ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

TRẦN THÁI HUY
PHAN QUỐC ĐẠT

ĐỒ ÁN CHUYÊN NGÀNH

HỆ THỐNG PHÁT HIỆN XÂM NHẬP MẠNG TRONG PHƯƠNG TIỆN GIAO THÔNG SỬ DỤNG HỌC LIÊN KẾT

INTRUSION DETECTION SYSTEM FOR IN-VEHICLE NETWORK SECURITY USING FEDERATED LEARNING

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

TP. Hồ Chí Minh, 2025

ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

TRẦN THÁI HUY - 22520580
PHAN QUỐC ĐẠT - 22520233

ĐỒ ÁN CHUYÊN NGÀNH

HỆ THỐNG PHÁT HIỆN XÂM NHẬP MẠNG TRONG PHƯƠNG TIỆN GIAO THÔNG SỬ DỤNG HỌC LIÊN KẾT

INTRUSION DETECTION SYSTEM FOR IN-VEHICLE NETWORK SECURITY USING FEDERATED LEARNING

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN:
TS. Nguyễn Tấn Cầm
ThS. Nghi Hoàng Khoa

TP. Hồ Chí Minh - 2025

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Tấn Cầm và thầy Nghi Hoàng Khoa vì sự hướng dẫn tận tình, những nhận xét quý báu và sự hỗ trợ xuyên suốt quá trình thực hiện đồ án chuyên ngành.

Trong suốt quá trình làm đồ án, chúng em cũng cảm ơn thầy vì các lời nhận xét quý báu và những lời giải đáp thắc mắc. Nhờ đó, chúng em có thể hoàn thành đồ án cũng như trang bị những kiến thức cần thiết cho quá trình học tập và làm việc sau này.

Trân trọng.

Trần Thái Huy **Phan Quốc Đạt**

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

TÓM TẮT ĐỒ ÁN

Tính cần thiết của đề tài nghiên cứu:

Các cuộc tấn công mạng nói chung và trong lĩnh vực IoT - Internet of Things đang ngày càng trở nên tinh vi và thay đổi không ngừng, áp dụng nhiều thủ đoạn và chiến lược khác nhau. Điều này gây ra nhiều khó khăn cho các hệ thống IDPS (Intrusion Detection and Prevention System) trong việc phát hiện kịp thời các hành vi xâm nhập này.

Bài báo "FedMSE: Semi-supervised federated learning approach for IoT network intrusion detection" của tác giả Nguyen Van Tuan và Razvan Beuran đề ra một mô hình học liên kết bán giám sát. Mô hình này áp dụng Shrink Autoencoder và bộ phân loại đơn lớp Centroid (SAE-CEN), cùng với đó là thuật toán tổng hợp (aggregation) MSE-Avg dựa trên mean squared error [1].

Đồ án của nhóm chúng em tập trung vào việc ứng dụng mô hình này trong ngữ cảnh học liên kết phân tán (decentralized) nhằm tăng cường mức độ bảo mật thông qua việc loại bỏ global aggregator và dựa vào các gateway để thực hiện việc tổng hợp mô hình. Cùng với đó, nhóm cũng đã áp dụng các phương thức kiểm tra, xác thực mô hình ở mỗi gateway nhằm tránh các cuộc tấn công trực tiếp lên mô hình.

CHƯƠNG 1. TỔNG QUAN

1.1. Giới thiệu vấn đề

Trong bối cảnh các phương tiện giao thông hiện đại ngày càng được tích hợp nhiều hệ thống điều khiển điện tử và các thiết bị thông minh (ECU, cảm biến, camera, radar, ...). Tuy nhiên, việc kết nối giữa các thiết bị này thông qua mạng cũng tạo ra những lỗ hổng bảo mật nghiêm trọng, khiến các phương tiện trở thành mục tiêu tấn công hấp dẫn của tin tặc. Do đó, xây dựng hệ thống phát hiện xâm nhập (Intrusion Detection System – IDS) hiệu quả, đảm bảo khả năng phát hiện hành vi bất thường trong mạng nội bộ của xe, là một yêu cầu cấp thiết trong ngành công nghiệp ô tô hiện nay.

Các IDS hiện tại gặp khó khăn trong việc phát hiện kịp thời các xâm nhập do tính chất và quy mô các cuộc tấn công thay đổi nhanh chóng. Do đó, việc áp dụng học máy (machine-learning) đang ngày càng trở nên phổ biến.

Học liên kết (Federated Learning) là một hướng tiếp cận hay trong việc ứng dụng machine-learning vào việc phát triển IDS trong lĩnh vực IoT (Internet of Things) nói chung và IoV (Internet of Vehicles) nói riêng. FL cho phép các phương tiện (gateways) có thể tự huấn luyện mô hình của mình dựa trên dữ liệu nội bộ và gửi trọng số về một server/gateway làm nhiệm vụ tổng hợp (aggregator). Việc chỉ gửi trọng số giảm thiểu khả năng tiết lộ thông tin nhạy cảm liên quan đến người dùng mà vẫn đảm bảo lượng dữ liệu sử dụng trong quá trình huấn luyện.

Tuy vậy, trong thực tế, việc áp dụng học liên kết trong môi trường IoV vẫn gặp nhiều thách thức. Các vấn đề như sự không đồng đều về dữ liệu giữa các phương tiện (non-IID), sự thiếu hụt dữ liệu bất thường trong quá trình huấn luyện, hay giới hạn tài nguyên tính toán trên các thiết bị đang khiến quá trình

phát hiện xâm nhập trở nên phức tạp hơn. Bên cạnh đó, cách thức tổng hợp mô hình toàn cục (global aggregation) truyền thống vẫn chưa tối ưu, có thể ảnh hưởng đến độ chính xác và khả năng phát hiện các hành vi tấn công [1].

1.2. Cơ sở lý thuyết và các khái niệm liên quan

Trong quá trình nghiên cứu và triển khai mô hình, nhóm đã tìm hiểu và vận dụng một số kỹ thuật học máy. Phần dưới đây sẽ trình bày những kiến thức, công nghệ được nhắc đến và áp dụng:

- Internet of Vehicles (IoV): là một bộ phận của Internet of Things (IoT) mà trong đó các phương tiện giao thông (xe otô, xe buýt, xe tải, ...) có khả năng giao tiếp với nhau, với con người và với cơ sở hạ tầng.

- Hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS): là hệ thống giám sát lưu lượng mạng, từ đó phát hiện ra các hành vi bất thường (xâm nhập trái phép, tấn công từ chối dịch vụ (DoS),...IDS có các hướng tiếp cận như: phát hiện theo chữ ký (signature-based), phát hiện bất thường (anomaly-based), specification-based.

- Học liên kết (Federated Learning): là phương pháp học máy phân tán, trong đó các thiết bị (gateway) tự huấn luyện mô hình sử dụng bộ dữ liệu cục bộ của mình và chỉ gửi những thông tin cần thiết như trọng số về để tổng hợp. Điều này giúp bảo vệ quyền riêng tư, giảm dữ liệu truyền đi và giảm số lượng tính toán (phù hợp với ngữ cảnh IoV với tài nguyên hạn chế).

CHƯƠNG 2. MÔ HÌNH VÀ KỸ THUẬT ĐÃ SỬ DỤNG TRONG BÀI BÁO GỐC

2.1. Phương pháp

Bài báo "FedMSE: Semi-supervised federated learning approach for IoT network intrusion detection" đề xuất phương pháp học liên kết bán giám sát kết hợp Shrink Autoencoder (SAE) và Centroid One-Class Classifier (CEN) cùng thuật toán MSEAvg dựa trên mean-squared error.

2.2. Các kỹ thuật sử dụng

2.2.1. Thuật toán MSEAvg (Mean Squared Error Averaging

Thuật toán này đánh giá chất lượng mô hình dựa trên sai số tái tạo (reconstruction error) của các mô hình cục bộ (local). Các mô hình có sai số thấp hơn được gán trọng số lớn hơn (MSE Weight) trong quá trình tổng hợp, đảm bảo mô hình toàn cục (global) phản ánh chính xác đặc điểm của dữ liệu.

2.2.2. Thuật toán FedAvg (Federated Averaging)

FedAvg là thuật toán tổng hợp thường dùng trong học liên kết, được thiết kế để sử dụng trong mô hình có nhiều thiết bị. Server tổng hợp mô hình tiến hành tính trung bình các trọng số được gửi về và cập nhật mô hình toàn cục [2].

2.2.3. Thuật toán FedProx (Federated Proximal

FedProx là thuật toán mở rộng của FedAvg, được thiết kế để xử lý vấn đề dữ liệu không đồng nhất và sự khác biệt về tài nguyên giữa các thiết bị. FedProx

thêm một proximal term vào loss function để giữ model cục bộ gần với toàn cục.

2.2.4. Shrink Autoencoder (SAE)

SAE là một dạng của mạng neuron Autoencoder (A,E). AE học bằng cách nén dữ liệu mạng đầu vào xuống một không gian có số chiều nhỏ hơn (latent space), rồi lại giải nén để tái tạo lại dữ liệu ban đầu. Quá trình huấn luyện này sử dụng backpropagation để tối ưu hóa loss function [3] [1]:

$$\mathcal{L}_{\mathrm{AE}}(x^i, \hat{x}^i) = \frac{1}{n}\sum_{i=1}^{n} \|x^i - \hat{x}^i\|^2 \tag{2.1}$$

trong đó n là số lượng mẫu.

SAE được thiết kế để xử lý dữ liệu thưa thớt và đa chiều (high-dimensional). SAE hữu dụng khi dữ liệu huấn luyện ít. So với Autoencoder thông thường, SAE thêm một regularization term vào loss function. Regularization term sẽ giúp thu nhỏ các vector trong latent-space, dễ dàng phát hiện các bất thường[1].

$$\mathcal{L}_{\mathrm{SAE}}(\lambda; x^i, \hat{x}^i, h) = \frac{1}{n}\sum_{i=1}^{n} \|x^i - \hat{x}^i\|^2 + \lambda\frac{1}{n}\sum_{i=1}^{n} \|h^i\|^2 \tag{2.2}$$

2.2.5. Centroid (CEN)

Centroid hoạt động theo nguyên lý:

- Dữ liệu bình thường (normal): sẽ nằm gần trung tâm của cụm dữ liệu bình thường.
- Dữ liệu bất thường (abnormal): sẽ nằm xa trung tâm đó.

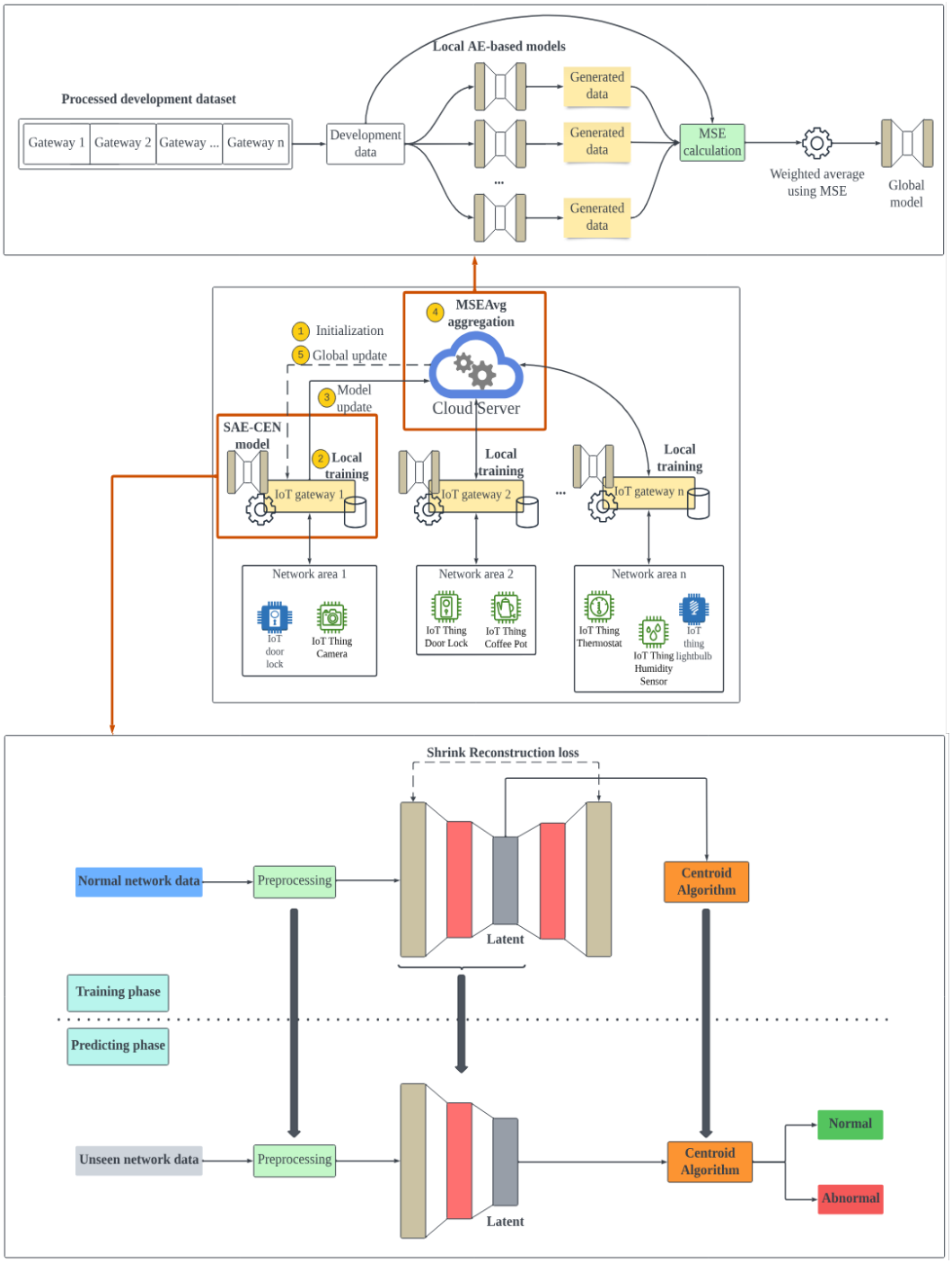
2.3. Mô hình

Mô hình FedMSE do tác giả đề ra gồm có hai thành phần chính:

- Cloud Server (server): làm nhiệm vụ tổng hợp các model sau mỗi vòng huấn luyện (rounds). Thí nghiệm bao gồm các thuật toán: FedAVG, FedProx, MSEAvg. Sau đó sẽ cập nhật model toàn cục cho các gateway.
- Gateways (client): làm nhiệm vụ huấn luyện các model cục bộ với dữ liệu của chúng.Sau mỗi vòng huấn luyện sẽ gửi trọng số về cho server và nhận lại mô hình đã được tổng hợp.

2.3.1. Quá trình huấn luyện

1. Server khởi tạo mô hình toàn cục ban đầu và gửi trọng số đến các gateway là các thiết bị IoT.
2. Các gateway nhận được trọng số và tiến hành huấn luyện model sử dụng dữ liệu cục bộ.
3. Sau khi training cục bộ sẽ gửi trọng số về lại Server.
4. Server nhận được trọng số sẽ tiến hành tổng hợp mô hình toàn cục sử dụng các trọng số được gởi đến.
5. Sau khi đã tổng hợp, server gửi lại mô hình cho các gateway và hoàn tất một vòng lặp.



Hình 2.1: *Mô hình FedMSE[1]*

2.4. Bộ dữ liệu (dataset)

Ở bài báo, tác giả sử dụng bộ dữ liệu N-BaIoT[4] được trích xuất lấy 5% dữ liệu normal và 0.5% dữ liệu abnormal. Sau đó giữ lại 40% lượng dữ liệu để kiểm thử. Các dữ liệu cũng được chia làm 2 nhóm là IID và Non-IID.

- IID: Jensen-Shannon distance = 0.01
- Non-IID: Jensen-Shannon distance = 0.83

ID	Device Name	Normal	Gafgyt	Mirai
0	Danmini_Doorbell	49548	652100	316650
1	Ecobee_Thermostat	13113	512133	310630
2	Philips_B120N10_Baby_Monitor	175240	312273	610714
3	Provision_PT_737E_Security_Camera	62154	330096	0
4	Provision_PT_838_Security_Camera	98514	309040	429337
5	Samsung_SNH_1011_N_Webcam	52150	323072	0
6	SimpleHome_XCS7_1002_WHT_Security_Camera	46585	303223	513248
7	SimpleHome_XCS7_1003_WHT_Security_Camera	19528	316438	514860
8	Ennio_Doorbell	39100	316400	0

Bảng 2.1: *Bảng phân chia dữ liệu ở các thiết bị theo Normal, Gafgyt, và Mirai[1]*

2.5. Quá trình thí nghiệm

Bài báo thực hiện thí nghiệm trên 2 loại model là Autoencoder truyền thống và SAE-CEN (Shrink Autoencoder kết hợp với Centroid one-class classifier). Với mỗi loại model tương ứng với 3 thuật toán tổng hợp: FedAvg, FedProx và MSEAvg.

2.5.1. Kết quả thí nghiệm

Kết quả cho thấy FedMSE, kết hợp SAE-CEN với MSEAvg, đạt hiệu suất cao hơn so với FedAvg và FedProx trong trường hợp không đồng nhất thấp và cao. FedMSE duy trì độ chính xác cao (lên đến 99,92% trong trường hợp không

đồng nhất thấp và 98,52% trong kịch bản không đồng nhất cao) cùng phương sai (standard deviation) thấp qua các tỷ lệ gateway và quy mô mạng khác nhau.

Các biểu đồ kết quả được lấy từ bài báo gốc "FedMSE: Semi-supervised federated learning approach for IoT network intrusion detection" [1]

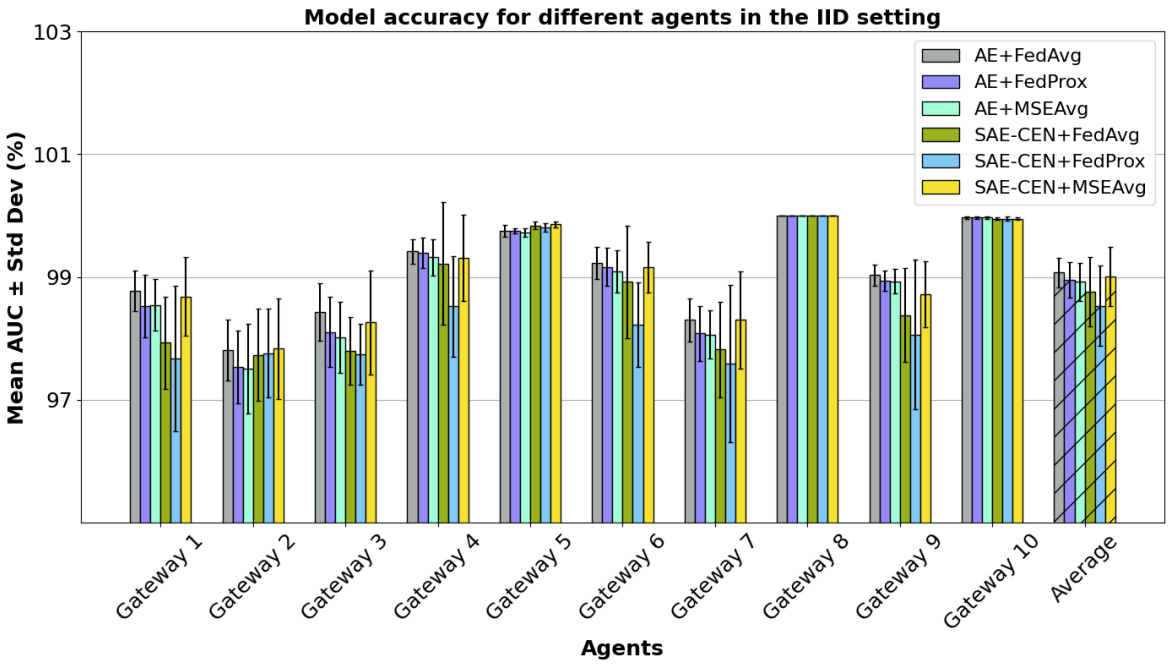
- AUC tăng từ 93.98 ± 2.90% (FedProx + AE) lên 97.30 ± 0.49% với FedMSE
- Giảm đáng kể chi phí huấn luyện với chỉ khoảng 50% gateway tham gia.
- Mô hình thể hiện độ bền cao trong các kịch bản mạng lớn và phi đồng nhất.

Gateway ratio	Autoencoder			SAE-CEN		
	FedAvg	FedProx	MSEAvg	FedAvg	FedProx	MSEAvg
a) Low non-IIDness						
50%	**99.07±0.24**	98.95±0.29	98.92±0.31	98.76±0.56	98.53±0.65	99.01±0.48
60%	98.48±0.44	98.12±0.45	98.47±0.37	98.76±0.42	98.85±0.56	**98.96±0.68**
70%	98.13±0.70	98.17±0.56	98.04±0.54	98.51±0.74	**98.69±0.37**	98.44±0.67
80%	97.96±0.93	97.77±0.69	97.66±0.78	98.77±0.85	**98.79±0.54**	98.71±0.80
90%	97.24±0.86	97.27±0.85	97.06±0.89	98.24±1.39	**98.70±0.71**	98.60±0.56
100%	97.61±0.52	97.34±0.36	97.29±0.63	**98.72±0.56**	98.61±0.64	98.69±0.69
b) High non-IIDness						
50%	94.74±2.69	93.98±2.90	92.87±1.26	96.93±0.70	97.28±0.84	**97.30±0.49**
60%	92.85±1.05	93.36±1.26	93.43±1.05	**97.31±0.68**	96.92±1.10	97.08±0.88
70%	95.28±1.45	94.39±1.36	94.09±1.97	**97.04±0.80**	97.03±0.74	97.00±1.09
80%	93.14±1.29	92.32±1.22	93.63±1.70	97.06±0.77	97.13±0.81	**97.21±0.75**
90%	92.83±1.18	92.61±0.97	92.70±1.17	**97.24±1.21**	97.15±0.73	97.11±0.88
100%	94.06±1.31	93.65±1.35	93.60±1.05	**97.30±0.71**	97.29±0.46	97.21±0.63

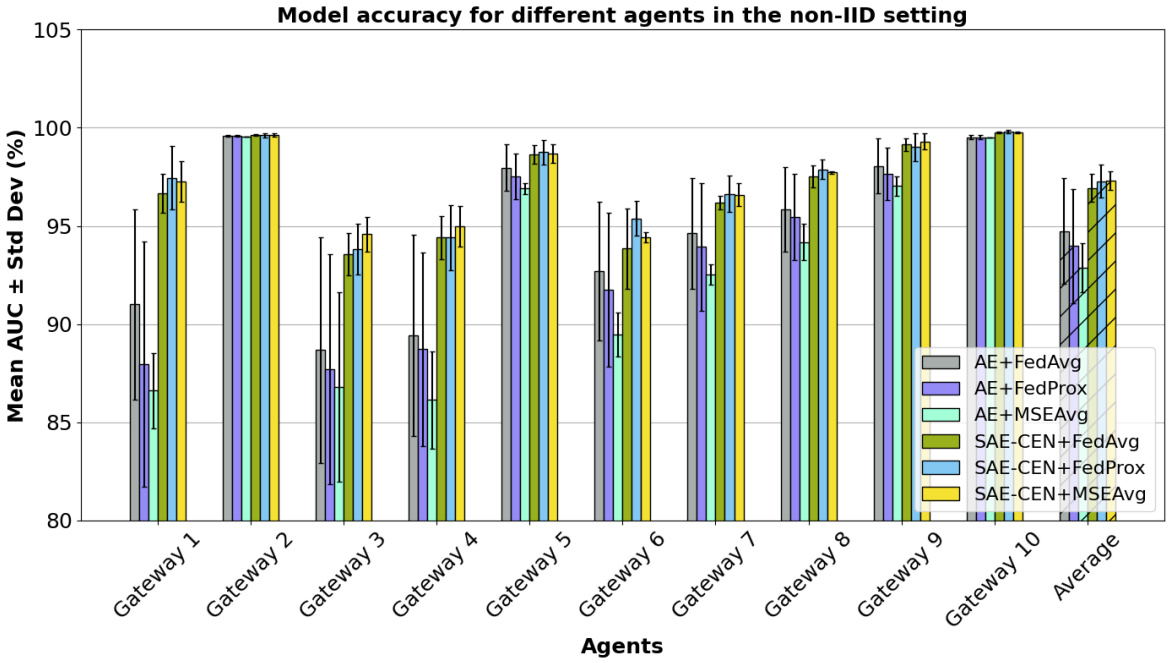
Hình 2.2: *So sánh hiệu suất với các tỉ lệ tham gia khác nhau*

Network scale	Autoencoder			SAE-CEN		
	FedAvg	FedProx	MSEAvg	FedAvg	FedProx	MSEAvg
a) Low non-IIDness						
10-gateway	**99.07±0.24**	98.95±0.29	98.92±0.31	98.76±0.56	98.53±0.65	99.01±0.48
20-gateway	98.43±0.26	98.51±0.21	98.56±0.31	98.59±0.37	**99.02±0.39**	98.54±0.37
30-gateway	97.40±0.67	97.45±0.47	97.37±0.57	98.27±0.43	**98.34±0.50**	98.34±0.55
40-gateway	97.05±0.97	97.16±0.66	96.95±0.95	98.38±0.42	98.38±0.42	**98.45±0.39**
50-gateway	97.01±0.75	96.74±0.56	97.20±0.82	**98.33±0.63**	98.16±0.37	98.20±0.56
b) High non-IIDness						
10-gateway	94.74±2.69	93.98±2.90	92.87±1.26	96.93±0.70	97.28±0.84	**97.30±0.49**
20-gateway	95.72±1.42	95.62±1.18	95.68±1.11	97.17±0.90	97.19±0.83	**97.29±0.96**
30-gateway	96.95±1.51	96.96±1.19	96.98±1.43	97.54±0.61	97.65±0.54	**97.73±0.56**
40-gateway	95.99±1.83	95.78±1.78	95.72±1.80	**97.81±0.94**	97.58±0.84	97.77±1.01
50-gateway	96.30±0.98	96.39±0.85	96.42±2.37	98.41±0.72	98.36±0.65	**98.52±0.51**

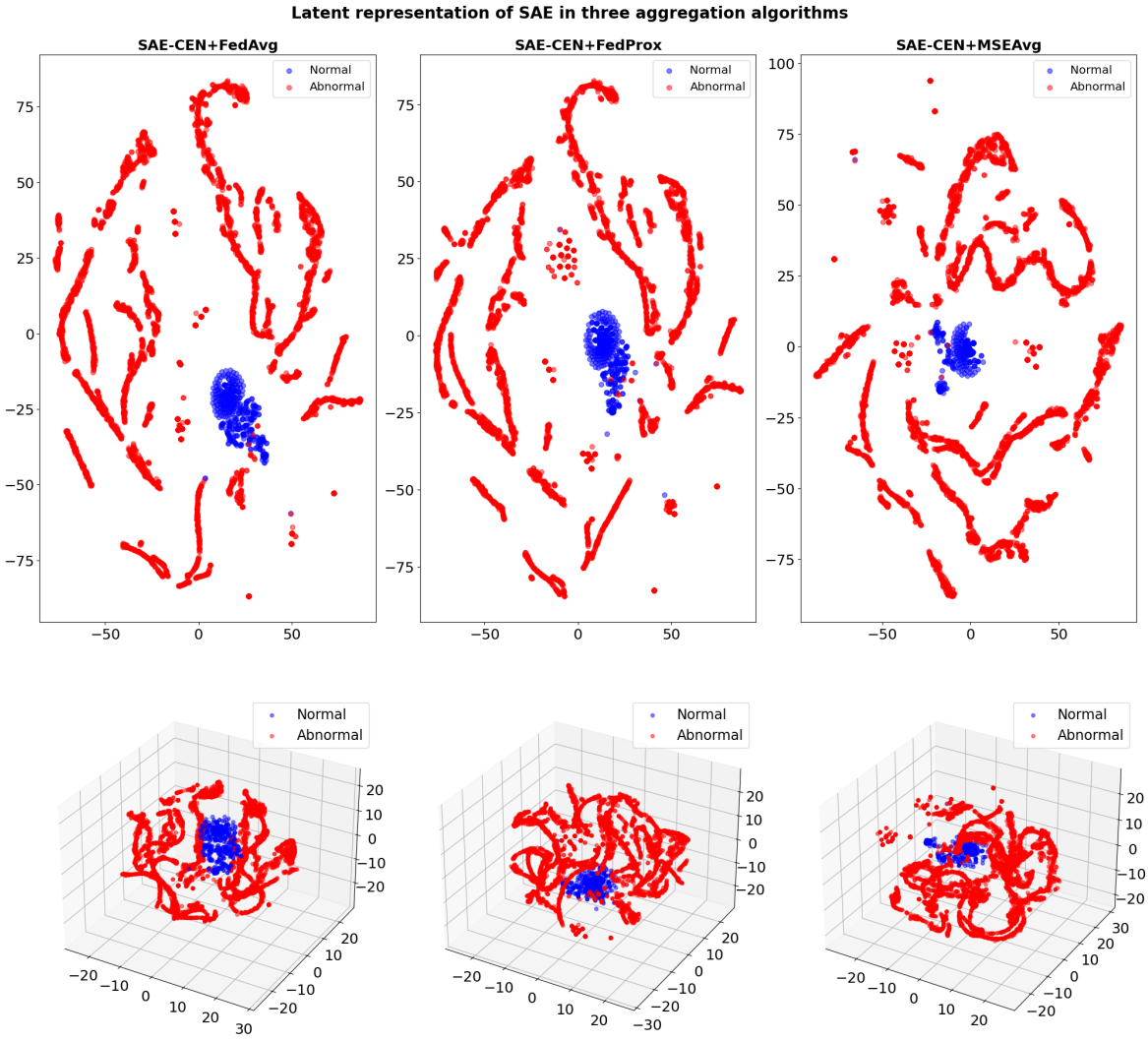
Hình 2.3: *So sánh hiệu suất với các quy mô mạng khác nhau*



Hình 2.4: *Accuracy của các dạng model và thuật toán tổng hợp trên các gateway trong ngữ cảnh IID*



Hình 2.5: *Accuracy của các dạng model và thuật toán tổng hợp trên các gatgeway trong ngữ cảnh Non-IID*



Hình 2.6: *Minh họa Latent Space cho FedMSE*

CHƯƠNG 3. CHUẨN BỊ BỘ DỮ LIỆU KITSUNE NETWORK ATTACK

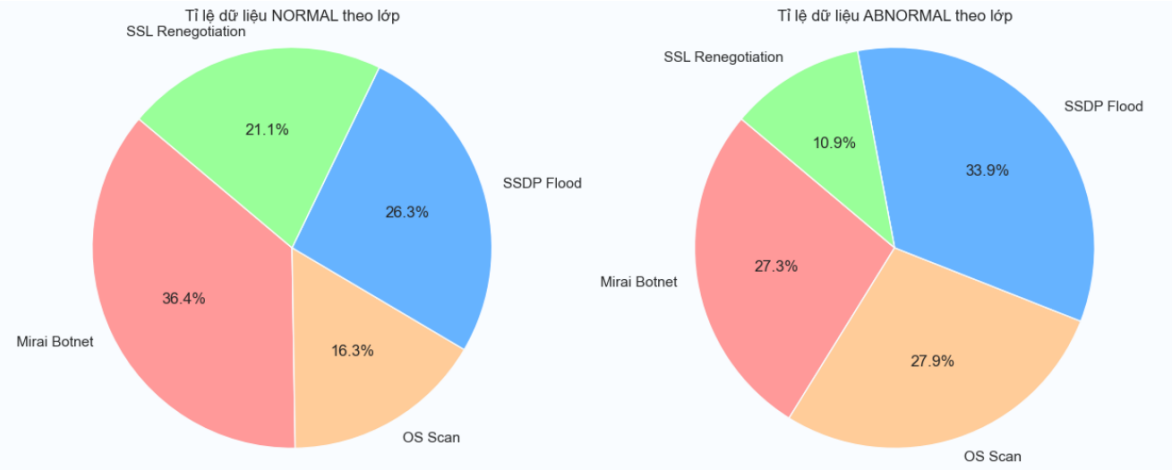
Ở chương này chúng em sẽ thực hiện trích xuất từ bộ dữ liệu Kitsune Network Attack [5] nhằm kiểm tra lại kết quả của bài báo song song với N-BaIoT. Đồng thời bộ dữ liệu này cũng được sử dụng vào việc kiểm thử mô hình training peer-to-peer ở Chương 4.

3.1. Phương pháp trích xuất dữ liệu

3.1.1. Tập dữ liệu

Trong bộ dữ liệu gốc gồm loại tấn công lên mô hình mạng: ARP MitM, Active Wiretap, Fuzzing, Mirai Botnet, OS Scan, SSDP Flood, SSL Renegotiation, SYN DoS, Video Injection.

Ở đây chúng em trích xuất ra 4 loại tấn công: Mirai Botnet, SSDP Flood, SSL Renegotiation, OS Scan.



Hình 3.1: *Phân bố dữ liệu theo loại tấn công*

Sau đó dữ liệu được chia làm các phần theo bảng sau:

Type	Tỷ lệ	Mục đích
train_dataset	40%	Huấn luyện
valid_dataset	10%	Validation khi training
dev_dataset	40%	Validation khi update
test_dataset	10%	Test mô hình

Bảng 3.1: *Phân chia bộ dữ liệu*

3.1.2. FedArtML

Thư viện FedArtML được sử dụng để phân phối dữ liệu cho học liên kết, hỗ trợ tạo các tập dữ liệu dành riêng cho từng client thông qua phân phối Dirichlet. Trong thí nghiệm, dữ liệu được chia cho 10 client *(num_clients=10)*.

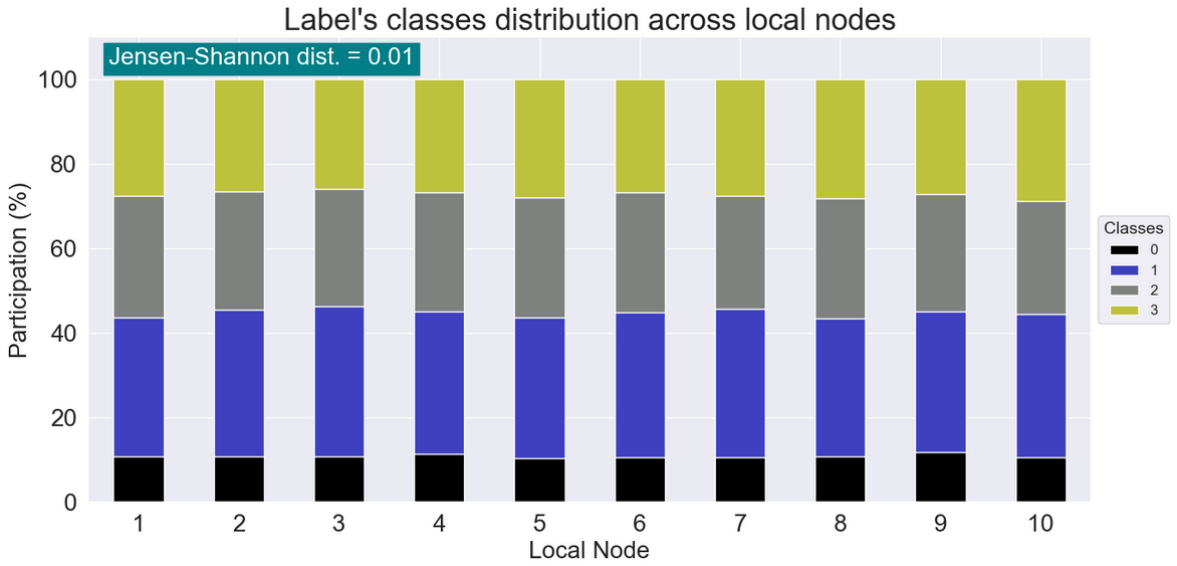
3.1.3. Phân phối Dirichlet

Phân phối Dirichlet sử dụng tham số α để kiểm soát mức độ không đồng đều của dữ liệu giữa các client:

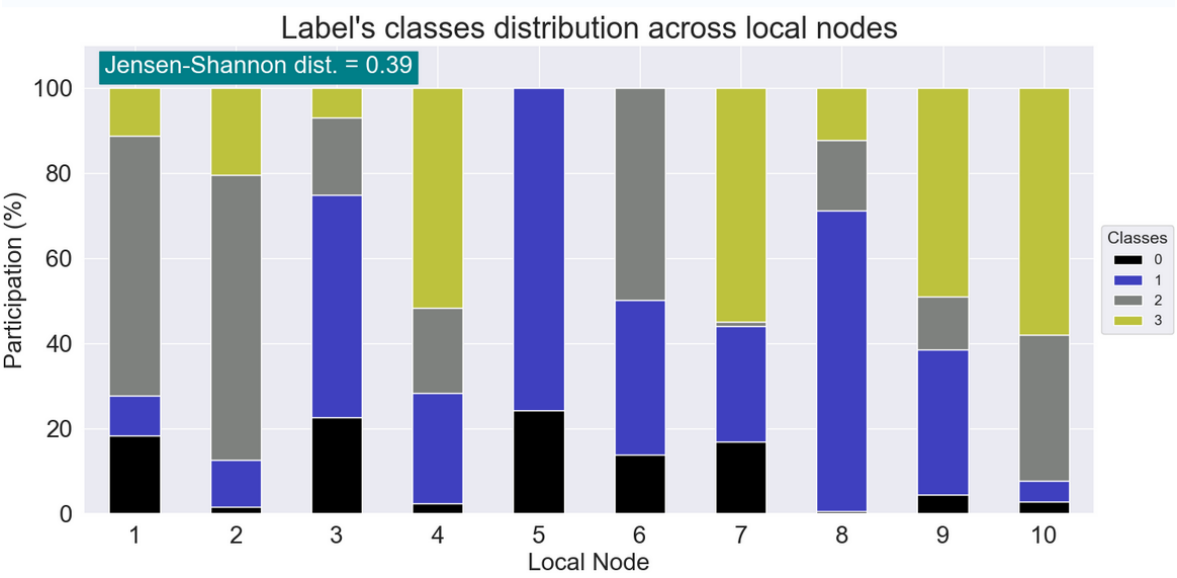
- $\alpha = 1$: Tạo phân phối không đồng đều (non-IID), một số client có thể nhận nhiều mẫu của một loại tấn công cụ thể, trong khi các client khác nhận ít hoặc không có mẫu nào.
- $\alpha = 1000$: Tạo phân phối đồng đều (IID), một số client nhận được số lượng mẫu gần giống nhau, tương tự như một tập dữ liệu tập trung được chia đều.

Với dữ liệu normal thì chúng em sẽ chia với alpha=1000 để chia dữ liệu đồng đều cho tất cả client.

Với dữ liệu normal thì chúng em sẽ chia 2 kịch bản là alpha=1 với alpha=1000 để chia dữ liệu đồng đều và không đồng đều cho tất cả client.



Hình 3.2: *Phân phối các lớp dữ liệu tấn công tấn công IID*



Hình 3.3: *Phân phối các lớp dữ liệu tấn công tấn công Non-IID*

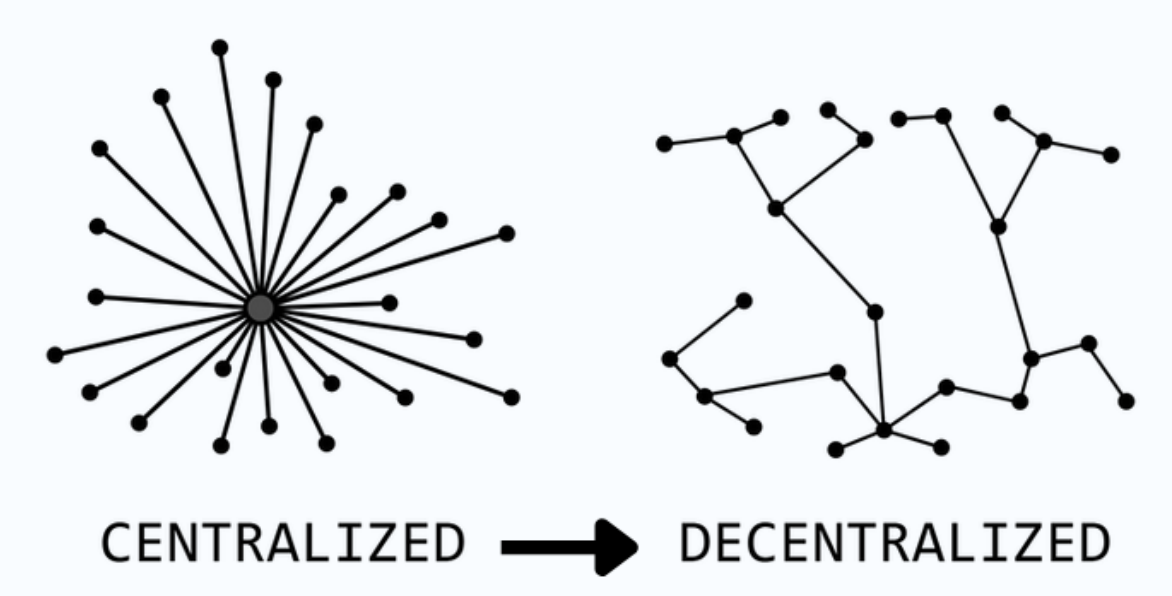
Với dữ liệu test được lấy ngẫu nhiên % số mẫu từ normal để tạo tập dữ liệu kiểm tra (tập normal sẽ bỏ các mẫu được lấy đi), sau đó chúng em chia dữ liệu kiểm tra với tỷ lệ nhãn cố định với mỗi nhãn tấn công.

CHƯƠNG 4. TRIỂN KHAI HỌC PHÂN TÁN THEO DẠNG PEER-TO-PEER

4.1. Đặt vấn đề

Trong mô hình FedMSE, sẽ luôn có một server làm nhiệm vụ điều phối quá trình training, tổng hợp model và phân phát (broadcast) model toàn cục mới đến các client ở mỗi vòng huấn luyện. Điều này tạo ra một nơi mà toàn bộ mô hình phụ thuộc vào (single point-of-failure). Nếu server bị tấn công thì sẽ khó có thể phát hiện hoặc ngăn chặn.

Ở chương này chúng em sẽ trình bày về hướng triển khai mô hình học phân tán (decentralized) theo hướng peer-to-peer nhằm loại bỏ điểm yếu là server tập trung thực hiện việc tổng hợp mô hình. Thay vào đó, các gateway (client) sẽ tự liên lạc và bầu chọn nơi tổng hợp ở mỗi round.

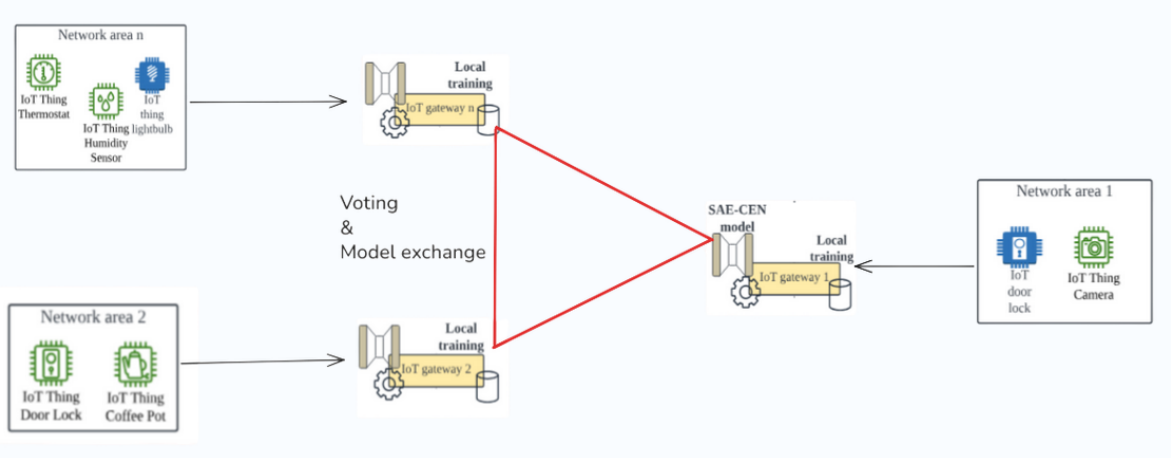


Hình 4.1: *Từ centralized sang decentralized*

4.2. Mô hình

4.2.1. Ý tưởng mô hình đơn giản

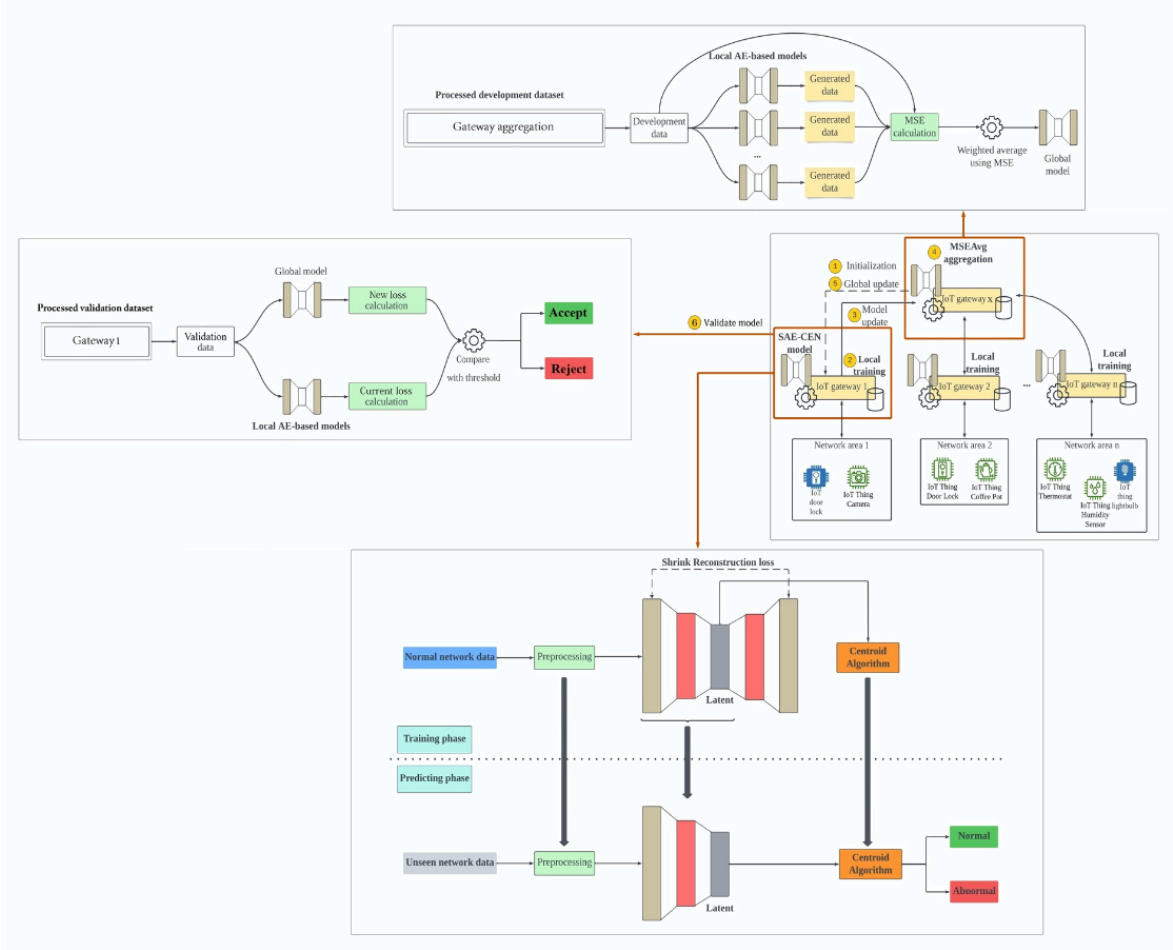
Về ý tưởng cơ bản thì chúng em sẽ thiết lập một mô hình mạng peer-to-peer. Trong đó, các client có khả năng tự giao tiếp khi mỗi vòng huấn luyện kết thúc. Các client sẽ tiến hành bầu chọn ra một gateway đáng tin cậy (ở đây chúng em dựa vào MSE score).



Hình 4.2: *Mô hình đơn giản*

4.2.2. Mô hình chi tiết

Về phần mô hình chi tiết, chúng em giữ lại quá trình training cơ bản so sánh 2 loại mô hình (Autoencoder, SAE-CEN) và 3 thuật toán tổng hợp (FedAvg, MSEAvg, FedProx). Tuy nhiên, tại mỗi client chúng em đã tích hợp chức năng tính MSE score cho các client khác, cơ chế bầu chọn (voting) và xác thực lại model vừa nhận.



Hình 4.3: *Mô hình chi tiết*

4.2.3. Cơ chế tính toán MSE score

Sau mỗi vòng huấn luyện, các client đều được tính MSE score trước khi tiến hành bầu chọn.

Thuật toán 4.1 Tính điểm MSE cho mô hình trên tập dữ liệu kiểm tra

Input: `model` — mô hình cần đánh giá `valid_loader` — bộ tải dữ liệu validation `device` — thiết bị thực thi; `client_id` — ID của client để ghi log

Output: Giá trị lỗi trung bình bình phương (MSE)

```
1  begin
2      Đặt mô hình sang chế độ đánh giá
3      Khởi tạo biến mse_loss = 0
4      Vô hiệu hóa gradient để tiết kiệm tài nguyên
5      foreach lô dữ liệu batch_input trong valid_loader do
6          if batch_input chứa giá trị NaN hoặc inf then
7              Ghi log cảnh báo với client_id return inf
8          Dữ liệu đầu ra generated_data ← model(batch_input)
9          if generated_data chứa giá trị NaN hoặc inf then
10             Ghi log cảnh báo với client_id return inf
11         Tính MSE giữa dữ liệu đầu vào và đầu ra. Cộng dồn giá trị MSE vào
           mse_loss
12     Chia tổng mse_loss cho số lượng batch. return Giá trị MSE trung bình
```

Thuật toán 4.2 Tính toán điểm MSE của các client trước khi tiến hành bầu chọn

Input: `selected_clients`, `selected_trainers`, `device`

Output: Danh sách các điểm MSE tương ứng với các client

begin

    Khởi tạo danh sách `mse_scores`

    foreach *client và trainer tại vị trí i trong danh sách đã chọn* **do**

        Gọi hàm `compute_mse_score` với:

        - `trainer.model`
        - `client["valid_loader"]`
        - `device`
        - `trainer.client_id`

        if *`mse_score > 3`* **then**

            Ghi log cảnh báo rằng client này có MSE cao bất thường và sẽ bị loại khỏi quá trình bầu chọn `continue`

        Thêm `(i, mse_score)` vào danh sách `mse_scores`

        Ghi log thông tin điểm MSE của client

    return *`mse_scores`*

4.2.4. Cơ chế xác thực model

Sau khi aggregator tổng hợp mô hình (aggregate_models), mô hình tổng hợp này sẽ được gửi tới tất cả các client. Trước khi client cập nhật mô hình mới này, nó sẽ kiểm tra xem mô hình mới có thực sự tốt hơn (hoặc không tệ hơn quá nhiều) so với mô hình hiện tại của mình không.

1. Tính loss (thường là MSE) của mô hình hiện tại trên validation set.
2. Load state_dict của mô hình mới (mô hình tổng hợp) vào model.

3. Tính loss của mô hình mới trên validation set.

4. So sánh loss mới với loss cũ:

- Nếu loss mới <= loss cũ × (1 + threshold) → Chấp nhận mô hình mới.
- Nếu loss mới >= loss cũ × (1 + threshold) → Từ chối mô hình mới, giữ lại mô hình cũ.

Tổng kết:

- Mỗi client chỉ nhận mô hình tổng hợp nếu mô hình đó không làm tăng loss validation quá nhiều.
- Nếu không đạt, client giữ lại mô hình cũ.

Cơ chế này giúp bảo vệ chất lượng mô hình của từng client trong quá trình federated learning.

Thuật toán 4.3 Xác thực mô hình mới dựa trên MSE validation loss

Input: `new_model_state`, `valid_loader`, `threshold`

Output: `True` nếu mô hình được chấp nhận, ngược lại `False`

begin

Ghi log bắt đầu xác thực Lưu trạng thái hiện tại vào `current_model_state`, khởi tạo loss

`// Tính loss hiện tại`

`self.model.eval()`; `with no_grad`: **foreach** *batch trong* ***`valid_loader`*** **do**

if *batch hoặc loss chứa* ***`NaN/inf`*** **then**

Ghi log, **return** `False`

Cộng dồn `current_loss`

Chia trung bình `current_loss`

`// Tính loss mô hình mới`

Load `new_model_state`; `with no_grad`: **foreach** *batch trong* ***`valid_loader`*** **do**

if *batch hoặc loss chứa* ***`NaN/inf`*** **then**

Ghi log, khôi phục mô hình cũ, **return** `False`

Cộng dồn `new_loss`

Chia trung bình `new_loss`

`// So sánh`

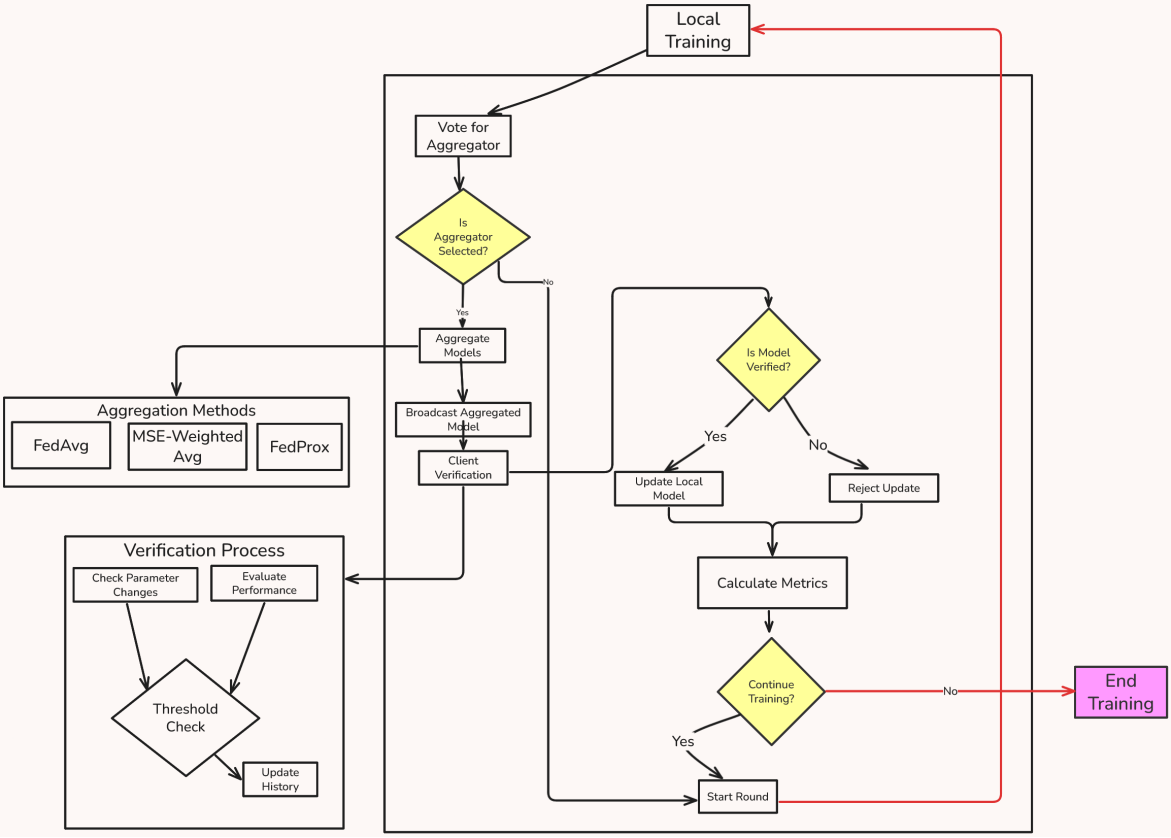
if ***`new_loss`*** *nhỏ hơn ngưỡng* **then**

Ghi log chấp nhận, **return** `True`

else

Ghi log từ chối, khôi phục model, **return** `False`

4.3. Quy trình huấn luyện model



Hình 4.4: *Sơ đồ chi tiết cho một vòng huấn luyện*

4.3.1. Chuẩn bị dữ liệu và khởi tạo client

1. Đọc file cấu hình JSON để lấy danh sách thiết bị, đường dẫn dữ liệu, v.v.
2. Với mỗi thiết bị (client):
   (a) Đọc dữ liệu thường, bất thường, và dữ liệu test mới.
   (b) Chia dữ liệu thành các phần: train, valid, dev, test.
   (c) Chuẩn hóa dữ liệu với *IoTDataProccessor.*
   (d) Tạo các *IoTDataset* và *DataLoader* cho từng phần dữ liệu.
   (e) Lưu thông tin vào *client_info.*

Đặt các tùy chọn cho thí nghiệm: num_participants, epoch, num_rounds, lr_rate, shrink_lambda, network_size, data_seed, no_Exp, num_runs, batch_size, new_device, min_val_loss, global_patience, global_worse, metric, dim_features, scene_name, config_file [1]

4.3.2. Khởi tạo mô hình và trainer cho từng client

1. Tạo mô hình toàn cục (*global_model*), ví dụ: Shrink_Autoencoder hoặc Autoencoder.
2. Với mỗi client, tạo một trainer:
   (a) *ClientTrainer(model, train_loader, ...)*
   (b) Mỗi trainer giữ một bản sao mô hình toàn cục.
3. Cập nhật danh sách peers cho mỗi trainer (các client khác).

4.3.3. Federated Rounds

Lặp qua số vòng (num_rounds)

1. Chọn client tham gia
   - Chọn ngẫu nhiên một tỷ lệ client (num_participants) để tham gia vòng này.
2. Huấn luyện cục bộ (Local Training)
   - Mỗi client được huấn luyện trên dữ liệu của mình: *trainer.run(train_loader, valid_loader)*
3. Sau khi huấn luyện, mỗi client được đánh giá trên tập validation: *mse_score = compute_mse_score(trainer.model, client["valid_loader"], device, trainer.client_id)*

4.3.4. Voting

Diễn ra trên mỗi client:

- Nếu không có client hợp lệ, chọn ngẫu nhiên.
- Nếu có, chọn client có MSE thấp nhất làm aggregator.

4.3.5. Aggregator được chọn thực hiện tổng hợp mô hình

Aggregator gọi: *aggregate_ models(valid_ loader, received_ models)*

- Nhận models từ các client khác: *receive_ models(selected_ client_ ids)*
- Tổng hợp model theo kiểu avg, fedprox, mse_avg

Thuật toán 4.4 Tổng hợp mô hình từ các client khác

Input: `valid_loader`, `received_models`

begin
- Ghi log bắt đầu tổng hợp mô hình với thuật toán `update_type`
- `own_model` ← mô hình hiện tại của client `all_models` ← `[own_model]` nối với `received_models`
- **if** *`update_type`* *là* *`"avg"`* **then**
  - Gọi `fed_avg(all_models)`
- **else if** *`update_type`* *là* *`"mse_avg"`* **then**
  - Gọi `fed_mse_avg(all_models, valid_loader)`
- **else if** *`update_type`* *là* *`"fedprox"`* **then**
  - Gọi `fedprox(all_models, mu = ...)`
- **else**
  - Báo lỗi loại tổng hợp không được hỗ trợ
- Ghi log hoàn tất cập nhật mô hình tổng hợp

4.3.6. Phân phối mô hình tổng hợp cho tất cả client

1. Mô hình tổng hợp được gửi tới tất cả client.
2. Mỗi client kiểm tra mô hình mới bằng: validate_model(aggregated_model, valid_loader, threshold=0.1)
3. Nếu loss tăng quá nhiều thì giữ mô hình cũ.

4.3.7. Đánh giá mô hình và lưu kết quả

Đánh giá mô hình toàn cục trên tập validation và test:

Thuật toán 4.5 Đánh giá mô hình Autoencoder trên tập test

Input: `test_loader`, `train_loader (tùy chọn)`

Output: AUC hoặc F1 score tùy theo `self.metric`

```
begin
    Đặt mô hình về chế độ đánh giá
    if mô hình là autoencoder then
        Khởi tạo danh sách: anomaly_score, test_label with no_grad: foreach batch trong test_loader do
            Dự đoán đầu ra từ mô hình  Tính recon_loss giữa input và output  Tính điểm bất thường và lưu nhãn thực tế
        Ghép các nhãn thành vector test_label
        if metric là "AUC" then
            Tính AUC từ nhãn và điểm bất thường  Ghi log AUC  return AUC
        if metric là "classification" then
            Tính F1, precision, recall từ nhãn và điểm bất thường  Ghi log F1, precision, recall  return F1
```

4.3.8. Cơ chế dừng khi đạt đủ tiêu chuẩn

Trong quá trình huấn luyện hệ thống tiến hành kiểm tra global_loss (validating loss của aggregator trên valid_loader) để quyết định có dừng sớm hay không.

Nếu global_loss không cải thiện trong global_patience vòng (mặc định global_patience= hệ thống dừng sớm (kết thúc round).

CHƯƠNG 5. KẾT QUẢ THÍ NGHIỆM

5.1. Kịch bản triển khai đánh giá

Kịch bản sẽ được chia làm 2 kịch bản chính là IID và non IID.

Với mỗi kịch bản, nhóm sẽ tiến hành thí nghiệm với các phần trăm số client tham gia training mỗi round.

1. 50% số lượng client tham gia
2. 60% số lượng client tham gia
3. 70% số lượng client tham gia
4. 80% số lượng client tham gia
5. 90% số lượng client tham gia
6. 100% số lượng client tham gia

5.1.1. Các tham số của thí nghiệm

Trong đó:

- num_participants: thay đổi tương ứng với thí nghiệm.
- epoch, num_rounds, shrink_lambda, ... thay đổi tùy ý để tinh chỉnh và tối ưu model.
- batch_size: 64. Ở đây trong quá trình dùng với hardware-acceleration sử dụng CUDA thì 64 là số lượng tối đa mà máy em có thể load dữ liệu trước khi bị tràn bộ nhớ.
- metric: Tiêu chí đánh giá của thí nghiệm là AUC

Tham số	**Ý nghĩa**
num_participants	Tỷ lệ client tham gia mỗi round (VD: 0.9 = 90% số client sẽ được chọn ngẫu nhiên mỗi round)
epoch	Số epoch huấn luyện cục bộ trên mỗi client trong mỗi round
num_rounds	Số vòng federated learning (số lần lặp toàn cục)
lr_rate	Learning rate cho optimizer (Adam)
shrink_lambda	Tham số regularization cho mô hình Shrink_Autoencoder
network_size	Số lượng client trong mạng
data_seed	Seed cho randomizer để đảm bảo thí nghiệm có thể được thực hiện lại
no_Exp	Tên thí nghiệm (lưu vào checkpoint), tự động sinh dựa trên các tham số trên
num_runs	Số lần lặp lại toàn bộ thí nghiệm để lấy kết quả trung bình
batch_size	Batch size cho DataLoader khi huấn luyện
new_device	Nếu True, sẽ thêm dữ liệu test mới vào test set
min_val_loss	Giá trị loss nhỏ nhất toàn cục (dùng cho early stopping)
global_patience	Số vòng cho phép không cải thiện loss trước khi dừng sớm (early stopping)
global_worse	Đếm số vòng không cải thiện loss
metric	Chỉ số đánh giá: "AUC" hoặc "classification"

Bảng 5.1: *Các tham số và ý nghĩa của chúng trong quá trình huấn luyện mô hình*

5.2. Cấu hình máy tính chạy thí nghiệm

- Hệ điều hành: Microsoft Windows 11 Pro, 64-bit
- CPU: AMD Ryzen 7 3750H (4 nhân, 8 luồng, 2.3 GHz)
- RAM: 16 GB DDR4
- GPU: NVIDIA GeForce GTX 1650, 4 GB VRAM
- Driver: 576.80
- CUDA: 12.9
- Python: 3.12.9

Đồng thời, nhóm cũng có kết hợp chạy song song thí nghiệm trên Google Colab để rút ngắn thời gian chạy.

5.2.1. Thời gian chạy thí nghiệm

Xét ở kịch bản IID, 100% client tham gia thì thí nghiệm tốn khoảng 6 giờ (có hardware-acceleration) và khoảng 4 giờ trên Google Colab.

5.3. Kết quả thí nghiệm

MSEAvg có hiệu quả cao nhất trong việc học biểu diễn tiềm ẩn cho phân biệt dữ liệu bình thường và bất thường.

FedProx cho thấy hiệu quả phân biệt kém hơn, có thể do đặc tính regularization của nó không phù hợp với SAE-CEN trong trường hợp này.

FedAvg là trung gian — cho kết quả ổn nhưng không bằng MSEAvg.

5.3.1. Nhận xét về hiệu suất giữa các mô hình

5.3.1.1. Nhận xét về hiệu suất giữa các mô hình với dữ liệu IID

SAE-CEN đạt kết quả cao hơn Autoencoder:

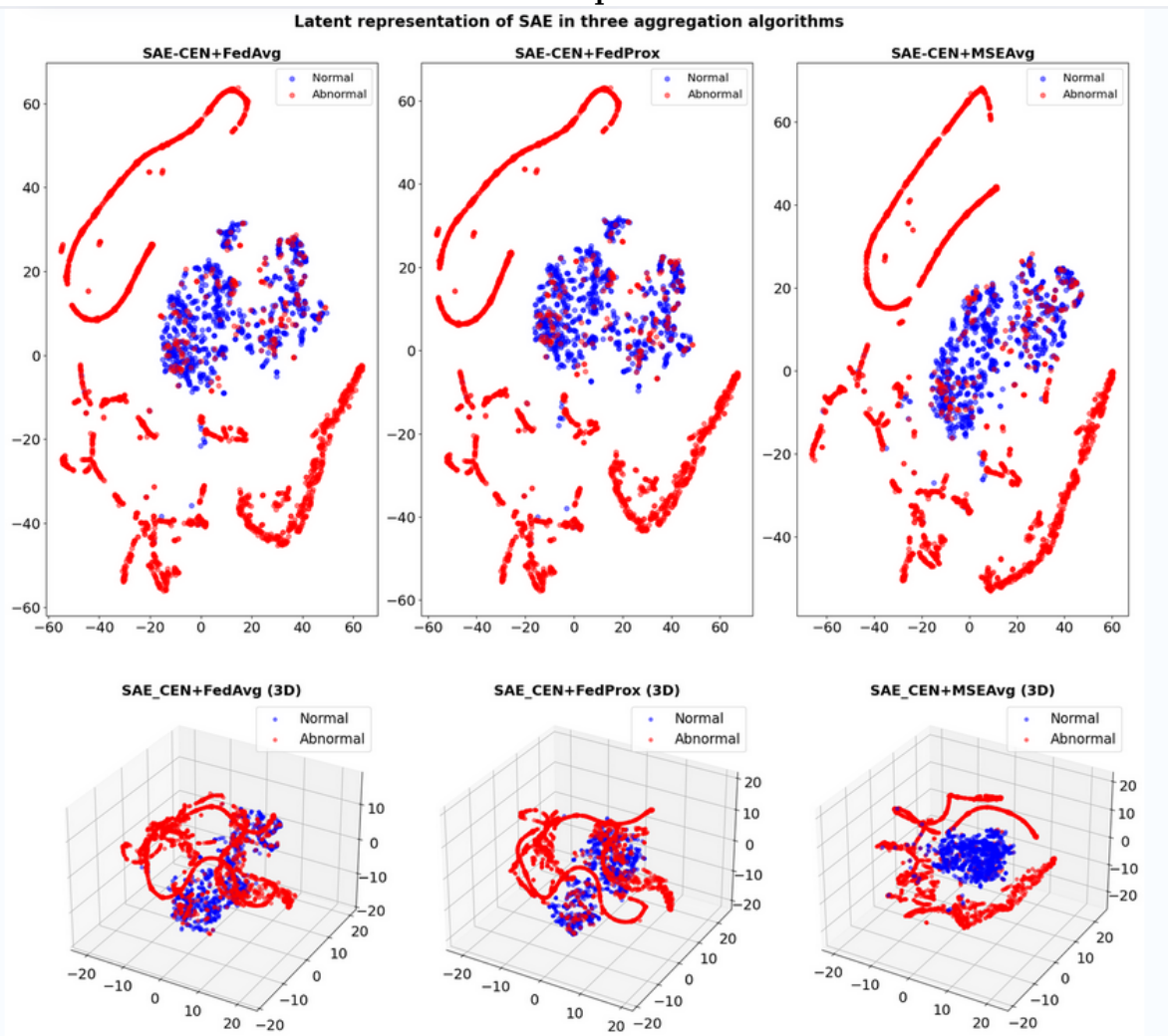
Mean AUC của SAE-CEN (93.33–93.77%) cao hơn Autoencoder (92.04–92.97%) trên tất cả các thuật toán tổng hợp. Sự khác biệt dao động từ khoảng 0.36% (MSEAvg) đến 0.73% (FedProx).

5.3.1.2. Nhận xét về hiệu suất giữa các mô hình với dữ liệu non-IID

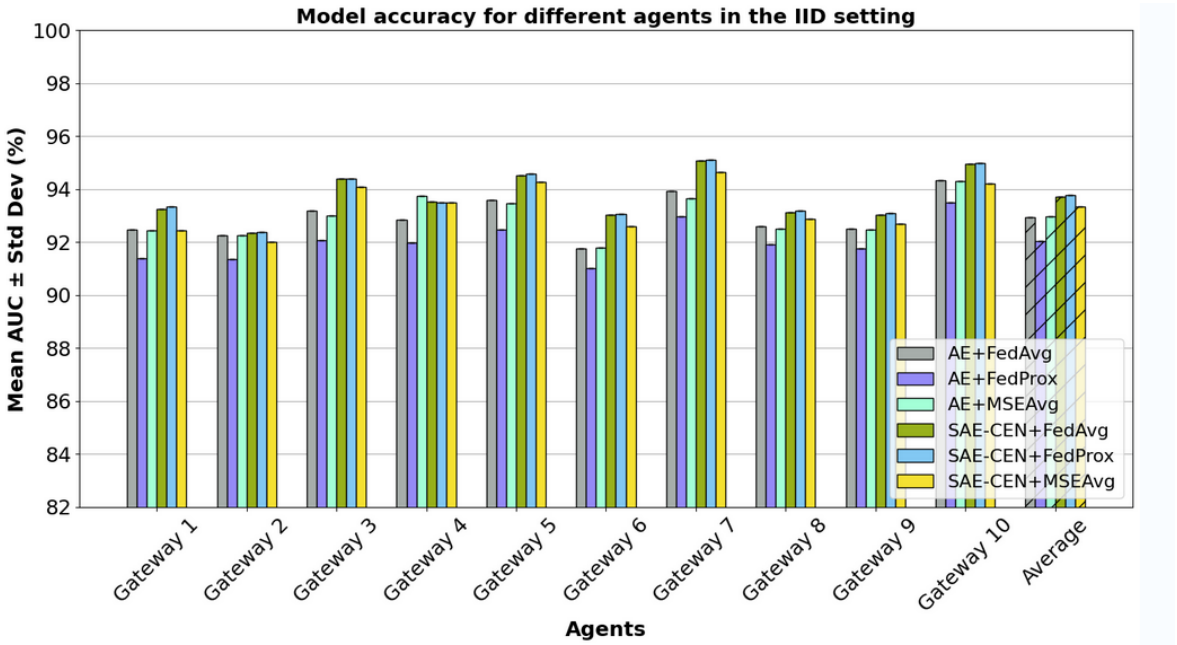
SAE-CEN có kết quả cao hơn Autoencoder:

Mean AUC của SAE-CEN (93.41–93.53%) cao hơn Autoencoder (92.36–92.66%) trên tất cả các thuật toán, với sự khác biệt dao động từ 0.05% (FedProx) đến 0.87% (FedAvg). Điều này cho thấy SAE-CEN có thể phù hợp hơn trong điều kiện dữ liệu không đồng nhất.

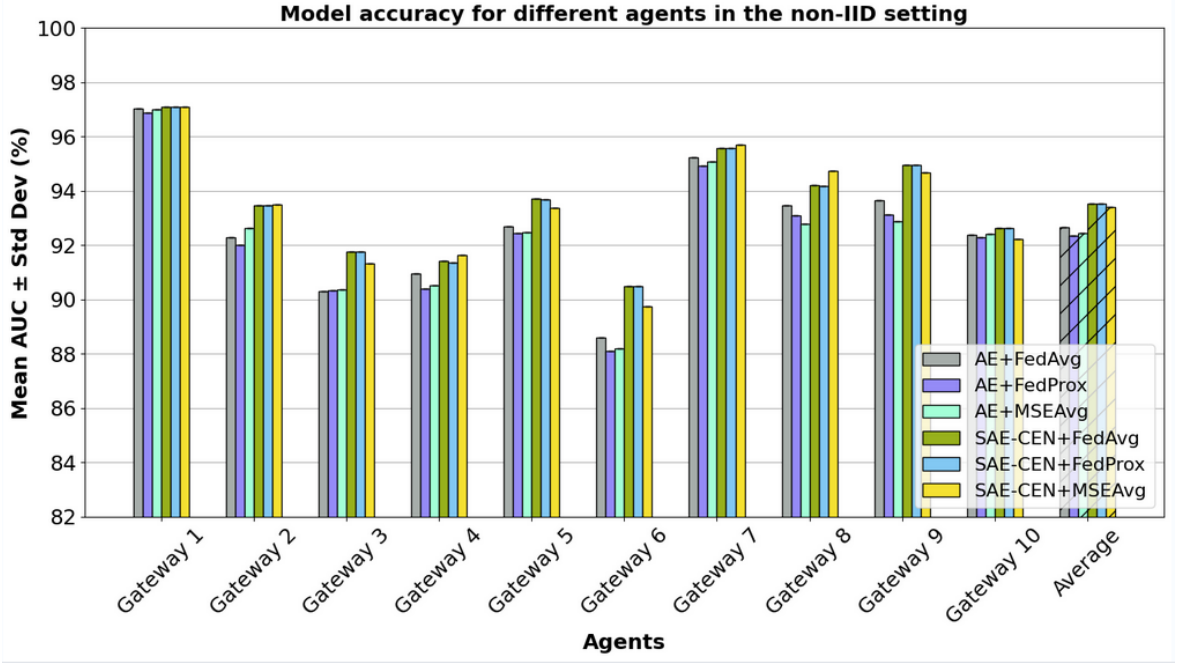
Latent Expression:



Hình 5.1: *Latent Expression của model SAE-CEN sử dụng các thuật toán tổng hợp*



Hình 5.2: *Accuracy ở các gateway trong môi trường IID*



Hình 5.3: *Accuracy ở các gateway trong môi trường Non-IID*

5.3.2. Nhận xét về hiệu suất giữa các thuật toán

5.3.2.1. Nhận xét về hiệu suất giữa các thuật toán tổng hợp với dữ liệu IID

Autoencoder:

- FedAvg (92.95%) và MSEAvg (92.97%) có hiệu suất gần như tương đương, với sự khác biệt nhỏ (0.02%), cho thấy cả hai thuật toán đều hoạt động ổn định trong bối cảnh này.
- FedProx (92.04%) có mean AUC thấp hơn đáng kể (khoảng 0.91% so với FedAvg), có thể do FedProx kém ổn định hoặc ít phù hợp với kiến trúc Autoencoder trong trường hợp này.

SAE-CEN:

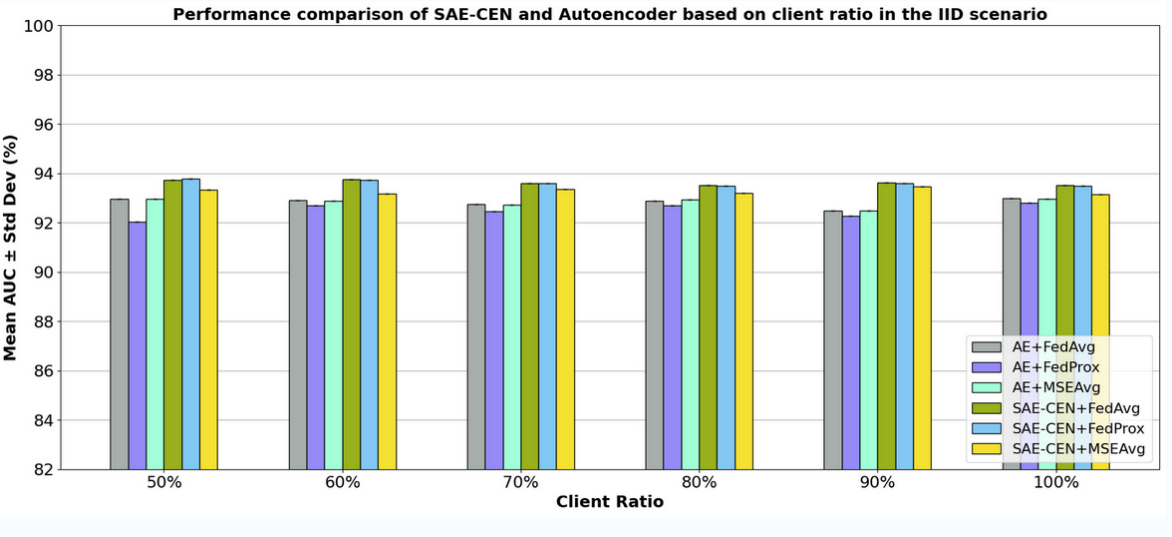
- FedProx (93.77%) đạt mean AUC cao nhất, vượt qua FedAvg (93.73%) và MSEAvg (93.33%) với chênh lệch lần lượt là 0.04% và 0.44%.
- MSEAvg (93.33%) có hiệu suất thấp nhất trong ba thuật toán, điều này có thể liên quan đến cách tổng hợp chưa tối ưu với đặc điểm của mô hình SAE-CEN trong trường hợp này.

Sự khác biệt giữa các thuật toán trong SAE-CEN nhỏ hơn so với Autoencoder, gợi ý rằng SAE-CEN ít nhạy cảm hơn với sự thay đổi của thuật toán tổng hợp.

5.3.3. Hiệu suất giữa các mô hình ở các tỉ lệ tham gia huấn luyện khác nhau

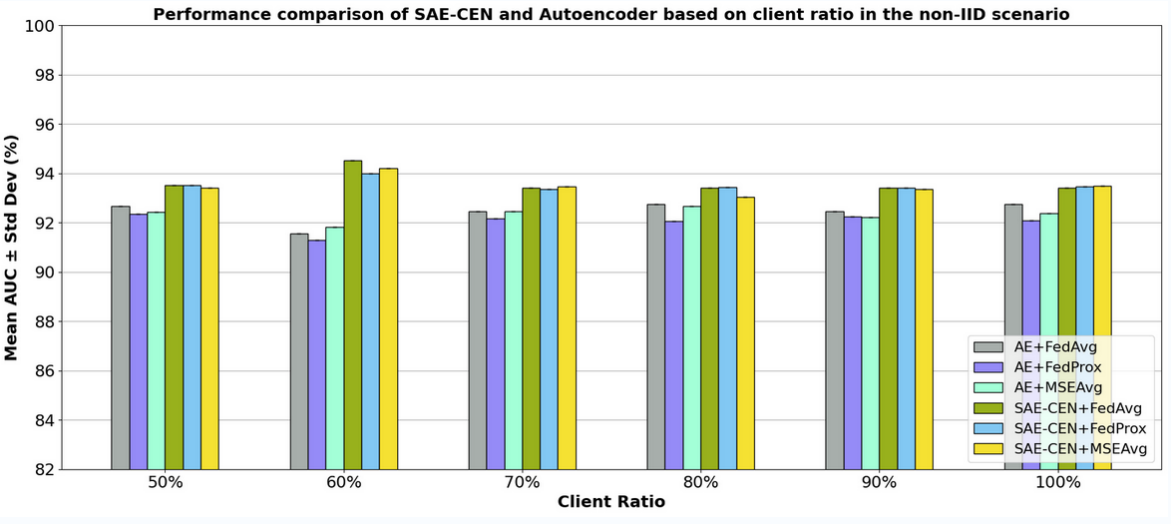
5.3.3.1. Nhận xét về tỉ lệ tham gia train local với cả 2 kịch bản IID và nonIID

SAE-CEN có Mean AUC cao hơn Autoencoder trong cả hai kịch bản, với chênh lệch trung bình khoảng 0.78% (IID) và 1.09% (non-IID). SAE-CEN cho thấy hiệu suất ổn định hơn trong điều kiện dữ liệu không đồng nhất, trong khi Autoencoder bị ảnh hưởng nhiều hơn.



Hình 5.4: *Accuracy của mô hình ở các tỉ lệ client tham gia huấn luyện (IID)*

FedAvg là thuật toán tổng hợp đạt Mean AUC cao nhất cho cả hai mô hình trong cả hai kịch bản. Tuy nhiên, MSEAvg cải thiện hiệu suất khi chuyển sang kịch bản non-IID, đặc biệt với SAE-CEN, cho thấy mô hình này có khả năng thích ứng với dữ liệu không đồng nhất tốt hơn.



Hình 5.5: *Accuracy của mô hình ở các tỉ lệ client tham gia huấn luyện (non-IID)*

5.4. Kết luận

Tổng hợp hiệu suất theo từng kịch bản:

- Trong kịch bản IID, SAE-CEN đạt mean AUC từ 93.33% đến 93.77%, cao hơn so với Autoencoder (92.04% đến 92.97%).
- Trong kịch bản non-IID, SAE-CEN duy trì mean AUC từ 93.41% đến 93.53%, cao hơn Autoencoder (92.36% đến 92.66%).

Khả năng duy trì hiệu suất trong điều kiện dữ liệu không đồng nhất, kết hợp với thuật toán MSEAvg, cho thấy mô hình SAE-CEN có tiềm năng ứng dụng trong các hệ thống phân tán, nơi dữ liệu giữa các thiết bị IoT không đồng nhất.

TÀI LIỆU THAM KHẢO

Tiếng Anh:

[1] Van Tuan Nguyen and Razvan Beuran (2025), "FedMSE: Semi-supervised federated learning approach for IoT network intrusion detection", *Computers Security*, 151, p. 104337, ISSN: 0167-4048, DOI: https://doi.org/10.1016/j.cose.2025.104337, URL: https://www.sciencedirect.com/science/article/pii/S0167404825000264.

[2] H. Brendan McMahan et al. (2016), "Federated Learning of Deep Networks using Model Averaging", *CoRR*, abs/1602.05629, arXiv: 1602.05629, URL: http://arxiv.org/abs/1602.05629.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016), *Deep Learning*, http://www.deeplearningbook.org, MIT Press.

[4] Yair Meidan et al. (2018), "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders", *IEEE Pervasive Computing*, 17 (3), pp. 12–22, DOI: 10.1109/MPRV.2018.03367731.

[5] UCI Machine Learning Repository (2019), "Kitsune Network Attack", DOI: https://doi.org/10.24432/C5D90Q.