.514	0.182	0.53
GradCAM-Krum	1.0	1.0	1.0	1.0	1.0	0.625	0.625	0.925	0.625	0.792
GradCAM-AE	1.0	1.0	1.0	1.0	1.0	1.0	0.943	0.986	0.961	0.992

score (the harmonic mean of Precision and Recall) are taken as performance measures, which can be obtained by relevant calculation through a confusion matrix. Moreover, we further calculate the second level detection index AUC (area under the receiver operating characteristic curve, a single scalar value between 0 and 1 that summarizes the overall ability of the defense model to discriminate between the malicious clients and benign clients, a larger value means the defense model is more effective at separating the malicious clients and benign clients) to measure the detection performance of the defense approaches. Unless otherwise specified, the number of attackers is fixed in each communication round.

5.3.2 Performance under IID Datasets

Figure 5.4 shows the comparison of FL test accuracy under our proposed GradCAM-AE defense framework with existing benchmarks, where the DNN model of ResNet-18 performs image classification on CIFAR-10 and GTSRB with IID settings, respectively. GradCAM-AE can achieve the highest test accuracy of the global model as the autoencoder can accurately exclude the abnormal GradCAM heat maps such that more benign local model updates are involved in the FL training process. While GradCAM-Krum only selects an optimal local model update as the global model to train FL, it also achieves higher test accuracy (0.79) compared with other benchmarks. The reason is that the data is IID, and the difference among the local model updates is not significant. Multi-Krum, AUROR and Trimmed-mean all directly detect millions, even many more, parameters of local model updates based on Euclidean distance. The Euclidean distance between the crafted malicious local model updates and benign ones is within the threshold designated by the server. This indicates that the malicious local model updates can elude the detection of the server and participate in the FL training process through multiple communication rounds, resulting in the global model being corrupted. Table 5.1 shows that although the ACCs of Euclidean distance-based defense schemes are relatively high (0.859, 0.736 and 0.816), the Precision is close to zero. In other words, they can correctly classify a few benign clients but fail to identify malicious clients. Similar performances are observed on GTSRB dataset, the detection rates of GrdaCAM-Krum still outperform the benchmarks, as shown in Table 5.2. The detailed detection rates of MobileNetV3-Large on CIFAR-10 and GTSRB with IID settings, are given in Table 5.3 and Table 5.4 respectively.

FAA-DL, a one-class SVM-based method, also directly classifies the local model updates aggregated by the server as benign and malicious. However, there are two key reasons why FAA-DL fails in the experiments. First, the local model updates have the characteristics of high-dimensional

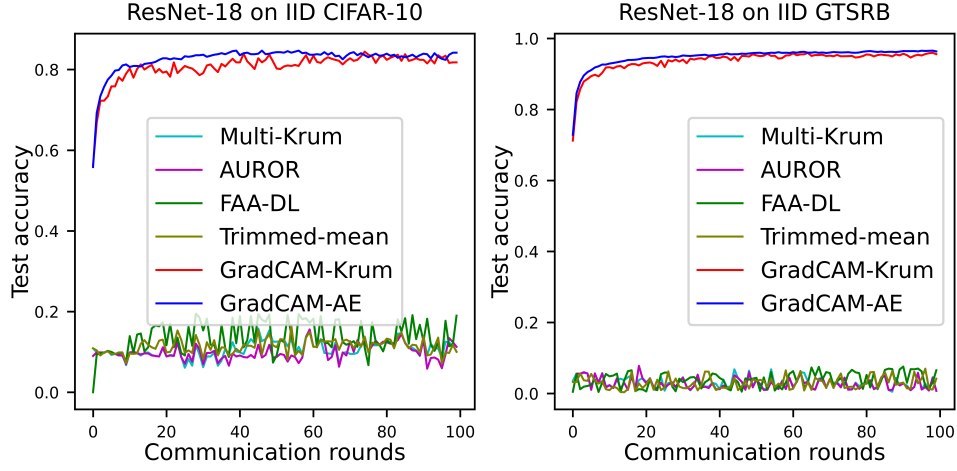


Figure 5.4: Test accuracy vs. communication rounds for **ResNet-18** on IID CIFAR-10 and GT-SRB.

Table 5.2: Detection rates of ResNet-18 on IID and Non-IID GT-SRB.

ResNet-18	IID GT-SRB					Non-IID GT-SRB				
	Recall	Precision	ACC	F1 score	AUC	Recall	Precision	ACC	F1 score	AUC
Multi-Krum	0.045	0.09	0.859	0.06	0.497	0.045	0.08	0.857	0.06	0.497
AUROR	0.035	0.016	0.739	0.02	0.427	0.035	0.016	0.755	0.0229	0.435
Trimmed-mean	0.08	0.08	0.816	0.08	0.489	0.07	0.07	0.814	0.07	0.483
FAA-DL	0.685	0.092	0.295	0.162	0.468	0.43	0.093	0.517	0.152	0.478
GradCAM-Krum	0.625	0.625	0.925	0.625	0.792	0.55	0.55	0.91	0.55	0.75
GradCAM-AE	1.0	1.0	1.0	1.0	1.0	1.0	0.990	0.998	0.994	0.999

Table 5.3: Detection rates of MobileNetV3-Large on IID and Non-IID CIFAR-10.

MobileNetV3-Large	IID CIFAR-10					Non-IID CIFAR-10				
	Recall	Precision	ACC	F1 score	AUC	Recall	Precision	ACC	F1 score	AUC
Multi-Krum	0.05	0.1	0.860	0.06	0.5	0.035	0.07	0.857	0.047	0.492
AUROR	0.02	0.0095	0.74	0.013	0.42	0.015	0.006	0.736	0.009	0.416
Trimmed-mean	0.1	0.1	0.82	0.1	0.50	0.50	0.08	0.816	0.08	0.489
FAA-DL	0.735	0.092	0.257	0.163	0.469	0.7	0.094	0.309	0.166	0.483
GradCAM-Krum	0.86	0.86	0.972	0.86	0.922	0.94	0.94	0.988	0.94	0.967
GradCAM-AE	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 5.4: Detection rates of MobileNetV3-Large on IID and Non-IID GT-SRB.

MobileNetV3-Large	IID GT-SRB					Non-IID GT-SRB				
	Recall	Precision	ACC	F1 score	AUC	Recall	Precision	ACC	F1 score	AUC
Multi-Krum	0.03	0.06	0.856	0.04	0.489	0.135	0.135	0.827	0.135	0.519
AUROR	0.035	0.019	0.745	0.02	0.43	0.015	0.0083	0.7375	0.011	0.416
Trimmed-mean	0.08	0.08	0.816	0.08	0.489	0.065	0.065	0.813	0.065	0.481
FAA-DL	0.79	0.096	0.236	0.17	0.483	0.765	0.098	0.28	0.173	0.496
GradCAM-Krum	0.995	0.995	0.999	0.995	0.997	0.58	0.58	0.916	0.58	0.767
GradCAM-AE	1.0	1.0	1.0	1.0	1.0	1.0	0.990	0.998	0.994	0.998

feature spaces. The curse of dimensionality can lead to data sparsity, making it challenging for FAA-DL to find a suitable margin that separates benign from malicious local model updates. Second, FAA-DL is sensitive to class imbalance (18 benign local model updates and two malicious local model updates), which means the FAA-DL biases its decision boundary towards the majority class, making it struggle to detect malicious local model updates effectively. Notwithstanding, we replace the DNN model with MobileNetV3-Large. Similar performances are observed; see Figure 5.5. This demonstrates that GradCAM-AE is applicable to different DNN models for detecting malicious local model updates.

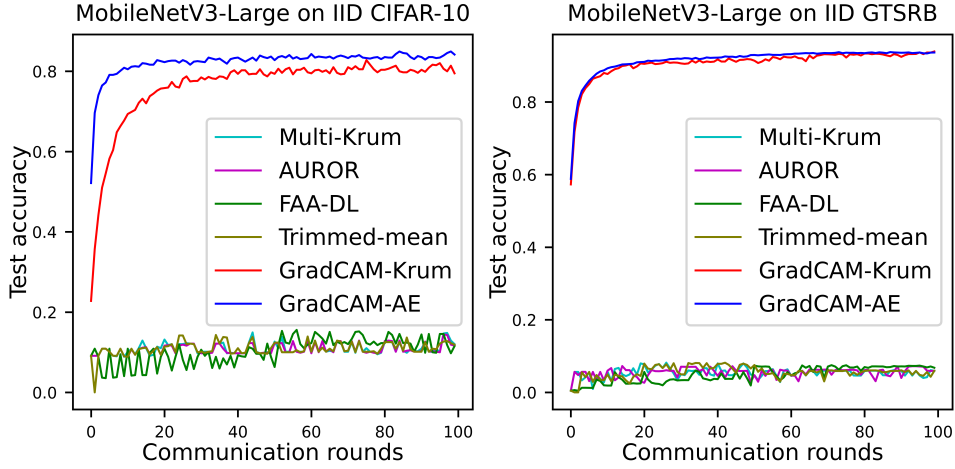


Figure 5.5: Test accuracy vs. communication rounds for **MobileNetV3-Large** on IID CIFAR-10 and GTSRB.

5.3.3 Performance under Non-IID Datasets

Compared with IID, the Non-IID setting is much more challenging for the server to identify the malicious local model updates. This reason is that the local model updates tend to be more divergent under non-IID settings; in turn, the generated heat maps on the server are more diverse. To simulate the statistical data heterogeneity in practical FL scenarios, we adopt one of the Non-IID partition schemes, i.e., latent Dirichlet allocation (LDA), which are widely used in the latest literature (Zhang et al., 2023b; Wang et al., 2022a). As a biased probability, the β parameter of LDA controls the distribution difference of the local training data. A larger β denotes a higher level of Non-IID among the local training data. Here, we set $\beta = 0.5$.

Figure 5.6 illustrates the test accuracy of ResNet-18 in non-IID CIFAR-10 and non-IID GTSRB. Under the Non-IID CIFAR10 setting, the test accuracy of GradCAM-AE is much higher than that of GradCAM-Krum. GradCAM-AE can quickly converge (around the 30th communication round) as it involves more benign clients in FL training. This indicates that GradCAM-AE can accurately filter malicious model updates, as can also be confirmed by the detection indicators in Table 5.1. Although GradCAM-Krum can restrict malicious clients from participating in FL, it sacrifices accuracy and robustness, especially under non-IID CIFAR-10. The test accuracy of

GradCAM-Krum fluctuates significantly and likely diverges, as it only selects one local model update as the global model. The more divergent the model updates, the more diverse the GradCAM heat maps. GradCAM-Krum struggles to screen malicious model updates, which coincides with the precision of 0.625 for GradCAM-Krum on Non-IID CIFAR-10 in Table 5.1. The remaining defense methods still undergo poor anomaly detection as on the IID dataset. The detection rates of MobileNetV3-Large on CIFAR-10 and GTSRB under IID and Non-IID settings are provided in Table 5.3 and Table 5.4 respectively.

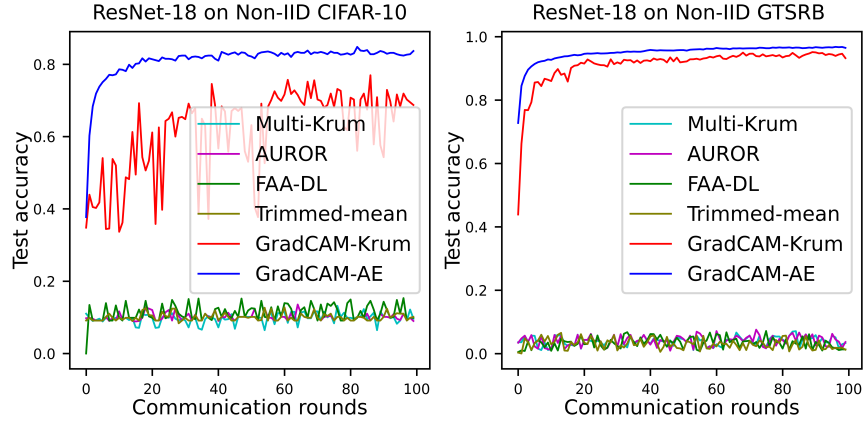


Figure 5.6: Test accuracy vs. communication rounds for **ResNet-18** on Non-IID CIFAR-10 and GTSRB.

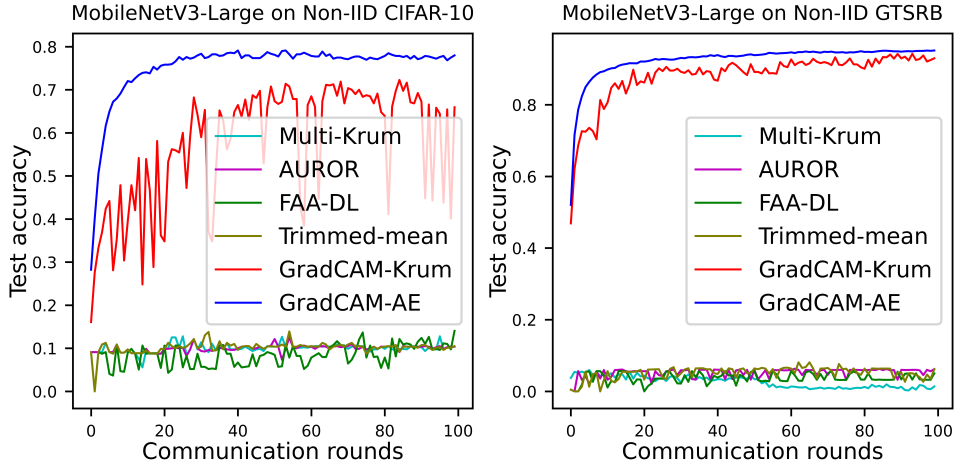


Figure 5.7: Test accuracy vs. communication rounds for **MobileNetV3-Large** on Non-IID CIFAR-10 and GTSRB.

We replace the ResNet-18 model with MobileNetV3-Large, and the changing trend of the test accuracy with the communication round is consistent with Figure 5.6, as shown in Figure 5.7. This means that the complexity of the DNN model itself has a limited impact on the malicious detection of GradCAM-Krum, while the complexity of local data plays a crucial role. GTSRB

has fewer image features than CIFAR-10, and the local model updates learned by FL have less diversity, making GradCAM-Krum converge.

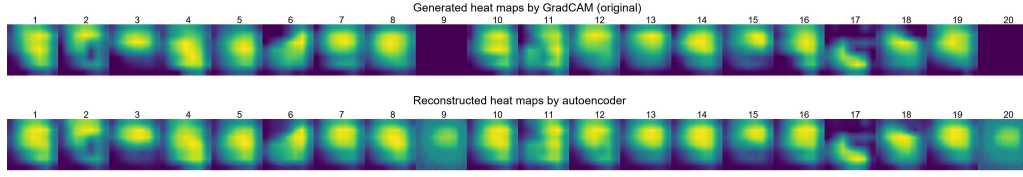


Figure 5.8: Comparison between generated heat maps by GradCAM and reconstructed heat maps by the autoencoder.

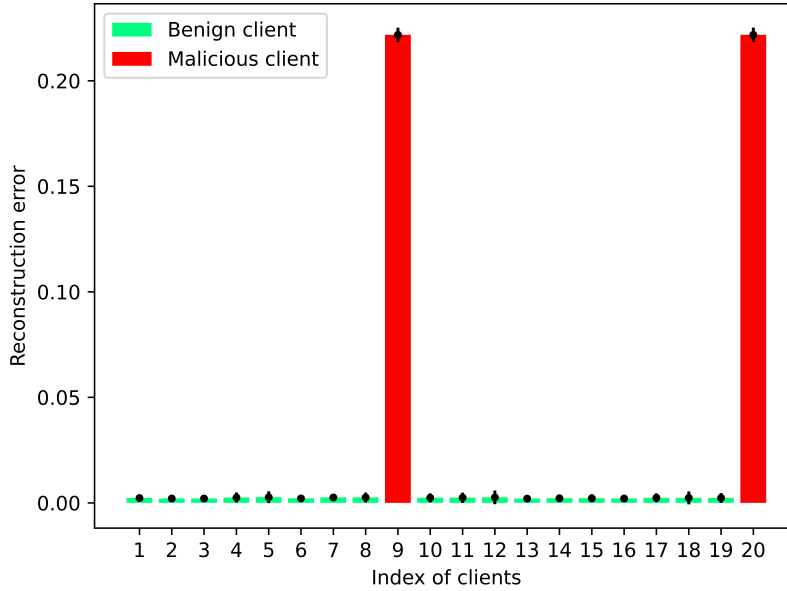


Figure 5.9: Reconstruction errors for ResNet18 on Non-IID CIFAR-10.

This significant difference is quantified in Figure 5.9, where the server averages the reconstruction errors of 20 clients in 100 communication rounds one by one. Figure 5.8 shows the original GradCAM heat maps and the reconstructed heat maps by the autoencoder under ResNet-18 on Non-IID CIFAR-10 setting. Strikingly, it can be noticed that the reconstruction errors of client 9 and client 20 are obviously larger than those of other clients.

To evaluate the scalability of GradCAM-AE, we define the attack rate as the ratio of the number of attackers to the total number of local clients, i.e., $\text{Att} = \frac{\# \text{attackers}}{\# \text{all clients}}$. GradCAM-AE and GradCAM-Krum are compared under Non-IID CIFAR-10 with 2 and 8 attackers. The total number of local clients is 20 (fixed), i.e., $\text{Att} = 0.1$ and $\text{Att} = 0.4$. Figure 5.10 shows that as the attack rate increases, GradCAM-Krum fails to identify attackers. The maximum test accuracy gap of 0.652 appears in the 61st communication round. In contrast, the test accuracy of GradCAM-AE changes, and the maximum gap of 0.076 appears in the 61st communication round. Similarly, in

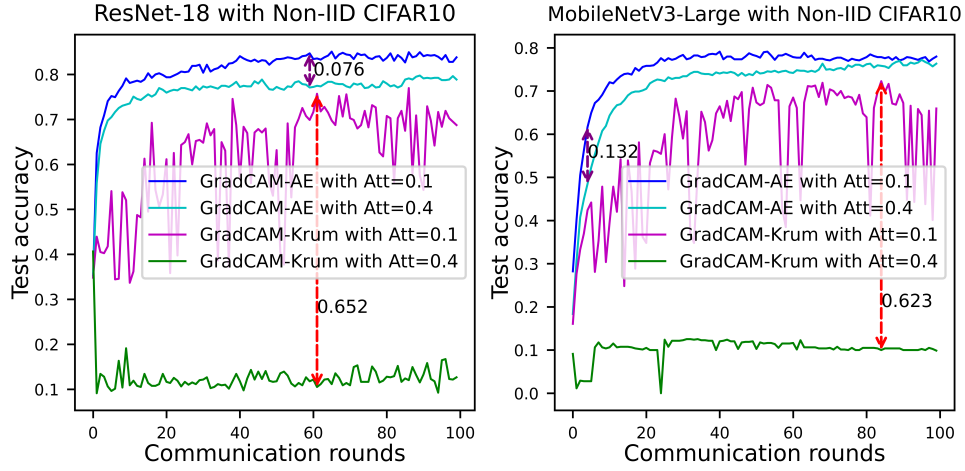
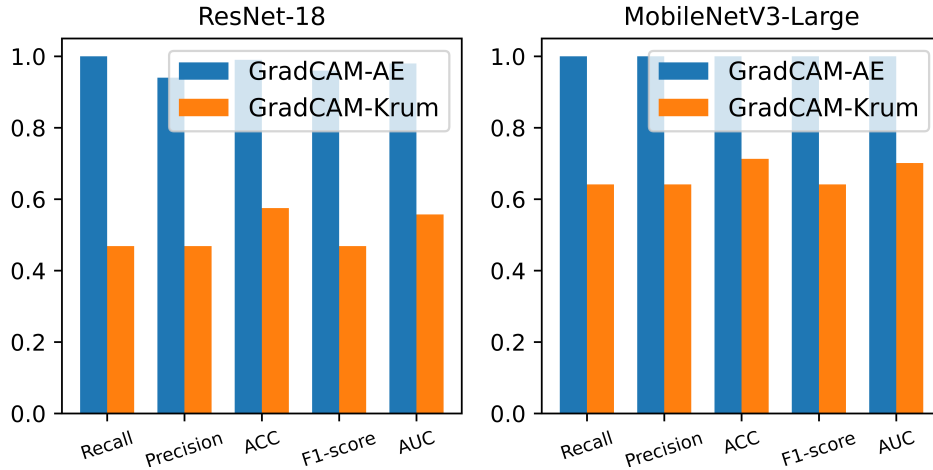


Figure 5.10: Scalability of GradCAM-AE vs. GradCAM-Krum

the case of MobileNetV3-Large, the tendency of test accuracy is consistent, offering scalability and robustness. The detection rates under $\text{Att} = 0.4$ are given in Figure 5.11, which shows detection rates of ResNet-18 and MobileNetV3-Large on Non-IID CIFAR-10 with $\text{Att} = 0.4$, respectively. The results indicate that the detection rates of GradCAM-AE outperform GradCAM-Krum no matter the number of attackers, offering scalability and robustness.

Figure 5.11: Detection rates of ResNet-18 and MobileNetV3-Large on Non-IID CIFAR-10 with $\text{Att} = 0.4$

5.4 Chapter Summary and Concluding Remarks

In this chapter, we propose a novel and robust mechanism, GradCAM-AE, which combines the strengths of GradCAM and autoencoder (AE), to shield defense against poisoning attacks on FL. This integration provides an enhanced detection capability compared to existing Euclidean

distance-based approaches. GradCAM generates heat maps for each uploaded local model, transforming them into lower-dimensional, visualizable representations. This facilitates the identification of malicious model updates by isolating anomalous GradCAM heat maps. Simultaneously, an autoencoder is employed to refine the GradCAM heat maps, emphasizing hidden features and improving their distinguishability. This improves the success rate of identifying anomalous heat maps and malicious local models. We conduct ablation studies, that is, GradCAM-Krum anomaly detection scheme in which Krum instead of AE is applied to identify the malicious GradCAM heat maps that are generated at the server. Experimental results demonstrate the indispensability of autoencoder. To further mitigate potential misclassifications, we also design a voting algorithm, where model updates that consistently manifest as potentially malicious over several communication rounds are conclusively identified as malicious and removed from FL.

Given the inherent complexity of deep neural networks, ensuring theoretical guarantees for detection effectiveness poses a significant challenge. While we have reviewed the original GradCAM paper and found no explicit theoretical guarantee, GradCAM and autoencoder technologies are widely acknowledged and proven effective through extensive empirical experiments. We extensively evaluate the GradCAM-AE framework using both IID and Non-IID settings on the CIFAR-10 and GTSRB datasets. Our assessment encompasses two prominent deep learning models, i.e., ResNet-18 and MobileNetV3-Large. The substantially experimental results demonstrate the robust detection effectiveness of GradCAM-AE, providing empirical validation for its efficacy in practical scenarios.

In term of the computational costs of GradCAM-AE, apart from the specific computational overhead required for FL to train local models, clients do not require extra computing overhead. Moreover, the GradCAM-AE framework is implemented on the server, which has sufficient computing capability to train the autoencoder.

Chapter 6

Conclusion and Future Work

In this chapter, we summarize the findings and results of this dissertation and discuss potential future research directions. In particular, we summarize our contributions and draw some conclusions in Section 6.1. We outline the limitations of the work and possible areas for improvement in Section 6.2.

6.1 Conclusion

In this dissertation, first of all, we investigated resource allocation for FL optimization to balance the trade-off between energy consumption of the edge clients and the learning accuracy of FL. We proposed FedAECS and FL-DLT3 frameworks to achieve the optimal learning accuracy of FL and energy consumption of clients balance under offline and online respectively. Specially, FedAECS recursively assesses the potential clients in FL to balance the energy consumption and learning accuracy in each communication round, the numerical results show the advantage of our proposed FedAECS; FL-DLT3 achieves the optimal learning accuracy of FL and energy consumption of clients balance in a continuous domain, LSTM is leveraged in FL-DLT3 to predict the time-varying data size, bandwidth, channel gain, and the remaining energy of the edge clients in each communication round. Numerical results demonstrate that the proposed FL-DLT3 achieves fast convergence (less than 100 communication rounds) while the FL accuracy-to-energy consumption ratio is improved by 51.8% compared to the existing state-of-the-art benchmark FedAECS.

Then, to design a new attack method that requires no knowledge of FL training data and achieves both effectiveness and undetectability, we proposed a novel, data-agnostic, model poisoning attack on FL, by designing a new adversarial GAE-based framework. In particular, the GAE is designed to generate malicious local models solely based on the benign local models overheard and capturing the correlation features of the benign local and global models while keeping the malicious local models undetectable. We demonstrate the convergence of FL under GAE-based attack with a considerably large optimality gap. The attack performance and the invisibility of the proposed GAE-based are evaluated with MNIST, Fashion-MNIST and CIFAR-10, respectively.

Last but not least, to defend the GAE-based model poisoning attacks, we proposed a new defense mechanism, namely, GradCAM-AE, which integrates GradCAM and AE to offer a substantially more powerful detection capability compared to existing Euclidean distance-based approaches. Particularly, GradCAM is applied at the server to create GradCAM heat maps for every uploaded model update. An autoencoder is applied to reconstruct the heat maps, while magnifying the discernible features of the heat maps. The reconstruction errors of the GradCAM heat maps are measured, and a threshold is created based on the statistics of the reconstruction errors. A reconstructed GradCAM heatmap with a reconstruction error surpassing the threshold is categorized as atypical, and the corresponding model update as malicious. Furthermore, to mitigate potential misclassifications, we also design a voting algorithm, where model updates that consistently manifest as potentially malicious over several communication rounds are conclusively identified as malicious and removed from FL. A comprehensive evaluation of the proposed GradCAM-AE framework is conducted using the CIFAR-10 and GTSRB datasets under both IID and Non-IID settings. The ResNet-18 and MobileNetV3-Large models are tested. The results substantiate that GradCAM-AE offers superior detection rates and FL test accuracy, juxtaposed with contemporary state-of-the-art methods.

6.2 Future Work

In this section, we acknowledge the limitations of our work and suggest ways in which it could be improved and extended. We believe that this topic can be further explored and developed, and we encourage future research in these directions. Below, we highlight the limitations and potential areas for improvement.

6.2.1 Resource Allocation for FL

In this work, we studied resource allocation for FL under offline and online respectively. However, we assume that clients' information is fully observed and we did not implement FL training. Moreover, we simplified the accuracy of FL to be proportional to the size of the data. In practical the clients' information is partially observed and the accuracy of FL model may be greatly affected by the quality of its training data. We believe that the quality of local data for FL should be taken into consideration, and the clients' information is partially observable during the FL training process. For example, only the local model size is known in advance, but the size and quality of the local data and the client's remaining energy are unknown in prior for server. Based on this, implementing online resource allocation for FL is worth studying.

It is worth mentioning that our work has attracted widespread attention and discussion in the community (Yuan et al., 2023; Kim et al., 2023; Ji et al., 2023; Bai and Chen, 2023; Li et al., 2023a,a). Particularly, (Ji et al., 2022) pointed out that our work mainly focus on improving FL accuracy in individual round while putting less emphasis on comprehensively considering the latency, energy consumption and accuracy under all communication rounds. We taken for granted that conventional wireless optimization approaches can be directly applied to FL, and

treat each round or time slot independently and identically. However, each communication round is not actually independent from each other. One question which particularly interests us is how to balance communication cost and accuracy within the long-term energy constrained over all communication rounds. We firmly believe that our work can inspire more relevant research efforts.

6.2.2 Data-Agnostic Poisoning Attacks against on FL

In this work, we assume that the adversary can eavesdrop all local model updates sent by participating benign clients. However, in practice local models are often encrypted, in the future, we will investigate applying our data-agnostic poisoning attacks to encrypted (e.g., Data Encryption Standard, Advanced Encryption Standard-based algorithm, Rivest–Shamir–Adleman) local model updates by generating keys through key generation (Wei et al., 2022), which may not have a significant effect. In addition, It is necessary to consider cases that are closely related to the real situation, such as the attack effects that occur when only a certain percentage of updates can be captured by adversary. We will further explore the susceptibility of NNs to the new attack model in our future work.

6.2.3 Defense Strategies for FL

Although our proposed new defense GradCAM-AE can suppress some malicious attacks on FL, we strongly believe that no defensive measure is a panacea. Recently, a variety of new backdoor attacks have emerged (Nguyen et al., 2024), such as semantic backdoor (Yoo and Kwak, 2022), frequency-injection based backdoor attack (Feng et al., 2022), there is no doubt that backdoor attacks in FL pose a significant threat to the security and privacy of FL systems. Whether our proposed GradCAM-AE defense system is also effective against these backdoor attacks remains to be verified. In addition, this work lacks consideration of defense against adaptive attacks (Yu et al., 2023) or spoofing attacks (Bi et al., 2024). For example, the attackers may bypass the voting mechanism by performing fewer poisoning rounds within every ξ communication rounds, we are actively exploring new defense measures to counter such intermittent or adaptive attacks, aiming to enhance the robustness of our defense system.

In the future, it will be urgent to continue developing more robust defense methods that are effective against semantic backdoor attacks and adaptive attacks, improving the efficiency of defense mechanisms, investigating the effectiveness of defenses under realistic attack scenarios, examining the impact of data heterogeneity on backdoor/adaptive attacks and defenses, and studying the effect of system-level factors on backdoor/adaptive attacks and defenses.

Bibliography

- Sawsan Abdulrahman, Hanine Tout, Azzam Mourad, and Chamseddine Talhi. 2021a. FedMCCS: Multicriteria Client Selection Model for Optimal IoT Federated Learning. *IEEE Internet of Things Journal* 8, 6 (2021), 4723–4735. <https://doi.org/10.1109/JIOT.2020.3028742>
- Sawsan Abdulrahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. 2021b. A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet of Things Journal* 8, 7 (2021), 5476–5497. <https://doi.org/10.1109/JIOT.2020.3030072>
- Tran The Anh, Nguyen Cong Luong, Dusit Niyato, Dong In Kim, and Li-Chun Wang. 2019. Efficient Training Management for Mobile Crowd-Machine Learning: A Deep Reinforcement Learning Approach. *IEEE Wireless Communications Letters* 8, 5 (2019), 1345–1348. <https://doi.org/10.1109/LWC.2019.2917133>
- Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. 2016. DROWN: Breaking TLS Using SSLv2. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 689–706. <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/aviram>
- Loukia Avramelou, Paraskevi Nousi, Nikolaos Passalis, and Anastasios Tefas. 2024. Deep reinforcement learning for financial trading using multi-modal features. *Expert Systems with Applications* 238 (2024), 121849. <https://doi.org/10.1016/j.eswa.2023.121849>
- Sana Awan, Bo Luo, and Fengjun Li. 2021. Contra: Defending against poisoning attacks in federated learning. In *Computer Security—ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I* 26. Springer, 455–475.
- Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*. PMLR, 2938–2948.

- Jingpan Bai and Yuan Chen. 2023. The Node Selection Strategy for Federated Learning in UAV-Assisted Edge Computing Environment. *IEEE Internet of Things Journal* 10, 15 (2023), 13908–13919. <https://doi.org/10.1109/JIOT.2023.3263359>
- Kilian Batzner, Lars Heckler, and Rebecca König. 2024. EfficientAD: Accurate Visual Anomaly Detection at Millisecond-Level Latencies. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 128–138.
- Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*. PMLR, 634–643.
- Siguo Bi, Kai Li, Shuyan Hu, Wei Ni, Cong Wang, and Xin Wang. 2024. Detection and Mitigation of Position Spoofing Attacks on Cooperative UAV Swarm Formations. *IEEE Transactions on Information Forensics and Security* 19 (2024), 1883–1895. <https://doi.org/10.1109/TIFS.2023.3341398>
- Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 118–128.
- Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems* 1 (2019), 374–388.
- Stephen Boyd and Lieven Vandenbergh. 2004. *Convex Optimization*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511804441>
- Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. 2018. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics* 112 (2018), 59–67.
- Debora Caldarola, Massimiliano Mancini, Fabio Galasso, Marco Ciccone, Emanuele Rodola, and Barbara Caputo. 2021. Cluster-Driven Graph Federated Learning Over Multiple Domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2749–2758.
- Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. 2019. Understanding Distributed Poisoning Attack in Federated Learning. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. 233–239. <https://doi.org/10.1109/ICPADS47876.2019.00042>
- Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2021. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society.

- Xiaoyu Cao and Neil Zhenqiang Gong. 2022. Mpaf: Model poisoning attacks to federated learning based on fake clients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3396–3404.
- Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. 2023. FedRecover: Recovering from Poisoning Attacks in Federated Learning using Historical Information. In *2023 IEEE Symposium on Security and Privacy (SP)*. 1366–1383. <https://doi.org/10.1109/SP46215.2023.10179336>
- Xiaoyu Cao, Zaixi Zhang, Jinyuan Jia, and Neil Zhenqiang Gong. 2022. FLCert: Provably Secure Federated Learning Against Poisoning Attacks. *IEEE Transactions on Information Forensics and Security* 17 (2022), 3691–3705. <https://doi.org/10.1109/TIFS.2022.3212174>
- Xiangyi Chen, Yuanguo Bi, Guangjie Han, Dongyu Zhang, Minghan Liu, Han Shi, Hai Zhao, and Fengyun Li. 2022. Distributed computation offloading and trajectory optimization in multi-uav-enabled edge computing. *IEEE Internet of Things Journal* 9, 20 (2022), 20096–20110.
- Zhenzhu Chen, Anmin Fu, Yinghui Zhang, Zhe Liu, Fanjian Zeng, and Robert H. Deng. 2021a. Secure Collaborative Deep Learning Against GAN Attacks in the Internet of Things. *IEEE Internet of Things Journal* 8, 7 (2021), 5839–5849. <https://doi.org/10.1109/JIOT.2020.3033171>
- Zheyi Chen, Pu Tian, Weixian Liao, and Wei Yu. 2021b. Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning. *IEEE Transactions on Network Science and Engineering* 8, 2 (2021), 1070–1083. <https://doi.org/10.1109/TNSE.2020.3002796>
- Prateek Chhikara, Rajkumar Tekchandani, Neeraj Kumar, Mohsen Guizani, and Mohammad Mehedi Hassan. 2021. Federated learning and autonomous UAVs for hazardous zone detection and AQI prediction in IoT environment. *IEEE Internet of Things Journal* 8, 20 (2021), 15456–15467.
- Yae Jee Cho, Jianyu Wang, and Gauri Joshi. 2022. Towards understanding biased client selection in federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 10351–10375.
- Shunfeng Chu, Jun Li, Jianxin Wang, Zhe Wang, Ming Ding, Yijin Zhang, Yuwen Qian, and Wen Chen. 2022. Federated Learning Over Wireless Channels: Dynamic Resource Allocation and Task Scheduling. *IEEE Transactions on Cognitive Communications and Networking* 8, 4 (2022), 1910–1924. <https://doi.org/10.1109/TCCN.2022.3196009>
- Will Cukierski. 2013. CIFAR-10 - Object Recognition in Images. <https://kaggle.com/competitions/cifar-10>

- Li Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142. <https://doi.org/10.1109/MSP.2012.2211477>
- Canh T. Dinh, Nguyen H. Tran, Minh N. H. Nguyen, Choong Seon Hong, Wei Bao, Albert Y. Zomaya, and Vincent Gramoli. 2021. Federated Learning Over Wireless Networks: Convergence Analysis and Resource Allocation. *IEEE/ACM Transactions on Networking* 29, 1 (2021), 398–409. <https://doi.org/10.1109/TNET.2020.3035770>
- Daniel Díaz-Sánchez, Andrés Marín-Lopez, Florina Almenárez Mendoza, Patricia Arias Cabarcos, and R. Simon Sherratt. 2019. TLS/PKI Challenges and Certificate Pinning Techniques for IoT and M2M Secure Communications. *IEEE Communications Surveys & Tutorials* 21, 4 (2019), 3502–3531. <https://doi.org/10.1109/COMST.2019.2914453>
- Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*. 1605–1622.
- Yu Feng, Benteng Ma, Jing Zhang, Shanshan Zhao, Yong Xia, and Dacheng Tao. 2022. FIBA: Frequency-Injection based Backdoor Attack in Medical Image Analysis. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 20844–20853. <https://doi.org/10.1109/CVPR52688.2022.02021>
- Zhiying Feng, Xu Chen, Qiong Wu, Wen Wu, Xiaoxi Zhang, and Qianyi Huang. 2023. FedDD: Toward Communication-efficient Federated Learning with Differential Parameter Dropout. *IEEE Transactions on Mobile Computing* (2023), 1–18. <https://doi.org/10.1109/TMC.2023.3311188>
- Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X. Liu, and Ting Wang. 2022. Label Inference Attacks Against Vertical Federated Learning. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 1397–1414. <https://www.usenix.org/conference/usenixsecurity22/presentation/fu-chong>
- Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 1587–1596. <https://proceedings.mlr.press/v80/fujimoto18a.html>
- Jiqiang Gao, Boyu Hou, Xiaojie Guo, Zheli Liu, Ying Zhang, Kai Chen, and Jin Li. 2023. Secure Aggregation is Insecure: Category Inference Attack on Federated Learning. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 147–160. <https://doi.org/10.1109/TDSC.2021.3128679>

- Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems* 33 (2020), 16937–16947.
- Bimal Ghimire and Danda B. Rawat. 2022. Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things. *IEEE Internet of Things Journal* 9, 11 (2022), 8229–8249. <https://doi.org/10.1109/JIOT.2022.3150363>
- Xueluan Gong, Yanjiao Chen, Huayang Huang, Yuqing Liao, Shuai Wang, and Qian Wang. 2022. Coordinated Backdoor Attacks against Federated Learning with Model-Dependent Triggers. *IEEE Network* 36, 1 (2022), 84–90. <https://doi.org/10.1109/MNET.011.2000783>
- Prajwal Gupta, Krishna Yadav, Brij B Gupta, Mamoun Alazab, and Thippa Reddy Gadekallu. 2023. A Novel Data Poisoning Attack in Federated Learning based on Inverted Loss Function. *Computers & Security* 130 (2023), 103270.
- Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. 2022. ADBench: Anomaly Detection Benchmark. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 32142–32159. https://proceedings.neurips.cc/paper_files/paper/2022/file/cf93972b116ca5268827d575f2cc226b-Paper-Dataset_s_and_Benchmarks.pdf
- Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated Learning for Mobile Keyboard Prediction. *CoRR* abs/1811.03604 (2018). arXiv:1811.03604 <http://arxiv.org/abs/1811.03604>
- Morghan Hartmann, Umair Sajid Hashmi, and Ali Imran. 2022. Edge computing in smart health care systems: Review, challenges, and research directions. *Transactions on Emerging Telecommunications Technologies* 33, 3 (2022), e3710.
- Ali Hatamizadeh, Hongxu Yin, Pavlo Molchanov, Andriy Myronenko, Wenqi Li, Prerna Dogra, Andrew Feng, Mona G Flores, Jan Kautz, Daguang Xu, and Holger R. Roth. 2023. Do Gradient Inversion Attacks Make Federated Learning Unsafe? *IEEE Transactions on Medical Imaging* 42, 7 (2023), 2044–2056. <https://doi.org/10.1109/TMI.2023.3239391>
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Sven Hebrok, Simon Nachtigall, Marcel Maehren, Nurullah Erinola, Robert Merget, Juraj Somorovsky, and Jörg Schwenk. 2023. We Really Need to Talk About Session Tickets: A Large-Scale Analysis of Cryptographic Dangers with TLS Session Tickets. In *32nd USENIX Security*

- Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 4877–4894. <https://www.usenix.org/conference/usenixsecurity23/presentation/hebrok>
- Hadi Hojjati and Narges Armanfard. 2023. DASVDD: Deep Autoencoding Support Vector Data Descriptor for Anomaly Detection. *IEEE Transactions on Knowledge and Data Engineering* (2023), 1–12. <https://doi.org/10.1109/TKDE.2023.3328882>
- Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. 2019. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
- Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. 2023. Edge Computing with Artificial Intelligence: A Machine Learning Perspective. *ACM Comput. Surv.* 55, 9, Article 184 (jan 2023), 35 pages. <https://doi.org/10.1145/3555802>
- Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.
- Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR* abs/1602.07360 (2016). arXiv:1602.07360 <http://arxiv.org/abs/1602.07360>
- Mustafa Musa Jaber, Mohammed Hasan Ali, Sura Khalil Abd, Ahmed Alkhayyat, RQ Malik, et al. 2023. Application of edge computing-based information-centric networking in smart cities. *Computer Communications* 211 (2023), 46–58.
- Malhar S. Jere, Tyler Farnan, and Farinaz Koushanfar. 2021. A Taxonomy of Attacks on Federated Learning. *IEEE Security & Privacy* 19, 2 (2021), 20–28. <https://doi.org/10.1109/MS-EC.2020.3039941>
- Yun Ji, Zhoubin Kou, Xiaoxiong Zhong, Hangfan Li, Fan Yang, and Sheng Zhang. 2022. Client Selection and Bandwidth Allocation for Federated Learning: An Online Optimization Perspective. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 5075–5080. <https://doi.org/10.1109/GLOBECOM48099.2022.10001492>
- Yun Ji, Xiaoxiong Zhong, Zhoubin Kou, Sheng Zhang, Hangfan Li, and Yuanyuan Yang. 2023. Efficiency-Boosting Federated Learning in Wireless Networks: A Long-Term Perspective. *IEEE Transactions on Vehicular Technology* 72, 7 (2023), 9434–9447. <https://doi.org/10.1109/TVT.2023.3250273>
- Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134. [https://doi.org/10.1016/S0004-3702\(98\)00023-X](https://doi.org/10.1016/S0004-3702(98)00023-X)

- Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.
- Hamed Karimi, Julie Nutini, and Mark Schmidt. 2016. Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-Łojasiewicz Condition. In *Machine Learning and Knowledge Discovery in Databases*, Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken (Eds.). Springer International Publishing, Cham, 795–811.
- Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*. PMLR, 5132–5143.
- Latif U Khan, Shashi Raj Pandey, Nguyen H Tran, Walid Saad, Zhu Han, Minh NH Nguyen, and Choong Seon Hong. 2020a. Federated learning for edge networks: Resource optimization and incentive mechanism. *IEEE Communications Magazine* 58, 10 (2020), 88–93.
- Latif U. Khan, Ibrar Yaqoob, Nguyen H. Tran, S. M. Ahsan Kazmi, Tri Nguyen Dang, and Choong Seon Hong. 2020b. Edge-Computing-Enabled Smart Cities: A Comprehensive Survey. *IEEE Internet of Things Journal* 7, 10 (2020), 10200–10232. <https://doi.org/10.1109/JIOT.2020.2987070>
- Doyeon Kim, Seungjae Shin, Jaewon Jeong, and Joohyung Lee. 2023. Joint Edge Server Selection and Dataset Management for Federated Learning-enabled Mobile Traffic Prediction. *IEEE Internet of Things Journal* (2023), 1–1. <https://doi.org/10.1109/JIOT.2023.3301019>
- Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
- Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- Harrison Kurunathan, Hailong Huang, Kai Li, Wei Ni, and Ekram Hossain. 2023. Machine Learning-Aided Operations and Communications of Unmanned Aerial Vehicles: A Contemporary Survey. *IEEE Communications Surveys & Tutorials* (2023), 1–1. <https://doi.org/10.1109/COMST.2023.3312221>
- Dohyun Kwon, Joohyung Jeon, Soohyun Park, Joongheon Kim, and Sungrae Cho. 2020. Multi-agent DDPG-Based Deep Learning for Smart Ocean Federated Learning IoT Networks. *IEEE Internet of Things Journal* 7, 10 (2020), 9895–9903. <https://doi.org/10.1109/JIOT.2020.2988033>

- Guillaume Lample and Devendra Singh Chaplot. 2017. Playing FPS games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.
- Kenneth Lange and Kenneth Lange. 2010. Singular value decomposition. *Numerical analysis for statisticians* (2010), 129–142.
- Sunwoo Lee, Anit Kumar Sahu, Chaoyang He, and Salman Avestimehr. 2023. Partial model averaging in federated learning: Performance guarantees and benefits. *Neurocomputing* 556 (2023), 126647.
- Khaled B. Letaief, Yuanming Shi, Jianmin Lu, and Jianhua Lu. 2022. Edge Artificial Intelligence for 6G: Vision, Enabling Technologies, and Applications. *IEEE Journal on Selected Areas in Communications* 40, 1 (2022), 5–36. <https://doi.org/10.1109/JSAC.2021.3126076>
- Hao Li, Chengcheng Li, Jian Wang, Aimin Yang, Zezhong Ma, Zunqian Zhang, and Dianbo Hua. 2023b. Review on security of federated learning and its application in healthcare. *Future Generation Computer Systems* 144 (2023), 271–290.
- Jingyi Li, Guocheng Liao, Lin Chen, and Xu Chen. 2023c. Roulette: A Semantic Privacy-Preserving Device-Edge Collaborative Inference Framework for Deep Learning Classification Tasks. *IEEE Transactions on Mobile Computing* (2023).
- Kai Li, Billy Pik Lik Lau, Xin Yuan, Wei Ni, Mohsen Guizani, and Chau Yuen. 2023a. Toward Ubiquitous Semantic Metaverse: Challenges, Approaches, and Opportunities. *IEEE Internet of Things Journal* 10, 24 (2023), 21855–21872. <https://doi.org/10.1109/JIOT.2023.3302159>
- Kai Li, Wei Ni, Lingjie Duan, Mehran Abolhasan, and Jianwei Niu. 2018. Wireless Power Transfer and Data Collection in Wireless Sensor Networks. *IEEE Transactions on Vehicular Technology* 67, 3 (2018), 2686–2697. <https://doi.org/10.1109/TVT.2017.2772895>
- Kai Li, Wei Ni, Eduardo Tovar, and Mohsen Guizani. 2020. Optimal Rate-Adaptive Data Dissemination in Vehicular Platoons. *IEEE Transactions on Intelligent Transportation Systems* 21, 10 (2020), 4241–4251. <https://doi.org/10.1109/TITS.2019.2939292>
- Kai Li, Wei Ni, Eduardo Tovar, and Mohsen Guizani. 2021b. Joint Flight Cruise Control and Data Collection in UAV-Aided Internet of Things: An Onboard Deep Reinforcement Learning Approach. *IEEE Internet of Things Journal* 8, 12 (2021), 9787–9799. <https://doi.org/10.1109/JIOT.2020.3019186>
- Kai Li, Wei Ni, Eduardo Tovar, and Abbas Jamalipour. 2019a. On-Board Deep Q-Network for UAV-Assisted Online Power Transfer and Data Collection. *IEEE Transactions on Vehicular Technology* 68, 12 (2019), 12215–12226. <https://doi.org/10.1109/TVT.2019.2945037>

- Kai Li, Wei Ni, Xin Yuan, Alam Noor, and Abbas Jamalipour. 2022. Deep-Graph-Based Reinforcement Learning for Joint Cruise Control and Task Offloading for Aerial Edge Internet of Things (EdgeIoT). *IEEE Internet of Things Journal* 9, 21 (2022), 21676–21686. <https://doi.org/10.1109/JIOT.2022.3182119>
- Kai Li, Chau Yuen, Branislav Kusy, Raja Jurdak, Aleksandar Ignjatovic, Salil S. Kanhere, and Sanjay Jha. 2019b. Fair Scheduling for Data Collection in Mobile Sensor Networks with Energy Harvesting. *IEEE Transactions on Mobile Computing* 18, 6 (2019), 1274–1287. <https://doi.org/10.1109/TMC.2018.2861384>
- Mushu Li, Jie Gao, Lian Zhao, and Xuemin Shen. 2021a. Adaptive Computing Scheduling for Edge-Assisted Autonomous Driving. *IEEE Transactions on Vehicular Technology* 70, 6 (2021), 5318–5331. <https://doi.org/10.1109/TVT.2021.3062653>
- X. Li, Z. Qu, S. Zhao, B. Tang, Z. Lu, and Y. Liu. 2023d. LoMar: A Local Defense Against Poisoning Attack on Federated Learning. *IEEE Transactions on Dependable and Secure Computing* 20, 01 (jan 2023), 437–450. <https://doi.org/10.1109/TDSC.2021.3135422>
- Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). <http://arxiv.org/abs/1509.02971>
- Wei Yang Bryan Lim, Jianqiang Huang, Zehui Xiong, Jiawen Kang, Dusit Niyato, Xian-Sheng Hua, Cyril Leung, and Chunyan Miao. 2021. Towards federated learning in uav-enabled internet of vehicles: A multi-dimensional contract-matching approach. *IEEE Transactions on Intelligent Transportation Systems* (2021).
- Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 2031–2063. <https://doi.org/10.1109/COMST.2020.2986024>
- Wenxin Liu, Hui Lin, Xiaoding Wang, Jia Hu, Georges Kaddoum, Md. Jalil Piran, and Atif Alamri. 2023. D2MIF: A Malicious Model Detection Mechanism for Federated-Learning-Empowered Artificial Intelligence of Things. *IEEE Internet of Things Journal* 10, 3 (2023), 2141–2151. <https://doi.org/10.1109/JIOT.2021.3081606>
- Hide Lycklama, Lukas Burkhalter, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. 2023. Rofl: Robustness of secure federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 453–476.

- Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. 2022. Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems* (2022).
- Xinchen Lyu, Chenshan Ren, Wei Ni, Hui Tian, Ren Ping Liu, and Eryk Dutkiewicz. 2019. Optimal Online Data Partitioning for Geo-Distributed Machine Learning in Edge of Wireless Networks. *IEEE Journal on Selected Areas in Communications* 37, 10 (2019), 2393–2406. <https://doi.org/10.1109/JSAC.2019.2934002>
- Pavel Mach and Zdenek Becvar. 2017. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1628–1656. <https://doi.org/10.1109/COMST.2017.2682318>
- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. <https://openreview.net/forum?id=rJzIBfZAb>
- Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. 2017. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2322–2358. <https://doi.org/10.1109/COMST.2017.2745201>
- Yuyi Mao, Jun Zhang, and Khaled B. Letaief. 2016. Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications* 34, 12 (2016), 3590–3605. <https://doi.org/10.1109/JSAC.2016.2611964>
- Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.
- Guangxu Mei, Ziyu Guo, Shijun Liu, and Li Pan. 2019. SGNN: A Graph Neural Network Based Federated Learning Approach by Hiding Structure. In *2019 IEEE International Conference on Big Data (Big Data)*. 2560–2568. <https://doi.org/10.1109/BigData47090.2019.9005983>
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
- Ihab Mohammed, Shadha Tabatabai, Ala Al-Fuqaha, Faissal El Bouanani, Junaid Qadir, Basheer Qolomany, and Mohsen Guizani. 2021. Budgeted Online Selection of Candidate IoT Clients

- to Participate in Federated Learning. *IEEE Internet of Things Journal* 8, 7 (2021), 5938–5952. <https://doi.org/10.1109/JIOT.2020.3036157>
- Jason J Molitierno. 2016. *Applications of combinatorial matrix theory to Laplacian matrices of graphs*. CRC Press.
- Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems* 115 (2021), 619–640.
- Thuy Dung Nguyen, Tuan Nguyen, Phi Le Nguyen, Hieu H Pham, Khoa D Doan, and Kok-Seng Wong. 2024. Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions. *Engineering Applications of Artificial Intelligence* 127 (2024), 107166.
- Thien Duc Nguyen, Phillip Rieger, Roberta De Viti, Huili Chen, Björn B Brandenburg, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, et al. 2022. {FLAME}: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*. 1415–1432.
- Thien Duc Nguyen, Phillip Rieger, Markus Miettinen, and Ahmad-Reza Sadeghi. 2020. Poisoning attacks on federated learning-based IoT intrusion detection system. In *Proc. Workshop Decentralized IoT Syst. Secur.(DISS)*. 1–7.
- Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.
- Florian Nuding and Rudolf Mayer. 2022. Data Poisoning in Sequential and Parallel Federated Learning. In *Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics (Baltimore, MD, USA) (IWSPA '22)*. Association for Computing Machinery, New York, NY, USA, 24–34. <https://doi.org/10.1145/3510548.3519372>
- Mícheál O’Searcoid. 2006. *Metric spaces*. Springer Science & Business Media.
- Shirui Pan, Ruiqi Hu, Sai-fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Learning graph embedding with adversarial training methods. *IEEE transactions on cybernetics* 50, 6 (2019), 2475–2487.
- Sharnil Pandya, Gautam Srivastava, Rutvij Jhaveri, M Rajasekhara Babu, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, Spyridon Mastorakis, Md Jalil Piran, and Thippa Reddy Gadekallu. 2023. Federated learning for smart cities: A comprehensive survey. *Sustainable Energy Technologies and Assessments* 55 (2023), 102987.
- Kai Peng, Peichen Liu, Muhammad Bilal, Xiaolong Xu, and Edoardo Prezioso. 2023. Mobility and Privacy-Aware Offloading of AR Applications for Healthcare Cyber-Physical Systems in

- Edge Computing. *IEEE Transactions on Network Science and Engineering* 10, 5 (2023), 2662–2673. <https://doi.org/10.1109/TNSE.2022.3185092>
- Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3803–3810.
- Jakob Puchinger, Günther R Raidl, and Ulrich Pferschy. 2010. The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing* 22, 2 (2010), 250–265.
- Adnan Qayyum, Kashif Ahmad, Muhammad Ahtazaz Ahsan, Ala Al-Fuqaha, and Junaid Qadir. 2022. Collaborative Federated Learning for Healthcare: Multi-Modal COVID-19 Diagnosis at the Edge. *IEEE Open Journal of the Computer Society* 3 (2022), 172–184. <https://doi.org/10.1109/OJCS.2022.3206407>
- Chenyang Qiu, Zhaoci Huang, Wenzhe Xu, and Huijia Li. 2022. Fast Community Detection based on Graph Autoencoder Reconstruction. In *2022 7th International Conference on Big Data Analytics (ICBDA)*. IEEE, 265–271.
- Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2020. Designing Network Design Spaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 10425–10433. <https://doi.org/10.1109/CVPR42600.2020.01044>
- Sawsan Abdul Rahman, Hanine Tout, Chamseddine Talhi, and Azzam Mourad. 2020. Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning? *IEEE Network* 34, 6 (2020), 310–317. <https://doi.org/10.1109/MNET.011.2000286>
- Nikolaos V Sahinidis. 2019. Mixed-integer nonlinear programming 2018. , 301–306 pages.
- Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.
- Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.
- Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*.
- Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1354–1371.

- Shiqi Shen, Shruti Tople, and Prateek Saxena. 2016. Auror: Defending against Poisoning Attacks in Collaborative Deep Learning Systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications* (Los Angeles, California, USA) (ACSAC '16). Association for Computing Machinery, New York, NY, USA, 508–519. <https://doi.org/10.1145/2991079.2991125>
- Siping Shi, Chuang Hu, Dan Wang, Yifei Zhu, and Zhu Han. 2022. Federated Anomaly Analytics for Local Model Poisoning Attack. *IEEE Journal on Selected Areas in Communications* 40, 2 (2022), 596–610. <https://doi.org/10.1109/JSAC.2021.3118347>
- David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*. PMLR, 387–395.
- Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). <http://arxiv.org/abs/1409.1556>
- Mengkai Song, Zhibo Wang, Zhifei Zhang, Yang Song, Qian Wang, Ju Ren, and Hairong Qi. 2020. Analyzing User-Level Privacy Attack Against Federated Learning. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2430–2444. <https://doi.org/10.1109/JSAC.2020.3000372>
- J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 32 (2012), 323–332. <https://doi.org/10.1016/j.neunet.2012.02.016> Selected Papers from IJCNN 2011.
- Gan Sun, Yang Cong, Jiahua Dong, Qiang Wang, Lingjuan Lyu, and Ji Liu. 2022. Data Poisoning Attacks on Federated Machine Learning. *IEEE Internet of Things Journal* 9, 13 (2022), 11365–11375. <https://doi.org/10.1109/JIOT.2021.3128646>
- Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. 2021. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9311–9319.
- Yuwei Sun, Hideya Ochiai, and Jun Sakuma. 2023. Attacking Distance-aware Attack: Semi-targeted Model Poisoning on Federated Learning. *IEEE Transactions on Artificial Intelligence* (2023), 1–15. <https://doi.org/10.1109/TAI.2023.3280155>
- Dhivya Suresh, Abdullatif Aydin, Stuart James, Kamran Ahmed, and Prokar Dasgupta. 2023. The role of augmented reality in surgical training: a systematic review. *Surgical Innovation* 30, 3 (2023), 366–382.

- Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6105–6114. <http://proceedings.mlr.press/v97/tan19a.html>
- Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. 2022. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *Comput. Surveys* 55, 8 (2022), 1–35.
- Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *Computer Security—ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 480–501.
- Ruben Rodriguez Torrado, Philip Bontrager, Julian Togelius, Jialin Liu, and Diego Perez-Liebana. 2018. Deep Reinforcement Learning for General Video Game AI. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. 1–8. <https://doi.org/10.1109/CIG.2018.8490422>
- Cuong Tran, Ferdinando Fioretto, and Pascal Van Hentenryck. 2021. Differentially private and fair deep learning: A lagrangian dual approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9932–9939.
- Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. 61–66.
- Fei Wang, Ethan Hugh, and Baochun Li. 2023. More than enough is too much: Adaptive defenses against gradient leakage in production federated learning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.
- Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. 2020b. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 16070–16084. https://proceedings.neurips.cc/paper_files/paper/2020/file/b8ffa41d4e492f0fad2f13e29e1762eb-Paper.pdf
- Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. 2022a. Federated Unlearning via Class-Discriminative Pruning. In *Proceedings of the ACM Web Conference 2022 (Virtual Event, Lyon, France) (WWW '22)*. Association for Computing Machinery, New York, NY, USA, 622–632. <https://doi.org/10.1145/3485447.3512222>

- Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. 2022b. Protect privacy from gradient leakage attack in federated learning. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 580–589.
- Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. 2019b. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221. <https://doi.org/10.1109/JSAC.2019.2904348>
- Tian Wang, Lei Qiu, Arun Kumar Sangaiah, Anfeng Liu, Md Zakirul Alam Bhuiyan, and Ying Ma. 2020a. Edge-computing-based trustworthy data collection model in the internet of things. *IEEE Internet of Things Journal* 7, 5 (2020), 4218–4227.
- Yingfeng Wang, Biyun Xu, Myungjae Kwak, and Xiaoqin Zeng. 2020c. A simple training strategy for graph autoencoder. In *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*. 341–345.
- Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019a. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2512–2520. <https://doi.org/10.1109/INFOCOM.2019.8737416>
- Bo Wei, Weitao Xu, Kai Li, Chengwen Luo, and Jin Zhang. 2022. i2 Key: A Cross-sensor Symmetric Key Generation System Using Inertial Measurements and Inaudible Sound. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 183–194. <https://doi.org/10.1109/IPSN54338.2022.00022>
- Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.
- Wenchao Xia, Tony Q. S. Quek, Kun Guo, Wanli Wen, Howard H. Yang, and Hongbo Zhu. 2020. Multi-Armed Bandit-Based Client Scheduling for Federated Learning. *IEEE Transactions on Wireless Communications* 19, 11 (2020), 7108–7123. <https://doi.org/10.1109/TWC.2020.3008091>
- Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
- Liang Xiao, Xiaoyue Wan, Xiaozhen Lu, Yanyong Zhang, and Di Wu. 2018. IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Processing Magazine* 35, 5 (2018), 41–49. <https://doi.org/10.1109/MSP.2018.2825478>

- Chulin Xie, Keli Huang, Pin Yu Chen, and Bo Li. 2020. DbA: Distributed backdoor attacks against federated learning. In *8th International Conference on Learning Representations, ICLR 2020*.
- Yuanhao Xiong, Ruochen Wang, Minhao Cheng, Felix Yu, and Cho-Jui Hsieh. 2023. Feddm: Iterative distribution matching for communication-efficient federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16323–16332.
- Bo Xu, Wenchao Xia, Jun Zhang, Tony Q. S. Quek, and Hongbo Zhu. 2021. Online Client Scheduling for Fast Federated Learning. *IEEE Wireless Communications Letters* 10, 7 (2021), 1434–1438. <https://doi.org/10.1109/LWC.2021.3069541>
- Ming Yang, Hang Cheng, Fei Chen, Ximeng Liu, Meiqing Wang, and Xibin Li. 2023. Model poisoning attack in differential privacy-based federated learning. *Information Sciences* 630 (2023), 158–172.
- Zhaohui Yang, Mingzhe Chen, Walid Saad, Choong Seon Hong, and Mohammad Shikh-Bahaei. 2021. Energy Efficient Federated Learning Over Wireless Communication Networks. *IEEE Transactions on Wireless Communications* 20, 3 (2021), 1935–1949. <https://doi.org/10.1109/TWC.2020.3037554>
- Benshun Yin, Zhiyong Chen, and Meixia Tao. 2020. Joint User Scheduling and Resource Allocation for Federated Learning over Wireless Networks. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 1–6. <https://doi.org/10.1109/GLOBECOM42002.2020.9348225>
- Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. PMLR, 5650–5659.
- Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. 2021. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16337–16346.
- KiYoon Yoo and Nojun Kwak. 2022. Backdoor attacks in federated learning by rare embeddings and gradient ensembling. *arXiv preprint arXiv:2204.14017* (2022).
- Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, and Koji Yamamoto. 2020. MAB-based Client Selection for Federated Learning with Uncertain Resources in Mobile Networks. In *2020 IEEE Globecom Workshops (GC Wkshps)*. 1–6. <https://doi.org/10.1109/GCWkshps50303.2020.9367421>
- Yi Yu, Yufei Wang, Wenhan Yang, Shijian Lu, Yap-Peng Tan, and Alex C Kot. 2023. Backdoor attacks against deep image compression via adaptive frequency trigger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12250–12259.

- Shijing Yuan, Beiyu Dong, Hongtao Lvy, Hongze Liu, Hongyang Chen, Chentao Wu, Song Guo, Yue Ding, and Jie Li. 2023. Adaptive Incentivize for Cross-Silo Federated Learning in IIoT: A Multi-agent Reinforcement Learning Approach. *IEEE Internet of Things Journal* (2023), 1–1. <https://doi.org/10.1109/JIOT.2023.3315770>
- Kai Yue, Richeng Jin, Chau-Wai Wong, Dror Baron, and Huaiyu Dai. 2023. Gradient obfuscation gives a false sense of security in federated learning. In *32nd USENIX Security Symposium (USENIX Security 23)*. 6381–6398.
- Yufeng Zhan, Peng Li, and Song Guo. 2020a. Experience-Driven Computational Resource Allocation of Federated Learning by Deep Reinforcement Learning. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 234–243. <https://doi.org/10.1109/IPDPS47924.2020.00033>
- Yufeng Zhan, Peng Li, Zhihao Qu, Deze Zeng, and Song Guo. 2020b. A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal* (2020).
- Jiale Zhang, Bing Chen, Xiang Cheng, Huynh Thi Thanh Binh, and Shui Yu. 2021. PoisonGAN: Generative Poisoning Attacks Against Federated Learning in Edge Computing Systems. *IEEE Internet of Things Journal* 8, 5 (2021), 3310–3322. <https://doi.org/10.1109/JIOT.2020.3023126>
- Jiale Zhang, Junjun Chen, Di Wu, Bing Chen, and Shui Yu. 2019. Poisoning Attack in Federated Learning using Generative Adversarial Nets. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 374–380. <https://doi.org/10.1109/TrustCom/BigDataSE.2019.00057>
- Jianyi Zhang, Ang Li, Minxue Tang, Jingwei Sun, Xiang Chen, Fan Zhang, Changyou Chen, Yiran Chen, and Hai Li. 2023a. Fed-cbs: A heterogeneity-aware client sampling mechanism for federated learning via class-imbalance reduction. In *International Conference on Machine Learning*. PMLR, 41354–41381.
- Yihuan Zhang, Liang Wang, Xuhui Jiang, Yong Zeng, and Yifan Dai. 2022c. An efficient LiDAR-based localization method for self-driving cars in dynamic environments. *Robotica* 40, 1 (2022), 38–55.
- Yifei Zhang, Dun Zeng, Jinglong Luo, Zenglin Xu, and Irwin King. 2023b. A Survey of Trustworthy Federated Learning with Perspectives on Security, Robustness and Privacy. In *Companion Proceedings of the ACM Web Conference 2023 (Austin, TX, USA) (WWW '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 1167–1176. <https://doi.org/10.1145/3543873.3587681>

- Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2022a. FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2545–2555.
- Zhiyuan Zhang, Qi Su, and Xu Sun. 2022b. Dim-Krum: Backdoor-Resistant Federated Learning for NLP with Dimension-wise Krum-Based Aggregation. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 339–354. <https://doi.org/10.18653/V1/2022.FINDINGS-EMNLP.25>
- Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. 2020. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 737–744. <https://doi.org/10.1109/SSCI47803.2020.9308468>
- Yue Zhao, Ryan Rossi, and Leman Akoglu. 2021. Automatic Unsupervised Outlier Model Selection. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 4489–4502. https://proceedings.neurips.cc/paper_files/paper/2021/file/23c894276a2c5a16470e6a31f4618d73-Paper.pdf
- Jingjing Zheng, Kai Li, Naram Mhaisen, Wei Ni, Eduardo Tovar, and Mohsen Guizani. 2022. Exploring Deep-Reinforcement-Learning-Assisted federated learning for Online Resource Allocation in Privacy-Preserving EdgeIoT. *IEEE Internet of Things Journal* 9, 21 (2022), 21099–21110.
- Jingjing Zheng, Kai Li, Naram Mhaisen, Wei Ni, Eduardo Tovar, and Mohsen Guizani. 2023a. Federated Learning for Online Resource Allocation in Mobile Edge Computing: A Deep Reinforcement Learning Approach. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–6. <https://doi.org/10.1109/WCNC55385.2023.10118940>
- Jingjing Zheng, Kai Li, Naram Mhaisen, Wei Ni, Eduardo Tovar, and Mohsen Guizani. 2023b. Federated Learning for Online Resource Allocation in Mobile Edge Computing: A Deep Reinforcement Learning Approach. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–6. <https://doi.org/10.1109/WCNC55385.2023.10118940>
- Jingjing Zheng, Kai Li, Eduardo Tovar, and Mohsen Guizani. 2021. Federated Learning for Energy-balanced Client Selection in Mobile Edge Computing. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*. 1942–1947. <https://doi.org/10.1109/IWCMC51323.2021.9498853>
- Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2921–2929.

- Chong Zhou and Randy C Paffenroth. 2017. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 665–674.
- Dali Zhu, Yuchen Ma, and Yinlong Liu. 2020. Anomaly Detection with Deep Graph Autoencoders on Attributed Networks. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. 1–6. <https://doi.org/10.1109/ISCC50000.2020.9219684>
- Zheqi Zhu, Shuo Wan, Pingyi Fan, and Khaled B. Letaief. 2022. Federated Multiagent Actor–Critic Learning for Age Sensitive Mobile-Edge Computing. *IEEE Internet of Things Journal* 9, 2 (2022), 1053–1067. <https://doi.org/10.1109/JIOT.2021.3078514>
- Haomin Zhuang, Yihua Zhang, and Sijia Liu. 2023. A Pilot Study of Query-Free Adversarial Attack Against Stable Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2384–2391.

Appendix A

Appendix

A.1 PROOF OF THEOREM 1

Based on the Taylor expansion and the assumption that the gradient of the global loss function is L –Lipschitz continuous, it follows that

$$F(\mathbf{W}_g^{t+1}) - F(\mathbf{W}_g^t) \leq \nabla F(\mathbf{W}_g^t) (\mathbf{W}_g^{t+1} - \mathbf{W}_g^t) + \frac{L}{2} \|\mathbf{W}_g^{t+1} - \mathbf{W}_g^t\|^2 \quad (\text{A.1})$$

By substituting (4.1) into (A.1), we obtain (A.2),

$$\begin{aligned} F(\mathbf{W}_g^{t+1}) - F(\mathbf{W}_g^t) &\leq \nabla F(\mathbf{W}_g^t) \left[\frac{1}{D} \sum_{k=1}^K D_k (\mathbf{W}_k^{t+1} - \mathbf{W}_k^t) + \frac{D_{k'}}{D} (\mathbf{W}_{k'}^{t+1} - \mathbf{W}_{k'}^t) \right] \\ &\quad + \frac{L}{2} \left\| \frac{1}{D} \sum_{k=1}^K D_k (\mathbf{W}_k^{t+1} - \mathbf{W}_k^t) + \frac{D_{k'}}{D} (\mathbf{W}_{k'}^{t+1} - \mathbf{W}_{k'}^t) \right\|^2 \\ &= \frac{\eta^2 L}{2} \left\| \frac{1}{D} \sum_{k=1}^K D_k \nabla F_k(\mathbf{W}_k^t) + \frac{D_{k'}}{D} \nabla F_{k'}(\mathbf{W}_{k'}^t) \right\|^2 \\ &\quad - \eta \nabla F(\mathbf{W}_g^t) \left[\frac{1}{D} \sum_{k=1}^K D_k \nabla F_k(\mathbf{W}_k^t) + \frac{D_{k'}}{D} \nabla F_{k'}(\mathbf{W}_{k'}^t) \right] \end{aligned} \quad (\text{A.2})$$

By substituting $\mathbf{W}_k^{t+1} = \mathbf{W}_k^t - \eta \nabla F_k(\mathbf{W}_k^t)$ and $\mathbf{W}_{k'}^{t+1} = \sum_{k=1}^K \frac{D_k}{D} \mathbf{W}_k^t + \frac{D_{k'}}{D} (\mathbf{W}_{k'}^{t+1} - \mathbf{W}_{k'}^t)$ into (A.2) the expectation of $F(\mathbf{W}_g^{t+1})$ can be given by

$$\mathbb{E}[F(\mathbf{W}_g^{t+1})] \leq F(\mathbf{W}_g^t) - \eta \|\nabla F(\mathbf{W}_g^t)\|^2 + \frac{\eta^2 L}{2} \mathbb{E} \left[\left\| \sum_{k=1}^K \frac{D_k}{D} \nabla F_k(\mathbf{W}_k^t) + \frac{D_{k'}}{D} \nabla F_{k'}(\mathbf{W}_{k'}^t) \right\|^2 \right].$$

It is generally assumed in FL that

$$\mathbb{E} \left[\sum_{k=1}^K \frac{D_k}{D} \nabla F_k(\mathbf{W}_k^t) + \frac{D_{k'}}{D} \nabla F_{k'}(\mathbf{W}_{k'}^t) \right] = \nabla F(\mathbf{W}_g^t).$$

According to Jensen's inequality, $\mathbb{E}(x^2) \leq \mathbb{E}^2(x)$. Then,

$$\mathbb{E} \left[\left\| \sum_{k=1}^K \frac{D_k}{D} \nabla F_k(\mathbf{W}_k^t) + \frac{D_{k'}}{D} \nabla F_{k'}(\mathbf{W}_{k'}^t) \right\|^2 \right] \leq \|\nabla F(\mathbf{W}_g^t)\|^2. \quad (\text{A.3})$$

As a result, $\mathbb{E} [F(\mathbf{W}_g^{t+1})]$ is bounded by

$$\mathbb{E} [F(\mathbf{W}_g^{t+1})] \leq \mathbb{E} [F(\mathbf{W}_g^t)] + \left(\frac{\eta^2 L}{2} - \eta \right) \|\nabla F(\mathbf{W}_g^t)\|^2 \stackrel{\eta=\frac{1}{L}}{=} \mathbb{E} [F(\mathbf{W}_g^t)] - \frac{\eta}{2} \|\nabla F(\mathbf{W}_g^t)\|^2. \quad (\text{A.4})$$

Next, we derive the relationship between $\nabla F(\mathbf{W}_g^t)$ and $\nabla F(\mathbf{W}^t)$. Since $\mathfrak{D}(\mathbf{W}_{k'}, \mathbf{W}_g) = \|\mathbf{W}_{k'} - \mathbf{W}_g\| \leq r$, we have

$$\begin{aligned} \|\mathbf{W}_{k'}^{t+1} - \mathbf{W}_{k'}^t\| &\leq \|\mathbf{W}_{k'}^{t+1} - \mathbf{W}_g^{t+1}\| + \|\mathbf{W}_g^{t+1} - \mathbf{W}_{k'}^t\| = r + \|\mathbf{W}_g^{t+1} - \mathbf{W}_{k'}^t\| \\ &\leq r + \|\mathbf{W}_g^{t+1} - \mathbf{W}_g^t\| + \|\mathbf{W}_g^t - \mathbf{W}_{k'}^t\| = 2r + \|\mathbf{W}_g^{t+1} - \mathbf{W}_g^t\| \end{aligned} \quad (\text{A.5})$$

where (A.5) is based on the triangle inequality. Likewise, we also have

$$\begin{aligned} \|\mathbf{W}_g^{t+1} - \mathbf{W}_g^t\| &\leq \left\| \left(\mathbf{W}^{t+1} + \frac{D_{k'}}{D} \mathbf{W}_{k'}^{t+1} \right) - \left(\mathbf{W}^t + \frac{D_{k'}}{D} \mathbf{W}_{k'}^t \right) \right\| \leq \|\mathbf{W}^{t+1} - \mathbf{W}^t\| + \frac{D_{k'}}{D} \|\mathbf{W}_{k'}^{t+1} - \mathbf{W}_{k'}^t\| \\ &\leq \|\mathbf{W}^{t+1} - \mathbf{W}^t\| + \frac{D_{k'}}{D} (2r + \|\mathbf{W}_g^{t+1} - \mathbf{W}_g^t\|) \end{aligned} \quad (\text{A.6})$$

By reorganizing (A.6), it follows that

$$\|\mathbf{W}_g^{t+1} - \mathbf{W}_g^t\| \leq \frac{D}{D - D_{k'}} \times \|\mathbf{W}^{t+1} - \mathbf{W}^t\| + \frac{2rD_{k'}}{D - D_{k'}}. \quad (\text{A.7})$$

By substituting (A.7) into (A.5), it follows that

$$\|\mathbf{W}_{k'}^{t+1} - \mathbf{W}_{k'}^t\| \leq \frac{D}{D - D_{k'}} \times \|\mathbf{W}^{t+1} - \mathbf{W}^t\| + \frac{2rD_{k'}}{D - D_{k'}}. \quad (\text{A.8})$$

By taking expectation on both sides of (A.7), we have

$$\mathbb{E} [\|\mathbf{W}_g^{t+1} - \mathbf{W}_g^t\|] \leq \frac{D}{D - D_{k'}} \times \mathbb{E} [\|\mathbf{W}^{t+1} - \mathbf{W}^t\|] + \frac{2rD_{k'}}{D - D_{k'}}. \quad (\text{A.9})$$

Note that $\mathbb{E} [\|\mathbf{W}_g^{t+1} - \mathbf{W}_g^t\|] = \eta \nabla F(\mathbf{W}_g^t)$ and $\mathbb{E} [\|\mathbf{W}^{t+1} - \mathbf{W}^t\|] = \nabla F(\mathbf{W}^t)$. By substituting

them into both sides of (A.9) and then reorganizing (A.9), we obtain

$$\|\nabla F(\mathbf{W}_g^t)\| \leq \frac{D}{D-D_{k'}} \nabla F(\mathbf{W}^t) + \frac{2rD_{k'}}{\eta(D-D_{k'})} \quad (\text{A.10})$$

By substituting (A.10) into (A.4), it follows

$$\begin{aligned} \mathbb{E}[F(\mathbf{W}_g^{t+1})] - \mathbb{E}[F(\mathbf{W}_g^t)] &\leq -\frac{\eta}{2} \|\nabla F(\mathbf{W}_g^t)\|^2 \\ &\leq -\frac{\eta}{2} \left(\frac{D}{D-D_{k'}} \nabla F(\mathbf{W}^t) + \frac{2rD_{k'}}{\eta(D-D_{k'})} \right)^2 \\ &\leq \frac{\eta D^2}{2(D-D_{k'})^2 \|\nabla F(\mathbf{W}^t)\|^2}. \end{aligned} \quad (\text{A.11})$$

Considering the Polyak-Lojasiewicz condition, we have

$$\|\nabla F(\mathbf{W}^t)\|^2 \geq 2\rho (F(\mathbf{W}^t) - F(\mathbf{W}^*)). \quad (\text{A.12})$$

Given the convex loss function of SVM models, it follows:

$$F(\mathbf{W}_g^t) \leq F(\mathbf{W}^t) + F\left(\frac{D_{k'}}{D} \mathbf{W}_{k'}^t\right) \leq F(\mathbf{W}^t) + \frac{D_{k'}}{D} F(\mathbf{W}_{k'}^t) \quad (\text{A.13})$$

By substituting (A.12) and (A.13) into (A.11), we have

$$\begin{aligned} \mathbb{E}[F(\mathbf{W}_g^{t+1})] - \mathbb{E}[F(\mathbf{W}_g^t)] &\leq -\frac{\eta\rho D^2}{\eta(D-D_{k'})^2} (F(\mathbf{W}^t) - F(\mathbf{W}^*)) \\ &\leq -\frac{\eta\rho D^2}{(D-D_{k'})^2} \left(F(\mathbf{W}_g^t) - \frac{D_{k'}}{D} F(\mathbf{W}_{k'}^t) - F(\mathbf{W}^*) \right) \end{aligned} \quad (\text{A.14})$$

By restructuring (A.14), we have

$$\begin{aligned} \mathbb{E}[F(\mathbf{W}_g^{t+1})] - \mathbb{E}[F(\mathbf{W}^*)] &\leq \left(1 - \frac{\eta\rho D^2}{(D-D_{k'})^2} \right) (\mathbb{E}[F(\mathbf{W}_g^t)] - F(\mathbf{W}^*)) + \frac{\eta\rho D^2}{(D-D_{k'})^2} F(\mathbf{W}_{k'}^t) \\ &\leq (1 - \zeta) (\mathbb{E}[F(\mathbf{W}_g^t)] - F(\mathbf{W}^*)) + \frac{\eta\rho D^2}{(D-D_{k'})^2} F^{max} \end{aligned} \quad (\text{A.15})$$

where $\zeta = 1 - \frac{\rho\eta D^2}{(D-D_{k'})^2}$, and the second inequality is due to the constrained problem in (4.3a) and (4.3b) has a maximum value, i.e., $F(\mathbf{W}_{k'}^t) \leq F^{max}$.

Finally, applying mathematical induction upon (A.15) gives

$$\begin{aligned} F(\mathbf{W}_g^t) - F(\mathbf{W}^*) &\leq [\mathbb{E}[F(\mathbf{W}_g^0)] - F(\mathbf{W}^*)] \zeta^t + \frac{1 - \zeta^t}{1 - \zeta} \cdot \frac{\rho\eta DD_{k'}}{(D-D_{k'})^2} F^{max} \\ &\leq \theta \zeta^t + \frac{1 - \zeta^t}{1 - \zeta} \cdot \frac{\rho\eta DD_{k'}}{(D-D_{k'})^2} F^{max}, \end{aligned} \quad (\text{A.16})$$

which concludes this proof.