

TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

KHÓA LUẬN TỐT NGHIỆP
SO SÁNH HIỆU QUẢ CỦA CÁC
THUẬT TOÁN HUẤN LUYỆN MÔ
HÌNH HỌC LIÊN KẾT

Giảng viên hướng dẫn: ThS. Huỳnh Thành Lộc

Sinh viên thực hiện: Lương Tiến Đạt – 22DH114497

Nguyễn Anh Huy – 22DH111251

TP. HỒ CHÍ MINH – 08/2025

TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

**KHÓA LUẬN TỐT NGHIỆP
SO SÁNH HIỆU QUẢ CỦA CÁC
THUẬT TOÁN HUẤN LUYỆN MÔ
HÌNH HỌC LIÊN KẾT**

Giảng viên hướng dẫn: ThS. Huỳnh Thành Lộc

Sinh viên thực hiện: Lương Tiến Đạt – 22DH114497

Nguyễn Anh Huy – 22DH111251

TP. HỒ CHÍ MINH – 08/2025

Lời cảm ơn

Khóa luận tốt nghiệp là một cột mốc quan trọng, thể hiện sự trưởng thành và nỗ lực bền bỉ của chúng em trong suốt quá trình học tập tại Trường Đại học Ngoại ngữ – Tin học TP. Hồ Chí Minh. Để có được kết quả này, nhóm em xin gửi lời tri ân sâu sắc đến tất cả các cá nhân và tổ chức đã luôn đồng hành, hỗ trợ và tạo mọi điều kiện thuận lợi để em hoàn thành đề tài.

Với tất cả lòng kính trọng và biết ơn sâu sắc, chúng em xin được gửi lời cảm ơn và lời bày tỏ lòng biết ơn chân thành nhất tới thầy Huỳnh Thành Lộc, người đã trực tiếp hướng dẫn và đồng hành cùng chúng em trong suốt quá trình thực hiện đề tài. Thầy không chỉ là người định hướng nghiên cứu, mà còn là người truyền cảm hứng, luôn tạo điều kiện tốt nhất để em có thể phát huy hết khả năng của mình. Em đặc biệt cảm ơn thầy đã luôn dành thời gian quý báu để giải đáp mọi thắc mắc, đưa ra những lời khuyên sâu sắc và kịp thời. Sự hỗ trợ về phần cứng trong quá trình thực nghiệm của thầy là một sự giúp đỡ vô cùng quý giá, giúp em có thể vượt qua những khó khăn về tài nguyên và hoàn thành các mô phỏng phức tạp một cách hiệu quả nhất.

Không có sự hướng dẫn và động viên của thầy, đề tài này đã không thể hoàn thành một cách trọn vẹn. Em xin kính chúc thầy luôn dồi dào sức khỏe, hạnh phúc và gặt hái được nhiều thành công hơn nữa trong sự nghiệp trồng người.

Em cũng xin gửi lời tri ân sâu sắc đến toàn thể quý thầy cô trong Khoa Công nghệ Thông tin. Em xin cảm ơn các thầy cô đã giảng dạy, trang bị cho em những kiến thức nền tảng vững chắc trong suốt những năm học qua. Chính những bài giảng tâm huyết và sự tận tình của quý thầy cô đã tạo nên hành trang kiến thức, giúp em có được nền tảng cần thiết để tiếp cận và giải quyết các vấn đề chuyên sâu trong đề tài này.

Một lần nữa, em xin kính chúc quý thầy cô dồi dào sức khỏe, thành công trong sự nghiệp và hạnh phúc trong cuộc sống.

Chúng em xin chân thành cảm ơn!

Lời cam đoan

Nhóm xin cam đoan rằng, toàn bộ nội dung trong khóa luận này là kết quả của quá trình học tập, nghiên cứu và thực hiện một cách độc lập của nhóm. Các số liệu, kết quả, cũng như những phân tích và đánh giá được trình bày trong đây là hoàn toàn trung thực và chưa từng được công bố trong bất kỳ công trình nghiên cứu nào khác.

Chúng em đã nghiêm túc tuân thủ các quy định về học thuật, đảm bảo rằng mọi tài liệu tham khảo, trích dẫn, và các nguồn thông tin khác đã được sử dụng một cách chính xác và được liệt kê đầy đủ trong danh mục tài liệu tham khảo của khóa luận.

Chúng em xin chịu hoàn toàn trách nhiệm về tính trung thực và chính xác của các thông tin, dữ liệu và kết quả nghiên cứu được trình bày trong khóa luận này.

Tóm tắt

Trong bối cảnh dữ liệu từ các thiết bị biên như thiết bị di động, IoT ngày càng tăng, việc tận dụng nguồn dữ liệu khổng lồ này để huấn luyện các mô hình học máy trở thành một nhu cầu cấp thiết. Tuy nhiên, phương thức huấn luyện truyền thống yêu cầu tập trung dữ liệu, tiềm ẩn nhiều rủi ro về quyền riêng tư, an toàn thông tin và chi phí truyền tải. Học liên kết (Federated Learning - FL) là một giải pháp đột phá, cho phép huấn luyện mô hình trên dữ liệu cục bộ của từng thiết bị và chỉ truyền các tham số đã cập nhật về máy chủ trung tâm. Điều này giúp bảo vệ quyền riêng tư của người dùng một cách hiệu quả và tối ưu hóa tài nguyên mạng.

Đề tài này tập trung vào việc phân tích, triển khai và đánh giá ba thuật toán FL tiêu biểu: FedAvg, FedProx và FedAdam. Ba thuật toán này đại diện cho ba hướng tiếp cận khác nhau trong việc giải quyết các thách thức đặc thù của FL, đặc biệt là vấn đề dữ liệu không đồng nhất (non-IID) - một kịch bản phổ biến trong thực tế.

Phân tích kết quả tập trung vào các tiêu chí then chốt: độ chính xác, hàm mất mát và thời gian thực thi. Bằng cách so sánh hiệu suất trung bình trong 20 vòng huấn luyện cuối cùng, đề tài cung cấp một cái nhìn khách quan về sự đánh đổi giữa hiệu quả mô hình và chi phí tính toán. Các thử nghiệm được triển khai trên nền tảng mã nguồn mở Flower, đảm bảo tính khoa học và khả thi về tài nguyên.

Kết quả của đề tài có ý nghĩa khoa học và thực tiễn rõ rệt. Về mặt học thuật, nó cung cấp một so sánh hệ thống ba thuật toán có ảnh hưởng lớn trong FL. Về mặt thực tiễn, đề tài cung cấp cơ sở để lựa chọn thuật toán phù hợp cho các hệ thống FL thực tế, tối ưu hóa chi phí triển khai và ứng dụng trong các bài toán thị giác máy tính phân tán.

Mục lục

Mục lục	IV
Danh mục từ viết tắt.....	VII
Danh mục bảng	VIII
Danh mục hình	IX
CHƯƠNG 1: MỞ ĐẦU.....	1
1.1. Lý do chọn đề tài	1
1.2. Lịch sử nghiên cứu vấn đề.....	1
1.3. Mục đích nghiên cứu	2
1.4. Đối tượng nghiên cứu.....	3
1.5. Phạm vi nghiên cứu	4
1.6. Phương pháp nghiên cứu	5
1.6.1. Phân tích lý thuyết	5
1.6.2. Thực nghiệm mô phỏng.....	6
1.6.3. Mục tiêu của phương pháp.....	7
1.7. Ý nghĩa khoa học và thực tiễn	8
1.7.1. Ý nghĩa khoa học.....	8
1.7.2. Ý nghĩa thực tiễn	9
1.8. Bố cục của khóa luận.....	9
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	12
2.1. Mô hình máy học truyền thống	12
2.2. Mô hình học liên kết.....	12
2.2.1. Tóm tắt nội dung các công trình nghiên cứu liên quan	12
2.2.2. Kiến trúc mô hình.....	13

2.3.	Phân loại Federated Learning	17
2.3.1.	<i>Bài toán học liên kết tập trung</i>	17
2.3.2.	<i>Bài toán học liên kết phi tập trung</i>	17
2.3.3.	<i>Bài toán học liên kết không đồng nhất</i>	18
2.4.	Các thuật toán học liên kết	19
2.4.1.	<i>Thuật toán FedAvg</i>	19
2.4.2.	<i>Thuật toán FedProx</i>	22
2.4.3.	<i>Thuật toán FedAdam</i>	26
2.5.	Các tiêu chí đánh giá mô hình trong Federated Learning	30
2.5.1.	<i>Các tiêu chí đánh giá hiệu suất mô hình:</i>	31
2.5.2.	<i>Hàm mất mát (Loss Function)</i>	32
2.5.3.	<i>Các chỉ số thời gian và hiệu quả hệ thống</i>	33
2.5.4.	<i>Vai trò trong đánh giá mô hình học liên kết</i> :	33
CHƯƠNG 3:	XÂY DỰNG MÔ HÌNH	34
3.1.	Môi trường và công cụ thực nghiệm	34
3.1.1.	<i>Ngôn ngữ lập trình và nền tảng</i>	34
3.1.2.	<i>Thư viện và framework sử dụng</i>	34
3.1.3.	<i>Môi trường thực thi và phần cứng</i>	35
3.2.	Tập dữ liệu và thiết kế kịch bản non-IID	35
3.2.1.	<i>Tập dữ liệu CIFAR10 và Fashion-MNIST</i>	35
3.2.2.	<i>Thiết kế kịch bản non-IID</i>	37
3.3.	Mô hình ResNet18 sử dụng trong Federated Learning	38
3.3.1.	<i>Cấu hình tổng quan mô hình Resnet18</i>	38
3.3.2.	<i>Cấu hình mô hình Resnet trên từng tập dữ liệu</i>	39

3.4.	Cấu hình thực nghiệm và tham số chạy mô phỏng	40
3.5.	Điều chỉnh tham số cho FedProx và FedAdam	41
3.5.1.	<i>Điều chỉnh tham số cho FedProx</i>	41
3.5.2.	<i>Điều chỉnh tham số cho FedAdam</i>	43
CHƯƠNG 4: THỰC NGHIỆM CHƯƠNG TRÌNH		46
4.1.	Kết quả huấn luyện của từng chiến lược	46
4.1.1.	<i>Kết quả huấn luyện trên dữ liệu Fashion-MNIST</i>	46
4.1.2.	<i>Kết quả huấn luyện trên dữ liệu CIFAR10</i>	52
4.2.	So sánh hiệu quả và phân tích độ chính xác.....	57
4.2.1.	<i>So sánh hiệu quả của 3 thuật toán trên từng tập dữ liệu</i>	57
4.2.2.	<i>So sánh độ hiệu quả trên dữ liệu không đồng nhất</i>	62
4.2.3.	<i>So sánh tốc độ hội tụ trên mỗi vòng</i>	65
4.2.4.	<i>Hiệu quả của 3 thuật toán so với thời gian chạy</i>	66
4.3.	Nhận xét.....	69
4.3.1.	<i>Đánh giá tổng quan về từng thuật toán</i>	69
4.3.2.	<i>Tầm quan trọng của việc lựa chọn tham số</i>	70
4.3.3.	<i>Lựa chọn chiến lược phù hợp cho từng bài toán</i>	70
CHƯƠNG 5: KẾT LUẬN		71
5.1.	Tóm tắt kết quả đạt được	71
5.2.	Hạn chế của quá trình huấn luyện	72
5.3.	Hướng phát triển trong tương lai	73
TÀI LIỆU THAM KHẢO		75

Danh mục từ viết tắt

STT	Từ viết tắt	Từ đầy đủ	Nghĩa tiếng Việt
1	DL	Deep Learning	Học sâu
2	FL	Federared Learning	Học liên kết
3	IID	Independent and identically distributed	Dữ liệu độc lập và phân bố giống nhau
4	ML	Machine Learning	Máy học
5	non-IID	Non independent and identically distributed	Dữ liệu không độc lập và phân bố giống nhau
6	FedAdagrad	Federated Adaptive Gradient	Tối ưu hóa gradient thích nghi liên kết
7	FedAdam	Federated Adaptive Moment Estimation	Tối ưu hóa ước lượng mô men thích nghi liên kết
8	FedAvg	Federated Averaging	Trung bình hóa liên kết
9	FedProx	Federated Proximal	Tối ưu hóa liên kết với thuật ngữ xấp xỉ
10	FedSGD	Federated Stochastic Gradient Descent	Tối ưu hóa đạo hàm giảm ngẫu nhiên liên kết
11	FedYogi	Federated Yogi	Tối ưu hóa liên kết Yogi
12	SGD	Stochastic Gradient Descent	Tối ưu hóa đạo hàm giảm ngẫu nhiên
13	IoT	Internet of things	Internet vạn vật

Danh mục bảng

Bảng 3-1: Đặc tính tổng quan của hai tập dữ liệu	36
Bảng 3-2: Cấu hình tổng quan mô hình ResNet18_GroupNorm	39
Bảng 3-3: Cấu hình ở server và client	40

Danh mục hình

Hình 2-1: Trực quan hóa quy trình Federated Learning ba bước	15
Hình 2-2: Học liên kết tập trung	17
Hình 2-3: Học liên kết không tập trung.....	18
Hình 3-1: Logo Flower Framework.....	35
Hình 3-2: Phân phối dữ liệu với $\alpha = 0.9$ trên hai tập dữ liệu.....	37
Hình 3-3: : Phân phối dữ liệu với $\alpha = 0.9$ trên hai tập dữ liệu.....	38
Hình 3-4: Kết quả cho thấy $\mu = 0.01$ tốt nhất.....	42
Hình 3-5: Kết quả cho thấy server_lr = 0.03 là tốt nhất	44
Hình 3-6: Kết quả cho thấy client_lr = $10^{-2.4}$ là tốt nhất	44
Hình 4-1: Kết quả huấn luyện của FedAvg trên Fashion-MNIST	47
Hình 4-2: : Kết quả đánh giá của FedAvg trên Fashion-MNIST	47
Hình 4-3: Kết quả huấn luyện của FedProx trên Fashion-MNIST	48
Hình 4-4: Kết quả đánh giá của FedProx trên Fashion-MNIST	48
Hình 4-5: Kết quả huấn luyện của FedAdam trên Fashion-MNIST.....	49
Hình 4-6: Kết quả đánh giá của FedAdam trên Fashion-MNIST	49
Hình 4-7: So sánh thời gian chạy của 3 thuật toán trên Fashion-MNIST	50
Hình 4-8: So sánh thời gian trung bình của 3 thuật toán (Fashion-MNIST)...	51
Hình 4-9: Kết quả huấn luyện của FedAvg trên CIFAR10	53
Hình 4-10: Kết quả đánh giá của FedAvg trên CIFAR10	53
Hình 4-11: Kết quả huấn luyện của FedProx trên CIFAR10	54
Hình 4-12: Kết quả đánh giá của FedProx trên CIFAR10	54
Hình 4-13: Kết quả huấn luyện của FedAdam trên CIFAR10	55
Hình 4-14: Kết quả đánh giá của FedAdam trên CIFAR10	55
Hình 4-15: So sánh thời gian chạy của 3 thuật toán trên CIFAR10	56
Hình 4-16: So sánh thời gian trung bình của 3 thuật toán trên CIFAR10	56
Hình 4-17: So sánh kết quả huấn luyện 3 thuật toán (Fashion-MNIST).....	57
Hình 4-18: So sánh kết quả đánh giá 3 thuật toán (Fashion-MNIST).....	58
Hình 4-19: Độ chính xác trung bình 20 vòng cuối (Fashion-MNIST).....	59

Hình 4-20: So sánh kết quả huấn luyện 3 thuật toán (CIFAR10).....	59
Hình 4-21: So sánh kết quả đánh giá 3 thuật toán (CIFAR10).....	60
Hình 4-22: Độ chính xác trung bình 20 vòng cuối (CIFAR10).....	61
Hình 4-23: Hiệu quả huấn luyện ở độ alpha khác nhau (Fashion-MNIST)	62
Hình 4-24: Hiệu quả đánh giá ở độ alpha khác nhau (Fashion-MNIST)	63
Hình 4-25: So sánh hiệu quả huấn luyện ở độ alpha khác nhau (CIFAR10) ..	63
Hình 4-26: So sánh hiệu quả đánh giá ở độ alpha khác nhau (CIFAR10)	64
Hình 4-27: Tốc độ hội tụ của 3 thuật toán (Fashion-MNIST).....	65
Hình 4-28: Tốc độ hội tụ của 3 thuật toán (CIFAR10)	66
Hình 4-29: Độ chính xác so với thời gian (Fashion-MNIST)	67
Hình 4-30: Độ chính xác so với thời gian trong 20 vòng cuối (CIFAR10)....	68

CHƯƠNG 1: MỞ ĐẦU

1.1. Lý do chọn đề tài

Trong bối cảnh lượng dữ liệu được gia tăng mạnh mẽ từ các thiết bị biến như: Thiết bị di động, thiết bị IoT,... việc tận dụng nguồn dữ liệu khổng lồ đó cho các mô hình máy học (ML – Machine Learning) hoặc mô hình học sâu (DL – Deep Learning) là điều cần thiết. Tuy nhiên, nhu cầu bảo vệ quyền riêng tư ngày một tăng cao không chỉ từ phía người dùng mà còn từ các quy định pháp lý, tiêu chuẩn ngành và giảm thiểu chi phí truyền tải dữ liệu là một thách thức to lớn. Phương thức huấn luyện truyền thống yêu cầu tập trung dữ liệu để huấn luyện mô hình, sẽ tiềm ẩn nguy cơ rò rỉ thông tin của người dùng và tốn kém tài nguyên. Federated Learning (FL) giải quyết trực tiếp vấn đề này bằng cách sử dụng dữ liệu tại client để huấn luyện mô hình cục bộ trên thiết bị đó và chỉ truyền các tham số cập nhật về mô hình trung tâm điều này sẽ đáp ứng yêu cầu riêng tư, bảo vệ dữ liệu người dùng và tối ưu giao tiếp hiệu quả.

Việc lựa chọn đề tài so sánh ba thuật toán FedAvg, FedProx và FedAdam nhằm hướng đến mục tiêu hiểu sâu hơn về khả năng hội tụ, độ chính xác, thời gian thực thi và tính ổn định của từng chiến lược trong thực tế trong môi trường dữ liệu không đồng nhất (non-IID).

1.2. Lịch sử nghiên cứu vấn đề

Federated Learning (FL) lần đầu được giới thiệu bởi nhóm nghiên cứu của Google giới thiệu từ năm 2016 dưới hình thức nghiên cứu sơ khởi về huấn luyện phân tán nhằm bảo vệ quyền riêng tư dữ liệu. Mục tiêu ban đầu là tạo điều kiện cho các bên cộng tác đào tạo một mô hình chung mà không cần chia sẻ dữ liệu thô, giải quyết bài toán dữ liệu nhạy cảm hoặc phân tán lớn mà không cần tập trung hóa dữ liệu về một vị trí duy nhất. [1]

Các thuật toán lõi như FedSGD và FedAvg được phát triển ngay trong giai đoạn đầu của FL. FedSGD là phương pháp cơ bản, trong đó máy chủ phân phối mô hình ban đầu cho clients, clients tính toán gradient cục bộ và gửi gradient về máy chủ để

tổng hợp. Tuy nhiên, FedSGD yêu cầu số vòng huấn luyện rất lớn để đạt độ chính xác đáng kể. Sau đó, FedAvg được giới thiệu bởi McMahan và các cộng sự năm 2017, cho phép clients thực hiện nhiều bước descent cục bộ trước khi gửi các tham số mô hình về máy chủ để trung bình hóa. Đây là cải thiện lớn về hiệu quả giao tiếp và tốc độ hội tụ, đặc biệt đối với dữ liệu non-IID và số epoch cục bộ. [2]

Trong các kịch bản dữ liệu không đồng nhất, FedAvg gặp hạn chế về hội tụ ổn định. Để giải quyết điều này, FedProx đã được trình bày và giới thiệu vào 2018 với sự bổ sung của một thuật ngữ “điều chỉnh proximal” vào hàm mục tiêu cục bộ nhằm giới hạn sự “trôi dạt” của các bản cập nhật cục bộ và cải thiện độ ổn định khi client có lượng dữ liệu không đồng đều. Thuật toán này đã chứng minh được khả năng hội tụ bền vững hơn và cải thiện độ chính xác trung bình tuyệt đối lên đến khoảng 22 % so với FedAvg trong môi trường rất không đồng nhất. [2]

Các thuật toán tối ưu hóa thích ứng như FedAdagrad, FedAdam và FedYogi do Reddi và cộng sự đề xuất từ năm 2020 đưa khả năng thích ứng vào FL thông qua việc sử dụng thống kê gradient tại cả client và server. Những thuật toán này điều chỉnh tốc độ học một cách tự động và thường đạt tốc độ hội tụ ban đầu nhanh hơn. Sự kết hợp khách hàng thực hiện nhiều epoch với optimizer cục bộ và máy chủ cập nhật mô hình tổng quát bằng optimizer thích ứng đã tạo nên một bước tiến đáng kể trong tối ưu hóa FL. [2]

Framework mã nguồn mở như Flower, được giới thiệu vào năm 2020, đóng vai trò then chốt trong việc hỗ trợ nghiên cứu và triển khai FL quy mô lớn. Flower cho phép mô phỏng và thực nghiệm với hàng triệu clients, hỗ trợ môi trường đa, tương thích với nhiều framework và ngôn ngữ khác nhau, từ đó đẩy nhanh việc nghiên cứu FL trong điều kiện thực tế phân tán và độ heterogeneity cao. [3]

1.3. Mục đích nghiên cứu

Mục đích nghiên cứu tập trung vào việc phân tích, triển khai thực nghiệm và đánh giá ba chiến lược tối ưu hóa trong Federated Learning: FedAvg, FedProx và FedAdam. Thông qua việc áp dụng lên mô hình ResNet18 trên tập dữ liệu CIFAR-10, Fashion-MNIST cùng với kịch bản thực nghiệm với dữ liệu non-IID sẽ bao gồm các

độ lệch của nhãn dữ liệu trên từng clients để đánh giá rõ sự ảnh hưởng của phân phối dữ liệu đến hiệu quả huấn luyện. Các tham số như hệ số điều chỉnh chuẩn trong FedProx và tốc độ học trong FedAdam được điều chỉnh bằng grid search, tham khảo các giá trị tối ưu từ các bài báo liên quan để đảm bảo tính khả thi và khả năng so sánh chung phương pháp.

Phân tích kết quả tập trung vào các tiêu chí như độ hội tụ (số round cần đạt độ chính xác nhất định), độ chính xác hàm, mất mát, thời gian huấn luyện trong quá trình huấn luyện và đánh giá.

1.4. Đối tượng nghiên cứu

Xuất phát từ mục đích và làm rõ những vấn đề được đề cập ở mục đích nghiên cứu thì mục tiêu của đề tài là làm rõ, chi tiết và so sánh ba thuật toán tối ưu tiêu biểu của FL bao gồm có FedAvg, FedProx và FedAdam. Đây là ba thuật toán đại diện cho ba hướng tiếp cận phổ biến trong việc tối ưu hóa mô hình phân tán trên dữ liệu không đồng nhất. Việc nghiên cứu, phân tích và so sánh ba thuật toán này là cần thiết để hiểu rõ cơ chế hoạt động, ưu nhược điểm, điều kiện áp dụng, từ đó đưa ra lựa chọn phù hợp.

FedAvg (Federated Averaging) là một thuật toán nền tảng đầu tiên được đề xuất cho FL của McMahan et al. (2017). FedAvg sử dụng cơ chế đơn giản là mỗi client thực hiện nhiều bước huấn luyện cục bộ bằng thuật toán stochastic gradient descent (SGD), sau đó gửi mô hình lên máy chủ để tổng hợp thông qua trung bình tham số. Dù đơn giản và hiệu quả trong nhiều trường hợp, FedAvg gặp nhiều khó khăn khi dữ liệu giữa các client có sự khác biệt về đáng kể (Non-IID), khiến mô hình toàn cục dễ bị di chuyển ra xa khỏi tối ưu toàn cục (client drift).

FedProx (Federated Proximal Optimization) được đề xuất bởi Li et al. (2020) như một phần mở rộng của FedAvg nhằm giải quyết các hạn chế khi dữ liệu không đồng nhất hoặc khi một số client không hoàn tất huấn luyện cục bộ do giới hạn tài nguyên hệ thống. FedProx bổ sung một thành phần chính quy (proximal term) vào hàm mục tiêu cục bộ nhằm điều tiết độ lệch giữa mô hình cục bộ và mô hình toàn cục, giúp quá trình hội tụ trở nên ổn định hơn. Các nghiên cứu gần đây đã chứng minh

rằng FedProx có thể đạt được độ hội tụ tốt hơn trong các điều kiện không đồng nhất dữ liệu hoặc hệ thống.

FedAdam, thuộc nhóm thuật toán FedOpt được giới thiệu bởi Reddi et al. (2021), là một biến thể hiện đại của FedAvg. Thay vì sử dụng trung bình đơn giản tại server, FedAdam áp dụng bộ tối ưu hóa thích nghi Adam để điều chỉnh tốc độ cập nhật mô hình toàn cục theo lịch sử gradient của các client. Việc sử dụng Adam giúp giảm nhiễu trong quá trình huấn luyện và cải thiện khả năng hội tụ, đặc biệt trong môi trường dữ liệu phi IID và gradient nhiễu cao. FedAdam cũng là một phương pháp phù hợp với FL liên thiết bị (cross-device FL), nơi client không giữ trạng thái lâu dài và chỉ tham gia không đồng đều qua các vòng giao tiếp.

Việc chọn ba thuật toán trên thể hiện cho ba xu hướng tiếp cận khác nhau là:

- FedAvg: Trung bình đơn giản – nền tảng cơ bản cho FL.
- FedProx: Chính quy hóa – kiểm soát sai lệch mô hình trong dữ liệu không đồng nhất.
- FedAdam: Tối ưu hóa thích nghi – cải thiện tốc độ hội tụ và độ ổn định mô hình.

1.5. Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài là tập trung vào việc phân tích lý thuyết và đánh giá thực nghiệm ba thuật toán FedAvg, FedProx và FedAdam trong môi trường dữ liệu tương đối đồng nhất và dữ liệu không đồng nhất phù hợp với đánh giá thực tiễn. Đề tài chủ yếu thực hiện trong bối cảnh cross-device FL, nơi dữ liệu được phân phối trên nhiều thiết bị đầu cuối với khả năng tính toán khác nhau và không phải tất cả.

Phạm vi nghiên cứu được thành hai phần là lý thuyết và thực nghiệm:

- Phân tích lý thuyết: Tìm hiểu cấu trúc thuật toán, nguyên lý hoạt động, tính chất hội tụ, các giả định cần thiết cho mỗi phương pháp. Các đặc điểm và tính chất cho từng thuật toán. Các đặc điểm về sự phù hợp với

dữ liệu Non-IID, ảnh hưởng đến việc mô hình bị trôi ra xa khỏi tối ưu toán cục, khả năng thích ứng của thuật toán trên dữ liệu.

- Thực nghiệm mô phỏng: Áp dụng ba thuật toán vào tập dữ liệu phổ biến và được dùng là CIFAR-10, Fashion-MNIST với các cấu hình chuẩn bị và đánh giá thực tiễn, cụ thể với các chỉ số được đánh giá như độ chính xác, thời gian thực hiện, số vòng thực hiện, mức độ ổn định và các thông số cụ thể dựa trên biểu đồ được thiết kế.

Giới hạn nghiên cứu của đề tài gồm có:

- Không đi sâu và các khía cạnh bảo mật.
- Học liên kết cá nhân.
- Sử dụng nhiều tập dữ liệu khác nhau để thực hiện.
- Không sử dụng mô hình lớn vì vấn đề thời gian và phần cứng trong quá trình thực nghiệm.
- Không đánh giá nhiều khía cạnh trong quá trình huấn luyện FedOpt vì các thuật toán này cần quá trình thực nghiệm lâu dài để tìm ra bộ tham số phù hợp cho dữ liệu và mô hình.

Qua việc giới hạn rõ phạm vi, đề tài hướng đến việc cung cấp một góc nhìn cụ thể, khách quan, thực tế và nêu ra được các hướng cụ thể cho FL.

1.6. Phương pháp nghiên cứu

Đề tài được thực hiện theo phương pháp kết hợp giữa phân tích lý thuyết và thực nghiệm mô phỏng có kiểm soát, trên hai tập dữ liệu chuẩn là CIFAR-10 và Fashion-MNIST, sử dụng mô hình ResNet-18, triển khai thông qua thư viện Flower – một nền tảng mã nguồn mở chuyên dụng cho học liên kết.

1.6.1. Phân tích lí thuyết

Nghiên cứu được thực hiện với sự kết hợp giữa phân tích, nghiên cứu lí thuyết và thực nghiệm để tiến hành đánh giá những kết quả dựa trên lí thuyết. Nhằm đánh giá khách quan, có cơ sở khoa học và có khả năng kiểm chứng.

Việc tìm hiểu Federated Learning là gì rất quan trọng trong quá trình nghiên cứu và thực hiện nghiên cứu thuật toán, từ đây việc tìm hiểu FL và hiểu được nhờ vào các bài báo và các bài tổng quan.

Bắt đầu nghiên cứu và tìm hiểu cụ thể những thuật toán đang được chọn để nghiên cứu là gì, tài liệu được chọn là những bài báo gốc của những thuật toán nhằm mục đích hiểu rõ và hiểu chi tiết thuật toán để có những kiến thức và đánh giá chính xác, cụ thể. Với các bài báo gốc như:

- FedAvg được giới thiệu trong công trình kinh điển của McMahan et al. (2017), là thuật toán cơ bản đầu tiên của học liên kết, sử dụng cơ chế trung bình mô hình từ các client cục bộ.
- FedProx là một mở rộng được đề xuất bởi Li et al. (2020), giải quyết các vấn đề không đồng nhất trong dữ liệu và hệ thống bằng cách bổ sung thành phần chính quy vào hàm mục tiêu cục bộ.
- FedAdam thuộc nhóm thuật toán FedOpt, được Reddi et al. (2021) phát triển như một phương pháp sử dụng bộ tối ưu hóa thích nghi để cập nhật mô hình trên server, giúp cải thiện hội tụ và ổn định mô hình trong môi trường dữ liệu Non-IID.

Sau khi nghiên cứu từ lý thuyết chi tiết từng thuật toán, tiến hành đến việc đọc đánh giá và nghiên cứu về các bài báo tổng quan về từng thuật toán và các bài nghiên cứu so sánh của ba thuật toán để có cái nhìn chi tiết, tổng quan và sâu sắc nhất về ba thuật toán được thực hiện nghiên cứu.

Phân tích lý thuyết không chỉ giúp làm rõ cách hoạt động, ưu nhược điểm và điều kiện áp dụng từng thuật toán mà còn là cơ sở xác định các tiêu chí đánh giá như độ chính xác, tốc độ hội tụ, chi phí truyền thông và độ ổn định mô hình toàn cục.

1.6.2. Thực nghiệm mô phỏng

Trên cơ sở lý thuyết đã phân tích, để tài tiến hành mô phỏng thực nghiệm các thuật toán trên thư viện mã nguồn mở Flower (A Framework for Federated Learning), được phát triển nhằm hỗ trợ triển khai các kịch bản học liên kết tùy biến và hiệu quả.

Thư viện sử dụng: Flower – nền tảng mã nguồn mở hỗ trợ triển khai các thuật toán FL linh hoạt và tích hợp tốt với PyTorch và TensorFlow.

Mô hình học sâu được sử dụng là ResNet-18, một mạng nơ-ron tích chập sâu nổi bật với cơ chế residual learning, phù hợp với bài toán phân loại ảnh và được dùng phổ biến trong các nghiên cứu về thị giác máy tính.

Tập dữ liệu:

- CIFAR-10: gồm 60.000 ảnh màu (32×32 pixels) thuộc 10 lớp. Được dùng để đánh giá khả năng học của mô hình trên dữ liệu thị giác có tính phức tạp vừa phải.
- Fashion-MNIST: mở rộng từ tập dữ liệu MNIST, bao gồm các ảnh chữ viết tay, phân chia theo người dùng – phù hợp để mô phỏng dữ liệu phân tán phi IID điển hình trong học liên kết.

Chiến lược phân phối dữ liệu: Cả hai tập dữ liệu được chia không đồng nhất (non-IID) giữa các client để phản ánh thực tế rằng mỗi thiết bị người dùng có thể sở hữu dữ liệu với đặc trưng riêng biệt.

Cấu hình mô phỏng: Mỗi thuật toán sẽ được triển khai với cùng một thiết lập về số client, tỷ lệ tham gia mỗi vòng, số epoch local, batch size, learning rate, và số vòng giao tiếp để đảm bảo tính công bằng khi so sánh.

1.6.3. Mục tiêu của phương pháp

Việc thực hiện từ lý thuyết đến thực hành đem lại sự nghiên cứu chặt chẽ và chuẩn với những kiến thức được xem xét:

- Từ lý thuyết đến thực tiễn: Giúp đánh giá mức độ phù hợp của từng thuật toán trong các tình huống cụ thể – đặc biệt là khi dữ liệu không đồng nhất, điều kiện phổ biến trong các hệ thống FL thực tế như thiết bị di động, y tế, IoT.
- Đảm bảo khách quan và khả thi: việc sử dụng Flower kết hợp với mô hình ResNet-18 cho hai tập dữ liệu CIFAR-10 và Fashion-MNIST là sự lựa chọn phù hợp để triển khai trên nền tảng như Google Colab và Kaggle, đảm bảo tính khoa học và khả thi về mặt tài nguyên cho khóa luận do sinh viên thực hiện.

1.7. Ý nghĩa khoa học và thực tiễn

1.7.1. Ý nghĩa khoa học

Trong bối cảnh dữ liệu ngày càng phân tán giữa nhiều thiết bị cá nhân và hệ thống biên, học liên kết (Federated Learning – FL) nổi lên như một giải pháp hiệu quả nhằm bảo vệ quyền riêng tư người dùng mà vẫn đảm bảo chất lượng mô hình học máy. Tuy nhiên, FL phải đối mặt với nhiều thách thức đặc thù, trong đó nổi bật nhất là vấn đề dữ liệu không đồng nhất (non-IID), hạn chế truyền thông, và sự bất ổn của hệ thống phân tán.

Ba thuật toán FedAvg, FedProx và FedAdam là lần lượt đại diện cho ba hướng tiếp cận khác nhau trong giải quyết các vấn đề trên:

- FedAvg là thuật toán cơ sở, đơn giản, dễ triển khai, nhưng kém ổn định khi dữ liệu giữa các client không đồng đều.
- FedProx bổ sung thành phần chính quy hóa để giảm sự sai lệch giữa mô hình cục bộ và toàn cục, từ đó cải thiện hội tụ trong môi trường phi IID.
- FedAdam, thuộc họ FedOpt, sử dụng cơ chế tối ưu hóa thích nghi, giúp tăng tốc độ hội tụ và ổn định mô hình, đặc biệt khi client có cập nhật biến động lớn.
- Đề tài có ý nghĩa khoa học rõ ràng ở chỗ:
 - Tổng hợp, phân tích và so sánh có hệ thống ba thuật toán có ảnh hưởng lớn trong FL hiện đại, giúp làm sáng tỏ các mối quan hệ giữa lý thuyết và hành vi thực nghiệm.
 - Xây dựng môi trường mô phỏng kiểm soát trên nền tảng Flower – công cụ hiện đại, được khuyến nghị trong cộng đồng học thuật – giúp kiểm chứng lý thuyết bằng thực tế triển khai.
 - Đánh giá trên hai tập dữ liệu tiêu chuẩn (CIFAR-10 và Fashion-MNIS) cùng mô hình ResNet-18, tạo điều kiện để rút ra kết luận có giá trị tổng quát hơn, thay vì chỉ dựa trên một kịch bản cụ thể.

Từ đó, đề tài đóng góp cho hướng nghiên cứu đánh giá hiệu năng các thuật toán học liên kết trong điều kiện dữ liệu phân tán không đồng đều, đang được cộng đồng quan tâm rộng rãi.

1.7.2. Ý nghĩa thực tiễn

Bên cạnh giá trị lý thuyết, đề tài còn mang lại nhiều ý nghĩa ứng dụng thực tiễn rõ rệt:

- Hướng dẫn lựa chọn thuật toán phù hợp cho hệ thống FL thực tế: Việc hiểu rõ ưu – nhược điểm của từng thuật toán giúp người dùng có cơ sở lựa chọn chiến lược tối ưu phù hợp với điều kiện dữ liệu và hạ tầng thực tế.
- Tiết kiệm chi phí triển khai: So sánh cụ thể về hiệu suất, tốc độ hội tụ và chi phí truyền thông giữa các thuật toán giúp giảm thiểu quá trình thử sai và tối ưu tài nguyên trong các hệ thống FL thật.
- Khả năng ứng dụng trong thị giác máy tính phân tán: Với việc sử dụng mô hình ResNet-18 trên CIFAR-10 và Fashion-MNIS, đề tài mô phỏng được các kịch bản ứng dụng học liên kết trong nhận dạng hình ảnh tại thiết bị đầu cuối (edge device), như điện thoại, camera giám sát hoặc xe tự hành – nơi dữ liệu hình ảnh không được đồng nhất giữa các thiết bị.
- Khả năng mở rộng và triển khai: Kết quả được thực nghiệm để phù hợp với thực tế nhằm mục đích đem lại cho người đọc hiểu rõ và có để xem xét đánh giá chi tiết cho ra những cơ sở và lí luận chính xác.

Từ cả hai khía cạnh học thuật và thực tiễn, đề tài góp phần làm rõ hiệu quả và giới hạn của ba thuật toán học liên kết tiêu biểu trong các điều kiện khác nhau. Bằng cách kết hợp giữa nền tảng lý thuyết chắc chắn và thực nghiệm được kiểm soát, đề tài không chỉ giúp nâng cao hiểu biết về bản chất của các thuật toán FL, mà còn tạo tiền đề ứng dụng chúng vào các hệ thống học máy phân tán hiện đại, đặc biệt là trong các bài toán thị giác máy tính với dữ liệu phi tập trung.

1.8. Bố cục của khóa luận

Bố cục của khóa luận được trình bày gồm năm chương chính, mỗi chương có chức năng và nội dung cụ thể như sau:

Chương 1: Mở đầu

- Giới thiệu lý do chọn đề tài, nêu bật tầm quan trọng của Federated Learning trong bối cảnh dữ liệu hiện đại và những thách thức đang đối mặt.

Chương 1: Mở đầu

- Tổng quan về lịch sử nghiên cứu, mục đích, đối tượng, phạm vi và phương pháp nghiên cứu.
- Trình bày bô cục chi tiết của khóa luận.

Chương 2: Cơ sở lý thuyết

- Giới thiệu cơ bản về mô hình học máy truyền thống và học liên kết (Federated Learning).
- Phân tích kiến trúc mô hình FL và các phân loại bài toán liên quan.
- Trình bày các công trình nghiên cứu liên quan đến FedAvg, FedProx và FedAdam.
- Mô tả chi tiết ba thuật toán chính được sử dụng trong nghiên cứu: FedAvg, FedProx, FedAdam.
- Nêu rõ các tiêu chí đánh giá mô hình FL: độ chính xác, tốc độ hội tụ, tính công bằng, chi phí truyền thông.

Chương 3: Xây dựng mô hình

- Mô tả môi trường thực nghiệm (ngôn ngữ lập trình, framework, phần cứng sử dụng).
- Trình bày tập dữ liệu (CIFAR-10, Fashion-MNIST) và thiết kế kịch bản dữ liệu non-IID.
- Cấu hình mô hình học sâu ResNet18 phù hợp với học liên kết.
- Trình bày chi tiết cấu hình tham số cho từng thuật toán, đặc biệt với FedProx và FedAdam.
- Thiết lập quy trình huấn luyện mô phỏng và chuẩn bị cho đánh giá thực nghiệm.

Chương 4: Thực nghiệm chương trình

- Trình bày kết quả huấn luyện của từng thuật toán trên từng tập dữ liệu.
- So sánh hiệu quả mô hình dựa trên các tiêu chí đánh giá đã đề ra.
- Phân tích ảnh hưởng của dữ liệu không đồng nhất đến quá trình hội tụ.

Chương 1: Mở đầu

- Đánh giá tổng thể về ưu – nhược điểm từng thuật toán.
- Nhận xét tổng quan về kết quả.

Chương 5: Kết luận

- Tóm tắt, tổng kết các kết quả đạt được.
- Các hạn chế trong quá trình thực hiện.
- Đề xuất những hướng phát triển trong tương lai.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Mô hình máy học truyền thống

Trong các mô hình học máy truyền thống, dữ liệu từ nhiều nguồn sẽ thường được thu thập và tập trung về một máy chủ trung tâm, nơi toàn bộ quá trình huấn luyện mô hình được thực hiện. Cách tiếp cận này giúp đơn giản hóa quy trình tối ưu hóa và cho phép sử dụng hiệu quả các tài nguyên tính toán, đặc biệt là khi áp dụng trên các tập dữ liệu đồng nhất và có chất lượng cao.

Tuy nhiên, trong bối cảnh dữ liệu ngày càng phân tán và nhạy cảm – chẳng hạn như dữ liệu người dùng trên thiết bị cá nhân, cảm biến IoT hoặc bệnh án y tế – phương pháp tập trung bộc lộ nhiều hạn chế. Việc chuyển toàn bộ dữ liệu về trung tâm không chỉ gây lo ngại về quyền riêng tư và bảo mật mà còn làm gia tăng chi phí truyền tải và nguy cơ mất cân bằng dữ liệu (data imbalance). Ngoài ra, việc tập trung dữ liệu không còn khả thi khi mô thiết bị đầu cuối tăng mạnh.

Chính vì những thách thức đó, Federated Learning đã ra đời như một giải pháp thay thế hiệu quả. Trái ngược với mô hình truyền thống, FL cho phép quá trình huấn luyện diễn ra phân tán ngay tại các thiết bị cục bộ, trong khi chỉ chia sẻ các tham số mô hình với máy chủ trung tâm. Điều này mở ra hướng tiếp cận mới phù hợp hơn với yêu cầu bảo mật, phân quyền và tối ưu tài nguyên trong môi trường dữ liệu hiện đại.

2.2. Mô hình học liên kết

2.2.1. Tóm tắt nội dung các công trình nghiên cứu liên quan

Thuật toán FedAdam được đề xuất trong công trình của Reddi et al. [1] như một cải tiến thuộc họ FedOpt, sử dụng tối ưu hóa thích nghi tại server để tăng tốc độ hội tụ và cải thiện độ chính xác mô hình trong môi trường dữ liệu không IID. Trước đó, Li et al. [2] đã phát triển FedProx như một mở rộng của FedAvg nhằm giải quyết bài toán dữ liệu không đồng nhất bằng cách thêm thành phần ràng buộc vào hàm mất mát.

Beutel et al. [3] giới thiệu Flower – một framework mã nguồn mở hỗ trợ triển khai các thuật toán FL tiêu biểu, bao gồm FedAvg, FedProx và FedAdam. Tại Việt

Nam, VinBigData [4] cung cấp hướng dẫn ứng dụng FL vào thị giác máy tính, cho thấy tính thực tiễn cao của mô hình này. Bài báo của McMahan et al. [5] là công trình nền tảng đề xuất FedAvg – thuật toán cơ bản giúp giảm chi phí truyền thông và bảo vệ dữ liệu người dùng.

Khảo sát tổng quan của Yurdem et al. [6] mô tả các thuật toán, công cụ và ứng dụng FL hiện đại, trong khi bài viết phổ biến của McMahan và Ramage [7] trình bày khái niệm FL lần đầu ra công chúng qua ứng dụng Gboard. Công trình thực nghiệm của Baumgart et al. [6] cho thấy FedAdam có hiệu quả cao hơn FedAvg trong điều kiện dữ liệu phi IID. Song song đó, Chai et al. [9] đề xuất bộ tiêu chí đánh giá toàn diện cho FL, làm cơ sở để đánh giá mô hình trong nghiên cứu này.

Về mặt lý thuyết, bài báo [10] phân tích FedAvg như một trường hợp của thuật toán Expectation-Maximization, giúp hiểu rõ hơn cơ chế hội tụ. Cuối cùng, Yuan và Li [11] chứng minh FedProx vẫn hội tụ ổn định ngay cả khi dữ liệu không mượt hoặc có sự khác biệt cao giữa các client.

2.2.2. Kiến trúc mô hình

Federated Learning (FL) – hay học liên kết – là một mô hình học máy phi tập trung, trong đó quá trình huấn luyện mô hình được thực hiện trực tiếp trên các tập dữ liệu cục bộ và độc lập, mà không yêu cầu tập trung dữ liệu về một máy chủ trung tâm. Thay vì chia sẻ dữ liệu, các tác nhân học (gọi là clients) chỉ chia sẻ thông tin mô hình với một máy chủ điều phối (gọi là server). Server sau đó thực hiện tổng hợp các cập nhật từ client để cải thiện mô hình toàn cục. Mô hình này cho phép nhiều thực thể cùng hợp tác huấn luyện mô hình học máy mà không cần chia sẻ dữ liệu gốc.

Học liên kết mang những đặc trưng bản chất như là mô hình học máy phi tập trung, trong đó dữ liệu luôn được giữ nguyên tại các thiết bị cục bộ (client), không chia sẻ ra ngoài. Mỗi client hoạt động độc lập, có thể huấn luyện mô hình bất đồng bộ, không yêu cầu phần cứng đồng nhất hoặc kết nối liên tục.

Thông tin được trao đổi giữa client và server chỉ bao gồm trọng số hoặc gradient của mô hình, không bao giờ là dữ liệu gốc, từ đó đảm bảo quyền riêng tư và giảm chi

phi truyền tải. FL cũng không giả định dữ liệu phải IID – mỗi client có thể chứa dữ liệu phi đồng nhất, không cân bằng và có phân phối riêng biệt.

FL có thể triển khai linh hoạt trong nhiều môi trường như edge, cloud, hoặc mô phỏng, miễn là có khả năng huấn luyện cục bộ và tổng hợp toàn cục.

Trong mô hình Federated Learning (FL), toàn bộ quá trình huấn luyện mô hình học máy được tổ chức theo từng vòng lặp, gọi là communication rounds (vòng truyền thông). Mỗi vòng gồm nhiều bước phối hợp giữa server trung tâm và một số client được chọn, trong đó:

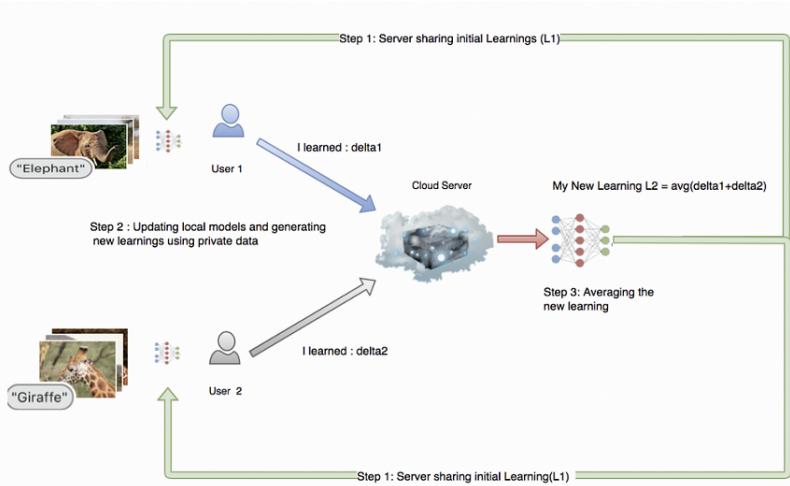
- Mô hình toàn cục (server), còn gọi là mô hình trung tâm hoặc mô hình tổng hợp, đóng vai trò khởi tạo trọng số mô hình và tổng hợp trọng số từ clients. Ban đầu, một phiên bản mô hình cơ bản được khởi tạo trên máy chủ trung tâm. Vào mỗi lần chạy từ mỗi vòng giao tiếp, máy chủ sẽ phân phối tham số weights của mô hình hiện tại ở server đến các client. Khi các client hoàn tất quá trình cập nhật và huấn luyện cục bộ, các bản cập nhật này sẽ được gửi trở lại máy chủ để tổng hợp và tạo ra mô hình toàn cầu mới thông qua các chiến lược FL khác nhau, đảm bảo phản ánh đặc tính dữ liệu của từng client một cách công bằng.
- Mô hình cục bộ (client) là mô hình đã được tạo ra từ toàn cục và được lưu trên mỗi thiết bị client để huấn luyện với dữ liệu nội bộ. Trong đó, ở mỗi lần chạy từ mỗi vòng giao tiếp, clients sẽ được nhận mô hình toàn cục từ máy chủ và thực hiện huấn luyện cục bộ với dữ liệu được sinh ra trong quá trình hoạt động của thiết bị đó. Trong quá trình huấn luyện cục bộ, mô hình sẽ thực thi một hoặc nhiều epoch giảm gradient, trong đó số epoch cục bộ (E) và kích thước minibatch (B) là các siêu tham số quan trọng ảnh hưởng đến hiệu suất và chi phí giao tiếp [5].

Được thực hiện qua sáu bước cơ bản:

1. Khởi tạo mô hình toàn cục tại server trung tâm.
2. Phân phối mô hình đến một tập con các client được chọn.

3. Huấn luyện cục bộ tại mỗi client với dữ liệu riêng.
4. Gửi cập nhật mô hình (trọng số hoặc gradient) từ client về server.
5. Tổng hợp mô hình toàn cục tại server từ các bản cập nhật.
6. Phân phối lại mô hình và lặp lại quy trình ở vòng tiếp theo.

Cập nhật mô hình và bắt đầu vòng mới (Model Redistribution): Mô hình toàn cục sau khi cập nhật lại được phân phối xuống client để bắt đầu vòng huấn luyện tiếp theo. Với mỗi vòng lặp như vậy, mô hình tiếp tục được cải thiện, ngày càng chính xác hơn và đồng thời không làm rò rỉ dữ liệu cá nhân.



Hình 2-1: Trực quan hóa quy trình Federated Learning ba bước

(Nguồn: website [VinBigData](#))

Mô tả quy trình Federated Learning ba bước trong (Hình 2-1), bao gồm: (1) server phân phối mô hình, (2) client huấn luyện cục bộ và gửi tham số, và (3) server tổng hợp để tạo mô hình mới. Hình thể hiện cơ chế phi tập trung, bảo vệ quyền riêng tư, và học từ dữ liệu không đồng nhất.

Học liên kết (Federated Learning – FL) là một phương pháp huấn luyện mô hình học máy phân tán, trong đó dữ liệu được giữ nguyên tại các thiết bị người dùng (client), và chỉ các tham số mô hình được trao đổi với máy chủ trung tâm (server). Để mô hình toàn cục có thể học hiệu quả từ dữ liệu phi tập trung, FL được xây dựng trên

một khuôn khổ tối ưu hóa tổng quát, nhằm kết hợp các cập nhật cục bộ từ các client thành một mục tiêu chung.

Mục tiêu tối ưu hóa toàn cục trong FL được phát biểu dưới dạng:

$$\min_w f(w) = \sum_{k=1}^K p_k F_k(w) = E_k[F_k(w)] \quad (2-1)$$

Trong đó:

- w : tham số mô hình toàn cục cần tối ưu.
- N : số lượng client trong hệ thống
- $F_k(w)$: hàm mất mát cục bộ của client thứ kkk, phản ánh lỗi huấn luyện trên dữ liệu riêng của thiết bị đó
- p_k : trọng số đóng góp của client kkk vào mục tiêu toàn cục, thường được xác định là $p_k = \frac{n_k}{n}$, với n_k là số lượng mẫu dữ liệu tại client k và $n = \sum_{k=1}^N n_k$ là tổng số mẫu trên toàn hệ thống.

Công thức này thể hiện rằng: FL tìm cách tối ưu trung bình có trọng số của các hàm mất mát cục bộ tại các client, sao cho mô hình toàn cục học được một cách tổng quát từ nhiều nguồn dữ liệu khác nhau mà không cần tập trung dữ liệu lại một nơi.

Hàm mất mát cục bộ $F_k(w)$ thường là giá trị kỳ vọng trên tập dữ liệu riêng của client:

$$F_k(w) = E_{x \sim D_k}[f_k(w; x)] \quad (2-2)$$

Trong đó:

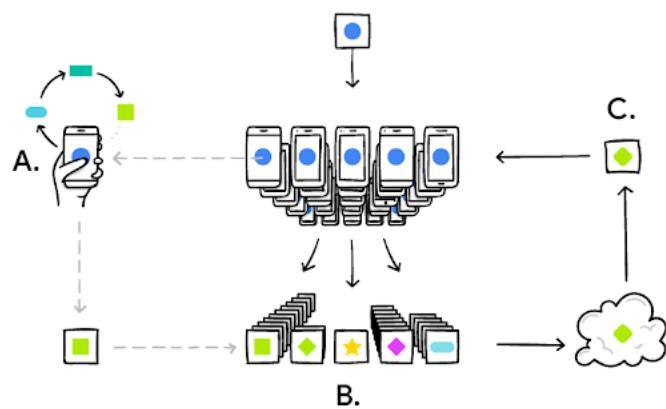
- D_k : phân phối dữ liệu tại client k
- $f_k(w; x)$: hàm mất mát cho một mẫu đơn x với mô hình tham số w .

Việc tối ưu hóa hàm $f(w)$ theo cách phân tán đòi hỏi sự phối hợp giữa các client và máy chủ thông qua một giao thức học gồm nhiều vòng lặp (federated rounds). Tại mỗi vòng, server gửi mô hình hiện tại cho một tập con client; mỗi client tối ưu hóa mô hình dựa trên dữ liệu của mình; rồi gửi tham số cập nhật về server để tổng hợp (thường là trung bình hóa có trọng số). Quá trình này lặp lại đến khi hội tụ.

2.3. Phân loại Federated Learning

2.3.1. Bài toán học liên kết tập trung

Học liên kết tập trung (centralized federated learning) yêu cầu kiến trúc trong đó một máy chủ trung tâm đóng vai trò điều phối quan trọng đối với quá trình đào tạo mô hình. Máy chủ này đảm nhiệm việc phân phối mô hình toàn cục ban đầu đến các client, tiến trình huấn luyện diễn ra tại các client với dữ liệu cục bộ, rồi các cập nhật (weights hoặc gradients) được gửi về để tổng hợp và cập nhật mô hình trung tâm. Các cập nhật tham số ở mô hình trung tâm sẽ tiếp tục gửi các trọng số đến client để cải thiện mô hình cục bộ và vòng lặp này tiếp tục diễn ra.



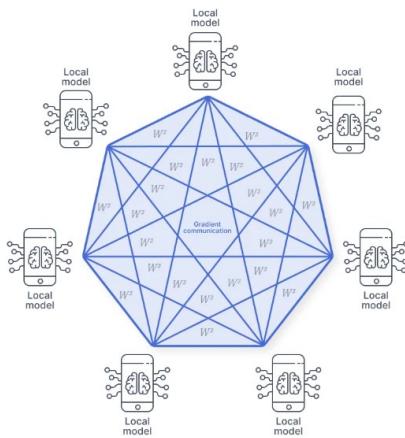
Hình 2-2: Học liên kết tập trung

(Nguồn: website [VinBigData](#))

Hệ thống này có ưu điểm là đơn giản và dễ triển khai, nhưng có thể có nguy cơ trở thành điểm nghẽn (nút cốt chai) nếu số lượng client tham gia lớn, đồng thời tồn tại nguy cơ bị tấn công hoặc lỗi. [4]

2.3.2. Bài toán học liên kết phi tập trung

Học liên kết phi tập trung (decentralized federated learning), không tồn tại máy chủ trung tâm điều phối, các client giao tiếp trực tiếp với nhau để phối hợp cập nhật mô hình toàn cục.



Hình 2-3: Học liên kết không tập trung

(*Nguồn: website VinBigData*)

Cách tiếp cận này tránh được rủi ro lỗi nút cỗ chai và cải thiện tính bền vững trong môi trường phân tán, tuy nhiên phụ thuộc vào mạng kết nối giữa các node và cách tổ chức trao đổi mô hình. Một số nghiên cứu đã ứng dụng blockchain hoặc kỹ thuật peer-to-peer để triển khai dạng học liên kết này, hỗ trợ tăng cường tính đồng thuận và an toàn phân tán.

2.3.3. Bài toán học liên kết không đồng nhất

Học liên kết không đồng nhất (heterogeneous federated learning) hướng đến xử lý trường hợp dữ liệu client khác biệt về phân phối non-IID, lượng dữ liệu và khả năng tính toán giữa các client không đồng đều. Trong thực tế, dữ liệu trên thiết bị người dùng hoặc cảm biến thường bị phân phối không đồng nhất, dẫn tới phân kỳ nếu sử dụng FedAvg mặc định.

Các thuật toán như FedProx, FedAdam, SCAFFOLD và FedDyn được đề xuất nhằm cải thiện độ ổn định hội tụ và độ chính xác bằng cách điều chỉnh hàm mục tiêu cục bộ hoặc sử dụng các biến điều khiển để giảm sai lệch hướng tới mô hình toàn cục. Những nghiên cứu này cho thấy cần áp dụng các chiến lược tối ưu hóa thích ứng khi đối mặt với dữ liệu phân tán bất đều và điều kiện tính toán khác nhau giữa client.

2.4. Các thuật toán học liên kết

2.4.1. Thuật toán FedAvg

Thuật toán Federated Averaging (FedAvg) lần đầu được giới thiệu trong công trình nổi bật của McMahan et al. (2016) [5], với mục tiêu cải thiện đáng kể hiệu quả giao tiếp trong FL so với FedSGD. Trong khi, FedSGD có thể được áp dụng một cách ngay thơ cho bài toán tối ưu hóa FL, một phép tính gradient local được thực hiện cho mỗi vòng giao tiếp. Cách tiếp cận này hiệu quả về mặt tính toán, nhưng yêu cầu một số lượng lớn các vòng giao tiếp để mô hình hội tụ đủ tốt, tạo nên một vấn đề là làm tăng đáng kể chi phí giao tiếp [1]. FedAvg đã ra đời để giải quyết trực tiếp vấn đề chính của FedSGD.

Federated Averaging (FedAvg) là một thuật toán học liên kết (FL) được thiết kế để huấn luyện các mô hình học máy và học sâu một cách hiệu quả về mặt giao tiếp trên dữ liệu phi tập trung. Đây cũng được xem là chiến lược FL phổ biến nhất vì không cần điều chỉnh nhiều tham số và triển khai đơn giản những hiệu quả trên dữ liệu phi tập trung và có độ non-IDD nhẹ.

Phương pháp thực hiện của FedAvg là mỗi client thực hiện các bước huấn luyện cục bộ trên tập dữ liệu của riêng mình, sau đó gửi các tham số cập nhật mô hình trở lại máy chủ trung tâm để tổng hợp.[5]

Mỗi client k nhận bản mô hình toàn cầu w_t từ server, huấn luyện cục bộ với dữ liệu nội bộ thông qua local SGD trong nhiều E epoch, bằng công thức:

$$w_{\{t+1\}}^k \leftarrow w_t - \eta \cdot \nabla F_k(w^k) \quad (2-3)$$

Trong đó:

- w_t : Trọng số của mô hình tại vòng thứ t
- $w_{\{t+1\}}^k$: Trọng số đã cập nhật của mô hình trên client
- η : Tốc độ học (learning rate)
- $\nabla F_k(w^k)$: Đạo hàm của hàm mất mát trên tập dữ liệu cục bộ của client k

Sau khi huấn luyện cục bộ, client được server yêu cầu gửi tham số về mô hình trung tâm để cập nhật server bằng công thức:

$$w_{\{t+1\}} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{\{t+1\}}^k \quad (2-4)$$

Trong đó:

- $w_{\{t+1\}}$: Trọng số đã cập nhật của mô hình server
- S_t : Tập hợp các client được chọn tham gia huấn luyện ở vòng t
- n_k : Số lượng mẫu trên dữ liệu của k
- m_t : $m_t \leftarrow \sum_{k \in S_t} n_k$ là tổng số lượng mẫu dữ liệu từ các client được chọn huấn luyện tại vòng t

Quy trình hoạt động của FedAvg được thực hiện theo từng vòng như sau:

1. Máy chủ khởi tạo mô hình toàn cầu w_0
2. Tại mỗi vòng t:
 - Chọn một tập con client S_t
 - Gửi bản sao mô hình toàn cầu w_t đến các client S_t
 - Mỗi client sử dụng local SGD để cập nhật mô hình trong E epoch $w_{\{t+1\}}^k \leftarrow w_t - \eta \cdot \nabla F_k(w^k)$
 - Mỗi client về $w_{\{t+1\}}^k$ server
 - Server thực hiện: $w_{\{t+1\}} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{\{t+1\}}^k$
3. Lặp lại cho đến khi dừng theo điều kiện hội tụ hoặc số round đã đặt trước

Mã giả cho thuật toán FedAvg:

Thuật toán 1: Federated Averaging. Gồm K client, được đánh chỉ mục bằng k. Các tham số bao gồm: C là tỷ lệ client tham gia trong mỗi vòng; B

là kích thước của các lô dữ liệu nhỏ (local minibatch size); E là số chu kỳ huấn luyện cục bộ (local epochs); và η là tốc độ học (learning rate).

Phía Server (Server executes):

```
1: khởi_tạo trọng số w0
2: for mỗi vòng t = 1, 2, ... làm:
3:     m ← max(C · K, 1)
4:     St ← (tập hợp ngẫu nhiên của m client)
5:     for mỗi client k ∈ St song song làm:
6:         wkt+1 ← ClientUpdate(k, wt)
7:     end for
7:     wtt+1 ← ∑_{k∈St} (nk/n) wkt+1
8: end for
```

ClientUpdate (k, w):

```
1: B ← (chia tập dữ liệu Pk thành các lô có kích thước B)
2: for mỗi epoch cục bộ i từ 1 đến E làm:
3:     for mỗi batch b ∈ B làm:
4:         w ← w - η ∇ℓ(w; b)
5:     end for
6: end for
7: return w về cho server
```

FedAvg có ưu điểm nằm ở khả năng bảo vệ quyền riêng tư và bảo mật dữ liệu do dữ liệu thô của người dùng không bao giờ rời khỏi thiết bị cục bộ, thay vào đó chỉ các bản cập nhật mô hình được chia sẻ với máy chủ.

Hiệu quả về giao tiếp là một điểm mạnh chính, khi thuật toán cho phép các client thực hiện nhiều bước tính toán cục bộ trước khi gửi bản cập nhật về server, do đó giảm đáng kể số lượng vòng giao tiếp cần thiết để đạt hội tụ. Những cải tiến này dẫn đến số vòng giao tiếp ít hơn hàng chục đến hàng trăm lần so với phương pháp truyền thống như FedSGD

Thêm vào đó, quá trình huấn luyện cục bộ trên client trong nhiều epoch và trung bình hóa mô hình toàn cục cũng mang lại sự đa dạng về dữ liệu từ nhiều nguồn client và có khả năng giúp xử lý dữ liệu không đồng nhất (non-IID) nhẹ. Ngoài ra, tăng khả

năng tổng quát hóa của mô hình toàn cầu một lợi ích điều chuẩn ngầm định, tương tự như hiệu ứng của dropout trong học sâu, giúp giảm overfitting và nâng cao khả năng khái quát hóa.

Tuy nhiên, FedAvg vẫn tồn tại một số hạn chế rõ rệt. Khi dữ liệu bị không đồng nhất nghiêm trọng, hiệu suất và tốc độ hội tụ đều có thể suy giảm đáng kể do sự lệch hướng giữa các mô hình cục bộ và mô hình toàn cục.

Hơn nữa, sự phụ thuộc vào một máy chủ trung tâm làm xuất hiện rủi ro trở thành điểm nghẽn hoặc lỗi duy nhất trong hệ thống, đặc biệt khi số lượng client lớn hoặc có sự cố mạng xảy ra.

Ngoài ra, FedAvg thường đòi hỏi các client cập nhật trong một khoảng thời gian nhất định trong trường hợp một số thiết bị không thể hoàn thành trong khung thời gian xác định, chúng sẽ bị loại ra tạo ra hiện tượng tượng “client drift”, dẫn tới mất mát thông tin huấn luyện hoặc tạo ra sự thiên vị trong quá trình chọn mẫu client.

2.4.2. Thuật toán FedProx

Một trong những thuật toán nền tảng của FL là FedAvg, dựa trên việc trung bình hóa các cập nhật từ local SGD. Tuy hoạt động tốt trong điều kiện lý tưởng, FedAvg lại gặp khó khăn khi triển khai thực tế: chỉ một phần nhỏ client tham gia mỗi vòng, dữ liệu thường không đồng nhất giữa client, và thiết bị có tài nguyên tính toán không đồng đều. Điều này khiến mô hình hội tụ kém ổn định và khó mở rộng. [10]

Để giải quyết các vấn đề trên, nhóm tác giả Tian Li et al. (2020) đã đề xuất FedProx như một phần mở rộng của FedAvg, nhằm tăng độ ổn định trong môi trường có đặc điểm dữ liệu và hệ thống. FedProx cải tiến quá trình huấn luyện cục bộ bằng cách ràng buộc mô hình client không được lệch xa mô hình toàn cục, đồng thời cho phép client thực hiện huấn luyện không đầy đủ (partial local work). Những cải tiến này giúp hệ thống hoạt động linh hoạt hơn trong điều kiện thực tế mà vẫn đảm bảo hiệu quả hội tụ.

FedProx (Federated Proximal) là một thuật toán tối ưu hóa phân tán trong học liên kết, được phát triển như một phần mở rộng tổng quát của FedAvg nhằm giải

quyết hiệu quả các vấn đề phổ biến trong môi trường thực tế – bao gồm sự không đồng nhất dữ liệu (non-IID) và đặc biệt hệ thống (heterogeneous devices) giữa các client. Trong khi FedAvg giả định dữ liệu phân phối đồng đều và client có khả năng xử lý tương đương, thì FedProx thừa nhận rằng các thiết bị trong hệ thống thường có dữ liệu riêng biệt và tài nguyên không đồng đều, gây ra hiện tượng sai lệch mô hình (client drift) và khó hội tụ ổn định. [10]

Với định hướng thực tiễn và được chứng minh về mặt lý thuyết, FedProx không chỉ cải thiện FedAvg mà còn đóng vai trò quan trọng trong sự phát triển của các thuật toán học liên kết hiện đại.

Trong FedProx, mục tiêu là tối ưu mô hình toàn cục w thông qua các bước huấn luyện cục bộ tại mỗi client, tương tự như FedAvg. Tuy nhiên, điểm khác biệt quan trọng là FedProx thêm một thành phần chính quy (proximal term) vào hàm mất mát cục bộ, nhằm điều chỉnh và kiểm soát mức độ lệch giữa mô hình local và mô hình global. Cụ thể, tại mỗi vòng huấn luyện, client k sẽ giải bài toán tối ưu:

$$\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2} \|w - w^{(t)}\|^2 \quad (2-5)$$

Trong đó:

- $F_k(w)$: hàm mất mát cục bộ của client k , được tính trên dữ liệu riêng của nó.
- w : tham số mô hình cục bộ được cập nhật tại client.
- $w^{(t)}$: tham số mô hình toàn cục do server gửi xuống tại vòng thứ t .
- μ : hệ số điều chỉnh mức ảnh hưởng của thành phần ràng buộc (proximal term).
- $\|w - w^{(t)}\|^2$: khoảng cách Euclid bình phương giữa mô hình local và global, đại diện cho độ lệch mô hình.

Thành phần $\|w - w^{(t)}\|^2$ là yếu tố then chốt giúp hạn chế sự "trôi mô hình" (client drift), đặc biệt trong điều kiện dữ liệu không đồng nhất giữa các client.

Khi giá trị mu (μ) bằng 0, công thức trên trở thành hàm mất mát của FedAvg – cho thấy rằng FedProx là một tổng quát hóa của FedAvg.

Mỗi client thực hiện huấn luyện mô hình của mình dựa trên công thức trên (có thể với số batch không đầy đủ nếu client yếu), sau đó gửi tham số w đã cập nhật về server để thực hiện bước tổng hợp trung bình (aggregation). Server sau đó cập nhật mô hình toàn cục $w^{(t+1)}$ như sau:

$$w^{(t+1)} = \sum_{k=1}^K \frac{n_k}{n} w_k^{(t)} \quad (2-6)$$

Trong đó:

- K : số lượng client được chọn trong vòng hiện tại
- n_k : số lượng mẫu dữ liệu tại client k
- $w_k^{(t)}$: mô hình được huấn luyện bởi client k ở vòng thứ t .

Quy trình hoạt động của FedProx được thực hiện theo từng vòng như sau:

1. Máy chủ khởi tạo mô hình toàn cầu w_0
2. Tại mỗi vòng lặp t :
 - Chọn client: Máy chủ chọn ngẫu nhiên một tập con client S_t từ tổng số N client (Theo phân phối xác suất p_k)
 - Gửi mô hình: Máy chủ gửi w^t đến tất cả các thiết bị được chọn
 - Cập nhật cục bộ: Mỗi client $k \in S_t$ cập nhật mô hình bằng cách giải bài toán tối ưu hóa cục bộ có ràng buộc proximal:

$$w_k^{(t+1)} \approx \arg \min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2} \|w - w^t\|^2$$
 (Client không cần giải chính xác bài toán, chỉ cần tìm nghiệm gần đúng)
 - Gửi kết quả: Mỗi client gửi mô hình cục bộ $w_k^{(t+1)}$ trở về máy chủ.
 - Tổng hợp mô hình: Máy chủ cập nhật mô hình toàn cục bằng cách trung bình các mô hình nhận được: $w^{(t+1)} = \frac{1}{|S_t|} \sum_{k \in S_t} w_k^{(t+1)}$
3. Dừng thuật toán: Lặp lại quy trình trên cho đến khi mô hình hội tụ hoặc đạt đến số vòng tối đa đã định trước.

Mã giả cho thuật toán FedProx:

Thuật toán Federated Proximal nhận vào các tham số đầu vào bao gồm:
 K là tổng số client trong hệ thống; C là tỷ lệ client được chọn trong mỗi vòng huấn luyện; B là kích thước của minibatch trong huấn luyện cục bộ;

E là số epoch cục bộ mỗi client thực hiện; η là tốc độ học; μ là hệ số ràng buộc proximal giúp ổn định hội tụ trong môi trường không đồng nhất; và T là số vòng lặp toàn cục. Ngoài ra, mô hình toàn cục ban đầu w_0 cũng là một đầu vào quan trọng được máy chủ khởi tạo trước khi bắt đầu huấn luyện.

Phía Server (Server executes):

```

1: khởi_tạo trọng số  $w_0$ 
2: for mỗi vòng  $t = 1, 2, \dots, T$  làm:
3:    $m \leftarrow \max(C \cdot K, 1)$ 
4:    $S_t \leftarrow$  (tập hợp ngẫu nhiên của  $m$  client)
5:   for mỗi client  $k \in S_t$  thực hiện song song:
6:      $w_k^{t+1} \leftarrow \text{ClientUpdate}(k, w_t)$ 
7:   end for
8:    $w_t^{t+1} \leftarrow \frac{1}{|S_t|} \sum_{k \in S_t}^K w_k^{(t+1)}$ 
9: end for

```

Cập nhật Client(k, w):

```

1:  $B \leftarrow$  (chia tập dữ liệu  $P_k$  thành các lô có kích thước  $B$ )
2:  $w_{\text{local}} \leftarrow w$ 
3: for mỗi epoch cục bộ  $i$  từ 1 đến  $E$  làm:
4:   for mỗi batch  $b \in B$  làm:
5:      $g \leftarrow \nabla \ell(w_{\text{local}} ; b)$ 
6:      $w_{\text{local}} \leftarrow w_{\text{local}} - \eta [g + \mu (w_{\text{local}} - w)]$ 
7: return  $w_{\text{local}}$  về cho server

```

FedProx là một thuật toán hiệu quả giúp giải quyết vấn đề không đồng nhất dữ liệu và hệ thống. Bằng cách thêm một thuật ngữ xấp xỉ (proximal term) vào hàm mất mát cục bộ, thuật toán này giới hạn mức độ sai lệch của mô hình cục bộ so với mô hình toàn cục. Điều này giúp duy trì sự ổn định và cải thiện độ chính xác của mô hình, đặc biệt trong môi trường dữ liệu không đồng nhất. Ngoài ra, FedProx hỗ trợ các thiết bị có tài nguyên yếu và không đồng đều bằng cách cho phép chúng chỉ huấn luyện một phần công việc thay vì phải hoàn thành toàn bộ như các thiết bị mạnh. Điều này giúp tránh việc loại bỏ các client và tận dụng tối đa dữ liệu phân tán. Nhờ có sự kiểm soát chặt chẽ giữa mô hình cục bộ và toàn cục, FedProx cũng giúp mô hình hội tụ ổn định hơn trong quá trình huấn luyện.

Mặc dù có nhiều ưu điểm, FedProx vẫn tồn tại một số hạn chế. Thứ nhất, việc thêm điều kiện ràng buộc xấp xỉ làm tăng khối lượng tính toán trên mỗi thiết bị, điều này có thể gây khó khăn cho các thiết bị có tài nguyên hạn chế. Thứ hai, việc lựa chọn tham số μ (Mu) phù hợp là rất quan trọng nhưng cũng rất phức tạp, đòi hỏi quá trình thử nghiệm và điều chỉnh tốn nhiều thời gian để đạt được hiệu quả tối ưu. Cuối cùng, dù cho phép các thiết bị yếu tham gia vào quá trình huấn luyện, thuật toán vẫn chưa có tiêu chuẩn cụ thể và chính xác để chọn các thiết bị này, đây là một vấn đề cần được nghiên cứu thêm để tối ưu hóa hiệu quả của FedProx.

2.4.3. Thuật toán FedAdam

Phương pháp tối ưu truyền thống như SGD thường được áp dụng cho FedAvg, trong đó các client tham gia thực hiện nhiều bước cập nhật bằng gradient (SGD) cục bộ và cập nhật mô hình trung tâm bằng cách trung bình hóa. Khi dữ liệu giữa clients là không đồng nhất, việc tính trung bình có trọng số theo trọng số của client cục bộ sẽ gây ra khả năng học thiên vị của mô hình. Ngoài ra, sẽ khó điều chỉnh các siêu tham số cho từng bài toán, dữ liệu có tính không đồng nhất cao khác. FedAdam được sinh ra để giải quyết hạn chế chính này bằng cách sử dụng các thống kê moments của gradient bằng phương pháp tối ưu Adam ở phía máy chủ.

FedAdam, viết tắt của Federated Adam, là phiên bản mở rộng của phương pháp tối ưu Adam dành riêng cho môi trường Federated Learning. Thuật toán này được giới thiệu trong bài báo Adaptive Federated Optimization [1] của Reddi et al. (2021) nhằm giải quyết những hạn chế trong hội tụ khi đối mặt với dữ liệu không đồng nhất và môi trường phân tán. Adam hiện tại vốn là một bộ tối ưu hóa tiêu chuẩn cho DL, nổi bật với khả năng cung cấp hiệu suất ổn định mà ít cần tinh chỉnh.

FedAdam là một thuật toán được cải tiến từ FedAvg, trong đó Adam đóng vai trò là bộ tối ưu hóa phía máy chủ, trong khi phía máy khách thường sử dụng SGD. Nó hoạt động bằng cách xem xét trung bình có trọng số theo kích thước của các cập nhật cục bộ của máy khách như một giả gradient (pseudo-gradient) cho bộ tối ưu hóa phía máy chủ [1]. Sự khác biệt chính giữa FedAdam và thuật toán Adam tiêu chuẩn

là việc sử dụng phương pháp trung bình liên kết để tính toán gradient trên tất cả các thiết bị. [6]

FedAdam có quá trình cập nhật ở client tương tự như FedAvg. Mỗi client được chọn sẽ được gửi trọng số w được cập nhật từ mô hình toàn cục w_t nhận từ server và thực hiện huấn luyện với bước tối ưu hóa cục bộ là SGD trên dữ liệu của mình với số E epoch. Sau mỗi vòng huấn luyện cục bộ, client có trọng số tiếp theo $w_{\{t+1\}}^k$ nhưng chưa gửi trọng số đó về mô hình trung tâm mà còn thực hiện thêm bước chuyển thành vector cập nhật delta của client đó là $\Delta_{\{t\}}^k$ thể hiện hướng và độ lớn của sự thay đổi:

$$\Delta_{k,t} = w_{\{t+1\}}^k - w_t \quad (2-7)$$

Trong đó:

- $\Delta_{k,t}$: vector delta của client

Điểm khác biệt cốt lõi của FedAdam so với FedAvg nằm ở phía server. Server không chỉ tổng hợp các mô hình mà còn duy trì các "moment" (tương tự như thuật toán Adam) để điều chỉnh tốc độ học cho từng tham số của mô hình một cách linh hoạt, bằng các công thức sau:

$$\Delta_t = \sum_{k \in S_t} \frac{n_k}{n} \Delta_{k,t} \quad (2-8)$$

Trong đó:

- n : Số lượng mẫu dữ liệu trên tất cả client trong tập S_t
- Δ_t : vector delta của server

Sau đó, server sử dụng Δ_t này để cập nhật các vector moment bậc nhất (m_t) và moment bậc hai (v_t) theo đúng công thức của Adam:

(1) Cập nhật moment bậc nhất (ước lượng trung bình của gradient):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \Delta_t \quad (2-9)$$

(2) Cập nhật moment bậc hai (ước lượng phương sai của gradient)

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \Delta_t^2 \quad (2-10)$$

Trong đó:

- β_1 : Hệ số phân rã (decay rate) thứ nhất kiểm soát mức độ tác động của các gradient trong quá khứ đối với trung bình động hiện tại
- β_2 : Hệ số phân rã (decay rate) thứ hai kiểm soát mức độ tác động của các gradient trong quá khứ đối với trung bình động hiện tại
- m_{t-1} : Là moment thứ nhất tại $t-1$
- v_{t-1} : Là moment thứ hai tại $t-1$
- Δ_t^2 : Bình phương của delta tại bước t

Cuối cùng là cập nhật trọng số mô hình toàn cục sau khi đã có các moment đã được hiệu chỉnh:

$$w_{\{t+1\}} \leftarrow w_{\{t\}} + \eta \cdot \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon} \quad (2-11)$$

Trong đó:

- \widehat{m}_t : $\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}$ là moment thứ nhất đã được hiệu chỉnh độ lệch tại t
- \widehat{v}_t : $\widehat{v}_t = \frac{v_t}{1 - \beta_2^t}$ là moment thứ hai đã được hiệu chỉnh độ lệch tại t
- ϵ : Một hằng số nhỏ để tránh chia cho 0

Quy trình hoạt động của FedAdam được thực hiện theo từng vòng như sau:

1. Máy chủ khởi tạo mô hình toàn cầu
 - Trọng số mô hình toàn cục w_0
 - Các moment : moment cấp 1 $m_0 = 0$, moment cấp 2 $v_0 = 0$
2. Tại mỗi vòng lặp t :
 - Chọn client: Máy chủ chọn ngẫu nhiên một tập con S_t gồm m client từ tổng số N client (theo phân phối xác suất p_k)
 - Gửi mô hình: Máy chủ gửi trọng số mô hình w_t đến tất cả các client được chọn

- Huấn luyện cục bộ: Mỗi client thực hiện huấn luyện mô hình trên dữ liệu cục bộ với E epoch, sử dụng thuật toán tối ưu SGD: $w \leftarrow w - \eta \cdot \nabla \ell(w; b)$ VỚI $\ell(w; b)$ là hàm mất mát trên minibatch $b \in B$.
 - Cập nhật moment và trọng số tại server: Máy chủ tổng hợp các moment bậc 1 và bậc 2: $m_t = \beta_1 m_{t-1} + (1 - \beta_1) \Delta_t$, $v_t = \beta_2 v_{t-1} + (1 - \beta_2) \Delta_t^2$
 - Cập nhật mô hình toàn cục: $w_{\{t+1\}} \leftarrow w_{\{t\}} + \eta \cdot \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \epsilon}$
3. Lặp lại cho đến khi mô hình hội tụ hoặc đạt đến số vòng lặp tối đa đã định.

Mã giả cho thuật toán FedAdam:

Thuật toán 2: Federated Adam (FedAdam). Các tham số bao gồm: K là tổng số client, C là tỷ lệ client tham gia, E là số chu kỳ cục bộ, τ là tốc độ học phía client, η là tốc độ học phía server, $\beta_1, \beta_2, \epsilon$ là các tham số của Adam.

Phía Server (Server executes):

- 1: khởi tạo trọng số w_0
- 2: khởi tạo các moment của server $m_0 \leftarrow 0, v_0 \leftarrow 0$
- 3: for mỗi vòng $t = 1, 2, \dots$ làm:
 - 4: $S_t \leftarrow$ (tập hợp ngẫu nhiên của m client)
 - 5: khởi tạo tập Δ_list rỗng
 - 5: for mỗi client $k \in S_t$ song song làm:
 - 6: $w_k^t \leftarrow ClientUpdate(k, w_{\{t-1\}}, \tau, E)$
 - 7: $\Delta_{k,t} \leftarrow w_{k,t} - w_{\{t-1\}}$
 - 8: thêm $\Delta_{k,t}$ vào Δ_list
 - 9: end for
 - 10: $\Delta_t \leftarrow$ Tổng hợp có trọng số các Δ trong Δ_list
 - 11: $m_t \leftarrow \beta_1 m_{\{t-1\}} + (1 - \beta_1) \Delta_t$
 - 12: $v_t \leftarrow \beta_2 v_{\{t-1\}} + (1 - \beta_2) \Delta_t^2$
 - 13: $hat(m)_t \leftarrow m_t / (1 - \beta_1^t)$
 - 14: $hat(v)_t \leftarrow v_t / (1 - \beta_2^t)$
 - 15: $w_t \leftarrow w_{\{t-1\}} + \eta \cdot hat(m)_t / (sqrt(hat(v)_t) + \epsilon)$
- 16: end for

Hàm ClientUpdate(k, w, τ, E):

- 1: $B \leftarrow$ (chia tập dữ liệu P_k thành các lô có kích thước B)
 - 2: for mỗi epoch cục bộ i từ 1 đến E làm:
-

```

3:      for mỗi batch  $b \in B$  làm:
4:           $w \leftarrow w - \eta \nabla \ell(w; b)$ 
6:      end for
7: end for
8: return  $w$  về cho server

```

FedAdam là một thuật toán hiệu quả trong học liên kết nhờ vào cơ chế cập nhật thích nghi. Nhờ kiểm soát được biến thiên trong cập nhật trọng số, thuật toán tạo ra các tham số ổn định hơn sau mỗi vòng lặp, từ đó cải thiện khả năng tổng quát của mô hình toàn cục trên các tập dữ liệu mới. Ngoài ra, FedAdam còn thích nghi tốt với các hệ thống không đồng nhất, không yêu cầu mỗi client cập nhật với cùng một số epoch hoặc lượng dữ liệu giống nhau. Thuật toán xử lý sự không đồng đều này bằng cách điều chỉnh mức độ ảnh hưởng của từng cập nhật thông qua các moment động, tăng tính thực tiễn khi triển khai trên các thiết bị có tài nguyên khác nhau.

Mặc dù có nhiều ưu điểm, tuy nhiên FedAdam vẫn tồn tại một số hạn chế. Thứ nhất, hiệu quả của thuật toán phụ thuộc rất lớn vào các tham số điều chỉnh. Việc tìm và lựa chọn bộ tham số phù hợp là một vấn đề cốt lõi, ảnh hưởng đến mọi khía cạnh từ tốc độ hội tụ đến khả năng học của mô hình. Việc chọn sai bộ tham số có thể dẫn đến kết quả không tốt và làm giảm chất lượng mô hình. Thứ hai, FedAdam có chi phí tính toán cao hơn so với FedAvg. Trong khi FedAvg chỉ đơn giản là trung bình hóa mô hình, FedAdam yêu cầu lưu trữ và cập nhật thêm hai moment bậc một và hai, làm tăng gánh nặng tính toán và bộ nhớ ở máy chủ trung tâm, đặc biệt trong các hệ thống có quy mô lớn. Cuối cùng, thuật toán thiếu cơ chế xử lý gradient nhiễu từ các client bị lỗi. Khác với FedProx đưa ra ràng buộc xấp xỉ để giảm ảnh hưởng của các client này, FedAdam vẫn tính trung bình các cập nhật mà không có bước chuẩn hóa hoặc lọc gradient. Điều này có thể làm suy giảm độ chính xác nếu client gửi kết quả không đúng hoặc không ổn định.

2.5. Các tiêu chí đánh giá mô hình trong Federated Learning

Trong quá trình nghiên việc đánh giá mô hình không chỉ dừng lại ở độ chính xác mà còn mở rộng sang các yếu tố đánh giá cho học liên kết như chi phí truyền

thông, tốc độ hội tụ, sự công bằng giữa các thiết bị và khả năng mở rộng. Để phản ánh đúng bản chất, những tiêu chí bên dưới được chọn để đánh giá trong quá trình nghiên cứu và thực nghiệm.

2.5.1. Các tiêu chí đánh giá hiệu suất mô hình:

Độ chính xác tổng thể (Accuracy) là thang đo được sử dụng để đánh giá khả năng học của mô hình, đây là thước đo đơn giản và phổ biến nhất, thể hiện tỷ lệ mẫu được phân loại đúng so với tổng số mẫu. Accuracy được tính theo công thức:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2-12)$$

Trong đó:

- TP (True Positive): Số lượng mẫu dương được dự đoán đúng.
- TN (True Negative): Số lượng mẫu âm được dự đoán đúng.
- FP (False Positive): Số lượng mẫu âm nhưng bị dự đoán nhầm là dương.
- FN (False Negative): Số lượng mẫu dương nhưng bị dự đoán nhầm là âm.

Ý nghĩa: Độ chính xác cao thể hiện mô hình có khả năng phân biệt tốt giữa hai lớp trong bài toán phân loại nhị phân. Tuy nhiên accuracy không phải lúc nào cũng là thước đo tốt, nhất là khi dữ liệu không cân bằng

Trong bài toán độ chính được đánh giá trên tập huấn luyện và tập kiểm thử.

Độ chính xác huấn luyện (Train Accuracy) : Tỷ lệ dự đoán đúng trên tập dữ liệu huấn luyện tại các client. Phản ánh khả năng ghi nhớ dữ liệu, nhưng không đủ để đánh giá khả năng tổng quát.

Độ chính xác trên tập kiểm tra (Validation Accuracy): Chỉ số quan trọng nhất phản ánh hiệu suất mô hình sau mỗi vòng FL. Tiêu chí đánh giá chủ yếu để so sánh giữa các thuật toán như FedAvg, FedProx và FedAdam.

2.5.2. Hàm mất mát (Loss Function)

Hàm mất mát là giá trị quan trọng trong việc đánh giá mô hình và khả năng học, độ ổn định trong quá trình huấn luyện. Cross-Entropy Loss là hàm mất mát được chọn để sử dụng vì sự phổ biến, phù hợp cho bài toán đa lớp. Đo lường sự khác biệt giữa phân phối xác suất dự đoán của mô hình và phân phối thực tế của nhãn. Công thức của hàm Cross-Entropy Loss được định nghĩa như sau:

Công thức:

$$L = \frac{-1}{N} \sum_{i=0}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c}) \quad (2-13)$$

Trong đó:

- N: Số lượng mẫu trong batch.
- C: Số lượng lớp
- $y_{i,c}$: Nhãn thực tế của mẫu i cho lớp c (1 nếu mẫu thuộc lớp c , 0 nếu không).
- $\hat{y}_{i,c}$: Xác suất dự đoán của mô hình cho mẫu i thuộc lớp c .

Ý nghĩa:

- Hàm loss sẽ cao nếu mô hình dự đoán sai (ví dụ, dự đoán xác suất cho nhãn đúng là rất thấp).
- Giúp huấn luyện mô hình bằng cách điều chỉnh các trọng số để giảm sai số giữa dự đoán và thực tế.

Hàm mất mát được đánh giá trên tập huấn luyện và tập kiểm thử

Hàm mất mát huấn luyện (Train Loss): Được tính trung bình từ tất cả các client trong mỗi vòng huấn luyện, biểu thị mức sai lệch của mô hình khi dự đoán trên dữ liệu cục bộ. Giá trị càng nhỏ thể hiện mô hình học tốt trên tập dữ liệu.

Hàm mất mát trên tập kiểm tra (Validation Loss): Giá trị mất mát khi mô hình dự đoán trên tập kiểm tra không thuộc về bất kỳ client cụ thể nào. Là thước đo quan trọng đánh giá khả năng tổng quát hóa.

2.5.3. Các chỉ số thời gian và hiệu quả hệ thống

Do đặc thù phân tán nên chi phí truyền thông và thời gian thực thi là yếu tố không thể thiếu trong đánh giá học liên kết.

Thời gian đánh giá mô hình trên tập kiểm tra sau mỗi vòng huấn luyện (Evaluation Duration). Cho thấy chi phí dự đoán phía server hoặc các client đại diện.

Thời gian tổng cộng để hoàn thành một vòng huấn luyện học liên kết (Round total time) bao gồm gửi mô hình, huấn luyện cục bộ, gửi lại kết quả và tổng hợp mô hình.

Ngoài ra còn có các tiêu chí khác trong tổng hợp thời gian huấn luyện như: Thời gian huấn luyện toàn bộ mô hình (Total runtime seconds), thời gian trung bình của mỗi vòng lặp (Average round time seconds), vòng lặp huấn luyện nhanh nhất (Fastest round seconds) đánh giá khả năng tối ưu của hệ thống, Vòng lặp huấn luyện chậm nhất (Slowest round seconds) xác định điểm nghẽn nếu có.

2.5.4. Vai trò trong đánh giá mô hình học liên kết:

Những chỉ số trên kết hợp cùng nhau đánh giá để cho ra được nhận định chính xác về hiệu quả mô hình trong môi trường học phân tán.

Mỗi tiêu chí đều quan trọng và nắm vai trò riêng trong việc hoàn thiện và đưa ra mô hình hiệu quả phù hợp với nhiều khía cạnh khác nhau.

CHƯƠNG 3: XÂY DỰNG MÔ HÌNH

3.1. Môi trường và công cụ thực nghiệm

Để triển khai và đánh giá các thuật toán học liên kết bao gồm FedAvg, FedProx và FedAdam, một môi trường thực nghiệm linh hoạt, phù hợp với người phát triển, người dùng và tối ưu hiệu suất đã được xây dựng dựa trên các thư viện mã nguồn mở phổ biến, mô hình học sâu tiêu chuẩn và nền tảng điện toán đám mây hỗ trợ GPU. Các công cụ được lựa chọn không chỉ đáp ứng yêu cầu kỹ thuật, mà còn hỗ trợ đầy đủ cho việc mô phỏng hệ thống với dữ liệu phân tán và không đồng nhất một cách dễ dàng và tối ưu cho quá trình thực hiện.

3.1.1. Ngôn ngữ lập trình và nền tảng

Ngôn ngữ sử dụng xuyên suốt là Python 3.10, nhờ khả năng tích hợp mạnh mẽ với các thư viện học sâu như PyTorch và TensorFlow, hỗ trợ mạnh mẽ và thích ứng với thư viện phát triển chính và hỗ trợ mô phỏng của đồ án. Python đồng thời cung cấp khả năng trực quan hóa, xử lý dữ liệu hiệu quả.

3.1.2. Thư viện và framework sử dụng

Flower là một framework mã nguồn mở hỗ trợ triển khai nhanh các thuật toán học liên kết. Với kiến trúc thiết kế linh hoạt, Flower cho phép mô phỏng hệ thống FL với số lượng lớn client, dữ liệu không đồng nhất (non-IID) và đa dạng chiến lược huấn luyện. Các thành phần chính được sử dụng trong nghiên cứu bao gồm:

- *flwr.client.NumPyClient, ClientApp*: Xây dựng logic huấn luyện phía client.
- *flwr_datasets*: Thư viện này được dùng để tải và phân phối dữ liệu một cách phân tán tới từng client.
- *flwr.server.ServerApp, ServerConfig*: Quản lý phía server và cấu hình huấn luyện.
- *flwr.server.strategy*: Gồm các thuật toán như FedAvg, FedProx, FedAdam.
- *run_simulation()*: Hàm khởi tạo quá trình mô phỏng tổng thể.



Hình 3-1: Logo Flower Framework

(*Nguồn: website flower.ai*)

3.1.3. Môi trường thực thi và phần cứng

Các mô phỏng được thực hiện trên nền tảng Google Colab và Kaggle Notebooks với hỗ trợ GPU miễn phí. Đây là hai nền tảng đám mây tiện lợi, dễ truy cập, và phù hợp cho mô hình FL nhỏ đến trung bình.

- GPU được sử dụng là NVIDIA Tesla T4, tất cả các thuật toán đều sử dụng chung GPU nhằm đưa ra những đánh giá và nhận xét đúng và chính xác về thời gian, có tính công bằng.
- Dung lượng RAM được sử dụng từ 15-16 GB do Google Colab và Kaggle cung cấp, trong quá trình thực nghiệm chưa từng xảy ra việc tràn bộ nhớ.

Phần cứng có những hạn chế nhất định như thời gian huấn luyện lâu, hao tốn nhiều tài nguyên, cần nhiều thời gian để có thể cho ra được kết quả và đưa ra nhận xét đúng. Khi sử dụng kỹ thuật “Grid Search” để tìm ra bộ tham số cực kì hao tốn tài nguyên.

3.2. Tập dữ liệu và thiết kế kịch bản non-IID

3.2.1. Tập dữ liệu CIFAR10 và Fashion-MNIST

Hai tập dữ liệu CIFAR10 và Fashion-MNIST được sử dụng trong nghiên cứu này do sự phổ biến và dễ dàng triển khai. Ngoài ra, sự khác nhau về đặc tính của hai tập dữ liệu này sẽ giúp đánh giá khả năng tổng quát hóa của các thuật toán trên các loại dữ liệu khác nhau về đặc tính và độ phức tạp.

Bảng 3-1: Đặc tính tổng quan của hai tập dữ liệu

Đặc tính	CIFAR10	Fashion-MNIST
Kích thước ảnh gốc	32×32 pixel	28×28 pixel
Số kênh	3 kênh màu	1 kênh màu
Số lớp phân loại	10	10
Tập huấn luyện	50.000	60.000
Tập kiểm tra	10.000	10.000
Loại dữ liệu	Ảnh tự nhiên (động vật, phương tiện)	Thời trang (quần áo, giày dép)
Độ phức tạp	Cao	Trung bình

Để phù hợp với kiến trúc ResNet18, cả hai tập dữ liệu được resize về kích thước lớn hơn:

- CIFAR-10 được resize lên 112×112 pixel
- Fashion-MNIST được resize lên 32×32 pixel.

Việc resize này giúp tận dụng tốt hơn khả năng trích xuất đặc trưng của mô hình pretrained trên ImageNet. Quá trình tiền xử lý dữ liệu bao gồm các kỹ thuật data augmentation như random flip và rotation để tăng tính đa dạng của dữ liệu, cùng với normalization phù hợp cho từng tập dữ liệu. DataLoader được cấu hình tối ưu cho môi trường federated learning với các tham số đảm bảo tính ổn định và hiệu suất trong quá trình huấn luyện phân tán.

Khác với các phương pháp học máy truyền thống sử dụng toàn bộ dữ liệu tập trung tại một nơi, FL yêu cầu dữ liệu được phân tán và lưu trữ cục bộ tại nhiều client khác nhau. Mỗi client chỉ có quyền truy cập vào phần dữ liệu của riêng mình và không thể chia sẻ dữ liệu trực tiếp với các client khác. Điều này tạo nên thách thức lớn về tính không đồng nhất (non-IID) của dữ liệu, khi mỗi client có thể có phân phối dữ liệu khác biệt đáng kể so với phân phối tổng thể.

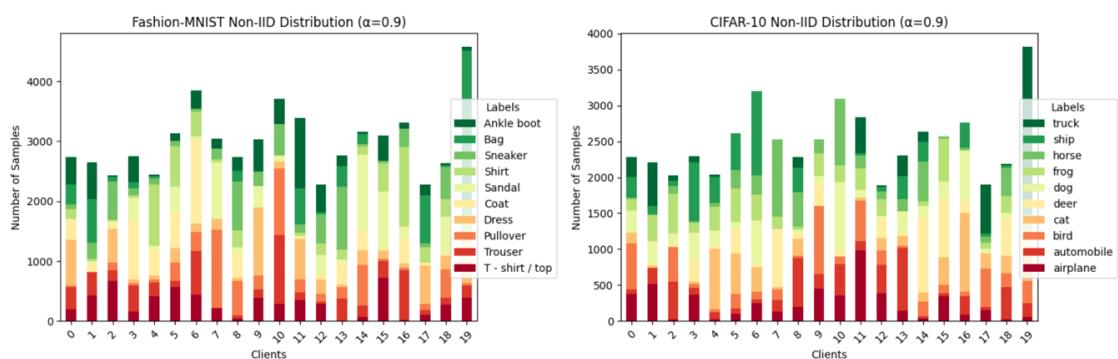
Trong thực nghiệm này, thực hiện việc phân phối dữ liệu cho 20 client trên từng tập dữ liệu (CIFAR10 và Fashion-MNIST). Quá trình phân phối dữ liệu cho các client được thực hiện thông qua thư viện Flower (Federated Learning Framework), một framework mã nguồn mở chuyên cho FL. Flower cung cấp lớp FederatedDataset cho phép tải và phân chia dữ liệu một cách linh hoạt, đồng thời hỗ trợ các chiến lược phân phối khác nhau thông qua các partitioner

3.2.2. Thiết kế kịch bản non-IID

Trong thực tế, dữ liệu của các client trong federated learning thường có phân phối không đồng nhất (non-IID), tạo ra thách thức lớn cho quá trình học. Vì thế, để mô phỏng tình huống Flower Framework đã cung cấp lớp Dirichlet Partitioner giúp giải quyết các thách thức kỹ thuật trong việc mô phỏng môi trường federated learning, bao gồm việc tạo ra các partition dữ liệu có độ lệch non-IID khác nhau, quản lý việc tải dữ liệu song song cho nhiều client, và đảm bảo tính reproducibility của các thực nghiệm. Phân phối dữ liệu dựa trên phân phối Dirichlet với tham số concentration α (alpha). Giá trị alpha càng nhỏ thì độ lệch non-IID càng nặng, nghĩa là mỗi client chứa số lượng loại nhãn khác nhau.

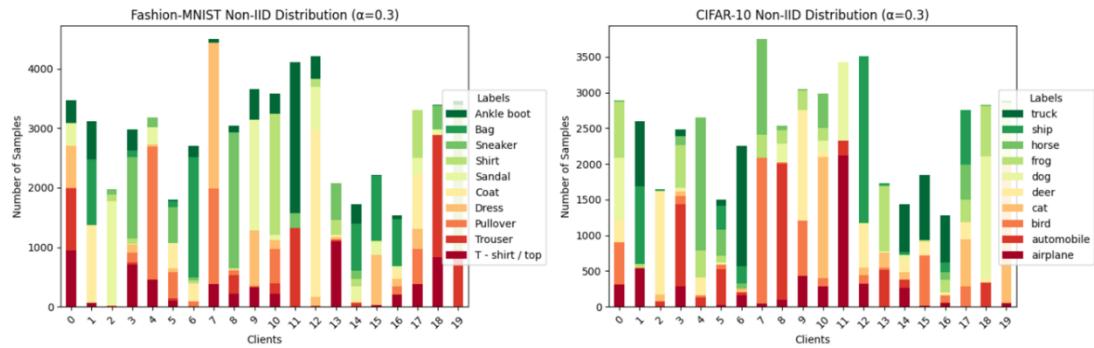
Trong thực nghiệm này, xây dựng hai kịch bản non-IID trên cả hai tập dữ liệu với hai giá trị alpha khác nhau:

- $\alpha = 0.9$ (non-IID nhẹ): Phân phối dữ liệu ít lệch hơn, mỗi client vẫn có đủ của hầu hết các nhãn nhưng với tỷ lệ không hoàn toàn cân bằng



Hình 3-2: Phân phối dữ liệu với $\alpha = 0.9$ trên hai tập dữ liệu

- $\alpha = 0.3$ (non-IID nặng): Phân phối dữ liệu của mỗi client có độ lệch cao, tập trung chủ yếu vào một số ít nhãn cụ thể.



Hình 3-3: : Phân phối dữ liệu với $\alpha = 0.9$ trên hai tập dữ liệu

Việc thiết kế các kịch bản non-IID này giúp đánh giá hiệu suất và độ bền của các thuật toán FL trong các điều kiện thực tế, đặc biệt là khả năng của chúng trong việc cải thiện tốc độ hội tụ và độ chính xác khi đối mặt với sự phân bố dữ liệu đa dạng và không đồng nhất giữa các client.

3.3. Mô hình ResNet18 sử dụng trong Federated Learning

3.3.1. Cấu hình tổng quan mô hình Resnet18

Trong nhiều nghiên cứu về federated learning (FL), ResNet18 được lựa chọn làm mô hình tin cậy, nhất là đối với các tập dữ liệu phân loại ảnh quy mô vừa như CIFAR-10, CIFAR-100 và FMNIST. Việc sử dụng một mô hình chuẩn giúp so sánh hiệu suất giữa các thuật toán khác nhau một cách nhất quán hơn. [1]

Trong bài báo Adaptive Federated Optimization, ResNet18 được sử dụng làm mô hình chuẩn cho CIFAR-10 và CIFAR-100 và được thay thế toàn bộ lớp batch normalization (BN) bằng group normalization (GN) vì lớp BN đã phát sinh nhiều vấn đề trong môi trường FL với dữ liệu cục bộ được phân phối không đồng nhất. Các thông số thống kê (mean, variance) của lớp BN không chủ yếu phản ánh đặc trưng tổng thể, nên có thể làm suy giảm độ chính xác trong FL do tập dữ liệu cục bộ có phân phối không đồng nhất. Một trong những giải pháp hiệu quả là thay thế BN bằng GN, nơi các tham số normalization được tính theo nhóm cố định thay vì batch. Sự

thay đổi này đã được chứng minh đem lại cải thiện đáng kể về độ chính xác trong các thiết lập FL với ResNet18, đặc biệt đối với CIFAR-100 non-IID. [1]

Trong nghiên cứu này mô hình ResNet18 được thiết kế bao gồm hai thành phần chính là backbone feature extractor sử dụng kiến trúc ResNet18 pretrained trên ImageNet, được thay thế toàn bộ lớp BN thành GN và classifier head được thiết kế với cơ chế dropout đa tầng nhằm giảm thiểu overfitting.

Bảng 3-2: Cấu hình tổng quan mô hình ResNet18_GroupNorm

Thành phần	Cấu hình
Backbone	ResNet18_Weights.IMAGENET1K_V1, thay tất cả BN bằng GN
Feature Extractor	512
Classifier Head	512→128→64→10
Normalization	Group Normalization
Activation	ReLU
Dropout	0.3
Weight Initialization	nonlinearity='relu'

3.3.2. Cấu hình mô hình Resnet trên từng tập dữ liệu

Mô hình ResNet18_GroupNorm được điều chỉnh riêng cho từng tập dữ liệu để phù hợp với đặc tính của dữ liệu đầu vào. Với tập dữ liệu CIFAR-10, sử dụng trực tiếp cấu hình ở Bảng 3-1 vì IMAGENET được huấn luyện để làm việc với ảnh 3 kênh màu. Còn tập dữ liệu Fashion-MNIST có sự thay đổi nhỏ, do đặc thù đặc tính của ảnh là 1 kênh màu, vì thế:

- Lớp convolution đầu tiên được thay đổi từ 3 kênh xuống 1 kênh.

- Trọng số của mô hình ResNet18 pretrained mới được tính bằng trung bình của 3 kênh RGB gốc, tạo ra một kênh đại diện cho thông tin grayscale.

3.4. Cấu hình thực nghiệm và tham số chạy mô phỏng

Để tiến hành các thử nghiệm, thư viện Flower cũng cấp những chức năng cần thiết để thiết lập môi trường ảo giúp người phát triển có thể mô phỏng môi trường FL bằng cách sử dụng các tham số cụ thể để cấu hình quá trình học liên kết (Federated Learning). Các tham số này được thiết lập nhằm tạo ra một môi trường thử nghiệm có kiểm soát, mô phỏng các điều kiện khác nhau của hệ thống.

Cụ thể, tổng cộng 20 phân vùng dữ liệu được tạo ra để mô phỏng 20 client tham gia. Trong mỗi vòng huấn luyện, 17 client (85% của 20) được chọn ngẫu nhiên để tham gia cập nhật mô hình, trong đó số lượng client tối thiểu phải là 7. Tương tự, để đánh giá hiệu suất mô hình sau mỗi vòng, 6 client (30% của 20) được chọn ngẫu nhiên, với số lượng tối thiểu là 10. Đáng chú ý, một vòng huấn luyện chỉ có thể bắt đầu khi có ít nhất 10 client sẵn sàng tham gia.

Mỗi client được khởi tạo mô hình cục bằng trọng số được lấy từ mô hình toàn cục là ResNet18, sau đó được giao nhiệm vụ huấn luyện mô hình cục bộ với 2 epoch và kích thước batch là 20 để tải dữ liệu. Toàn bộ quá trình được lặp lại trong 100 vòng huấn luyện toàn cục.

Để kiểm tra ảnh hưởng của dữ liệu, thử nghiệm được thực hiện trên hai cấu hình dữ liệu khác nhau, được điều chỉnh bằng tham số Alpha với giá trị 0.3 và 0.9. Giá trị Alpha càng thấp thì dữ liệu giữa các client càng không đồng nhất (non-IID), cho phép đánh giá hiệu suất thuật toán trong các kịch bản thực tế.

Bảng 3-3: Cấu hình ở server và client

Tham số	Mô tả	Giá trị
NUM_PARTITIONS	Số lượng phân vùng dữ liệu	20

FRACTION_FIT	Tỷ lệ client được chọn để huấn luyện trong mỗi vòng.	0.85
FRACTION_EVALUATE	Tỷ lệ client được chọn để đánh giá sau mỗi vòng.	0.3
MIN_FIT_CLIENTS	Số client tối thiểu bắt buộc tham gia huấn luyện trong một vòng.	7
MIN_EVALUATE_CLIENTS	Số client tối thiểu để thực hiện đánh giá mô hình sau mỗi vòng	10
MIN_AVAILABLE_CLIENTS	Số client cần khả dụng tối thiểu để server có thể khởi động vòng.	10
EPOCH	Số epoch huấn luyện cục bộ tại mỗi client.	2
BATCH	Kích thước batch size trong việc tải dữ liệu	20
ROUND	Số vòng lặp huấn luyện toàn cục.	100
ALPHA	Độ đồng nhất	0.3, 0.9
INITIAL_PARAMETERS	Khởi tạo mô hình ResNet18 GroupNorm toàn cục khi bắt đầu huấn luyện để gửi đến tất cả client đảm bảo đồng bộ	net

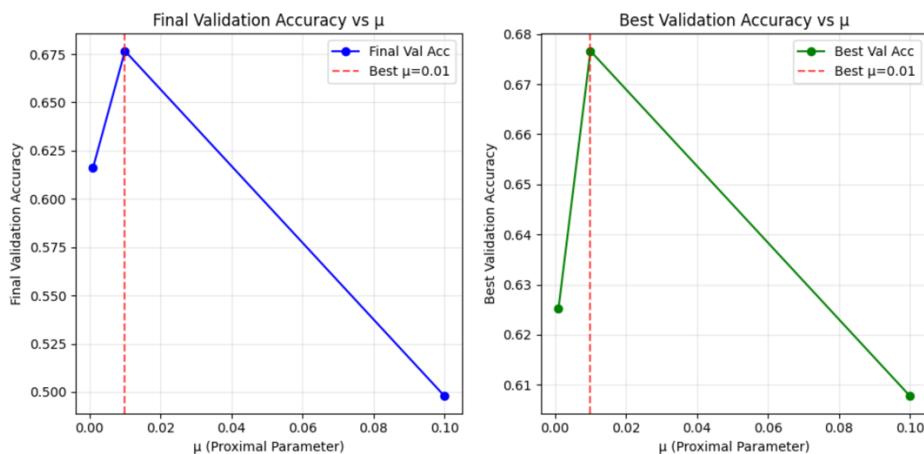
3.5. Điều chỉnh tham số cho FedProx và FedAdam

3.5.1. Điều chỉnh tham số cho FedProx

FedProx là một khung tối ưu hóa liên kết được thiết kế để xử lý tính không đồng nhất của hệ thống và dữ liệu. Nó cải thiện FedAvg bằng cách thêm một thuật ngữ xấp xỉ (proximal term) vào hàm mục tiêu cục của client. Vì thế tham số chính của FedProx cần điều chỉnh là tham số μ (mu).

Trong Section 5.3.1 của bài báo “Federated Optimization in Heterogeneous Networks”, tác giả Li và cộng sự đã chứng minh rằng khi $\mu = 0$, độ lệch của gradient local tăng dần cùng với mức độ không đồng nhất thông kê, dẫn đến hội tụ chậm hoặc thậm chí phân kỳ. Ngược lại, $\mu > 0$ giúp giảm đáng kể sự lệch này, nhờ đó FedProx có hiệu ứng ổn định hơn và cho phép kết hợp partial updates từ các thiết bị làm chậm mà không làm suy giảm độ chính xác mô hình toàn cục. Việc chọn $\mu > 0$ mang lại tính ổn định và khả năng hội tụ được cải thiện rõ rệt ngay cả khi dữ liệu ở các client không độc lập và phân phối khác nhau mang lại độ chính xác trung bình của mô hình toàn cục được cải thiện đáng kể. Li et al. cũng đã đề xuất grid search giá trị μ từ tập $\{0.001, 0.01, 0.1, 1\}$, với kết quả tốt nhất thường rơi vào khoảng 0.01 hoặc 1 khi dữ liệu rất không đồng nhất hoặc biến thiên gradient cao. [2]

Kết thừa từ những kinh nghiệm đó, sử dụng phương pháp tìm kiếm Grid Search để tìm ra giá trị μ phù hợp với bài toán. Trong thực nghiệm tìm tham số μ tốt nhất bằng phương pháp Grid Search trên CIFAR-10 với độ non-IID nặng ($\alpha = 0.3$) trải qua 30 vòng giao tiếp cùng với những cấu hình tương tự như ở phần 3.4, thử nghiệm grid search đã được thực hiện với giá trị μ là $\{0.001, 0.01, 0.1, 1\}$. Kết quả cho thấy $\mu = 0.01$ mang lại sự kết hợp tốt nhất giữa hội tụ nhanh và độ chính xác trung bình 10 vòng cuối, do đó giá trị này được chọn làm tiêu chuẩn cho quá trình huấn luyện FedProx tiếp theo.



Hình 3-4: Kết quả cho thấy $\mu = 0.01$ tốt nhất

3.5.2. Điều chỉnh tham số cho FedAdam

FedAdam là một thuật toán tối ưu hóa thích ứng được mở rộng cho FL, sử dụng Adam làm trình tối ưu hóa phía máy chủ (server optimizer). Các tham số chính của FedAdam cần điều chỉnh bao gồm:

- client_lr: Tốc độ học ở phía client
- server_lr: Tốc độ học ở phía server
- beta_1: Tham số phân rã β_1
- beta_2: Tham số phân rã β_2
- tau: Tham số thích ứng τ

Trong Section D của bài báo “Adaptive Federated Optimization”, tác giả Reddi và cộng sự đã thảo luận về nghiên cứu lựa chọn các tham số phù hợp cho các bài toán FedOpt nói chung và FedAdam nói riêng. Trong đó, bài báo nêu ra việc cài đặt mặc định các tham số $\beta_1 = 0.9$, $\beta_2 = 0.99$ cho hầu hết các thử nghiệm, tác giả cũng đã nghiên cứu và khuyến nghị $\tau = 10^{-3}$ thường hoạt động ổn định trong nhiều thử nghiệm mà không cần điều chỉnh thêm.

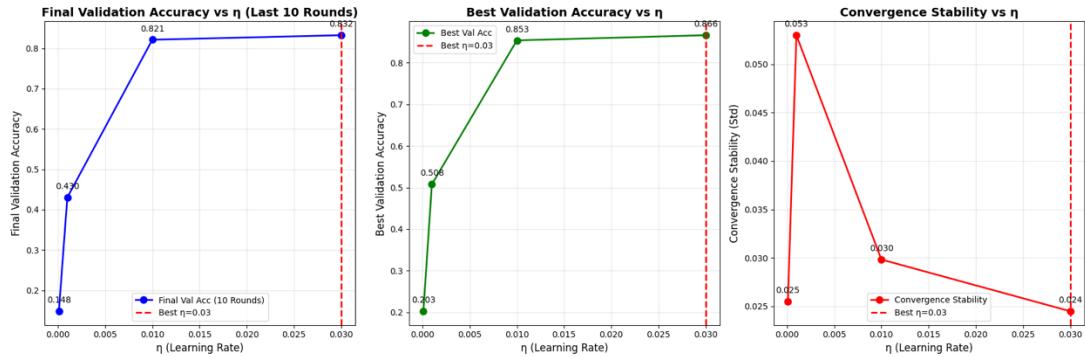
Các tham số client_lr và server_lr cũng đã được nghiên cứu trong bài báo đó. Tuy nhiên, với mỗi bài toán khác nhau thì mỗi bộ client_lr, server_lr đều khác nhau. Trong đó, đối với bộ dữ liệu CIFAR10 được thực nghiệm và cho ra kết quả trên trung bình độ chính xác trên tập dữ liệu Validation ở 100 vòng huấn luyện thì kết quả cho ra tốt nhất ở bộ $\text{client_lr} = 10^{-1.5}$, $\text{server_lr} = 10^{-2}$ với trung bình độ chính xác là 77.4%. Bài báo này cũng khuyến nghị nên dùng phương pháp grid search để tìm và xác định các giá trị tối ưu này. [1]

Kế thừa từ những kết quả nghiên cứu đó, thực nghiệm để tìm hai tham số bằng phương pháp grid search trên tập CIFAR-10 với những cấu hình tương tự nhưng thay đổi số lượng vòng giao tiếp trong mỗi lần thử là 50 và được tiến hành theo hai bước độc lập do giới hạn tài nguyên.

Trước hết, client_lr được cố định tạm thời ở $10^{-1.5}$ dựa trên quan sát từ ma trận nhầm lẫn trong bài báo “Adaptive Federated Optimization”, trong khi thực hiện

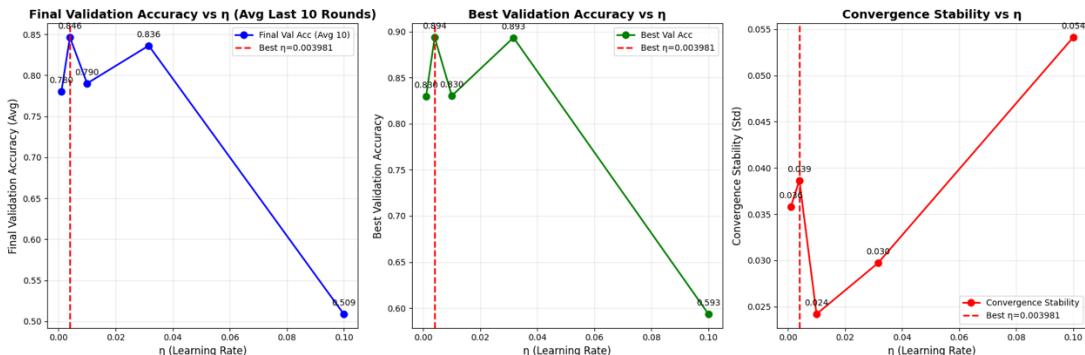
Chương 3: Xây dựng mô hình

grid search server_lr với các giá trị $\{0.01, 0.03, 0.001, 0.0001\}$. Kết quả thử nghiệm ghi nhận server_lr = 0.03 là giá trị tốt nhất trong trung bình 10 vòng cuối.



Hình 3-5: Kết quả cho thấy server_lr = 0.03 là tốt nhất

Sau đó, client_lr được tinh chỉnh trên tập giá trị $\{0.1, 0.01, 0.001, 10^{(-1.5)}, 10^{(-2.4)}\}$, trong đó client_lr = 0.003981 = $10^{(-2.4)}$ thể hiện hiệu suất hội tụ nhanh và độ chính xác trung bình 10 vòng cuối tốt nhất.



Hình 3-6: Kết quả cho thấy client_lr = $10^{(-2.4)}$ là tốt nhất

Sau khi thực hiện nhiều bước tìm những tham số tối ưu cho thuật toán FedAdam, đây là bộ tham số sẽ đưa vào thực nghiệm cho cả hai tập dữ liệu:

- client_lr: $10^{(-2.4)}$
- server_lr: 0.03
- beta_1: 0.9
- beta_2: 0.99
- tau: 10^{-3}

Bộ tham số này không chỉ đảm bảo tính ổn định trong quá trình huấn luyện mà còn giúp thuật toán FedAdam đạt được kết quả tốt hơn rõ rệt so với các thiết lập mặc

Chương 3: Xây dựng mô hình

định ban đầu. Việc lựa chọn các giá trị học thích hợp cho cả client và server và hệ số điều chỉnh tỉ lệ thay đổi (tau), đóng vai trò then chốt trong việc nâng cao hiệu quả tối ưu liên tục theo từng vòng liên kết

CHƯƠNG 4: THỰC NGHIỆM CHƯƠNG TRÌNH

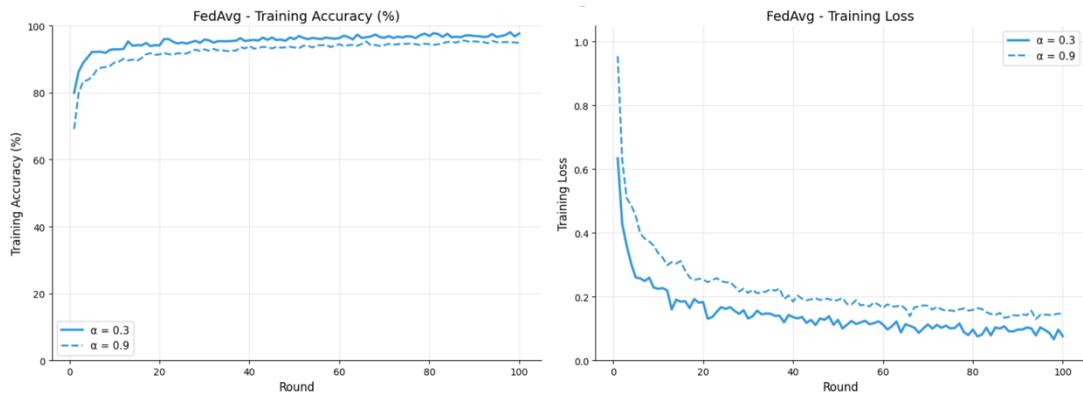
4.1. Kết quả huấn luyện của từng chiến lược

Các thí nghiệm được thực hiện trên hai tập dữ liệu Fashion-MNIST và CIFAR-10 với ba thuật toán học liên kết bao gồm FedAvg, FedProx và FedAdam. Mỗi thuật toán được triển khai với hai mức độ không đồng nhất dữ liệu là $\alpha = 0.3$ và $\alpha=0.9$. Quá trình huấn luyện kéo dài trong 100 vòng lặp, mỗi vòng gồm hai epoch cục bộ với kích thước batch là 20. Các thông số khác như tỉ lệ client tham gia huấn luyện và đánh giá, số lượng client tối thiểu,... và điều huấn luyện trên GPU T4 được giữ cố định trong suốt quá trình thực nghiệm để đảm bảo tính công bằng.

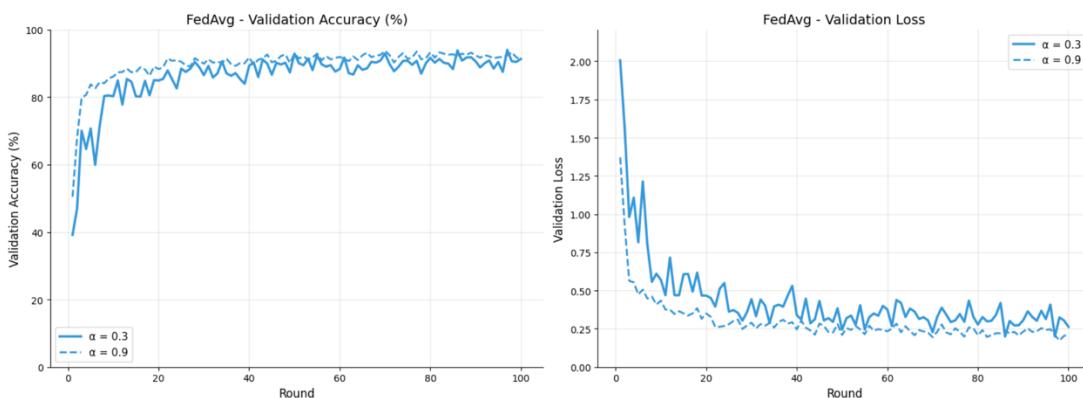
4.1.1. Kết quả huấn luyện trên dữ liệu Fashion-MNIST

Ba thuật toán học liên kết nổi bật là FedAvg, FedProx và FedAdam đã được triển khai trên tập dữ liệu Fashion-MNIST. Các thử nghiệm được tiến hành trên hai kịch bản dữ liệu khác nhau, được điều chỉnh bởi tham số không đồng nhất α với giá trị 0.3 và 0.9.

Sau khi chạy mô phỏng trong 100 vòng lặp, và hiệu suất của các thuật toán được trực quan hóa qua các biểu đồ. Những biểu đồ này minh họa diễn biến của độ chính xác và hàm mất mát trên cả tập huấn luyện và tập validation theo từng vòng. Mục đích là để so sánh trực quan và làm nổi bật những khác biệt về tốc độ hội tụ cũng như độ ổn định giữa ba chiến lược. Bên cạnh đó, thời gian thực thi tổng thể của từng thuật toán cũng được ghi lại. Phân tích này sẽ cung cấp một cái nhìn toàn diện về hiệu quả tính toán, giúp đánh giá không chỉ hiệu suất mô hình mà còn cả chi phí tài nguyên cần thiết cho mỗi phương pháp.



Hình 4-1: Kết quả huấn luyện của FedAvg trên Fashion-MNIST



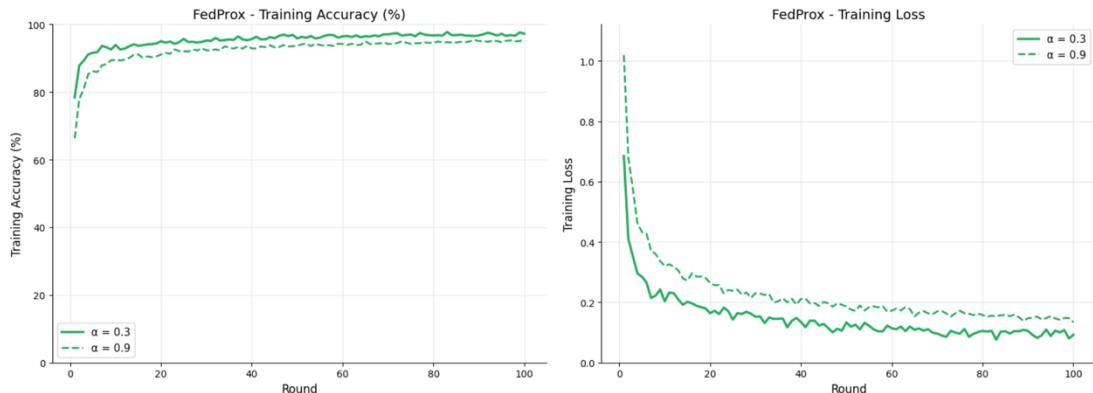
Hình 4-2: Kết quả đánh giá của FedAvg trên Fashion-MNIST

Kết quả huấn luyện (Hình 4-1), (Hình 4-2) trên tập dữ liệu Fashion-MNIST, thuật toán FedAvg cho thấy khả năng hội tụ nhanh và hiệu quả trong môi trường dữ liệu phi đồng nhất. Với $\alpha = 0.3$, độ chính xác huấn luyện đạt mức 97.69%, mô hình nhanh chóng tăng từ khoảng 70% lên gần 95% chỉ trong 20 vòng đầu, cho thấy tốc độ học án tượng.

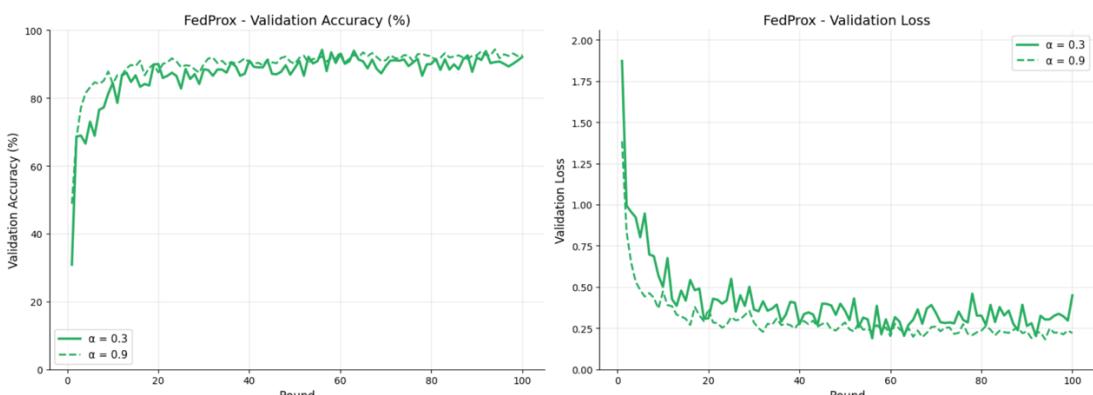
Hàm mất mát trong quá trình huấn luyện giảm đều và đạt giá trị thấp với giá trị 0.075 ở epoch thứ 100, phản ánh khả năng tối ưu ổn định mà không có dấu hiệu overfitting. Trên tập kiểm thử, mô hình đạt độ chính xác 91.33%, cho thấy năng lực tổng quát hóa tốt.

Khi α tăng lên 0.9, mặc dù độ chính xác huấn luyện giảm còn 94.90% và mất mát tăng lên khoảng 0.145, nhưng đường học biểu diễn mượt và ổn định hơn do dữ liệu đồng đều hơn. Tuy nhiên, độ chính xác kiểm thử không chênh lệch nhiều so với $\alpha = 0.3$, đạt 91.24%.

Qua đó cho thấy FedAvg hoạt động khá tốt trong điều kiện dữ liệu không đồng đều, nhưng vẫn tồn tại dao động và thiếu cơ chế ổn định hóa mạnh mẽ trong quá trình cập nhật mô hình toàn cục.



Hình 4-3: Kết quả huấn luyện của FedProx trên Fashion-MNIST



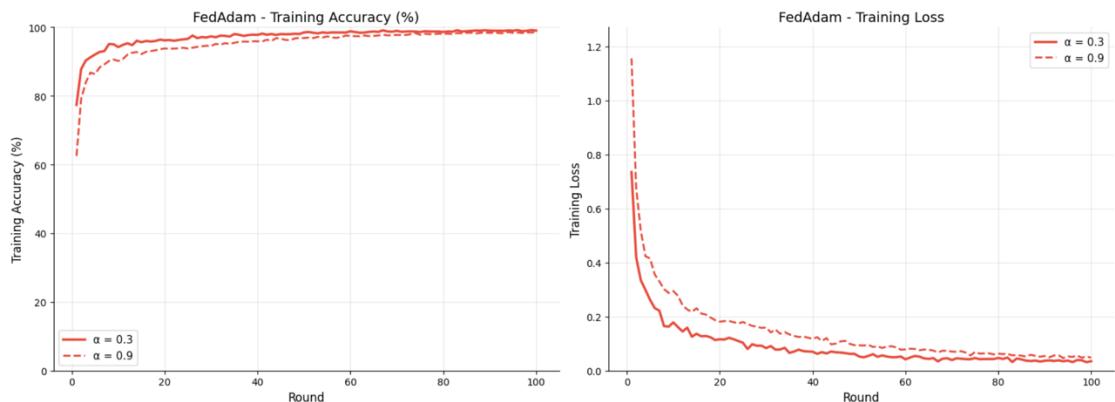
Hình 4-4: Kết quả đánh giá của FedProx trên Fashion-MNIST

Trong khi đó, FedProx (Hình 4-3), (Hình 4-4) tỏ ra là lựa chọn ưu việt hơn trong bối cảnh dữ liệu không đồng nhất nhờ vào việc bổ sung thành phần ràng buộc proximal trong hàm mục tiêu tối ưu cục bộ.

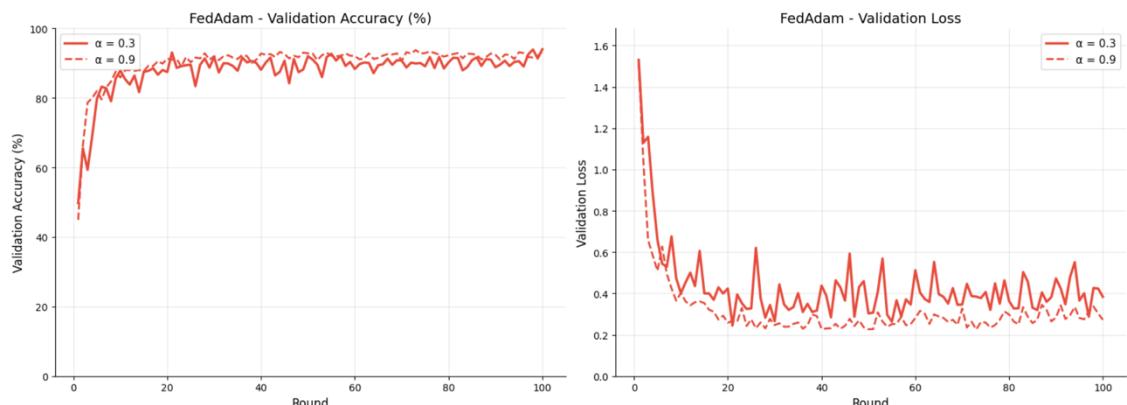
Khi $\alpha = 0.3$, độ chính xác huấn luyện đạt 97.20%, thấp hơn một chút so với FedAvg, nhưng quá trình học diễn ra ổn định hơn rõ rệt, không có những dao động bất thường. Trên tập kiểm thử, mô hình đạt 92.11% độ chính xác – cao hơn FedAvg ở cùng mức α .

Khi $\alpha = 0.9$, mô hình tiếp tục cho kết quả tốt với độ chính xác huấn luyện đạt 95.50% và độ chính xác kiểm thử lên đến 92.66%.

Sự ổn định của FedProx thể hiện rõ ở các biểu đồ, trong đó hàm mất mát giảm đều và các đường cong độ chính xác không có dao động mạnh, chứng minh rằng thuật toán này tổng quát hóa tốt và thích nghi hiệu quả trong môi trường phân phối dữ liệu không đồng đều.



Hình 4-5: Kết quả huấn luyện của FedAdam trên Fashion-MNIST

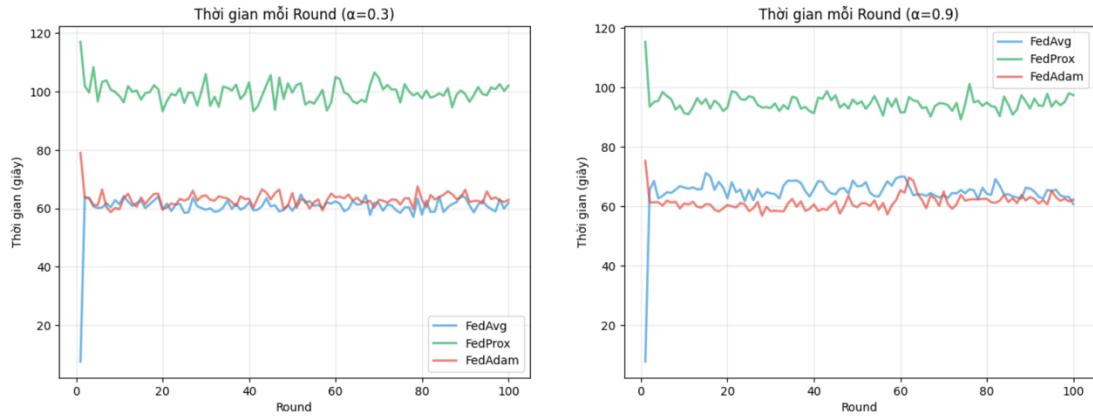


Hình 4-6: Kết quả đánh giá của FedAdam trên Fashion-MNIST

FedAdam (Hình 4-5), (Hình 4-6) nổi bật hơn cả với hiệu suất học vượt trội trên cả hai mức α . Khi $\alpha = 0.3$, mô hình đạt độ chính xác huấn luyện lên tới 98.99% và kiểm thử 93.99%, cao nhất trong tất cả các thuật toán.

Biểu đồ học cho thấy mô hình hội tụ rất nhanh và mượt, hàm mất mát giảm mạnh và không có biến động lớn. Ở $\alpha = 0.9$, mặc dù dữ liệu đồng đều hơn, FedAdam vẫn giữ hiệu năng cao với độ chính xác huấn luyện 98.58% và kiểm thử 93.32%, cho thấy khả năng thích ứng tuyệt vời.

Đặc biệt, mô hình duy trì sự ổn định tốt trên cả hai tập huấn luyện và kiểm thử, khẳng định ưu thế của cơ chế cập nhật thích nghi mà FedAdam áp dụng tại server. Từ đó, có thể kết luận rằng FedAdam là thuật toán hiệu quả nhất trong ba phương pháp, vừa đảm bảo độ chính xác cao, vừa duy trì sự ổn định trong quá trình huấn luyện với dữ liệu phi đồng nhất.



Hình 4-7: So sánh thời gian chạy của 3 thuật toán trên Fashion-MNIST

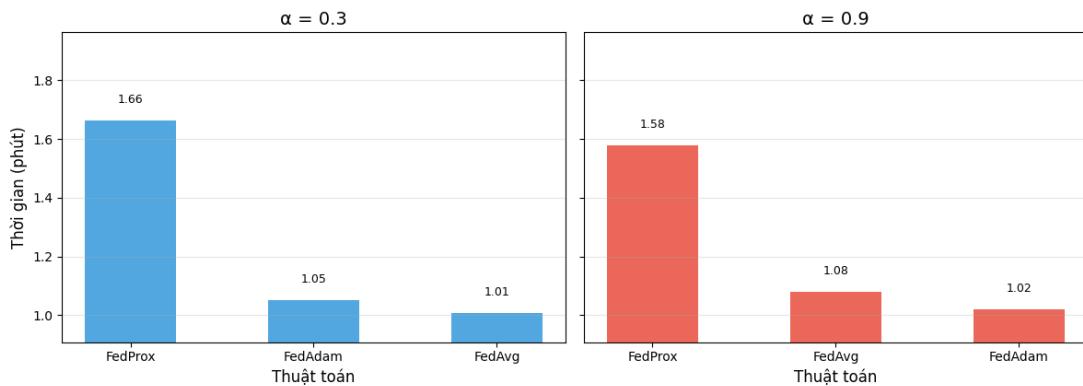
Biểu đồ (Hình 4-7) cung cấp cái nhìn trực quan về thời gian thực hiện mỗi vòng (Round) huấn luyện tương ứng với hai mức phân phối dữ liệu: $\alpha = 0.3$ (non-IID mạnh) và $\alpha = 0.9$ (gần IID).

Khi $\alpha = 0.3$:

- FedProx có thời gian mỗi vòng lớn nhất, dao động quanh 100 giây. Nguyên nhân đến từ việc FedProx có thêm ràng buộc proximal, làm tăng chi phí tính toán ở phía client.
- FedAvg và FedAdam có thời gian ổn định hơn và thấp hơn, khoảng 60–65 giây mỗi vòng. FedAdam mặc dù áp dụng tối ưu hóa server-side (Adam), nhưng thời gian cập nhật không tăng đáng kể so với FedAvg, do hầu hết thời gian nằm ở quá trình huấn luyện phía client.

Khi $\alpha = 0.9$:

- Mọi thuật toán đều có thời gian huấn luyện mỗi vòng ổn định hơn so với $\alpha = 0.3$, do dữ liệu được phân phối đồng đều, giúp client xử lý nhanh và nhất quán hơn.
- FedProx vẫn là thuật toán tốn nhiều thời gian nhất (gần 95 giây), trong khi FedAdam là nhanh nhất (gần 60 giây), tương đương FedAvg.



Hình 4-8: So sánh thời gian trung bình của 3 thuật toán (Fashion-MNIST)

Biểu đồ cột này (Hình 4-8) cho thấy thời gian tổng thể để hoàn thành 100 vòng huấn luyện, từ đó đánh giá hiệu năng thực thi toàn cục của từng thuật toán.

Khi $\alpha = 0.3$:

- FedAvg là nhanh nhất với tổng thời gian huấn luyện chỉ gần 1.01 phút/vòng, tức gần 1.68 giờ tổng cộng.
- FedAdam tiêu tốn nhiều thời gian hơn một chút (gần 1.05 phút/vòng, gần 1.75 giờ).
- FedProx có thời gian cao nhất (gần 1.66 phút/vòng, tức gần 2.77 giờ), khẳng định rằng chi phí tính toán tăng đáng kể do cấu trúc tối ưu hóa có ràng buộc.

Khi $\alpha = 0.9$:

- Cả ba thuật toán đều có thời gian thấp hơn một chút so với $\alpha = 0.3$ do dữ liệu đồng đều hơn.
- FedAdam tiếp tục duy trì thời gian thấp nhất (gần 1.08 phút/vòng), còn FedProx vẫn là chậm nhất (gần 1.58 phút/vòng).

FedAvg nằm giữa, tiêu tốn khoảng 1.08 phút/vòng.

FedProx có độ trễ cao và cần nhiều thời gian hơn để hoàn thành quá trình huấn luyện, nhưng đổi lại là độ ổn định mô hình cao trong môi trường dữ liệu không đồng nhất. Ngược lại, FedAdam đạt được cân bằng tối ưu giữa hiệu năng học, tính ổn định và hiệu quả thời gian, là lựa chọn phù hợp trong các hệ thống cần tốc độ và độ chính xác cao. FedAvg vẫn là thuật toán cơ bản, dễ triển khai và đạt kết quả tốt trong điều kiện dữ liệu tương đối đồng đều.

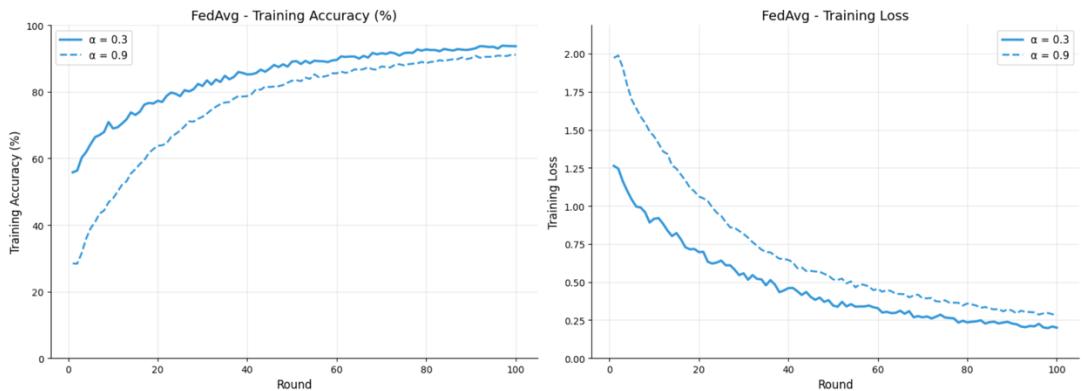
Từ những biểu đồ và nhận xét trên, cho thấy:

- FedAvg là thuật toán đơn giản, dễ triển khai, cho kết quả tốt trong môi trường dữ liệu gần đồng nhất ($\alpha = 0.9$), nhưng kém ổn định khi dữ liệu phi đồng nhất ($\alpha = 0.3$), biểu hiện ở sự dao động và hội tụ chậm.
- FedProx giúp cải thiện độ ổn định khi dữ liệu phân mảnh nhờ vào hàm phạt proximal, tuy nhiên phải đánh đổi bằng chi phí tính toán và thời gian huấn luyện cao hơn.
- FedAdam đạt hiệu suất cao nhất ở cả hai mức α , với khả năng hội tụ nhanh, chính xác cao và ổn định. Đây là thuật toán thích hợp trong các môi trường FL thực tế với dữ liệu không đồng nhất. FedAdam là lựa chọn toàn diện, FedProx phù hợp khi ưu tiên sự ổn định, còn FedAvg vẫn là nền tảng cơ bản cho các mô hình đơn giản. Lựa chọn thuật toán nên cân nhắc đến tính chất dữ liệu và điều kiện triển khai cụ thể

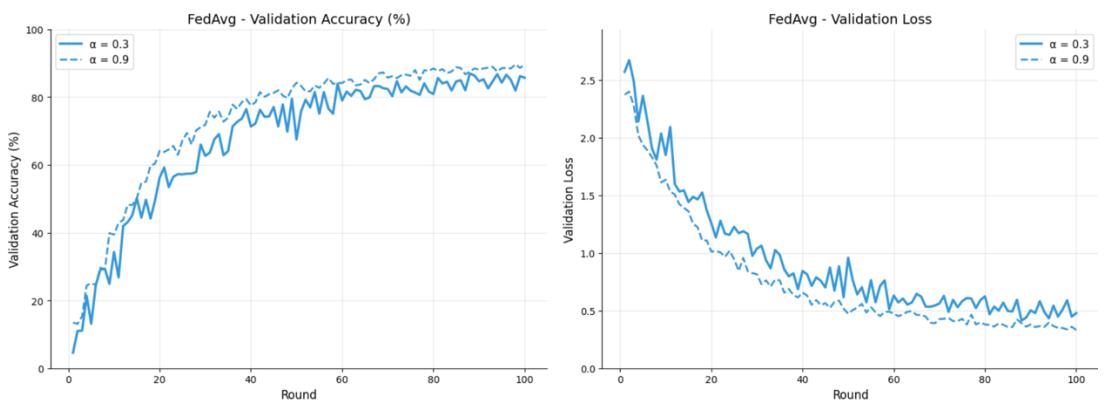
4.1.2. Kết quả huấn luyện trên dữ liệu CIFAR10

Để cung cấp một cái nhìn toàn diện hơn về hiệu quả của các thuật toán học liên kết, thực hiện thử nghiệm các thuật toán được áp dụng trên tập dữ liệu CIFAR10, một bộ dữ liệu phức tạp hơn so với Fashion-MNIST.

Tương tự như thử nghiệm trước, các mô hình được huấn luyện trong 100 vòng lặp và kết quả được phân tích chi tiết. Điều này sẽ làm rõ sự khác biệt trong hành vi và khả năng hội tụ của mỗi thuật toán khi đối mặt với dữ liệu phức tạp để được ưu điểm và hạn chế của từng phương pháp trong các điều kiện thực tế khác nhau.



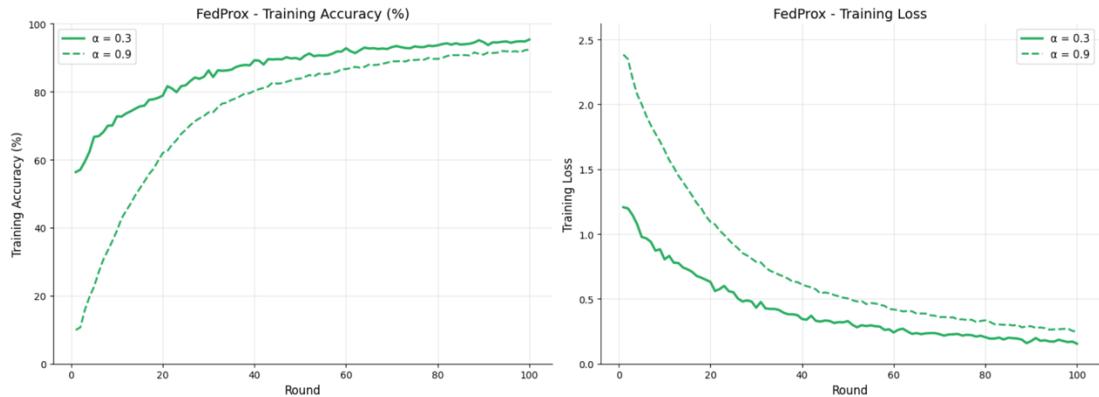
Hình 4-9: Kết quả huấn luyện của FedAvg trên CIFAR10



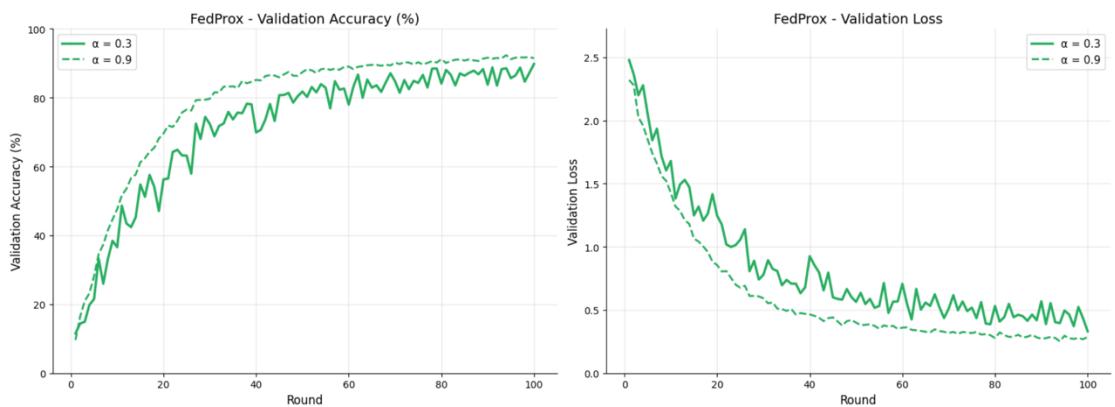
Hình 4-10: Kết quả đánh giá của FedAvg trên CIFAR10

Ở cả hai mức phân phối dữ liệu $\alpha = 0.3$ và $\alpha = 0.9$ được thể hiện ở biểu đồ (Hình 4-9), (Hình 4-10) FedAvg thể hiện xu hướng hội tụ ổn định qua 100 vòng huấn luyện. Với $\alpha = 0.3$, độ chính xác huấn luyện đạt 93.70% và hàm mất mát giảm về mức ~ 0.22 . Khi dữ liệu phân phối đồng đều hơn ($\alpha = 0.9$), mô hình đạt 91.14% độ chính xác huấn luyện, với hàm mất mát khoảng 0.3 – cao hơn so với $\alpha = 0.3$. Tốc độ hội tụ nhìn chung tốt trong khoảng 30 vòng đầu tiên, sau đó duy trì ổn định.

Trên tập kiểm thử, mô hình đạt độ chính xác 85.71% ($\alpha = 0.3$) và 89.56% ($\alpha = 0.9$). Tuy nhiên, với $\alpha = 0.3$, các chỉ số kiểm thử thể hiện sự dao động nhất định, phản ánh sự nhạy cảm của FedAvg khi xử lý dữ liệu không đồng nhất. Điều này cho thấy mặc dù FedAvg có thể hoạt động tốt trong môi trường phân phối dữ liệu phi đồng nhất, song vẫn còn hạn chế về độ ổn định và khả năng khai quật hóa.



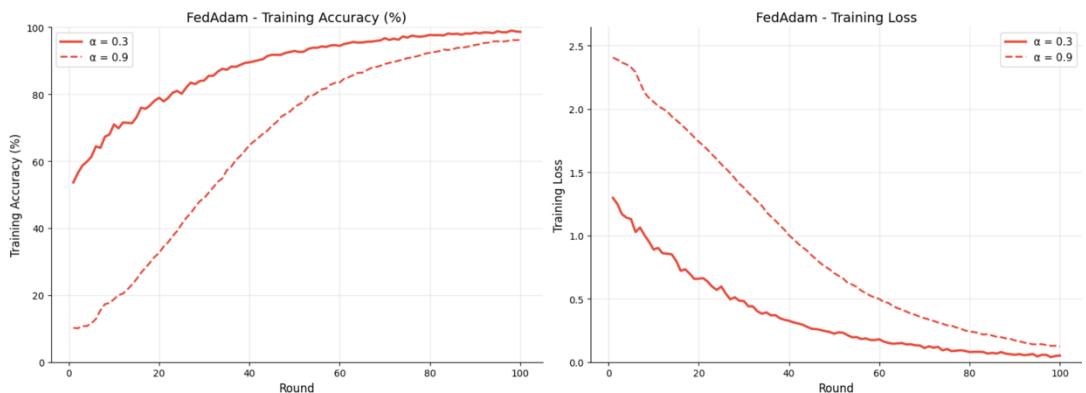
Hình 4-11: Kết quả huấn luyện của FedProx trên CIFAR10



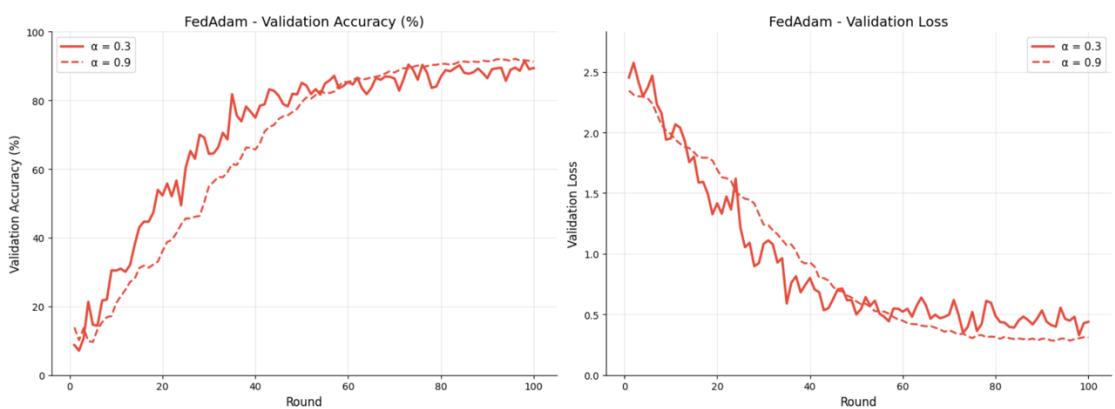
Hình 4-12: Kết quả đánh giá của FedProx trên CIFAR10

FedProx (Hình 4-11), (Hình 4-12) cải thiện rõ rệt hiệu suất huấn luyện và kiểm thử so với FedAvg. Ở $\alpha = 0.3$, độ chính xác huấn luyện đạt 95.33%, với hàm mất mát huấn luyện giảm xuống dưới 0.18 – tốt hơn đáng kể so với FedAvg. Với $\alpha = 0.9$, độ chính xác huấn luyện là 92.41%, và quá trình hội tụ diễn ra đều đặn, ít dao động.

Về mặt kiểm thử, FedProx thể hiện hiệu năng nổi bật với độ chính xác 89.79% ($\alpha = 0.3$) và 91.46% ($\alpha = 0.9$), là cao nhất trong cả ba thuật toán tại hai mức phân phối dữ liệu. Hàm mất mát kiểm thử ổn định và thấp, chứng tỏ hiệu quả của hàm ràng buộc proximal trong việc giảm sự sai lệch cục bộ và cải thiện khả năng tổng quát hóa. FedProx thể hiện rõ vai trò của một phương pháp được thiết kế dành riêng cho môi trường dữ liệu không đồng nhất.



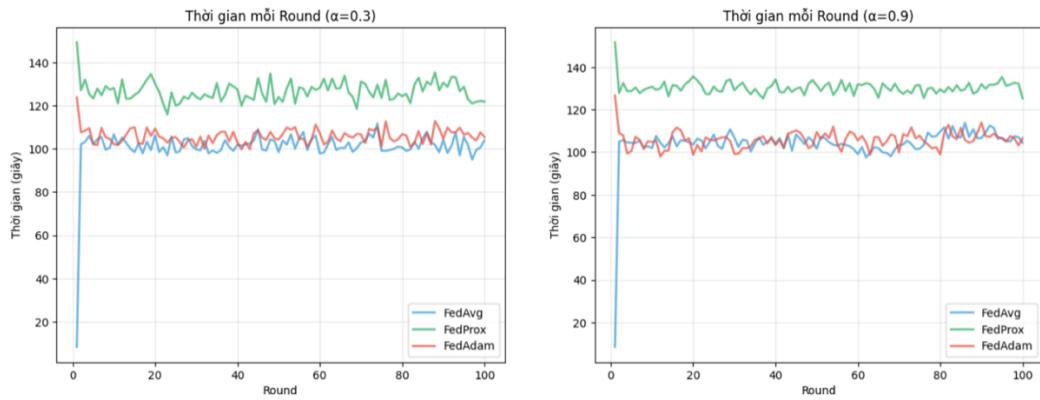
Hình 4-13: Kết quả huấn luyện của FedAdam trên CIFAR10



Hình 4-14: Kết quả đánh giá của FedAdam trên CIFAR10

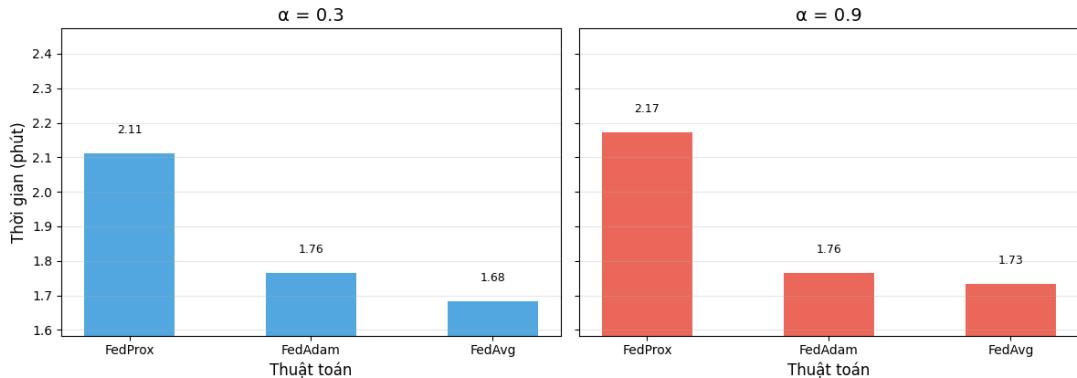
FedAdam đạt độ chính xác huấn luyện cao nhất trong cả ba thuật toán: 98.60% với $\alpha = 0.3$ và 96.19% với $\alpha = 0.9$. Thuật toán thể hiện tốc độ hội tụ nhanh – chỉ trong 30 vòng đầu tiên đã vượt mốc 90%. Hàm mất mát giảm nhanh và ổn định, chạm mức thấp nhất (< 0.15), phản ánh hiệu quả trong việc tối ưu hóa mô hình.

Trên tập kiểm thử, FedAdam đạt độ chính xác 89.38% ($\alpha = 0.3$) và 91.24% ($\alpha = 0.9$). Mặc dù kém hơn một chút so với FedProx ở $\alpha = 0.9$, kết quả kiểm thử vẫn rất cao và ổn định. Điều này khẳng định khả năng của FedAdam trong việc xử lý dữ liệu không đồng nhất và duy trì hiệu năng tốt ở cả hai mức phân phối dữ liệu. FedAdam đặc biệt phù hợp với các môi trường yêu cầu tối ưu hóa mạnh mẽ và ổn định.



Hình 4-15: So sánh thời gian chạy của 3 thuật toán trên CIFAR10

Cả hai biểu đồ (Hình 4-15) cho thấy rằng FedProx có thời gian huấn luyện mỗi vòng cao nhất, trung bình khoảng 120 giây. FedAvg và FedAdam đều tiêu tốn ít thời gian hơn (~105s với FedAdam, ~100s với FedAvg). Điều này phản ánh chi phí tính toán tăng do thành phần ràng buộc proximal của FedProx



Hình 4-16: So sánh thời gian trung bình của 3 thuật toán trên CIFAR10

FedProx tiếp tục tiêu tốn tổng thời gian nhiều nhất (2.11 phút với $\alpha = 0.3$ và 2.17 phút với $\alpha = 0.9$). FedAvg tiêu tốn thời gian ít nhất, trong khi FedAdam giữ mức trung bình. Việc tối ưu hóa sử dụng moment như trong FedAdam giúp giảm số vòng cần thiết để đạt độ chính xác cao, từ đó tiết kiệm thời gian hơn FedProx.

Qua thí nghiệm với ba thuật toán FedAvg, FedProx và FedAdam trên tập dữ liệu CIFAR-10, các thuật toán đều cho thấy những điểm mạnh và điểm yếu riêng:

- FedAdam đạt hiệu năng vượt trội cả ở huấn luyện và kiểm thử, với độ chính xác cao nhất (gần 100% train acc và ~90% val acc), tốc độ hội tụ

nhanh và ít dao động, đặc biệt hiệu quả trong môi trường dữ liệu phi đồng nhất ($\alpha = 0.3$).

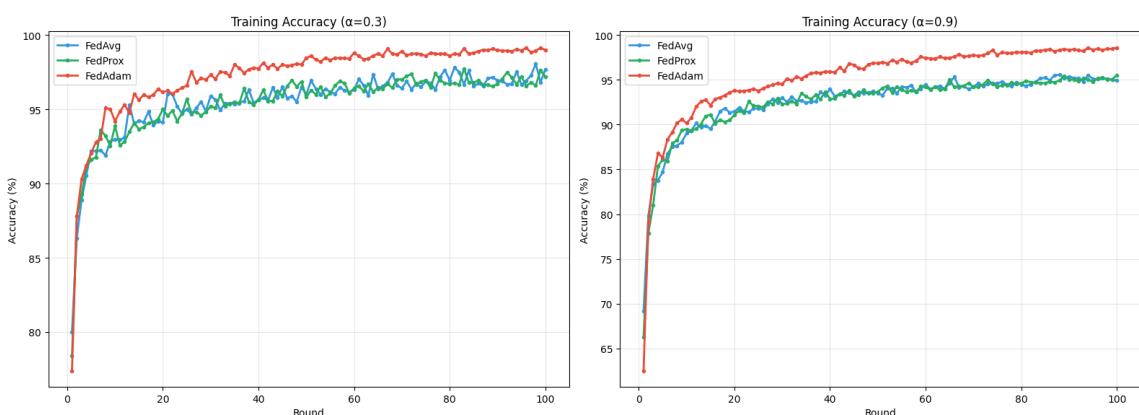
- FedProx thể hiện tốt khi dữ liệu không đồng nhất, giúp mô hình học ổn định và giảm dao động, mặc dù có thời gian huấn luyện lâu nhất.
- FedAvg đơn giản và nhanh nhất về thời gian, nhưng kém ổn định hơn khi dữ liệu phân phối không đều, đặc biệt là với $\alpha = 0.3$.

Tổng kết được là FedAdam là lựa chọn tối ưu nhất nếu yêu cầu chính xác và độ ổn định cao. FedProx phù hợp khi dữ liệu lệch phân phối, còn FedAvg thích hợp cho hệ thống giới hạn tài nguyên.

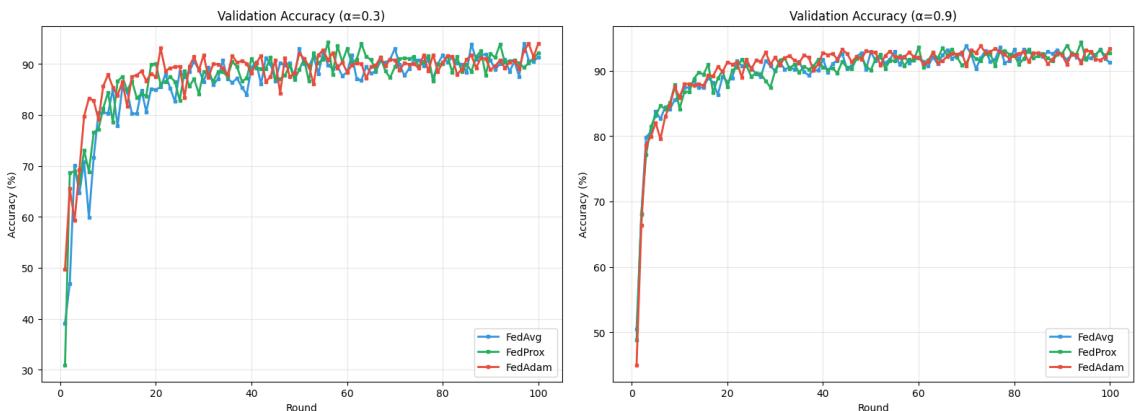
4.2. So sánh hiệu quả và phân tích độ chính xác

4.2.1. So sánh hiệu quả của 3 thuật toán trên từng tập dữ liệu

Nhằm nắm rõ hơn sự khác biệt của từng thuật toán giữa kết quả trong quá trình chạy mô phỏng việc huấn luyện và đánh giá mô hình. Sử dụng phương pháp trực quan hóa kết quả thông qua biểu đồ đường. Kết quả được trình bày trong một chuỗi các biểu đồ, với hai biểu đồ chính tương ứng với hai tập dữ liệu đã sử dụng. Mỗi biểu đồ chính lại bao gồm bốn biểu đồ con, thể hiện đồng thời ba đường biểu thị cho ba thuật toán FedAvg, FedProx và FedAdam. Cách sắp xếp này cho phép không chỉ so sánh hiệu suất giữa các thuật toán một cách trực quan mà còn theo dõi xu hướng hội tụ của chúng qua từng vòng lặp, từ đó đưa ra những nhận định chi tiết về ưu điểm và hạn chế của mỗi phương pháp



Hình 4-17: So sánh kết quả huấn luyện 3 thuật toán (Fashion-MNIST)

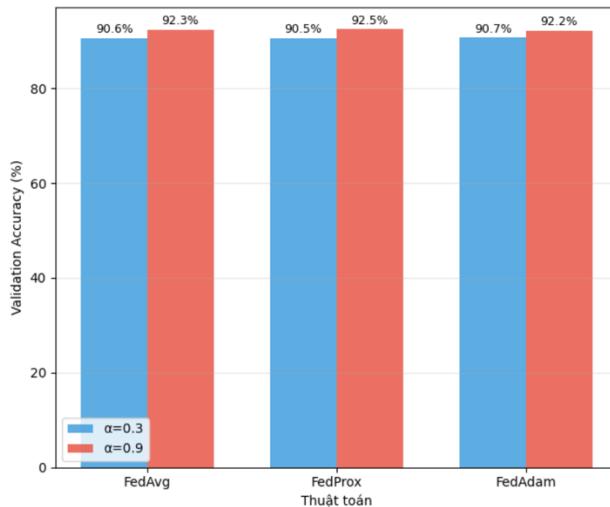


Hình 4-18: So sánh kết quả đánh giá 3 thuật toán (Fashion-MNIST)

Biểu đồ đường ở (Hình 4-17), (Hình 4-18) cho thấy, sự khác biệt về hiệu quả giữa ba chiến lược FedAvg, FedProx và FedAdam trong suốt quá trình huấn luyện trên tập dữ liệu Fashion-MNIST, xét theo hai mức độ không đồng nhất dữ liệu là $\alpha = 0.3$ và $\alpha = 0.9$. Ở cả hai trường hợp, thuật toán FedAdam duy trì lợi thế rõ rệt về độ chính xác trên tập huấn luyện, đạt mức cao nhất trong toàn bộ quá trình. Cụ thể, khi $\alpha = 0.3$, độ chính xác huấn luyện của FedAdam chạm ngưỡng gần 99%, vượt trội so với FedAvg (97.69%) và FedProx (97.20%). Tương tự, khi $\alpha = 0.9$, FedAdam vẫn đạt 98.58% so với FedProx (95.50%) và FedAvg (94.90%).

Đối với tập validation, sự khác biệt cũng được phản ánh rõ nét, đặc biệt trong giai đoạn hội tụ ổn định. Khi $\alpha = 0.3$, FedAdam đạt độ chính xác cuối cùng 93.99%, cao hơn đáng kể so với FedProx (92.11%) và FedAvg (91.33%). Với $\alpha = 0.9$, FedAdam tiếp tục giữ vị trí dẫn đầu với 93.32%, theo sau là FedProx (92.66%) và FedAvg (91.24%).

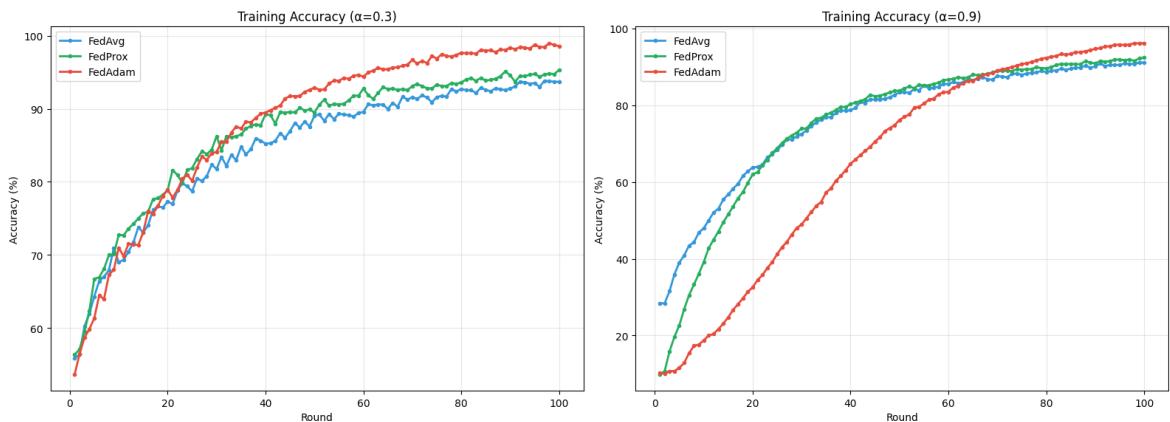
Tuy nhiên, do đường biểu diễn ở biểu đồ validation có thể dao động tại một số thời điểm, việc chỉ đánh giá theo giá trị cuối cùng chưa phản ánh đầy đủ đặc trưng mô hình. Do đó, ở phần tiếp theo sẽ sử dụng giá trị trung bình của 20 vòng cuối cùng để có cái nhìn tổng quát, ổn định và chính xác hơn về hiệu quả thực tế của từng chiến lược.



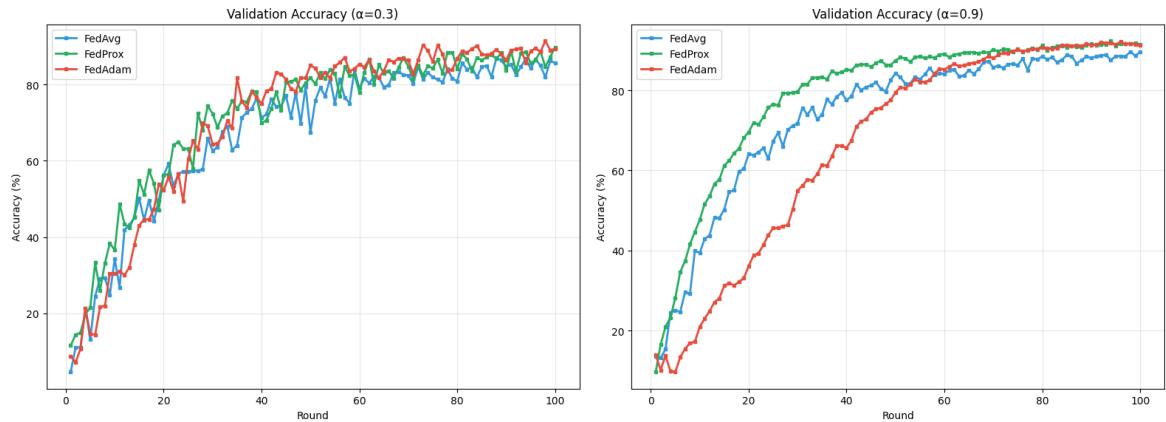
Hình 4-19: Độ chính xác trung bình 20 vòng cuối (Fashion-MNIST)

Biểu đồ (Hình 4-19) trình bày độ chính xác trung bình trên tập validation trong 20 vòng cuối cùng của quá trình huấn luyện, từ đó loại bỏ ảnh hưởng của dao động ngẫu nhiên và phản ánh ổn định thực sự của từng thuật toán. Trong cả hai mức độ không đồng nhất:

- FedProx đạt kết quả cao nhất với 92.5% ($\alpha = 0.9$) và 92.3% ($\alpha = 0.3$).
- FedAdam theo sát ngay sau với độ chính xác lần lượt là 92.2% và 90.7%.
- FedAvg đạt kết quả thấp hơn, dao động quanh mức 90.5–90.6%.



Hình 4-20: So sánh kết quả huấn luyện 3 thuật toán (CIFAR10)



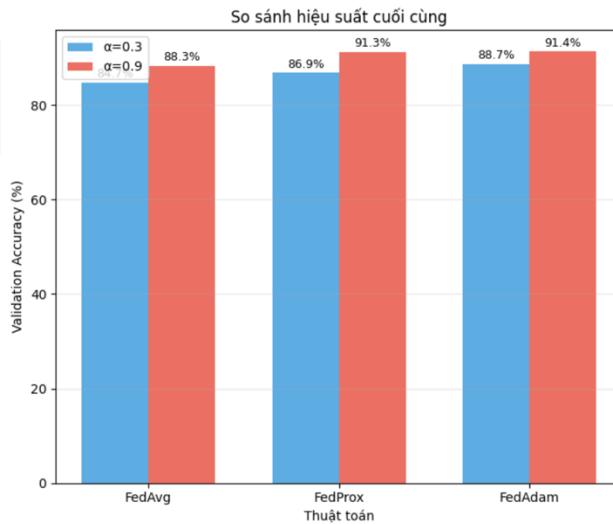
Hình 4-21: So sánh kết quả đánh giá 3 thuật toán (CIFAR10)

Quan sát từ (Hình 4-20), (Hình 4-21) cho thấy rõ sự khác biệt trong tốc độ hội tụ và độ chính xác huấn luyện giữa ba thuật toán. Ở mức $\alpha = 0.3$, FedAdam tăng trưởng đều đặn và nhanh chóng đạt gần 99% độ chính xác trên tập huấn luyện, với mức Final Train Acc đạt 98.60%. FedProx giữ nhịp hội tụ ổn định, đạt 95.33%, trong khi FedAvg thấp hơn một chút ở 93.70%.

Trên tập validation, các đường biểu diễn cho thấy cả ba thuật toán đều đạt hiệu quả tiệm cận sau khoảng 60 vòng. Tuy nhiên, FedProx và FedAdam có độ chính xác cao hơn, phản ánh qua các kết quả cuối cùng lần lượt là 89.79% và 89.38%, so với 85.71% của FedAvg. Đặc biệt, sự chênh lệch rõ ràng về khả năng tổng quát hóa có thể quan sát được qua khoảng cách giữa đường train và val của FedAvg, vốn lớn hơn đáng kể so với hai thuật toán còn lại.

Khi $\alpha = 0.9$, biểu đồ thể hiện điểm khác biệt rõ hơn: FedAdam hội tụ chậm ở giai đoạn đầu (do tham số được tối ưu cho $\alpha = 0.3$), nhưng vượt lên mạnh mẽ ở cuối quá trình, đồng thời đạt Final Train Acc lên đến 96.19%. Dù vậy, FedProx vẫn là thuật toán dẫn đầu về hiệu suất kiểm tra, đạt Val Acc là 91.46%, cao nhất trong toàn bộ thí nghiệm. Đường biểu diễn của FedProx cũng mượt và ổn định hơn, cho thấy tính nhất quán cao trong điều kiện dữ liệu không đồng nhất. Ngược lại, FedAvg tiếp tục bị ảnh hưởng rõ bởi phân phối dữ liệu, thể hiện qua mức Val Acc chỉ đạt 89.56%, thấp hơn cả FedProx và FedAdam.

Để đánh giá hiệu quả một cách toàn diện và tránh bị ảnh hưởng bởi các dao động ngẫu nhiên giữa các vòng, biểu đồ Hình 3-20 tổng hợp độ chính xác trung bình trên tập validation trong 20 vòng cuối cùng.



Hình 4-22: Độ chính xác trung bình 20 vòng cuối (CIFAR10)

Biểu đồ (Hình 4-22) minh họa độ chính xác trung bình trên tập validation trong 20 vòng cuối cùng, giúp làm rõ xu hướng hiệu quả của các chiến lược sau khi mô hình đạt trạng thái ổn định. Trong cả hai mức α , FedProx và FedAdam đều đạt hiệu suất cao hơn FedAvg. Khi $\alpha = 0.3$, FedProx dẫn đầu với 88.3%, nhỉnh hơn FedAdam (88.7%) và vượt trội so với FedAvg (84.7%). Trong khi đó, khi dữ liệu trở nên không đồng nhất hơn ($\alpha = 0.9$), FedAdam và FedProx gần như tương đương, đạt lần lượt 91.4% và 91.3%, trong khi FedAvg vẫn ở mức thấp hơn là 86.9%.

Tổng hợp lại tất cả, có thể nhận thấy rằng FedAdam thường đạt độ chính xác huấn luyện cao nhất, trong khi FedProx lại giữ ưu thế rõ rệt về độ chính xác khi đánh giá, đặc biệt trong môi trường không đồng nhất cao. Ngược lại, FedAvg dù ổn định nhưng thể hiện rõ sự suy giảm hiệu quả khi đổi mới với mức độ không đồng nhất tăng lên.

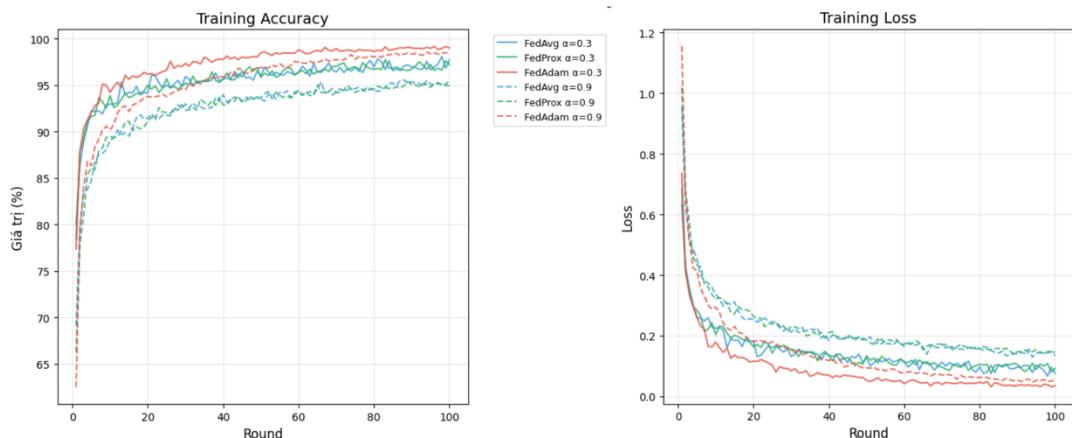
Từ những kết quả trên, có thể rút ra nhận định sơ bộ rằng FedAdam là chiến lược phù hợp khi ưu tiên tốc độ hội tụ và độ chính xác huấn luyện, còn FedProx là

lựa chọn đáng cân nhắc khi cần duy trì hiệu suất kiểm tra cao trong điều kiện dữ liệu phân tán và không đồng nhất.

4.2.2. So sánh độ hiệu quả trên dữ liệu không đồng nhất

Để đánh giá toàn diện khả năng của các thuật toán trong môi trường thực tế, phần này sẽ đi sâu vào việc phân tích hiệu suất của FedAvg, FedProx và FedAdam trên hai mức độ không đồng nhất của dữ liệu, được thể hiện qua tham số $\alpha=0.3$ (độ không đồng nhất cao) và $\alpha=0.9$ (độ không đồng nhất thấp).

Kết quả được trình bày thông qua một loạt các biểu đồ thể hiện quá trình của độ chính xác và hàm mất mát trên cả tập huấn luyện và tập validation, áp dụng cho hai tập dữ liệu Fashion-MNIST và CIFAR10. Mỗi biểu đồ chính bao gồm hai biểu đồ con, mỗi biểu đồ con lại chứa sáu đường biểu thị. Sáu đường này tương ứng với ba thuật toán đã nêu, mỗi thuật toán được thử nghiệm trên hai giá trị α khác nhau. Cách trình bày này cho phép người đọc so sánh trực quan hiệu suất của mỗi thuật toán khi đối mặt với các mức độ phân tán dữ liệu khác nhau, nhằm rút ra kết luận về khả năng thích ứng trên các loại dữ liệu không đồng nhất của từng phương pháp.

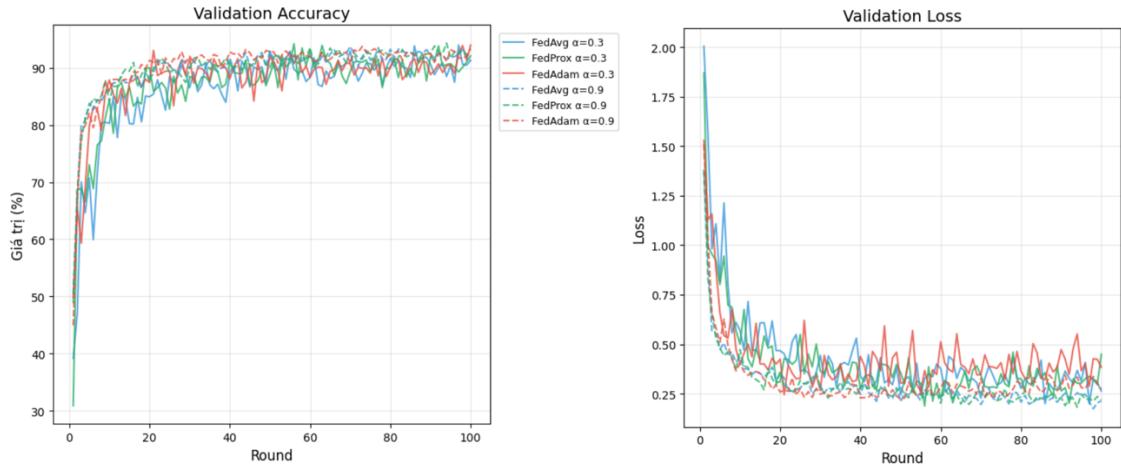


Hình 4-23: Hiệu quả huấn luyện ở độ alpha khác nhau (Fashion-MNIST)

Biểu đồ huấn luyện (Hình 4-23) trên tập Fashion-MNIST, cho thấy FedAdam luôn đạt độ chính xác huấn luyện cao hơn rõ rệt so với FedAvg và FedProx ở cả hai mức α . Đặc biệt, sự chênh lệch càng rõ ở $\alpha = 0.3$ khi FedAdam nhanh chóng đạt gần 99% sau khoảng 40 vòng, trong khi hai thuật toán còn lại dao động quanh mức 96–

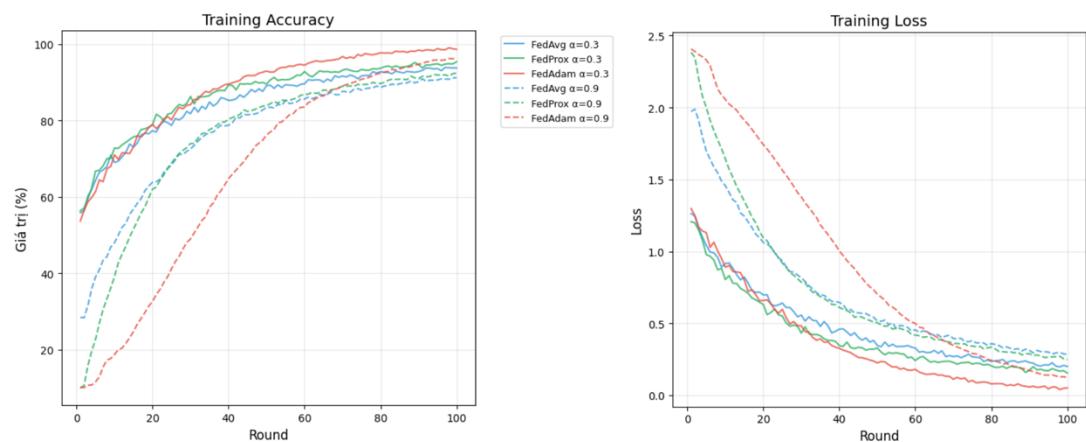
Chương 4: Thực nghiệm chương trình

97%. Kết quả huấn luyện này đồng thời đi kèm với mức mất mát thấp hơn ổn định của FedAdam, cho thấy mô hình không chỉ học tốt mà còn hội tụ ổn định hơn



Hình 4-24: Hiệu quả đánh giá ở độ alpha khác nhau (Fashion-MNIST)

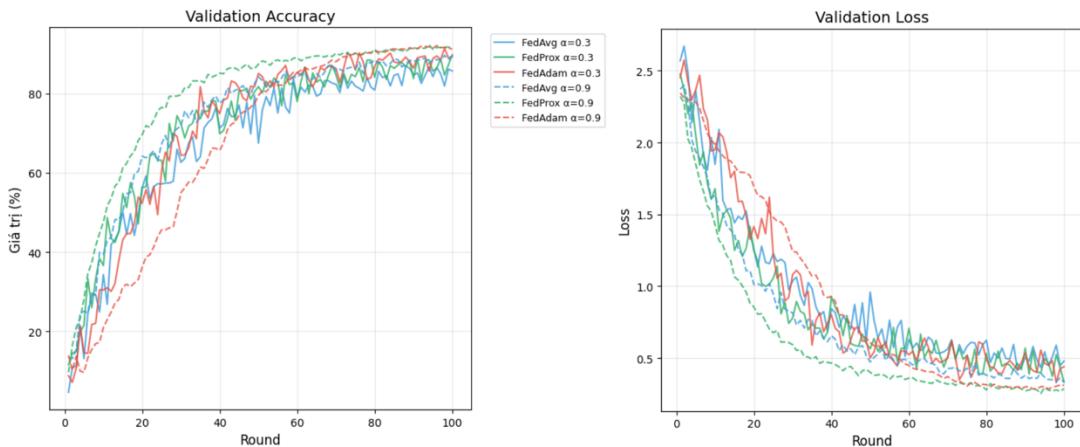
Biểu đồ đánh giá (Hình 4-24) so với (Hình 4-23) đã có sự khác biệt rõ, các thuật toán lại thể hiện hiệu suất tiệm cận gần nhau hơn. Cả ba thuật toán đều đạt độ chính xác đánh giá dao động quanh ngưỡng 91–93%, ít có sự khác biệt lớn về mặt tổng thể. Mặc dù FedAdam có xu hướng nhỉnh hơn một chút, nhưng độ dao động của loss trong giai đoạn cuối lại cho thấy một mức độ bất ổn nhẹ so với FedProx. Điều này gợi ý rằng FedAdam dù học nhanh nhưng có thể dễ bị ảnh hưởng bởi đặc tính dữ liệu không đồng nhất trong quá trình đánh giá.



Hình 4-25: So sánh hiệu quả huấn luyện ở độ alpha khác nhau (CIFAR10)

Quan sát (Hình 4-25), có thể thấy rằng khi $\alpha = 0.3$ (mức không đồng nhất cao), FedAdam đạt độ chính xác huấn luyện vượt trội so với FedAvg và FedProx, đồng

thời duy trì mức măt măt thấp hơn xuyên suốt quá trình huấn luyện. Ngược lại, khi α tăng lên 0.9, lợi thế ban đầu của FedAdam bị thu hẹp, cả ba thuật toán đều hội tụ đến các mức độ chính xác tương tự, cho thấy mức phân tán thấp giúp cân bằng hiệu quả huấn luyện giữa các phương pháp.



Hình 4-26: So sánh hiệu quả đánh giá ở độ alpha khác nhau (CIFAR10)

Biểu đồ (Hình 4-26), FedProx cho thấy khả năng tổng quát hóa tốt hơn khi $\alpha = 0.9$, vượt trội nhẹ so với hai thuật toán còn lại. Tuy nhiên, khi $\alpha = 0.3$, sự dao động trong đường chính xác của cả ba thuật toán nhiều hơn, đặc biệt là với FedAvg. Điều này phản ánh rằng khi dữ liệu bị phân mảnh nặng, khả năng thích nghi của từng thuật toán trở nên quan trọng hơn. Đồng thời, mức măt măt đánh giá ở $\alpha = 0.3$ cũng cao và dao động hơn đáng kể so với $\alpha = 0.9$, đặc biệt là ở giai đoạn đầu huấn luyện, cho thấy thách thức lớn hơn trong việc đồng bộ hóa mô hình toàn cục khi dữ liệu cục bộ thiếu tính đại diện.

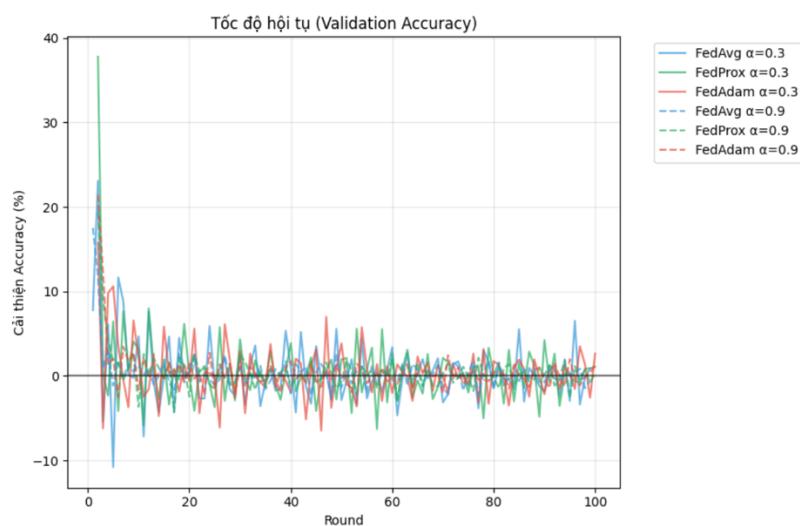
So sánh tổng quan giữa hai mức α hoạt động trên hai tập dữ liệu Fashion-MNIST và CIFAR10, một điểm đáng chú ý đã được ghi nhận. Trong quá trình huấn luyện cục bộ (train), mô hình dường như đạt độ chính xác cao hơn khi dữ liệu có độ không đồng nhất cao ($\alpha = 0.3$). Điều này cho thấy dữ liệu càng non-IID thì mô hình học càng chính xác trên tập dữ liệu cục bộ của từng client.

Tuy nhiên, khi đánh giá trên tập dữ liệu validation tổng thể, kết quả lại cho thấy một xu hướng ngược lại. Các mô hình được huấn luyện trên dữ liệu có độ đồng nhất cao (α cao) lại đạt hiệu suất tổng quát tốt hơn so với các mô hình được huấn luyện

trên dữ liệu non-IID. Điều này nhấn mạnh rằng, mặc dù mô hình cục bộ có thể "ghi nhớ" dữ liệu của mình tốt hơn trong môi trường non-IID.

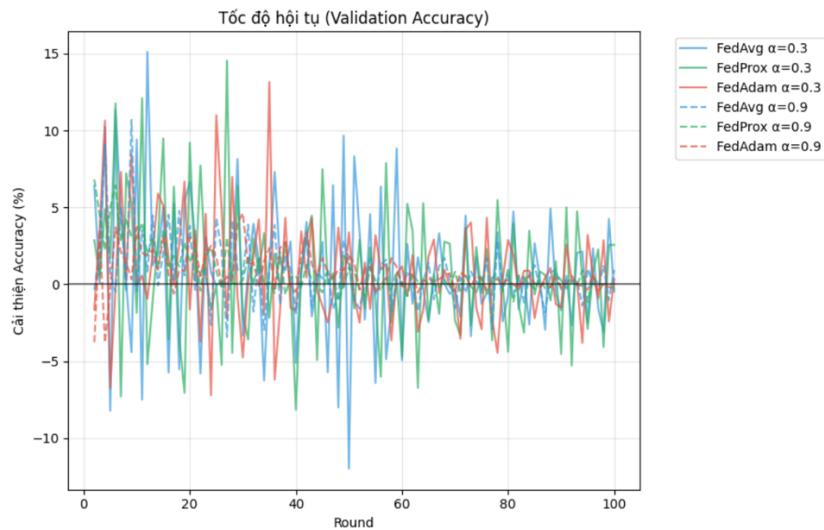
4.2.3. So sánh tốc độ hội tụ trên mỗi vòng

Tốc độ hội tụ là một yếu tố then chốt, phản ánh khả năng của thuật toán trong việc đạt được hiệu suất tối ưu một cách nhanh chóng. Phần này sẽ sử dụng các biểu đồ để trực quan hóa diễn biến của các chỉ số hiệu suất, từ đó so sánh sự khác biệt trong hành vi hội tụ của ba thuật toán FedAvg, FedProx và FedAdam trong từng độ non-IID trên hai tập dữ liệu.



Hình 4-27: Tốc độ hội tụ của 3 thuật toán (Fashion-MNIST)

Biểu đồ tốc độ hội tụ trên Fashion-MNIST (Hình 4-27), cho thấy các thuật toán có sự cải thiện độ chính xác (Validation Accuracy) rất nhanh chóng, với các đỉnh tăng cao đột biến. Sau đó, tốc độ hội tụ giảm dần và độ chính xác bắt đầu dao động quanh một giá trị ổn định hơn.



Hình 4-28: Tốc độ hội tụ của 3 thuật toán (CIFAR10)

Biểu đồ tốc độ hội tụ trên CIFAR10 (Hình 4-28), cho thấy sự biến động lớn hơn nhiều trong suốt 100 vòng huấn luyện. Điều này có thể là do CIFAR10 là một tập dữ liệu phức tạp hơn, đòi hỏi nhiều vòng lặp hơn để mô hình ổn định.

Kết luận từ hai biểu đồ (Hình 4-27), (Hình 4-28) ba thuật toán FedAvg, FedProx và FedAdam đều thể hiện tốc độ hội tụ ban đầu nhanh trên tập dữ liệu Fashion-MNIST, nhưng sau đó không có sự cải thiện đáng kể. Trên tập dữ liệu phức tạp hơn là CIFAR10, tất cả các thuật toán đều cho thấy sự biến động lớn, cho thấy mô hình chưa hội tụ ổn định sau 100 vòng. Tuy nhiên, dựa vào hai biểu đồ này, rất khó để đưa ra kết luận rõ ràng về thuật toán nào có tốc độ hội tụ vượt trội hơn so với hai thuật toán còn lại. Các đường biểu thị đều có xu hướng và mức độ biến động tương tự nhau, bất kể giá trị của tham số α .

4.2.4. Hiệu quả của 3 thuật toán so với thời gian chạy

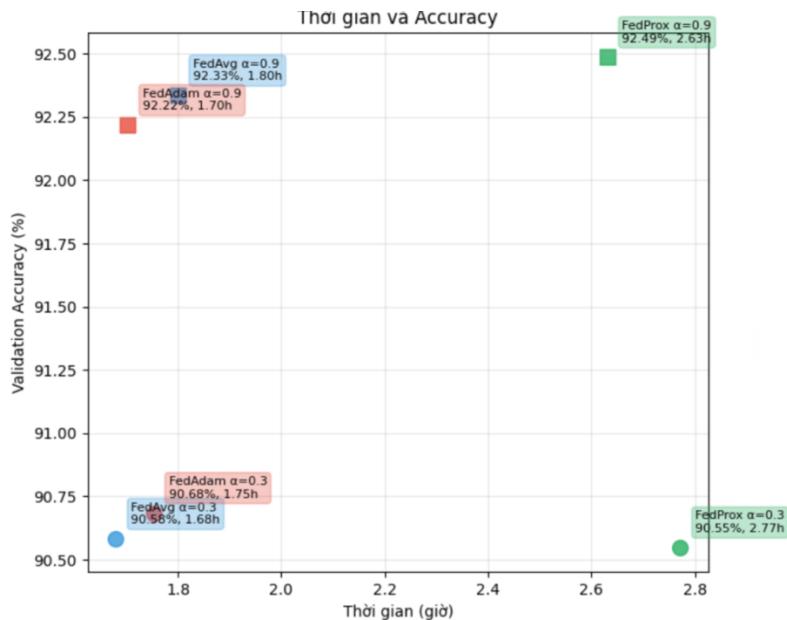
So sánh trên hiệu quả của thuật toán trên độ chính xác ở tập đánh giá cùng với thời gian đạt được độ chính xác đó của từng thuật toán trên các độ alpha khác nhau của hai tập dữ liệu sẽ cung cấp cái nhìn tổng quan về sự đánh đổi giữa hiệu suất của mô hình và chi phí tính toán cần thiết.

Để có một đánh giá khách quan và công bằng về hiệu quả của từng thuật toán, ở việc thực hiện so sánh giữa độ chính xác và thời gian thực thi. Thay vì chỉ dựa vào

Chương 4: Thực nghiệm chương trình

một giá trị cuối cùng, phân tích này tập trung vào hiệu suất trung bình trong 20 vòng huấn luyện cuối cùng.

Các kết quả được trực quan hóa trên các biểu đồ phân tán, với trực hoành thể hiện Tổng thời gian thực thi (tính bằng giờ) và trực tung thể hiện Độ chính xác trung bình (Average Validation Accuracy). Mỗi điểm trên biểu đồ đại diện cho kết quả của một thuật toán trong kết quả.

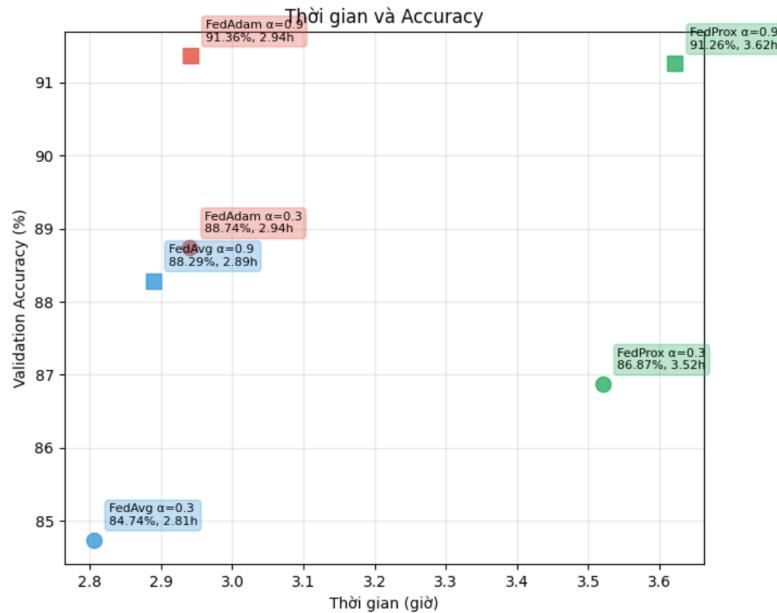


Hình 4-29: Độ chính xác so với thời gian (Fashion-MNIST)

Biểu đồ ở (Hình 4-29), cho thấy hiệu quả khi dữ liệu không đồng có độ α là 0.9 thì các thuật toán FedAvg và FedAdam đều thể hiện sự cân bằng tốt giữa hiệu suất và chi phí thời gian. FedAdam đạt độ chính xác 92.22% trong thời gian 1.70 giờ, là thuật toán nhanh nhất trong nhóm. Mặc dù FedAvg đạt độ chính xác cao hơn một chút (92.33%), nhưng lại mất nhiều thời gian hơn (1.80 giờ). Ngược lại, FedProx, mặc dù đạt độ chính xác cao nhất (92.49%), lại có thời gian thực thi lâu hơn đáng kể (2.63 giờ), cho thấy thuật toán này phải đánh đổi tốc độ để đạt hiệu suất cao.

Còn trong môi trường dữ liệu không đồng nhất cao ($\alpha=0.3$) FedAdam vẫn tiếp tục thể hiện sự vượt trội về mặt tốc độ, chỉ mất 1.75 giờ để hoàn thành huấn luyện. Độ chính xác của thuật toán này đạt 90.68%, ngang bằng với FedAvg (90.56% trong

1.68 giờ) và FedProx (90.55% trong 2.77 giờ) - đây là điểm bất ngờ khi FedAdam không phải trả giá bằng hiệu suất như FedProx.



Hình 4-30: Độ chính xác so với thời gian trong 20 vòng cuối (CIFAR10)

Biểu đồ ở (Hình 4-30), cho thấy hiệu quả khi dữ liệu không đồng có độ α là 0.9 thì các thuật toán khi đối mặt với tập dữ liệu phức tạp hơn, FedProx tiếp tục dẫn đầu về độ chính xác (91.26%) nhưng cũng là thuật toán chậm nhất (3.62 giờ). FedAdam thể hiện một sự đánh đổi hiệu quả là đạt được độ chính xác rất cao (91.36%) chỉ với 2.94 giờ, nhanh hơn đáng kể so với FedProx. Điều này cho thấy khả năng tổng quát hóa của FedAdam là rất tốt, không hề thua kém FedProx. FedAvg đạt độ chính xác thấp hơn (88.29%) nhưng lại có thời gian thực thi nhanh nhất (2.89 giờ).

Trong điều kiện dữ liệu khắc nghiệt nhất, tất cả các thuật toán đều bị ảnh hưởng. Tuy nhiên, FedAdam vẫn giữ được hiệu suất tương đối cao (88.74%), vượt trội so với FedAvg (84.74%). Đáng chú ý, FedAdam chỉ mất 2.94 giờ để hoàn thành, trong khi FedProx (86.87%) lại mất tới 3.52 giờ.

Từ những nhận xét trên, cho thấy trên tập dữ liệu đơn giản như Fashion-MNIST hoặc phức tạp như CIFAR10, FedAdam đã chứng tỏ khả năng ưu việt. Thuật toán này không chỉ duy trì độ chính xác cao mà còn tối ưu hóa thời gian thực thi, mang lại sự cân bằng tốt nhất giữa hiệu suất và chi phí tính toán trong cả hai kịch bản.

4.3. Nhận xét

Dựa trên toàn bộ kết quả phân tích và so sánh thực nghiệm đã trình bày ở phần 3.6, 3.7, có thể rút ra một số nhận định sâu sắc về hiệu quả và đặc điểm của ba thuật toán FedAvg, FedProx và FedAdam, đặc biệt là trong bối cảnh dữ liệu không đồng nhất (non-IID).

4.3.1. Đánh giá tổng quan về từng thuật toán

FedAvg đã đem lại nhiều ưu điểm về hiệu quả thời gian nhờ vào cơ chế trung bình hóa đơn giản. Trên cả hai tập dữ liệu, FedAvg thường là một trong những thuật toán có tốc độ chạy nhanh nhất. Tuy nhiên, hiệu quả của FedAvg giảm rõ rệt khi đối mặt với dữ liệu non-IID. Độ chính xác trên tập validation của FedAvg luôn thấp hơn so với hai thuật toán còn lại. Do dữ liệu non-IID khiến việc trung bình hóa trở nên kém hiệu quả hơn, dẫn đến khả năng tổng quát hóa mô hình toàn cục bị suy giảm.

Trong khi đó, FedProx đã chứng minh được khả năng giải quyết vấn đề non-IID một cách hiệu quả. Nhờ vào việc bổ sung thành phần điều chỉnh proximal, giúp kiểm soát sự sai lệch của các cập nhật mô hình cục bộ, từ đó cải thiện tính ổn định và duy trì độ chính xác cao nhất trên tập validation trong hầu hết các kịch bản. Tuy nhiên, sự ổn định này phải đánh đổi bằng chi phí thời gian. Kết quả thực nghiệm cho thấy FedProx luôn là thuật toán chậm nhất. Thành phần điều chỉnh proximal, mặc dù hiệu quả, lại làm tăng chi phí tính toán cục bộ và làm chậm tốc độ hội tụ tổng thể.

Cuối cùng, FedAdam thể hiện sự cân bằng vượt trội giữa hiệu suất và thời gian. Thuật toán này không chỉ đạt được độ chính xác rất cao, tiệm cận hoặc thậm chí vượt qua FedProx, mà còn duy trì tốc độ thực thi nhanh, gần như tương đương với FedAvg. Khả năng thích nghi của bộ tối ưu hóa Adam giúp FedAdam nhanh chóng đạt được độ chính xác cao trên tập huấn luyện và ổn định hiệu suất trên tập validation, ngay cả trong môi trường non-IID. Tuy nhiên, việc tối ưu hóa FedAdam không hề đơn giản. Hiệu quả của nó phụ thuộc rất lớn vào việc lựa chọn tham số học (learning rate) và các tham số của bộ tối ưu hóa Adam.

4.3.2. Tầm quan trọng của việc lựa chọn tham số

Kết quả thực nghiệm đã nhấn mạnh tầm quan trọng của việc điều chỉnh tham số cho từng thuật toán, đặc biệt là μ trong FedProx và learning rate trong FedAdam.

Tham số μ của FedProx: Tham số này quyết định mức độ "điều chỉnh" sự sai lệch của mô hình cục bộ. Một giá trị μ phù hợp sẽ giúp mô hình ổn định hơn, nhưng nếu quá lớn có thể làm giảm tốc độ học.

Tham số học của FedAdam: Tương tự như Adam trong học tập trung, việc chọn learning rate thích hợp là yếu tố sống còn để FedAdam có thể phát huy tối đa hiệu quả. Một learning rate không phù hợp có thể khiến quá trình học bị chậm hoặc thậm chí là không hội tụ.

4.3.3. Lựa chọn chiến lược phù hợp cho từng bài toán

Từ những phân tích trên, ta có thể rút ra một kết luận quan trọng là không có một thuật toán nào là tối ưu cho mọi trường hợp. Việc lựa chọn chiến lược phụ thuộc vào yêu cầu cụ thể của bài toán. FedAvg là lựa chọn tốt khi ưu tiên tốc độ và chi phí thấp, đặc biệt nếu dữ liệu có tính đồng nhất cao. FedProx là chiến lược đáng cân nhắc khi độ ổn định và độ chính xác cao là ưu tiên hàng đầu, ngay cả khi phải chấp nhận thời gian huấn luyện lâu hơn. FedAdam nổi lên như một lựa chọn cân bằng, phù hợp cho các bài toán yêu cầu hiệu suất cao và tốc độ hội tụ nhanh. Tuy nhiên, việc điều chỉnh tham số để khai thác tối đa sức mạnh của thuật toán này cũng là một cản trở不小的 của thuật toán này.

Tóm lại, việc hiểu rõ bản chất, ưu và nhược điểm của từng thuật toán sẽ giúp việc đưa ra quyết định sáng suốt, lựa chọn chiến lược FL phù hợp nhất với điều kiện dữ liệu khác nhau và yêu cầu của hệ thống, từ đó tối ưu hóa hiệu quả của quá trình học liên kết.

CHƯƠNG 5: KẾT LUẬN

5.1. Tóm tắt kết quả đạt được

Trong đề tài này, ba thuật toán học liên kết phổ biến là FedAvg, FedProx và FedAdam đã được lựa chọn và thực nghiệm nhằm đánh giá hiệu suất hoạt động trong bối cảnh dữ liệu không đồng nhất (non-IID), sử dụng hai tập dữ liệu là CIFAR-10 và Fashion-MNIST để đánh giá đem lại nhiều gốc nhìn nhất, đánh giá chính xác và khách quan nhất về ba thuật toán được chọn.

Các kết quả được thực nghiệm và so sánh với nhiều tiêu chí khác nhau gồm các tiêu chí: độ chính xác huấn luyện (train accuracy), độ chính xác kiểm thử (Validation accuracy), hàm mất mát (loss) và tốc độ, cùng một số tiêu chí hiệu năng khác, các kết quả được thực nghiệm trong môi trường công bằng, chuẩn và chính xác, đem lại những kết quả chính xác nhất và đúng nhất.

Các kết quả được thực nghiệm phù hợp với lí thuyết chuẩn của từng thuật toán và các bài báo liên quan về khả năng của các thuật toán FedAvg, FedProx và FedAdam.

Các thuật toán được triển khai trên nền tảng thư viện Flower, với kiến trúc mô phỏng nhiều client trong môi trường không đồng nhất về dữ liệu và điều kiện huấn luyện. Các tham số được chọn mang tính chất tương đối phù hợp với thực thế, đem lại những đánh giá khách quan và có giá trị thực tiễn.

Với tập dữ liệu Fashion-MNIST, thuật toán FedAdam nổi bật với khả năng đạt độ chính xác huấn luyện và kiểm thử cao nhất, lần lượt là 98.99% và 93.99% khi $\alpha = 0.3$. Trong khi đó, FedProx thể hiện khả năng xử lý tốt dữ liệu phân tán và giảm dao động đáng kể, với độ chính xác kiểm thử lên đến 92.66% khi $\alpha = 0.9$. FedAvg tuy đơn giản và có tốc độ hội tụ nhanh nhưng cho thấy độ dao động lớn hơn và độ chính xác thấp hơn so với hai phương pháp còn lại trong môi trường dữ liệu không đồng nhất.

Với tập dữ liệu CIFAR-10, một bài toán thị giác máy tính phức tạp hơn, FedProx và FedAdam tiếp tục thể hiện ưu thế rõ rệt. FedProx đạt 91.46% độ chính xác kiểm thử ở $\alpha = 0.9$ – cao nhất trong cả ba thuật toán nhờ vào thành phần ràng buộc proximal giúp ổn định quá trình cập nhật mô hình. FedAdam duy trì tốc độ hội tụ vượt trội và đạt độ chính xác huấn luyện cao nhất là 98.60% ($\alpha = 0.3$). FedAvg, mặc dù vẫn giữ được hiệu suất khá ổn định, nhưng thể hiện rõ hạn chế khi xử lý dữ liệu phân phối không đều, với độ chính xác kiểm thử dao động và hàm mất mát không ổn định.

Trên tập dữ liệu Fashion-MNIST và CIFAR-10, ba thuật toán FedAvg, FedProx và FedAdam đều cho thấy khả năng huấn luyện tốt, nhưng sự khác biệt rõ rệt xuất hiện khi dữ liệu được phân phối phi đồng nhất. FedAdam consistently đạt độ chính xác huấn luyện cao nhất (lên tới 98.99% với FMNIST và 98.60% với CIFAR-10 khi $\alpha = 0.3$), đồng thời duy trì hiệu suất kiểm thử ổn định. Kết quả này phản ánh khả năng tối ưu hóa mạnh mẽ và thích ứng tốt của FedAdam trong môi trường Non-IID.

Trong khi đó, FedProx cải thiện tính ổn định so với FedAvg nhờ thêm ràng buộc proximal, đặc biệt hiệu quả ở mức phân phối đồng đều hơn ($\alpha = 0.9$). FedAvg, tuy đơn giản và dễ triển khai, lại kém hiệu quả trong môi trường dữ liệu phi đồng nhất do dao động mạnh và tốc độ hội tụ không ổn định.

Từ các kết quả thực nghiệm, có thể kết luận rằng FedAdam là thuật toán hiệu quả nhất khi làm việc trong điều kiện Non-IID, vượt trội cả về độ chính xác và độ ổn định so với FedAvg và FedProx.

5.2. Hạn chế của quá trình huấn luyện

Trong quá trình thực hiện việc nhóm có xảy ra nhiều vấn đề liên quan đến phần cứng, học liên kết yêu cầu cần có phần cứng mạnh để có thể thực hiện sâu và chi tiết, từ đây mới cho ra được những đánh giá tốt nhất và chính xác nhất.

Việc FedAdam cần có phần cứng đủ mạnh để tiến hành thực nghiệm tìm kiếm bộ tham số phù hợp là vấn đề lớn và quan trọng do thời gian huấn luyện lâu với số lượng máy biến và số vòng huấn luyện nhiều.

Các tham số chưa được điều chỉnh ở mức cao vì không đủ phần cứng và thời gian huấn luyện. Trung bình mỗi lần tìm bộ tham số các máy cần phải có trung bình là 10 tiếng để hoàn thành mỗi lần tìm kiếm với tham số mới. Yêu cầu thực nghiệm nhiều lần với nhiều bộ để tìm ra bộ tham số tốt nhất và tối ưu nhất.

Chưa có điều kiện để nghiên cứu sâu về các kỹ thuật tối ưu nâng cao hoặc cải tiến cấu trúc thuật toán để nâng cao tốc độ hội tụ hoặc giảm độ dao động trong quá trình huấn luyện. Việc này đòi hỏi nhiều thời gian, tài nguyên, cũng như hiểu biết chuyên sâu hơn về lý thuyết tối ưu hóa trong môi trường phân tán.

Khả năng mở rộng và tính linh hoạt của mô hình trong điều kiện thực tế vẫn chưa được khai thác đầy đủ. Thí nghiệm được tiến hành trong môi trường mô phỏng với số lượng client cố định, dữ liệu được phân chia trước. Tuy nhiên, trong thực tế, client thường tham gia không đồng bộ, mất kết nối, hoặc có năng lực tính toán khác nhau, gây ra thách thức lớn hơn cho quá trình huấn luyện liên kết.

Chưa nghiên cứu và phát triển khả năng bảo mật và riêng tư, dù không phải là trọng tâm của nghiên cứu này, nhưng là vấn đề không thể bỏ qua trong các ứng dụng FL thực tế. Việc chia sẻ tham số mô hình giữa server và client vẫn có thể bị tấn công nếu không có các biện pháp bảo mật.

Bên cạnh đó, nghiên cứu cũng chưa có điều kiện để so sánh tốc độ thực thi của các thuật toán trên nhiều loại tài nguyên phần cứng khác nhau, như các loại GPU khác nhau, để đánh giá toàn diện hiệu quả của chúng trong các môi trường triển khai đa dạng.

5.3. Hướng phát triển trong tương lai

Dựa trên những kết quả đạt được cũng như các hạn chế đã được chỉ ra, hướng phát triển trong tương lai sẽ tập trung vào việc nâng cao hiệu quả huấn luyện, khả năng mở rộng, và tính ứng dụng thực tế của mô hình học liên kết (Federated Learning) trong các điều kiện môi trường đa dạng hơn.

Trước hết, cần mở rộng các thí nghiệm trên môi trường thực tế với client có tính chất không đồng bộ, như sự khác biệt về tài nguyên tính toán, khả năng kết nối mạng,

hoặc thời điểm tham gia huấn luyện. Việc mô phỏng các tình huống client mất kết nối, bị lỗi, hoặc chỉ tham gia một phần quá trình là rất cần thiết để đánh giá khả năng chịu lỗi và phục hồi của các thuật toán học liên kết.

Thực hiện tinh chỉnh chi tiết và tìm ra bộ tham số tốt nhất và tối ưu nhất với các phương pháp tìm kiếm tự động như grid search, random search. Tăng tốc độ hội tụ và độ chính xác tổng thể.

Nghiên cứu và mở rộng các thuật toán tối ưu hóa trong FL, đặc biệt là những phương pháp có khả năng giảm thiểu độ dao động do dữ liệu không đồng nhất gây ra, như FedDyn, FedNova, FedYogi. So sánh và tích hợp những thuật toán mới này với các mô hình hiện tại có thể đem lại hiệu quả huấn luyện vượt trội hơn nữa.

Sử dụng nhiều bộ dữ liệu có tính phức tạp hơn để có thể đánh giá chi tiết và rõ ràng hơn về khả năng học của từng client khi gặp phải những tập dữ liệu khó, tìm ra được nhưng vấn đề mới và hạn chế mới của các thuật toán.

Nên sử dụng mô hình lớn và mạnh mẽ hơn để xem xét và đánh giá về khả năng học với những dữ liệu đơn giản và dữ liệu phức tạp, thời gian hội tụ và khả năng phản ứng của mô hình lớn sẽ thế nào. Đánh giá xem việc đánh đổi thời gian và độ chính xác có là vấn đề không, phù hợp cho lĩnh vực nào và tại sao.

Xem xét và thử nghiệm cho bài toán về y thế vì học liên kết cho thấy khả năng tốt và nổi bật trong y tế và các vấn đề nhạy cảm về dữ liệu.

TÀI LIỆU THAM KHẢO

- Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, H. Brendan McMahan, "Adaptive Federated Optimization," in *ICLR2021*, 2020.
- Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, Virginia Smith, "FEDERATED OPTIMIZATION IN HETEROGENEOUS NETWORKS," 14 December 2018.
- Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, Nicholas D. Lane, "Flower: A Friendly Federated Learning Research Framework," p. 15, 5 March 2022.
- Vin Big Data, "Federated Learning trong thị giác máy tính: Hướng dẫn chi tiết từng bước," Vin, 15 Jun 2023. [Online]. Available: <https://vinbigdata.com/kham-pha/federated-learning-trong-thi-giac-may-tinh-huong-dan-chi-tiet-tung-buoc.html>.
- H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," p. 11, 17 Feb 2016.
- Betul Yurdem a, Murat Kuzlu b, Mehmet Kemal Gullu a, Ferhat Ozgur Catak c,* , Maliha Tabassum b, "Federated learning: Overview, strategies, applications, tools and future directions," *Heliyon*, vol. X, no. 19, 2024.
- Brendan McMahan, Daniel Ramage, "Federated Learning: Collaborative Machine Learning without Centralized Training Data," Google research, 6 April 2017. [Online]. Available: <https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data/>.

- Gustav A. Baumgart, Jaemin Shin, Ali Payani, Myungjin Lee, Ramana Rao
- [8] Kompella, "Not All Federated Learning Algorithms Are Created Equal: A Performance Evaluation Study," p. 12, 26 March 2024.
- Di Chai*, Leye Wang*, Liu Yang, Junxue Zhang, Kai Chen, Qiang Yang *
- [9] Equal Contribution, "A Survey for Federated Learning Evaluations: Goals and Measures," p. 20, 8 August 2021.
- [10] A. authors, "FEDERATED AVERAGING AS EXPECTACTION-MAXIMIZATION," *ICLR 2021*, 2020.
- [11] Xiao-Tong Yuan, Ping Li, "On Convergence of FedProx: Local Dissimilarity Invariant Bounds, Non-smoothness and Beyond," p. 14, 10 Jun 2022.