TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Bảo Long - Cao Tất Cường

Ứng dụng của Meta Learning và Personalization Layer trong hệ thống Federated Learning

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

Tp. Hồ Chí Minh, tháng 03/2022

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Bảo Long - 18120201
Cao Tất Cường - 18120296

Ứng dụng của Meta Learning và Personalized Layer trong hệ thống Federated Learning

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN
CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN HƯỚNG DẪN
GS.TS. Lê Hoài Bắc

Tp. Hồ Chí Minh, tháng 03/2022

Tóm tắt

Đề cương chi tiết

Thông tin chung

- Tên đề tài: Ứng dụng của Meta Learning và Personalization Layer trong hệ thống Federated Learning
- Giảng viên hướng dẫn: GS. TS. Lê Hoài Bắc
- Nhóm sinh viên thực hiện:
  - Nguyễn Bảo Long - MSSV: 18120201
  - Cao Tất Cường - MSSV: 18120296
- Thời gian thực hiện: Từ 09/2021 đến 03/2022
- Loại đề tài: Nghiên cứu

Nội dung thực hiện

Giới thiệu đề tài

Trong bối cảnh bùng nổ thông tin cũng như việc đề cao tính riêng tư dữ liệu người dùng như hiện nay, các mô hình huấn luyện tập trung truyền thống dần bộc lộ nhiều điểm yếu khiến chúng không còn phù hợp. Ba điểm yếu làm cho cách tiếp cận cũ này trở nên tốn kém, không năng suất và ảnh hưởng đến quyền riêng tư của người dùng có thể kể đến:

- Việc truyền dữ liệu từ máy người dùng về máy chủ để tiến hành huấn luyện tiềm ẩn nguy cơ lộ dữ liệu quan trọng của người dùng.
- Chi phí truyền dữ liệu từ người dùng về máy chủ để huấn luyện ngày càng lớn do lượng dữ liệu sinh ra tại thiết bị cuối ngày càng tăng cao.

- Cần một máy chủ thật mạnh mẽ để huấn luyện mô hình với lượng dữ liệu lớn như vậy.

Khái niệm federated learning (FL) được Google lần đầu giới thiệu trong nghiên cứu [18] với ý tưởng chính là huấn luyện mô hình máy học trên các tập dữ liệu riêng biệt được phân bố trên các thiết bị biên (được gọi là huấn luyện phân tán). Với ý tưởng này, việc triển khi mô hình máy học đến người dùng không còn gặp phải vấn đề về chi phí truyền tin, giúp bảo vệ quyền riêng tư dữ liệu và không đòi hỏi một máy chủ quá mạnh để huấn luyện mô hình. Tuy nhiên, dữ liệu của người dùng trong hệ thống thường không đồng nhất và có tính cá nhân hóa rất cao (tuân theo phân phối Non-IID). Điều này khiến cho hiệu suất của hệ thống FL suy giảm nghiêm trọng [28].

Một cách ngắn gọn, hiệu suất của mô hình bị giảm là do mô hình không khái quát tốt trên tập dữ liệu của người dùng. Các thuật toán meta learning (ML) được biết đến với khả năng thích ứng nhanh trên tập dữ liệu mới [12]. Điều này giải quyết chính xác vấn đề dữ liệu FL đang gặp phải. Song song với đó, để tăng thêm tính các nhân hóa mô hình cho từng người dùng, các nghiên cứu [2, 16] đề xuất sử dụng kỹ thuật personalization layer (PL), giúp tăng đáng kể hiệu suất trên hệ thống FL. Tuy nhiên, các phương pháp tối ưu kể trên vẫn tồn tại nhiều khuyết điểm và có khả năng bù trừ cho nhau. Do đó, chúng tôi tiến hành nghiên cứu về ML, PL. Từ đó, kết hợp chúng vào hệ thống FL để đạt được hiệu suất tốt hơn.

Mục tiêu đề tài

Mục tiêu chính của đề tài này bao gồm: (1) - nghiên cứu, khảo sát các thuật toán theo hướng FL, ML, PL và (2) - kết hợp cài đặt các thuật toán trên, giúp nâng cao hiệu suất của hệ thống FL khi đối mặt với dữ liệu dạng Non-IID.

Việc nghiên cứu, khảo sát các thuật toán nhằm đưa ra đánh giá về ưu, nhược điểm của từng thuật toán. Từ đó, biết cách kết hợp chúng để đạt hiệu suất tốt.

Việc kết hợp cài đặt nhằm mục đích chứng minh thực nghiệm tính hiệu quả của thuật toán đề xuất khi làm việc với dữ liệu Non-IID.

Phạm vi đề tài

Nghiên cứu [25] chỉ ra ba hướng nghiên cứu chính khi đề cập đến một hệ thống FL: (1) - Cải thiện hiệu suất của hệ thống FL, (2) - Cải thiện khả năng bảo mật của hệ thống FL, (3) - Cải thiện vấn đề về quyền riêng tư của người dùng trong hệ thống FL.

Về việc phân loại hệ thống FL, dựa trên dữ liệu đầu vào, hệ thống FL được chia thành ba loại [25]: (1) - Horizontal FL, (2) - Vertical FL, (3) - Federated transfer learning.

Về việc phân loại các kịch bản Non-IID, nghiên cứu [28] chỉ ra bốn kịch bản chính: (1) - Phân phối thuộc tính khác nhau giữa các máy khách, (2) - Phân phối nhãn khác nhau giữa các máy khách, (3) - Phân phối thời gian khác nhau giữa các máy khách, (4) - Các kịch bản khác.

Chúng tôi giới hạn phạm vi và xây dựng phương án giải quyết của đề tài dựa trên ba giả định sau:

- Loại hệ thống: Môi trường thí nghiệm (bao gồm các yếu tố như số lượng người dùng, dữ liệu, cấu hình, khả năng lưu trữ của thiết bị cuối,...) tuân theo đặc điểm của hệ thống Horizontal FL.
- Bảo mật & quyền riêng tư: Hệ thống đã đảm bảo tính bảo mật cũng như duy trì tốt quyền riêng tư của người dùng.
- Kịch bản Non-IID: Phân phối nhãn dữ liệu trên các máy khách là khác nhau.

Cách tiếp cận

Phương pháp chính.

Hệ thống FL huấn luyện trực tiếp các mô hình máy học trên các tập dữ liệu của từng người dùng, sau đó tiến hành tổng hợp tham số của mô hình thu được từ các máy khách này để thu được một mô hình toàn cục. Do đó, kiến trúc máy chủ - máy khách nghiễm nhiên trở được nghĩ đến và trở nên phổ biến. Nghiên cứu [25] nêu ra các đặc điểm chính của máy chủ và máy khách trong một hệ thống FL:

- Máy chủ: Điều phối các hoạt động huấn luyện mô hình và duy trì một bộ tham số toàn cục bằng cách tổng hợp các tham số mô hình do máy khách gửi về.

- Máy khách: Đóng vai trò là nơi huấn luyện các mô hình học theo sự chỉ đạo của máy chủ. Chúng nhận tham số toàn cục từ máy chủ, huấn luyện mô hình dựa trên bộ tham số này và gửi tham số của mô hình mới về máy chủ để tổng hợp.

Phương pháp đề xuất của chúng tôi nhằm tích hợp các thuật toán ML, PL vào hệ thống nêu trên. Trong đó, các thuật toán ML được sử dụng để tạo ra một khởi tạo tốt, giúp máy khách hội tụ mô hình nhanh chóng (chỉ sau một hoặc một vài bước huấn luyện); các thuật toán PL được thêm vào như một phương pháp giúp tăng tính cá nhân hóa của từng mô hình máy học phân bố trên máy khách.

Dữ liệu thực nghiệm. Chúng tôi sử dụng tập dữ liệu CIFAR-10 và tập dữ liệu MNIST trong tất cả các thí nghiệm. Cả hai tập dữ liệu đều sử dụng 25% số điểm dữ liệu để kiểm thử và 75% số điểm dữ liệu để huấn luyện.

Phương pháp đối sánh. Chúng tôi sử dụng thuật toán `FedAvg` làm mô hình baseline. Để so sánh công bằng, chúng tôi cho phép mô hình này fine-tune một hoặc một vài bước trên một phần tập dữ liệu kiểm thử (thuật toán `FedAvgMeta`) trước khi bước vào kiểm thử thực sự. Các kết quả của thuật toán đề xuất sẽ được đem so sánh với kết quả của `FedAvg` và `FedAvgMeta`.

Kết quả đề tài

Sau khi tiến hành nghiên cứu, chúng tôi kỳ vọng đạt được những kết quả sau:

- Nắm được ý tưởng huấn luyện mô hình của các thuật toán theo hướng FL, ML, PL. Cài đặt mô hình baseline.

- Cài đặt được hệ thống FL có tích hợp ML. So sánh độ chính xác thu được với mô hình baseline.

- Cài đặt hệ thống FL có tích hợp ML và PL. So sánh độ chính xác thu được với mô hình baseline.

Kế hoạch thực hiện

Kế hoạch thực hiện khóa luận bao gồm 4 giai đoạn, được trình bày trong bảng sau:

Bảng 1: Bảng phân chia công việc

Giai đoạn	Công việc	Người thực hiện
1 (01/09/2021 - 15/09/2021)	Tìm hiểu kiến thức nền tảng về ML	Nguyễn Bảo Long
	Tìm hiểu kiến thức nền tảng về FL	Cao Tất Cường
	Tìm hiểu kiến thức nền tảng về PL	Cao Tất Cường
	Trao đổi 2 mảng kiến thức	Cả hai
2 (16/09/2021 - 31/01/2022)	Cài đặt các thuật toán FedAvg, FedMeta(Meta-SGD)	Nguyễn Bảo Long
	Cài đặt các thuật toán FedAvg(Meta), FedAvg(MAML)	Cao Tất Cường
	Cài đặt hệ thống FL có tích hợp ML và PL	Nguyễn Bảo Long
	Phân tích và đánh giá kết quả	Cả hai
3 (01/02/2022 - 25/02/2022)	Viết luận văn	Nguyễn Bảo Long
	Làm slide thuyết trình	Cao Tất Cường
	Tập thuyết trình	Cả hai

Lời cảm ơn

Xin phép gửi lời cảm ơn chân thành nhất đến GS. TS. Lê Hoài Bắc, người đã cung cấp, chia sẻ tài liệu và kiến thức, cũng như đã trực tiếp tham gia giảng dạy, hướng dẫn chúng tôi hoàn thiện đồ án này.

Ngoài ra, chúng tôi đặc biệt cảm ơn TS. Nguyễn Tiến Huy vì những lời khuyên rất hữu ích mỗi khi chúng tôi gặp khó khăn.

Đồ án này sẽ không thể hoàn thiện được nếu không có sự hỗ trợ kiến thức đến từ chị Bùi Thị Cẩm Nhung, cựu sinh viên chương trình Cử nhân tài năng, khoa Công Nghệ Thông Tin, khóa 2017. Chúng tôi rất cảm ơn trước những đóng góp của chị.

Mục lục

Danh sách hình

Danh sách bảng

Chương 1

Giới thiệu

1.1 Đặt vấn đề & Động lực

Hiện nay, các thiết bị biên như điện thoại, máy tính bảng, thậm chí máy giặt, máy hút bụt thông minh có thể sinh ra lượng lớn dữ liệu trong quá trình hoạt động. Lượng dữ liệu này, nếu tận dụng được, có thể mang lại sự cải thiện rất lớn về độ chính xác cho các mô hình máy học hiện tại. Ví dụ, dữ liệu thu thập được từ bàn phím điện thoại có thể phục vụ tối ưu cho các mô hình ngôn ngữ; ảnh chụp được lưu trữ trong bộ nhớ điện thoại hoàn toàn có thể được sử dụng làm dữ liệu để huấn luyện cho mô hình nhận dạng ảnh; hay lịch sử duyệt web của người dùng có thể được dùng cho bài toán đề xuất sản phẩm. Những lý do trên trở thành một động lực to lớn, thúc đẩy việc tìm ra một phương pháp giúp tận dụng nguồn dữ liệu dồi dào này.

Việc ngày càng nhiều dữ liệu được sinh ra tại các thiết bị biên khiến cho phương pháp huấn luyện mô hình theo cách tiếp cận truyền thống (được gọi là huấn luyện tập trung) bộc lộ nhiều khuyết điểm. Ba điểm yếu khiến cho các tiếp cận này không còn mạnh mẽ có thể kể đến: sự vi phạm về quyền riêng tư dữ liệu, chi phí truyền tin và chi phí phần cứng máy chủ.

Sự vi phạm về quyền riêng tư dữ liệu. Phương pháp truyền thống đòi hỏi phải gửi dữ liệu người dùng về một máy chủ để tiến hành huấn luyện mô hình. Các thông tin nhạy cảm của người dùng hoàn toàn có thể bị nghe lén bởi kẻ tấn công hoặc bị khai thác khi máy chủ bị nhiễm mã độc. Điều này ảnh hưởng nghiêm trọng đến quyền riêng tư dữ liệu của người dùng - một vấn đề mà hiện nay đang nhận được rất nhiều sự quan tâm từ cả người dùng lẫn chính phủ.

Chi phí truyền tin. Dữ liệu sinh ra tại thiết bị biên đang ngày một tăng lên do văn hóa sử dụng và sự phát triển của công nghệ. Một người

dùng điện thoại thông minh giờ đây có thể thực hiện giao dịch tài chính, nghe nhạc, lướt web, xem phim ngay trên thiết bị của mình. Một máy hút bụi thông minh được trang bị các cảm biến nên hoàn toàn có thể sử dụng dữ liệu cảm biến này như một "time series". Chi phí truyền tin từ các thiết bị biên đến máy chủ để huấn luyện trở nên tốn kém và có thể gây mất thông tin, ảnh hưởng đến hiệu suất học của mô hình.

Chi phí phần cứng máy chủ. Sau khi dữ liệu được gửi về máy chủ, cần một cấu hình máy mạnh mẽ cùng khả năng lưu trữ lớn để có thể xử lý hết lượng dữ liệu khổng lồ trên trong thời gian giới hạn.

Việc các phương pháp tiếp cận máy học theo hướng truyền thống đang dần bộc lộ các nhược điểm về chi phí vận hành và bảo trì ngày càng cao, cũng như các mối nguy hiểm tiềm tàng có thể xảy ra đối với dữ liệu của người dùng, một lần nữa thúc đẩy việc nghiên cứu về một phương pháp huấn luyện giúp làm giảm chi phí phần cứng (sử dụng cho đường truyền và máy chủ), đồng thời đảm bảo tính riêng tư dữ liệu cho người dùng. Khái niệm federated learning (FL) và thuật toán FedAvg được đưa ra vào năm 2016 bởi Google trong nghiên cứu [18] nhằm mục đích huấn luyện mô hình máy học trên các tập dữ liệu riêng biệt được phân bố trên các thiết bị biên (được gọi là huấn luyện phân tán). Do đó, một hệ thống FL không cần một máy chủ quá mạnh để vận hành (thậm chí có thể sử dụng một máy khách để vận hành [25] , không đòi hỏi chi phí truyền tin quá lớn và đảm bảo được quyền riêng tư dữ liệu của người dùng vì không diễn ra bất cứ quá trình thu thập dữ liệu từ người dùng nào (điều mà mô hình huấn luyện tập trung bắt buộc phải làm). Dễ thấy rằng, phần lớn quá trình tính toán được chuyển đến các thiết bị biên. Tuy nhiên, khả năng lưu trữ và tính toán tại các thiết bị này ngày càng được cải thiện, khiến cho việc việc huấn luyện phân tán dần trở nên khả thi và đạt hiệu quả cao hơn.

Mặt khác, nghiên cứu [28] chỉ ra rằng, hệ thống FL hoạt động trên nền thuật toán FedAvg bị giảm hiệu suất nghiêm trọng khi xử lý dữ liệu đầu vào tuân theo phân phối Non-IID. Trong khi đó, dữ liệu phân bố trên máy khách là không đồng nhất và có tính cá nhân hóa tất cao. Nói cách khác, các tập dữ liệu này tuân theo phân phối Non-IID. Do đó, việc nghiên cứu và cải tiến hệ thống FL để thu được kết quả cao trên dữ liệu Non-IID là rất cần thiết. Đây chính là vấn đề mà khóa luận hướng tới giải quyết.

1.2 Phạm vi đề tài

Nghiên cứu [25] chỉ ra ba hướng nghiên cứu chính khi đề cập đến một hệ thống FL: (1) - Cải thiện hiệu suất của hệ thống FL, (2) - Cải thiện khả năng bảo mật của hệ thống FL, (3) - Cải thiện vấn đề về quyền riêng tư của người dùng trong hệ thống FL.

Về việc phân loại hệ thống FL, dựa trên dữ liệu đầu vào, hệ thống FL được chia thành ba loại [25]: (1) - Horizontal FL, (2) - Vertical FL, (3) - Federated transfer learning.

Về việc phân loại các kịch bản Non-IID, nghiên cứu [28] chỉ ra bốn kịch bản chính: (1) - Phân phối thuộc tính khác nhau giữa các máy khách, (2) - Phân phối nhãn khác nhau giữa các máy khách, (3) - Phân phối thời gian khác nhau giữa các máy khách, (4) - Các kịch bản khác.

Chúng tôi giới hạn phạm vi và xây dựng phương án giải quyết của đề tài dựa trên ba giả định sau:

- Loại hệ thống: Môi trường thí nghiệm (bao gồm các yếu tố như số lượng người dùng, dữ liệu, cấu hình, khả năng lưu trữ của thiết bị cuối,...) tuân theo đặc điểm của hệ thống Horizontal FL.
- Bảo mật & quyền riêng tư: Hệ thống đã đảm bảo tính bảo mật cũng như duy trì tốt quyền riêng tư của người dùng.
- Kịch bản Non-IID: Phân phối nhãn dữ liệu trên các máy khách là khác nhau.

1.3 Đóng góp chính

Chúng tôi chia các đóng góp chính của mình thành hai loại: đóng góp về mặt lý thuyết và đóng góp về mặt thực nghiệm.

1.3.1 Đóng góp lý thuyết

- Nghiên cứu hệ thống FL và thách thức về phân phối dữ liệu mà hệ thống Horizontal FL gặp phải.

- Khảo sát các phương pháp tối ưu hóa hệ thống Horizontal FL trên dữ liệu dạng Non-IID. Trong đó, tập trung nghiên cứu các phương pháp theo hướng Personalized Federated Averaging [8, 5] và Personalization Layer [16, 2].
- Phương pháp đề xuất của chúng tôi đã cho thấy khả năng đạt độ chính xác cao hơn trong quá trình kiểm thử với hai đối tượng người dùng (người dùng cục bộ và người dùng mới) so với các phương pháp trước đó (chỉ sử dụng `FedAvg`, chỉ sử dụng Personalized Federated Averaging, hoặc chỉ sử dụng Personalization Layer).

1.3.2 Đóng góp thực nghiệm

- Tổ chức bộ dữ liệu MNIST và CIFAR-10 theo hai hướng IID và Non-IID để tiến hành thí nghiệm.
- Cài đặt thuật toán `FedAvg`, `FedAvgMeta`, các thuật toán kết hợp giữa `FedAvg` và ML (thuật toán `FedMeta(MAML)`, `FedMeta(Meta-SGD)`).
- Cài đặt thuật toán kết hợp giữa `FedAvg` và PL (thuật toán `FedPer`, `LG-FedAvg`).
- Kết hợp các thuật toán ML và PL vào hệ thống FL.
- Fine-tune các siêu tham số như số lượng máy khách tham gia huấn luyện, số bước huấn luyện cục bộ, các siêu tham số học để mô hình đạt độ chính xác tốt nhất.

1.4 Bố cục

Trong luận văn này, chương 2 trình bày về tổng quan lý thuyết được sử dụng trong khóa luận, các lý thuyết này làm nền tảng cho nghiên cứu và đề xuất thuật toán; chương 3 đề xuất thuật toán giúp giải quyết vấn đề vừa nêu ở chương 1; chương 4 trình bày về cài đặt thực nghiệm để kiểm chứng tính hiệu quả của thuật toán; chương 5 đi vào phân tích kết quả đạt được; chương 6 nêu kết luận, những điều chưa làm được và hướng phát triển tương lai của khóa luận.

Chương 2

Tổng quan lý thuyết

2.1 Hệ thống Federated Learning

2.1.1 Định nghĩa

Định nghĩa về FL [24]: Giả sử có n máy khách, máy khách thứ i ký hiệu là c_i ($i \in [1, n]$), chứa tập dữ liệu D_i. FL là một quá trình học mà ở đó, các chủ sở hữu dữ liệu (ở đây có thể hiểu là các thiết bị biên) cùng hợp tác huấn luyện một mô hình $\mathcal{M}$ và đạt được độ chính xác f nhưng không có bất kỳ chủ sở hữu dữ liệu c_i nào chia sẻ tập dữ liệu $\mathcal{D}_i$ của chúng.

Gọi $\bar{\mathcal{M}}$ là mô hình máy học được huấn luyện trên tập dữ liệu $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup ... \cup \mathcal{D}_n$ và cho độ chính xác $\bar{f}$. f và $\bar{f}$ chỉ được phép chênh lệch nhau một khoảng nhỏ. Gọi δ là một giá trị thực không âm, nếu $| f - \bar{f} | < \delta$ ta nói mô hình $\mathcal{M}$ có δ - *accuracy loss*.

Định nghĩa về tính hợp lệ [14]: Ký hiệu $\mathcal{M}_i$ là mô hình được huấn luyện trên tập dữ liệu $\mathcal{D}_i$ và cho độ chính xác f_i. Mô hình $\mathcal{M}$ được gọi là hợp lệ nếu tồn tại $i \in [1, n]$ sao cho $f > f_i$.

2.1.2 Một hệ thống Federated Learning điển hình

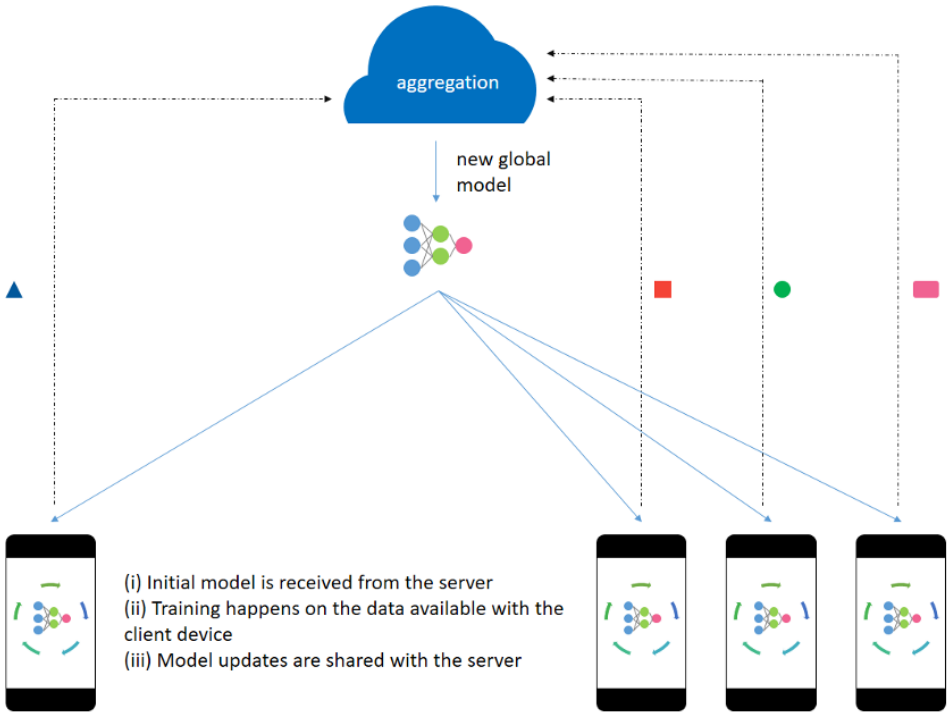
Thành phần và các tương tác trong hệ thống. Một hệ thống FL (Hình 2.1) thường bao gồm hai thành phần chính: máy chủ (đóng vai trò là đối tượng duy trì mô hình toàn cục) và máy khách (đóng vai trò là đối tượng nắm giữ dữ liệu huấn luyện). Hai thành phần này tương tác với nhau theo ba bước sau [17]:

- *Khởi tạo.* Máy chủ khởi tạo trọng số w_G^0 cho mô hình toàn cục và các siêu tham số cho quá trình huấn luyện. Thông tin này sau đó được gửi đến một tập hợp con các máy khách được chọn để tiến hành huấn luyện.

- *Huấn luyện và cập nhật mô hình cục bộ.* Tại bước huấn luyện thứ t, máy khách c_i nhận trọng số w_G^t từ máy chủ và tiến hành huấn luyện cục bộ trên tập dữ liệu $\mathcal{D}_i$. Tham số θ_i^{t+1} thu được sau quá trình huấn luyện (có thể là trọng số w_i^{t+1} hoặc đạo hàm hàm lỗi g_i^{t+1}) được máy khách gửi về máy chủ để tổng hợp.

- *Tổng hợp và cập nhật mô hình toàn cục.* Máy chủ nhận tham số θ_i^{t+1} gửi về từ các máy khách được chọn trước đó, tiến hành tổng hợp w_G^{t+1} - trọng số mới của mô hình toàn cục và gửi trọng số này đến một tập hợp con các máy khách khác để bắt đầu bước huấn luyện toàn cục mới.

Máy chủ sẽ lặp lại bước 2 và bước 3 cho đến khi độ lỗi hội tụ hoặc độ chính xác đạt đến một ngưỡng nhất định. Khi quá trình huấn luyện kết thúc, tham số của mô hình toàn cục sẽ được phân phối đến toàn bộ máy khách trong hệ thống.



Hình 2.1: Hai thành phần chính và quá trình tương tác giữa chúng trong hệ thống FL [4]

Mục tiêu của hệ thống FL. Chúng tôi khảo sát hai mục tiêu của hệ thống FL: (1) - Mục tiêu cục bộ; (2) - Mục tiêu toàn cục.

Các máy khách trong hệ thống hướng đến việc thực hiện mục tiêu cục bộ. Ban đầu, máy khách c_i nhận một trọng số toàn cục w_G từ máy chủ. Máy khách này sau đó sẽ cố gắng tìm kiếm một trọng số w_i^{t+1} giúp cực tiểu hóa hàm lỗi cục bộ. Nói cách khác, w_i^{t+1} phải thỏa mãn thỏa mãn:

$$w_i^{t+1} = \arg\min_{w_i} f_{local}(w_i^t) \tag{2.1}$$

Trong đó, $f_{local}(w_i)$ là hàm lỗi trên tập dữ liệu của c_i. Với α là siêu tham số học cục bộ, $w_{i(j)}$ là trọng số tại bước huấn luyện j của c_i, lời giải của phương trình 2.1 theo phương pháp SGD sau e bước huấn luyện có thể được viết:

$$\begin{cases} w_{i(0)}^t = w_G^t \\ w_{i(j)}^t = w_{i(j-1)}^t - \alpha \nabla f_{local}(w_{i(j-1)}^t) \\ w_i^{t+1} = w_{i(e)}^t \end{cases} \tag{2.2}$$

Hay:

$$w_i^{t+1} \leftarrow w_i^t - \alpha \nabla f_{local}(w_i^t) \tag{2.3}$$

Mặt khác, mục tiêu toàn cục, cũng là mục tiêu chính của hệ thống FL, được máy chủ thực hiện bằng cách tìm kiếm một trọng số w_G^* giúp tối thiểu hóa hàm lỗi của cả hệ thống [25]:

$$\begin{aligned} w_G^* &= \arg\min_{w_G} f_{global}(w_G) \\ &= \arg\min_{w_G} \frac{1}{n} \sum_{i=1}^{n} f_{local}(w_i) \end{aligned} \tag{2.4}$$

Trong đó, $f_{global}(w_G)$ là hàm lỗi toàn cục của hệ thống. Để giải phương trình 2.4, máy chủ thực hiện tổng hợp tham số gửi về từ máy khách bằng một trong hai cách: lấy trung bình trọng số [18, 1, 26] hoặc lấy trung bình đạo hàm [5, 19].

Đặt $n_i = |\mathcal{D}_i|$ là số điểm dữ liệu của tập $\mathcal{D}_i$, $N = \sum_{i=1}^{n} n_i$ là tổng số điểm dữ liệu có trong cả hệ thống. Phương pháp lấy trung bình trọng số

tính toán trọng số toàn cục tại bước huấn luyện thứ t từ các trọng số của máy khách như sau [18]:

$$w_G^{t+1} = \sum_{i=1}^{n} \frac{n_i}{N} w_i^{t+1} \tag{2.5}$$

Trái lại, phương pháp lấy trung bình đạo hàm đòi hỏi máy khách gửi về đạo hàm hàm lỗi sau khi kết thúc quá trình huấn luyện cục bộ. Với β là siêu tham số học toàn cục, quá trình tổng hợp tại bước huấn luyện t được biểu diễn theo công thức:

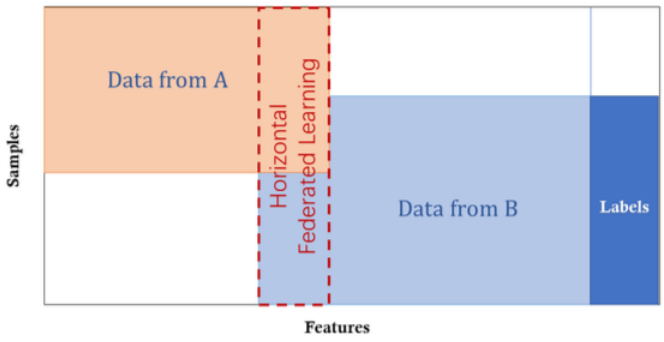
$$\begin{aligned} w_G^{t+1} &= w_G^t - \beta \sum_{i=1}^{n} \frac{n_i}{N} \nabla f_{local}(w_i^{t+1}) \\ &= w_G^t - \beta g^{t+1} \end{aligned} \tag{2.6}$$

Sau khi khảo sát cả hai phương pháp tổng hợp tham số của máy chủ, nghiên cứu [25] chỉ ra rằng, việc lấy trung bình trọng số giúp hệ thống có khả năng chịu được việc mất cập nhật, nhưng không đảm bảo việc hội tụ. Trái lại, việc lấy trung bình đạo hàm giúp hệ thống đảm bảo sự hội tụ nhưng tiêu tốn nhiều chi phí truyền tin hơn. Trong nghiên cứu này, chúng tôi tổng hợp trọng số toàn cục bằng phương pháp lấy trung bình trọng số để phù hợp hơn với giới hạn về chi phí giao tiếp và lưu trữ.

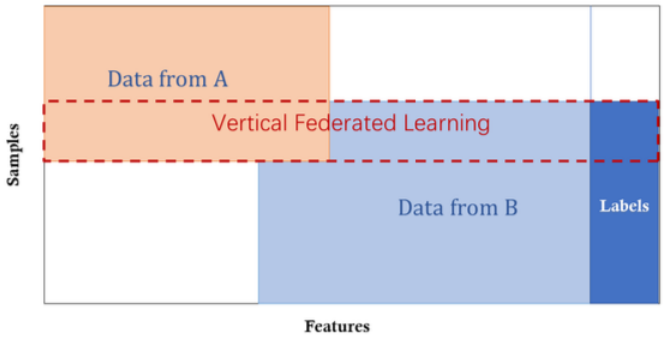
Phân loại hệ thống Federated Learning. Nghiên cứu [25] đề xuất các phân loại các hệ thống FL dựa trên phân bố dữ liệu đầu vào của chúng. Theo đó, ba phân bố dữ liệu: (1) - Phân bố dữ liệu theo chiều ngang (Horizontal data partitioning), (2) - Phân bố dữ liệu theo chiều dọc (Vertical data partitioning), (3) - Phân bố dữ liệu hỗn hợp (Hybrid data partitioning) sẽ ứng với ba loại hệ thống FL (Hình 2.2): (1) - Hệ thống FL theo chiều ngang (Horizontal FL), (2) - Hệ thống FL theo chiều dọc (Vertical FL), (3) - Hệ thống học chuyển giao tri thức (Federated Transfer Learning).

Hệ thống Horizontal FL. Phân bố dữ liệu theo chiều ngang là kiểu phân bố dữ liệu mà ở đó các bên tham gia vào hệ thống cùng sở hữu các đặc tính dữ liệu giống nhau nhưng giá trị định danh của mẫu dữ liệu của các bên là khác nhau. Ví dụ, khi các bên tham gia hệ thống là các trường đại học,

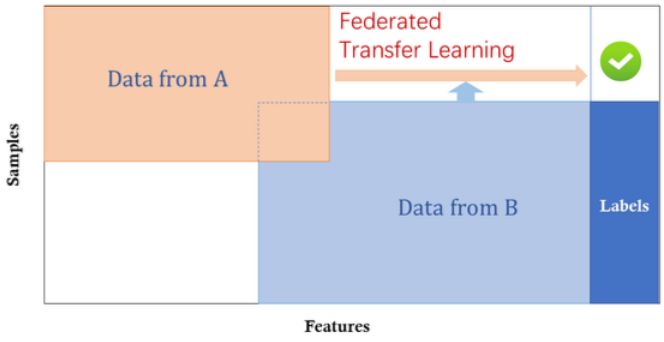
họ sẽ muốn quản lý các thông tin giống nhau về sinh viên như họ và tên, mã số sinh viên,... nhưng không có một sinh viên nào tham gia hai trường đại học cùng một lúc. Kiến trúc Horizontal FL rất phù hợp để huấn luyện mô hình học tuân theo phân phối này [25].



(a) Horizontal Federated Learning



(b) Vertical Federated Learning



(c) Federated Transfer Learning

Hình 2.2: Ba loại hệ thống FL với phân bố dữ liệu tương ứng [24]

Dựa vào kiến trúc giao tiếp, có thể chia Horizontal FL ra làm hai loại: Kiến trúc client-server và kiến trúc peer-to-peer (P2P). Kiến trúc client-server, hay còn gọi là kiến trúc FL tập trung, về cơ bản sẽ thực hiện các bước huấn luyện giống như đã trình bày trong phần **Thành phần và các tương tác trong hệ thống**. Trong khi đó, kiến trúc P2P, hay còn gọi là kiến trúc FL phân tán không có một máy chủ cố định. Tại mỗi bước huấn luyện toàn cục, một máy khách trong hệ thống được chọn làm máy chủ. Quá trình huấn luyện sau đó được thực hiện giống như kiến trúc client-server.

Một hệ thống Horizontal FL thường có số lượng máy khách rất lớn, khả năng lưu trữ và tính toán tại các máy khách không cần quá cao (ví dụ như điện thoại thông minh, máy tính bảng) và tần suất một máy khách tham gia huấn luyện là rất thấp.

Hệ thống Vertical FL. Đây là kiến trúc phù hợp với phân bố dữ liệu theo chiều dọc. Trong phân bố dữ liệu dạng này, các bên tham gia hệ thống sở hữu các đặc tính dữ liệu khác nhau nhưng giá trị định danh của mẫu dữ liệu của các bên là giống nhau. Ví dụ, khi các bên tham gia hệ thống là ngân hàng và trường đại học. Thuộc tính mà ngân hàng và trường đại học lưu trữ là rất khác nhau nhưng lại chứa thông tin của cùng một người dùng.

Hệ thống Federated Transfer Learning. Khi phân bố dữ liệu của các bên tham gia hệ thống không sở hữu chung các đặc tính dữ liệu hay giá trị định danh của từng mẫu, người ta gọi đây là phân bố dữ liệu hỗn hợp. Ví dụ, khi các bên tham gia hệ thống là một ngân hàng ở Hoa Kỳ và một trường đại học ở Việt Nam. Do cản trở địa lý và nhu cầu quản lý thông tin khác nhau, chủ sở hữu dữ liệu này sẽ không có chung thuộc tính hay giá trị định danh nào. Trong trường hợp này, FTL có thể được sử dụng để chuyển giao tri thức giữa các bên tham gia.

Dựa vào các đặc điểm phân loại nêu trên, chúng tôi xếp nghiên cứu của mình vào nhóm hệ thống Horizontal FL tập trung.

2.2 Khảo sát dữ liệu Non-IID

Dữ liệu tại các máy khách thường được sinh ra dựa trên nhu cầu của người dùng cuối. Do đó, loại dữ liệu này thường có tính cá nhân hóa cao và không đồng nhất. Nói cách khác, không có bất kỳ phân phối dữ liệu cục bộ nào có thể đại diện cho phân phối trên toàn bộ dữ liệu, phân phối dữ liệu trên hai máy khách khác nhau là khác nhau [28]. Đây chính là ý tưởng mà thuật ngữ *dữ liệu Non-IID* muốn truyền đạt.

Mặt khác, nghiên cứu [27] chỉ ra rằng hệ thống FL có thể bị giảm hiệu quả nghiêm trọng khi đối mặt với dữ liệu Non-IID. Để hiểu rõ vấn đề mình đang đối mặt, dựa trên nghiên cứu [28], chúng tôi tiến hành khảo sát các kịch bản về dữ liệu Non-IID.

Gọi (x, y) là cặp thuộc tính và nhãn dữ liệu. Theo kịch bản dữ liệu Non-IID, phân phối dữ liệu của hai máy khách c_i, c_j bất kỳ là khác nhau: $P_i(x, y) \neq P_j(x, y)$. Nghiên cứu [28] nêu ra bốn kịch bản dữ liệu Non-IID: (1) - Phân phối thuộc tính khác nhau giữa các máy khách, (2) - Phân phối nhãn khác nhau giữa các máy khách, (3) - Phân phối thời gian khác nhau giữa các máy khách, (4) - Các kịch bản khác.

2.2.1 Phân phối thuộc tính khác nhau giữa các máy khách

Với kịch bản này, phân phối thuộc tính $P(x)$ trên các máy khách là đôi một khác nhau. Không gian thuộc tính của các máy khách có thể khác nhau hoàn toàn, trùng lặp một vài thuộc tính hoặc trùng lặp hoàn toàn.

Các hệ thống Vertical FL thường rơi vào trường hợp đầu tiên. Ví dụ, trong trường hợp dữ liệu dạng bảng, máy khách A có thể quản lý các thuộc tính A_1, A_2, A_3 trong khi máy khách B quản lý các thuộc tính B_1, B_2 của cùng một người dùng.

Đối với trường hợp thứ hai, hai máy khách có thể cùng quản lý một số thuộc tính dữ liệu. Ví dụ, đối với dữ liệu của một hệ thống camera dám sát, hai camera bất kỳ có thể cùng lưu hình một người với các góc chụp khác nhau.

Trường hợp cuối chính là đặc điểm chính của hệ thống Horizontal FL. Tại đây, không gian thuộc tính của các máy khách là hoàn toàn giống nhau.

Ví dụ, tập dữ liệu MNIST chứa các mẫu chữ viết tay của nhiều người. Do đó, cùng là một chữ số, nhưng độ đậm nhạt, cách viết người dùng A sẽ khác với người dùng B.

2.2.2 Phân phối nhãn khác nhau giữa các máy khách

Đây là trường hợp dữ liệu Non-IID phổ biến nhất, gây hại nghiêm trọng cho hệ thống FL [28], cũng chính là trường hợp mà chúng tôi hướng tới giải quyết. Tại đây, phân phối nhãn của hai máy khách c_i, c_j bất kỳ là khác nhau: $P_i(y) \neq P_j(y)$ và xác suất thuộc tính x có nhãn dữ liệu y: $P(x|y)$ của các máy khách là như nhau. Một kịch bản thường thấy của trường hợp này được trình bày trong nghiên cứu [18]: Mỗi máy khách sẽ chỉ chứa các điểm dữ liệu thuộc về k nhãn. Trong đó, k là một siêu tham số biểu diễn độ mất cân bằng về nhãn. k càng nhỏ, hệ thống mất cân bằng nhãn càng mạnh. Trong khóa luận này, chúng tôi thiết đặt môi trường dữ liệu theo cách tương tự như vậy. `#todo: chèn thêm ảnh`

Ngoài ra, còn một trường hợp phổ biến khác liên quan đến việc dữ liệu Non-IID trên nhãn. Đối với trường hợp này, xác suất thuộc tính x được gán nhãn y: $P(y|x)$ là khác nhau giữa các máy khách. Ví dụ, với một bức ảnh trên mạng xã hội, người dùng A có thể gán nhãn "yêu thích", trong khi người dùng B gán nhãn "không yêu thích".

2.2.3 Phân phối thời gian khác nhau giữa các máy khách

Một ví dụ dễ hiệu cho trường hợp này là việc hai người dùng c_i, c_j thu thập dữ liệu trong hai khoảng thời gian khác nhau. Dẫn đến việc phân phối $P_i(x, y|t) \neq P_j(x, y|t)$, với t là một thời điểm nhất định. Một trường hợp khác của kịch bản này là phân phối $P(x, y|t)$ của một máy khách bị thay đổi liên tục theo thời gian. Ví dụ, một hệ thống camera giám sát có thể ghi nhận hình ảnh của rất nhiều người vào các ngày làm việc trong tuần nhưng lại có rất ít hình ảnh vào những ngày nghỉ.

2.2.4 Các kịch bản khác

Các kịch bản còn lại thường rơi vào hai trường hợp: (1) - Phân phối thuộc tính và nhãn là khác nhau giữa các máy khách, (2) - Số lượng dữ

liệu huấn luyện là khác nhau giữa các máy khách.

2.3 Tối ưu hệ thống Federated Learning trên dữ liệu Non-IID

2.3.1 Tối ưu dựa trên dữ liệu

Việc mô hình toàn cục làm việc với dữ liệu có phân phối không đồng nhất dẫn đến việc các lớp được học một cách không đồng đều (có một số lớp được học ít hơn/nhiều hơn các lớp khác), khiến cho mô hình bị giảm hiệu suất [28]. Hướng tối ưu dựa trên dữ liệu trực tiếp giải quyết vấn đề này bằng hai cách: (1) - Chia sẻ dữ liệu, (2) - Tăng cường dữ liệu.

Chia sẻ dữ liệu. Chia sẻ dữ liệu [27] được thực hiện bằng cách xây dựng một tập dữ liệu chứa dữ liệu của tất cả các nhãn theo phân phối đều. Dữ liệu trong tập này được thu thập trực tiếp từ các máy khách và gửi về máy chủ để kết hợp huấn luyện mô hình toàn cục.

Tăng cường dữ liệu. Cùng với ý tưởng cho phép mô hình toàn cục được học trên các tập dữ liệu có phân phối đều các nhãn trong hệ thống, tăng cường dữ liệu [21] nhằm mục đích gia tăng sự đa dạng của dữ liệu huấn luyện. Phương pháp này đòi hỏi các máy khách phải gửi phân phối dữ liệu của mình về máy chủ. Máy chủ theo đó yêu cầu các máy khách tạo ra ảnh mới [7] với số lượng và nhãn lớp biết trước, hoặc tự mình tạo ra ảnh mới bằng cách sử dụng GAN [28] để có thể huấn luyện trên một tập dữ liệu chứa tất cả các nhãn với phân phối trung bình.

Các phương pháp nêu trên đều giúp hệ thống FL "chống chịu" tốt trước dữ liệu Non-IID. Tuy nhiên, đòi hỏi máy khách gửi thông tin cá nhân về máy chủ là vi phạm mục tiêu ban đầu của hệ thống FL - bảo vệ quyền riêng tư dữ liệu của người dùng.

2.3.2 Tối ưu dựa trên thuật toán

Tinh chỉnh cục bộ

Tinh chỉnh cục bộ, hay local fine-tuning, là kỹ thuật mạnh mẽ trong việc cá nhân hóa mô hình học cho các tập dữ liệu riêng biệt. Kỹ thuật này

hướng đến việc tinh chỉnh mô hình học tại các máy khách sau khi nhận được mô hình từ máy chủ [23].

Một hướng tiếp cận phổ biến được đề ra là sử dụng ML trong việc tạo ra một mô hình toàn cục tốt, có thể thích ứng với tập dữ liệu mới trên máy khách một cách nhanh chóng. Các thuật toán theo hướng này [5, 8] sử dụng các kỹ thuật ML có khả năng tạo ra một khởi tạo tốt như Model-Agnostic Meta-Learning (MAML) [9] hay Meta-SGD [15] để huấn luyện mô hình toàn cục. Mô hình toàn cục này, trong quá trình chạy thực tế trên một máy khách mới, hoàn toàn có thể đạt hội tụ chỉ sau một hoặc một vài bước huấn luyện.

Lớp cá nhân hóa

Các thuật toán theo hướng này cho phép duy trì một phần của mạng học sâu trên máy khách. Cụ thể, thuật toán chia mạng học sâu thành hai thành phần: phần chung và phần riêng. Phần chung được hợp tác huấn luyện bởi các máy khách và được tổng hợp bởi máy chủ. Phần riêng tồn tại riêng biệt trên từng máy khách, được máy khách trực tiếp duy trì và huấn luyện.

Một thuật toán điển hình theo hướng tiếp cận này là FedPer [2]. FedPer quy định phần chung của mạng học sâu là các lớp rút trích đặc trưng, phần riêng của mạng là các lớp còn lại. Các thí nghiệm đã cho thấy thuật toán này đạt điệu quả cao hơn nhiều so với FedAvg khi làm việc trên dữ liệu Non-IID.

Thuật toán LG-FedAvg [16] cũng được xếp vào nhóm thuật toán sử dụng lớp cá nhân hóa. Tuy nhiên, ngược lại với FedPer, LG-FedAvg chỉ định phần riêng là các lớp rút trích đặc trưng trong mạng học sâu. Thực nghiệm cho thấy, LG-FedAvg đạt hiệu quả tốt hơn FedAvg trên cả các máy khách sẵn có trong hệ thống lẫn các máy khách chỉ vừa mới tham gia hệ thống.

Bằng các lớp cá nhân hóa, các thuật toán nêu trên đã giải quyết được sự khác nhau về dữ liệu giữa các máy khách, từ đó tránh được phần nào sự giảm hiệu suất trên dữ liệu Non-IID. Nhưng các lớp thuộc phần chung của mạng học sâu vẫn có thể bị thiên kiến (bias) do dữ liệu huấn luyện

không tuân theo phân phối đều. **Vậy có thể làm gì để các lớp này có thể thích ứng nhanh với tập dữ liệu của máy khách chỉ sau một vài bước huấn luyện?**

2.3.3 Tối ưu dựa trên hệ thống

Gom cụm người dùng

Các tiếp cận FL truyền thống giả định rằng hệ thống này chỉ bao gồm một máy chủ. Điều này làm cho việc học các đặc tính của tất cả các máy khách trong môi trường dữ liệu Non-IID là khó khả thi. Để giải quyết vấn đề này, một hệ thống huấn luyện với nhiều máy chủ được đề xuất. Câu hỏi đặt ra là: Làm sao để biết một máy khách nên huấn luyện cùng với máy chủ nào?

Trong ngữ cảnh đa máy chủ, thuật toán IFCA [10] trả lời câu hỏi trên bằng cách gửi trọng số của tất cả các máy chủ cho từng máy khách. Các máy khách theo đó tìm ra được trọng số cho độ lỗi nhỏ nhất trên tập dữ liệu cục bộ và gửi thông tin sau khi huấn luyện cục bộ của mình về máy chủ đó để cập nhật mô hình toàn cục. Việc gửi toàn bộ trọng số của các máy chủ đến một máy khách khiến cho thuật toán này tăng chi phí giao tiếp lên gấp k lần, với k là số lượng máy chủ của hệ thống.

Một cách khác để trả lời câu hỏi trên là đánh giá sự tương đồng của trọng số do máy khách gửi về [28]. Một thang độ đo sự tương đồng như độ đo cosine được máy chủ sử dụng trên các trọng số của máy khách. Từ đó biết được nên tổng hợp trọng số của các máy khách nào với nhau.

Với việc lượng dữ liệu và thiết bị biên ngày càng tăng lên, việc duy trì nhiều máy chủ là thực sự cần thiết khi đối mặt với nhu cầu nâng cấp hệ thống học. Tuy nhiên, chi phí giao tiếp và phương pháp gom cụm vẫn là những vấn đề rất lớn cần giải quyết.

Chương 3

Phương pháp đề xuất

Trong chương này, chúng tôi khảo sát hai và phân tích về ML và PL - hai hướng tiếp cận giúp cải thiện hiệu suất của hệ thống FL trên dữ liệu Non-IID. Từ đó đề xuất thuật toán `FedMeta-Per`, là sự kết hợp của hai kỹ thuật ML và PL vào hệ thống FL.

3.1 Tiếp cận hệ thống theo hướng Meta Learning

Meta Learning được áp dụng vào hệ thống FL như một phương pháp tối ưu thuộc nhóm "Tinh chỉnh cục bộ (local fine-tuning)" [28]. Các thuật toán ML được sử dụng trong hệ thống FL nhằm mục đích tạo ra một mô hình toàn cục tốt, giúp hội tụ nhanh trên tập dữ liệu phân bố trên các máy khách.

3.1.1 Diễn giải Meta Learning

ML là một phương pháp học mới, cho phép mô hình học có thể gia tăng kinh nghiệm qua việc thực hiện nhiều nhiệm vụ khác nhau trong cùng một phân phối nhiệm vụ. Dẫn đến việc các mô hình ML có khả năng thích ứng nhanh trên nhiệm vụ mới chỉ sau một vài bước huấn luyện và dữ liệu huấn luyện giới hạn. Đây là một phát kiến quan trọng, đóng vai trò trong việc đưa cách học tập của máy trở nên tiệm cận với cách học tập của con người [11].

Đối với phương pháp huấn luyện mô hình học truyền thống, chúng ta huấn luyện mô hình dự đoán $\hat{y} = f_\theta(x)$ trên tập dữ liệu $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ của nhiệm vụ T gồm các cặp thuộc tính và nhãn tương ứng. Ký hiệu $\mathcal{L}$ là hàm lỗi, ω là giả định ban đầu của hệ thống học, mục tiêu của việc học là tối thiểu hóa hàm lỗi trên tập dữ liệu $\mathcal{D}$ bằng cách tìm một bộ trọng số w^* thỏa mãn:

$$w^* = \arg\min_w \mathcal{L}(\mathcal{D}; w, \omega) \tag{3.1}$$

Hướng tiếp cận của ML nằm ở chỗ cố gắng học một giả định ban đầu ω thật tốt. Điều này đạt được thông qua việc học một phân phối các nhiệm vụ $p(T)$ [12]. Sau khi học được một giả định ban đầu tốt, có thể áp dụng giả định này cho các nhiệm vụ mới trong cùng phân phối nhiệm vụ: $T \sim p(T)$.

Về mặt công thức, ký hiệu $\mathcal{L}(\mathcal{D}, \omega)$ là hàm số biểu diễn sự hiệu quả việc sử dụng ω trong huấn luyện nhiệm vụ T có tập dữ liệu $\mathcal{D}$, chúng ta có thể biểu diễn hàm mục tiêu của ML như sau:

$$\min_{\omega} \mathop{\mathbb{E}}_{T \sim p(T)} \mathcal{L}(\mathcal{D}, \omega) \tag{3.2}$$

Trong thực tế, người ta thực hiện mục tiêu trên bằng cách huấn luyện mô hình học trên tập dữ liệu $\mathcal{D}_{train} = \{(\mathcal{D}^{support}_{train(i)}, \mathcal{D}^{query}_{train(i)})\}_{i=1}^{|\mathcal{D}_{train}|}$ và kiểm thử trên tập dữ liệu $\mathcal{D}_{test} = \{(\mathcal{D}^{support}_{test(i)}, \mathcal{D}^{query}_{test(i)})\}_{i=1}^{|\mathcal{D}_{test}|}$. Mục tiêu của việc huấn luyện là tìm ra một giá trị ω^*, sao cho khi sử dụng giá trị này trong huấn luyện một nhiệm vụ $T \sim p(T)$ thì đạt được hiệu quả cao:

$$\omega^* = \arg\max_{\omega} \log p(\omega | \mathcal{D}_{source}) \tag{3.3}$$

Trong quá trình kiểm thử, tham số ω^* được sử dụng trong việc huấn luyện mô hình giải quyết nhiệm vụ T_{new}: $w^* = \arg\max_w \log p(w|\omega^*, \mathcal{D}^{support}_{test(new)})$. Để đánh giá hiệu quả của việc sử dụng ω trong huấn luyện nhiệm vụ T_{new}, người ta dựa vào kết quả của w^* trên tập $\mathcal{D}^{query}_{test(new)}$.

Để giải phương 3.3, chúng tôi nhìn nhận ML dưới góc độ một bài toán tối ưu hai cấp độ [12]. Dưới góc nhìn này, phương trìnhh 3.3 được giải bằng cách đạt được mục tiêu tại hai cấp độ: (1) - Cấp độ thấp, (2) - Cấp độ cao. Đối với cấp độ thấp, mục tiêu là giải quyết nhiệm vụ T_i dựa vào ω:

$$w_i^*(\omega) = \arg\min_w \mathcal{L}^{task}(w, \omega, \mathcal{D}^{support}_{train(i)}) \tag{3.4}$$

Giải phương trình 3.4 bằng kỹ thuật SGD, ta được lời giải sau:

$$\begin{cases} w_{i(0)} = \omega \\ w_{i(j)} = w_{i(j-1)} - \alpha \nabla \mathcal{L}^{task}(w_{i(j-1)}, \omega, \mathcal{D}^{support}_{train(i)}) \end{cases} \tag{3.5}$$

Hay:

$$w_i \leftarrow w_i - \alpha \nabla \mathcal{L}^{task}(w_i) \tag{3.6}$$

Đối với cấp độ cao, mục tiêu là tìm ra tham số ω^* tối ưu, giúp việc học một nhiệm vụ mới $T_{new} \sim p(T)$ được thực hiện nhanh chóng và đạt hiệu suất cao:

$$\omega^* = \arg\min_{\omega} \sum_{i=1}^{|\mathcal{D}_{souece}|} \mathcal{L}^{meta}(w_i^*(\omega), \omega, \mathcal{D}^{query}_{train(i)}) \tag{3.7}$$

Áp dụng kỹ thuật SGD, lời giải cần tìm của bài toán ML được biểu diễn như sau:

$$\begin{cases} \omega_0 = \Omega, \Omega \text{ là giá trị khởi tạo ngẫu nhiên} \\ \omega_j = \omega_{j-1} - \beta \nabla \sum_{i=1}^{|\mathcal{D}_{souece}|} \mathcal{L}^{meta}\left(w_i^*(\omega), \omega, \mathcal{D}^{query}_{train(i)}\right) \end{cases} \tag{3.8}$$

Hay:

$$\begin{aligned} \omega &\leftarrow \omega - \beta \nabla \sum_{i=1}^{|\mathcal{D}_{souece}|} \mathcal{L}^{meta}\left(w_i^*(\omega)\right) \\ &\leftarrow \omega - \beta \nabla \sum_{i=1}^{|\mathcal{D}_{souece}|} \mathcal{L}^{meta}\left(w_i - \alpha \nabla \mathcal{L}^{task}(w_i)\right) \\ &\leftarrow \omega - \beta \sum_{i=1}^{|\mathcal{D}_{souece}|} \left(I - \alpha \nabla^2 \mathcal{L}^{task}(w_i)\right) \times \nabla \mathcal{L}^{meta}\left(w_i - \alpha \nabla \mathcal{L}^{task}(w_i)\right) \end{aligned} \tag{3.9}$$

3.1.2 Hệ thống Federated Learning và Meta Learning

Nhìn nhận hai phương trình 3.4 và 3.7, có thể thấy chúng dễ dàng đem thay thế các mục tiêu của một hệ thống FL truyền thống (được trình bày trong phần 2.1.2). Theo đó, mục tiêu cấp thấp và cấp cao trong ML có

thể lần lượt thay thế cho mục tiêu cục bộ và mục tiêu toàn cục trong hệ thống FL. Từ đây, có thể dễ dàng tích hợp ML vào hệ thống FL.

Thật vậy, nghiên cứu [8] đã tích hợp thuật toán MAML vào hệ thống FL và biểu diễn lại hàm mục tiêu toàn cục của hệ thống từ phương trình 2.4 như sau:

$$\min_{w_G} f_{global}(w_G) = \min_{w_G} \frac{1}{n} \sum_{i=1}^{n} f_{local}\left(w_i - \alpha \nabla f_{local}(w_i, \mathcal{D}_{train(i)}^{support}), \mathcal{D}_{train(i)}^{query}\right) \tag{3.10}$$

Trong hàm số 3.10, ta có thể nhận thấy sự xuất hiện của hai tập dữ liệu $\mathcal{D}_{train(i)}^{support}$ và $\mathcal{D}_{train(i)}^{query}$. Điều này có nghĩa là hệ thống FL huấn luyện theo hướng ML cần phải chia lại tập dữ liệu tại các máy khách theo kiểu ML. Một máy khách c_i tham gia huấn luyện bao gồm hai tập dữ liệu $\mathcal{D}_{train(i)}^{support}$ và $\mathcal{D}_{train(i)}^{query}$. Trong quá trình kiểm thử, dữ liệu của máy khách c_i cũng được chia thành hai tập $\mathcal{D}_{test(i)}^{support}$ và $\mathcal{D}_{test(i)}^{query}$. Trong đó, c_i cần thực hiện tinh chỉnh (fine-tune) mô hình trên tập $\mathcal{D}_{test(i)}^{support}$ và đánh giá mô hình trên tập $\mathcal{D}_{test(i)}^{query}$.

Từ phương trình tổng hợp mô hình toàn cục bằng phương pháp lấy trung bình trọng số 2.5, bằng phương pháp SGD như phương trình 3.9, phương trình tổng hợp mô hình toàn cục trong hệ thống FL tích hợp ML tại bước huấn luyện toàn cục thứ t có dạng:

$$\begin{aligned} w_G^{t+1} &= \sum_{i=1}^{n} \frac{n_i}{N} w_i^{t+1} \\ &= \sum_{i=1}^{n} \frac{n_i}{N} \left[w_i^t - \beta \left(I - \alpha \nabla^2 f_{local}(w_i^t)\right) \times \nabla f_{local}\left(w_i^t - \alpha \nabla f_{local}(w_i^t)\right)\right] \end{aligned} \tag{3.11}$$

Ngoài thuật toán MAML, dựa theo nghiên cứu [5], chúng tôi tích hợp thêm vào hệ thống của mình thuật toán Meta-SGD [15]. Tương tự như MAML, Meta-SGD được cấu trúc theo hướng tối ưu hai cấp độ. Tuy nhiên, có một thay đổi nhỏ giúp thuật toán này đạt độ chính xác cao hơn MAML: thuật toán coi siêu tham số học α là một tham số có thể học được và tiến hành tối ưu α trong mục tiêu cấp cao.

Tóm lại, cả hai nghiên cứu [5, 8] đã chứng minh được việc tích hợp ML vào hệ thống FL giúp đạt hiệu quả cao hơn FedAvg về độ chính xác trên

cả hai phương diện lý thuyết và thực nghiệm. Chúng tôi dựa theo nghiên cứu [5] để cài đặt hai thuật toán FedMetaMAML và FedMetaSGD với mục đích khảo sát khả năng xử lý dữ liệu Non-IID của các thuật toán ML khi được tích hợp vào hệ thống FL.

3.1.3 Thuật toán $FedMeta(MAML)$ và $FedMeta(Meta-SGD)$

Algorithm 1 FedMeta(MAML) và FedMeta(Meta-SGD) [5]

Server:
Khởi tạo w_G^0 cho MAML hoặc (w_G^0, α^0) cho Meta-SGD.
for $t = 0, 1, 2, ...$ **do**
    Chọn một tập C_t gồm m máy khách
    for $c_i \in C_t$ **do**
        Tính toán $g_i^{t+1} \leftarrow \text{ModelTrainingMAML}(c_i, w_G^t)$ cho MAML
        Tính toán $g_i^{t+1} \leftarrow \text{ModelTrainingMetaSGD}(c_i, w_G^t, \alpha^t)$ cho Meta-SGD

    Tính toán $n_i = \left|\mathcal{D}_{train(i)}^{query}\right|, N_m = \sum_{i=0}^{m} n_i$
    Cập nhật $w_G^{t+1} \leftarrow w_G^t - \beta \sum_{i=0}^{m} \frac{n_i}{N_m} g_i^{t+1}$ cho MAML
    Cập nhật $(w_G^{t+1}, \alpha^{t+1}) \leftarrow (w_G^t, \alpha^t) - \beta \sum_{i=0}^{m} \frac{n_i}{N_m} g_i^{t+1}$ cho Meta-SGD

ModelTrainingMAML(c_i, w_G^t)**:** ▷ Tại máy khách c_i
Chọn tập support $\mathcal{D}_{train(i)}^{support}$ và tập query $\mathcal{D}_{train(i)}^{query}$
$\hat{w}_i^{t+1} \leftarrow w_G^t - \alpha \nabla_{w_G^t} f_{local}(w_G^t, \mathcal{D}_{train(i)}^{support})$
$g_i^{t+1} \leftarrow \nabla_{w_G^t} f_{local}(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query})$
Gửi g_i^{t+1} về máy chủ

ModelTrainingMetaSGD(c_i, w_G^t, α^t)**:** ▷ Tại máy khách c_i
Chọn tập support $\mathcal{D}_{train(i)}^{support}$ và tập query $\mathcal{D}_{train(i)}^{query}$
$\hat{w}_i^{t+1} \leftarrow w_G^t - \alpha^t \circ \nabla_{w_G^t} f_{local}(w_G^t, \mathcal{D}_{train(i)}^{support})$
$g_i^{t+1} \leftarrow \nabla_{(w_G^t, \alpha^t)} f_{local}(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query})$
Gửi g_i^{t+1} về máy chủ

Nghiên cứu [5] đề xuất kết hợp MAML và Meta-SGD vào hệ thống FL của họ (thuật toán 1). Tuy nhiên, tác giả sử dụng kỹ thuật cập nhật bằng cách lấy trung bình đạo hàm của hàm lỗi. Trước hết, máy khách sau khi nhận

được trọng số toàn cục sẽ tiến hành tinh chỉnh (fine-tune) trọng số này trên tập dữ liệu support (phương trình 3.12 và 3.13). Như vậy, mô hình sau khi tinh chỉnh sẽ nắm bắt được các đặc trưng riêng của bộ dữ liệu. Từ đó thích ứng tốt với dữ liệu query.

$$\text{MAML: } \hat{w}_i^{t+1} \leftarrow w_G^t - \alpha \nabla_{w_G^t} f_{local}(w_G^t, \mathcal{D}_{train(i)}^{support}) \tag{3.12}$$

$$\text{Meta-SGD: } \hat{w}_i^{t+1} \leftarrow w_G^t - \alpha^t \circ \nabla_{w_G^t} f_{local}(w_G^t, \mathcal{D}_{train(i)}^{support}) \tag{3.13}$$

Trọng số sau khi tinh chỉnh được dùng để dự đoán phân lớp trên tập dữ liệu query và tính toán hàm lỗi dựa trên kết quả dự đoán:

$$\text{MAML: } g_i^{t+1} \leftarrow \nabla_{w_G^t} f_{local}(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query}) \tag{3.14}$$

$$\text{Meta-SGD: } g_i^{t+1} \leftarrow \nabla_{(w_G^t, \alpha^t)} f_{local}(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query}) \tag{3.15}$$

Kết quả đạo hàm trên được gửi trực tiếp về máy chủ để tổng hợp bộ trọng số toàn cục mới:

$$\text{MAML: } w_G^{t+1} \leftarrow w_G^t - \beta \sum_{i=0}^{m} \frac{n_i}{N_m} g_i^{t+1} \tag{3.16}$$

$$\text{Meta-SGD: } (w_G^{t+1}, \alpha^{t+1}) \leftarrow (w_G^t, \alpha^t) - \beta \sum_{i=0}^{m} \frac{n_i}{N_m} g_i^{t+1} \tag{3.17}$$

Trong nghiên cứu của mình, chúng tôi sử dụng kỹ thuật lấy trung bình trọng số trong việc tổng hợp trọng số toàn cục vì chi phí giao tiếp của việc truyền thông tin đạo hàm từ máy khách về máy chủ vượt quá giới hạn phần cứng được cung cấp. Đặt trong ngữ cảnh của ML, nếu bỏ qua việc mất gói tin trong quá trình giao tiếp giữa máy chủ và máy khách, việc sử dụng trọng số để truyền tin không ảnh hưởng đến kết quả của bài toán và được chứng minh đối với MAML như sau:

Tại bước huấn luyện toàn cục thứ t với sự tham gia của m máy khách, phương trình 3.11 trở thành:

$$
\begin{aligned}
w_G^{t+1} &= \sum_{i=1}^{m} \frac{n_i}{N_m} \left[w_i^t - \beta \left(I - \alpha \nabla^2 f_{local}(w_i^t) \right) \times \nabla f_{local} \left(w_i^t - \alpha \nabla f_{local}(w_i^t) \right) \right] \\
&= w_G^t - \beta \sum_{i=1}^{m} \frac{n_i}{N_m} \left(I - \alpha \nabla^2 f_{local}(w_i^t) \right) \times \nabla f_{local} \left(w_i^t - \alpha \nabla f_{local}(w_i^t) \right) \\
&= w_G^t - \beta \sum_{i=1}^{m} \frac{n_i}{N_m} g_c^{t+1}
\end{aligned}
\tag{3.18}
$$

Từ phương trình 3.18 và 3.16, ta suy ra điều cần chứng minh. Đối với Meta-SGD và các bước thực hiện như trên, ta thu được kết quả chứng minh tương tự.

Về phần thuật toán Meta-SGD, nghiên cứu [15] chỉ ra rằng việc sử dụng một siêu tham số học cục bộ nhỏ, được cố định theo thời gian hoặc một một siêu tham số cục bộ giảm dần theo thời gian chỉ phù hợp cho ngữ cảnh huấn luyện một mô hình với bộ dữ liệu lớn trong thười gian dài. Trong ngữ cảnh dữ liệu gắn nhãn có ít nhưng mô hình cần phải thích ứng nhanh với tập dữ liệu mới, phương pháp chọn siêu tham số như vậy không còn phù hợp nữa.

Nghiên cứu cũng đề ra một hướng tiếp cận mới cho phép tự điều chỉnh và tối ưu siêu tham số cục bộ. Theo đó, ngoài việc tối ưu tri thức tiên nghiệm (ω), thuật toán coi siêu tham số học tại cấp thấp α cũng là một tham số có thể tối ưu. Bằng việc khởi tạo α là một mảng siêu tham số có kích thước giống như w, thuật toán hướng tới việc cập nhật cả hướng đi lẫn bước học cho từng phần tử trọng số trong w bằng cách điều chỉnh α tại bước tối ưu cấp cao. Máy khách sau đó sử dụng α bằng cách nhân vô hướng đại lượng này với đạo hàm hàm lỗi cục bộ. Xét trong quan hệ với hệ thống FL, Meta-SGD vừa khiến cho các mô hình học thích ứng nhanh trên các tập dữ liệu cục bộ, vừa đóng góp lớn vào việc cá nhân hóa mô hình học cho từng người dùng.

3.2 Tiếp cận hệ thống theo hướng Personalization Layer

Chúng tôi tiến hành khảo sát kết quả của các kỹ thuật tối ưu hệ thống FL trên dữ liệu Non-IID và nhận thấy các thuật toán theo hướng PL đạt kết quả rất cao và có tiềm năng phát triển thêm (bảng 3.1). Các thuật toán theo hướng này chia mạng học sâu ra làm hai phần [28]: phần chung và phần riêng. Theo đó, phần chung được hợp tác huấn luyện bởi tất cả các máy khách trong hệ thống còn phần riêng được từng máy khách huấn luyện riêng biệt trên tập dữ liệu cục bộ. Chính các lớp học sâu trong phần riêng đã làm cho cách tiếp cận này trở nên mạnh mẽ trên dữ liệu có tính cá nhân hóa cao như dữ liệu Non-IID.

Bảng 3.1: Độ chính xác (%) của các hệ thống FL trên dữ liệu Non-IID của tập dữ liệu CIFAR-10 [20]. Các thuật toán PL được in đậm.

Thuật toán \ #clients	10	50	100
Per-FedAvg	76.65 ± 4.84	83.03 ± 0.25	80.19 ± 1.99
FedPer	87.27 ± 1.39	83.39 ± 0.47	80.99 ± 0.71
pFedMe	87.69 ± 1.93	86.09 ± 0.32	85.23 ± 0.58
LG-FedAvg	89.11 ± 2.66	85.19 ± 0.58	81.49 ± 1.56

Tuy nhiên, như câu hỏi đã được nêu ra trong phần 2.3.2, chúng tôi nhận thấy, hiệu suất của hệ thống FL cài đặt theo hướng PL vẫn có thể được cải thiện vì các lớp phần chung chưa thực sự mạnh mẽ và còn phụ thuộc nhiều vào dữ liệu. Cụ thể, các lớp phần chung này có thể bị bias do phân bố dữ liệu không đồng đều trong kịch bản Non-IID. Do đó, cần cải thiện cách huấn luyện của các lớp học sâu trong phần chung để chúng bớt phụ thuộc vào dữ liệu hơn. Nói cách khác, các lớp này cần có khả năng làm việc khách quan, "đối xử" công bằng hơn với các tập dữ liệu riêng biệt trong hệ thống FL.

Như đã thảo luận trước đó, các thuật toán ML có khả năng thích ứng nhanh trên tập dữ liệu mới thông qua việc học nhiều nhiệm vụ khác nhau. Hơn nữa, dựa vào phân tích tính tương đồng về hệ hàm mục tiêu của FL

và ML ở trên, ta có thể coi các nhiệm vụ khác nhau của ML chính là các máy khách trong hệ thống FL. Do đó, chúng tôi sử dụng phương pháp huấn luyện mô hình của ML vào huấn luyện các lớp phần chung nhằm thu được các lớp phần chung có khả năng ít bị bias trên dữ liệu huấn luyện và thích ứng tốt trên dữ liệu mới.

Hai thuật toán điển hình theo hướng PL là FedPer [2] và LG-FedAvg [16]. Trong phần này, chúng tôi tiến hành tìm hiểu, phân tích hai thuật toán trên và đưa ra các nhận xét về ưu, nhược điểm của từng thuật toán. Từ đó kết hợp chúng với các thuật toán FedMeta ở trên.

3.2.1 Thuật toán *FedPer*

#todo: chèn ảnh

Nghiên cứu [2] đề xuất kiến trúc hệ thống FL theo hướng PL bằng cách chia mạng học sâu ra làm hai phần. Phần chung được hợp tác huấn luyện bởi các máy khách trong hệ thống và được tổng hợp bởi máy chủ. Phần riêng được các máy khách độc lập huấn luyện trên dữ liệu của mình.

Các lớp học sâu trong phần chung là các lớp rút trích đặc trưng của mạng. Vì được hợp tác huấn luyện, các lớp phần chung này được tiếp xúc với đầy đủ các phân lớp dữ liệu. Do đó, sẽ có được khả năng rút trích được đặc trưng dữ liệu của tất cả các nhãn dữ liệu trong hệ thống. Điều này theo chúng tôi là rất quan trọng và là điểm khác biệt chính khi so sánh giữa FedPer và LG-FedAvg.

Phần riêng của mạng bao gồm các lớp tuyến tính ở mức cao. Phần này sẽ sử dụng các đặc trưng dữ liệu được rút trích ở trên để tính toán và quyết định một mẫu dữ liệu đầu vào sẽ thuộc phân lớp nào. Vì được duy trì riêng tại mỗi máy khách, các lớp phần riêng này được tối ưu riêng cho dữ liệu trên máy khách đó. Đây chính là điểm đáng giá của thuật toán FedPer khi nó giúp nắm bắt điểm riêng biệt trong phân phối dữ liệu của từng máy khách mà thuật toán FedAvg không thể nào làm được.

Ký hiệu $w_{P(i)}, w_B$ lần lượt là trọng số của các lớp phần riêng của máy khách c_i và phần chung của hệ thống. Hàm phân lớp cần huấn luyện tại máy khách c_i là $\hat{y}_i = g(x, w_B, w_{P(i)})$. Trong đó, trọng số w_B có nhiệm vụ rút trích các đặc trưng của dữ liệu đầu vào x còn $w_{P(i)}$ chịu trách nhiệm

phân lớp dữ liệu x cũng như lưu trữ tính cá nhân hóa của máy khách c_i. Theo đó, hàm mục tiêu của hệ thống có thể được biểu diễn:

$$\begin{aligned} &\min_{w_B, w_{P(1)}, \ldots, w_{P(n)}} f_{global}(w_B, w_{P(1)}, \ldots, w_{P(n)}) \\ &= \min_{w_B, w_{P(1)}, \ldots, w_{P(n)}} \frac{1}{n} \sum_{i=1}^{n} f_{local}(w_B, w_{P(i)}) \end{aligned} \tag{3.19}$$

Các hoạt động chính của máy chủ và máy khách trong hệ thống được trình bày trong thuật toán 3 và 2. Đầu tiên, máy chủ khởi tạo trọng số phần chung w_B^0 và gửi trọng số này đến các máy khách trong một bước huấn luyện. Máy khách c_i nhận w_B^0 từ máy chủ đồng thời khởi tạo trọng số phần riêng $w_{P(i)}^0$. Bộ trọng số $(w_B^0, w_{P(i)}^0)$ sau đó được máy khách c_i sử dụng để dự đoán và huấn luyện cục bộ. Tại bước huấn luyện toàn cục thứ t, ta có:

$$H = h(x, w_B^t) \tag{3.20}$$

$$\hat{y} = g(H, w_{P(i)}^t) \tag{3.21}$$

$$w_{B(i)}^{t+1} \leftarrow w_B^t - \alpha \nabla_{w_B^t} f_{local}(x, w_B^t, w_{P(i)}^t) \tag{3.22}$$

$$w_{P(n)}^{t+1} \leftarrow w_{P(n)}^t - \alpha \nabla_{w_{P(n)}^t} f_{local}(x, w_B^t, w_{P(n)}^t) \tag{3.23}$$

Kết thúc quá trình huấn luyện cục bộ, $w_{P(n)}^{t+1}$ được lưu trữ lại còn $w_{B(i)}^{t+1}$ được gửi về máy chủ để tổng hợp w_B^{t+1}:

$$w_B^{t+1} = \sum_{i=1}^{n} \frac{n_i}{N} w_{B(i)}^{t+1} \tag{3.24}$$

Algorithm 2 FEDPER-CLIENT(c_i, w_B^t) [2]

Require: Siêu tham số học α, trọng số w_B^t từ máy chủ

if $t = 0$ **then**
    Khởi tạo $w_{P(i)}^t$
else
    Tải lại $w_{P(i)}^t$ đã được lưu trữ trước đó
Dự đoán: $H = h(x, w_B^t)$; $\hat{y} = g(H, w_{P(i)}^t)$
Tính toán $(w_{B(i)}^{t+1}, w_{P(i)}^{t+1}) \leftarrow (w_B^t, w_{P(i)}^t) - \alpha \nabla_{(w_B^t, w_{P(i)}^t)} f_{local}(w_B^t, w_{P(i)}^t, \mathcal{D}_i)$
Gửi $w_{B(i)}^{t+1}$ về máy chủ và lưu trữ $w_{P(i)}^{t+1}$

Algorithm 3 FEDPER-SERVER [2]

Khởi tạo w_B^0
for $t = 0, 1, 2, ...$ **do**
    Chọn một tập C_t gồm m máy khách
    for $c_i \in C_t$ **do**
        Tính toán $(w_{B(i)}^{t+1}, n_i) \leftarrow$ FEDPER-CLIENT(c_i, w_B^t)
    Tính toán $n_i = |\mathcal{D}_i|$, $N_m = \sum_{i=1}^{m} n_i$
    Cập nhật $w_B^{t+1} \leftarrow \sum_{i=1}^{m} \frac{n_i}{N_m} w_{B(i)}^{t+1}$

Sau khi hoàn thành giai đoạn huấn luyện toàn cục, hệ thống thu được một trọng số phần chung và n trọng số phần riêng. Quá trình kiểm thử trên tập dữ liệu của máy khách mới được tiến hành bằng thông qua bộ tham số (w_B, w_P). Trong đó, w_P là trung bình cộng của các trọng số $(w_{P(1)}, ..., w_{P(n)})$ được tính bằng phương trình 3.25.

$$w_P = \sum_{i=1}^{n} \frac{n_i}{N} w_{P(i)} \tag{3.25}$$

Một lần nữa, trọng số của hệ thống có thể bị bias trên dữ liệu huấn luyện. Bộ trọng số $(w_{P(1)}, ..., w_{P(n)})$ có thể mang đến sự cá nhân hóa rất tốt trên các máy khách đã tồn tại từ lâu trong hệ thống; trọng số w_B được hợp tác huấn luyện nên có thể đã "quen thuộc" hơn đối với dữ liệu thuộc các nhãn khác nhau nhưng vẫn khó tránh khỏi hiện tượng bias trên dữ liệu huấn luyện. Trên các máy khách vừa tham gia hệ thống, hai trọng số này sẽ gặp tình trạng tương tự như thuật toán FedAvg: Bị giảm hiệu suất nghiêm trọng trên dữ liệu Non-IID.

3.2.2 Thuật toán $LG - FedAvg$

Ngoại trừ việc phần chung của mạng học sâu là các lớp tuyến tính còn phần riêng của mạng là các lớp rút trích đặc trưng, ý tưởng huấn luyện của thuật toán LG-FedAvg (thuật toán 4) gần như giống hoàn toàn với thuật toán FedPer. Các lớp phần riêng của thuật toán này được kỳ vọng sẽ học những đặc trưng dữ liệu của từng máy khách một cách riêng biệt. Tuy nhiên, khi đối mặt với các phân phối dữ liệu lạ trên các máy khách vừa tham gia vào hệ thống, các lớp phần riêng tỏ ra khó khăn trong việc nắm bắt các đặc trưng mới vì chúng đã được cá nhân hóa rất cao cho các đặc trưng cục bộ trước đó. Dẫn đến việc độ chính xác của hệ thống giảm từ 31% đến 34% khi hoạt động trên các máy khách mới khi so sánh với độ chính xác trên các máy khách cũ [16].

Algorithm 4 LG-FEDAVG [16]

Server:
Khởi tạo w_B^0
for $t = 1, 2, ...$ **do**
    Chọn một tập C_t gồm m máy khách
    for $c_i \in C_t$ **do**
        Tính toán $w_{B(i)}^{t+1} \leftarrow ClientUpdate(c_i, w_B^t)$
    Tính toán $n_i = |\mathcal{D}_i|, N_m = \sum_{i=1}^{m} n_i$
    Cập nhật $w_B^{t+1} \leftarrow \sum_{i=1}^{m} \frac{n_i}{N_m} w_{B(i)}^{t+1}$

ClientUpdate (c_i, w_B^t):
if $t = 0$ **then**
    Khởi tạo $w_{P(i)}^t$
else
    Tải lại $w_{P(i)}^t$ đã được lưu trữ trước đó
Dự đoán: $H = h(x, w_{P(i)}^t); \hat{y} = g(H, w_B^t)$
Tính toán $(w_{B(i)}^{t+1}, w_{P(i)}^{t+1}) \leftarrow (w_B^t, w_{P(i)}^t) - \alpha \nabla_{(w_B^t, w_{P(i)}^t)} f_{local}(w_B^t, w_{P(i)}^t, \mathcal{D}_i)$
Gửi $w_{B(i)}^{t+1}$ về máy chủ và lưu trữ $w_{P(i)}^{t+1}$

Điểm làm chúng tôi chú ý đến nghiên cứu này chính là kịch bản mà nó xây dựng trong quá trình kiểm thử rất phù hợp với các tình huống thực tế của một hệ thống client-server. Kịch bản kiểm thử của nghiên cứu [16]

chia người dùng ra làm hai loại: (1) - người dùng cũ, (2) - người dùng mới.

Đối với người dùng cũ, nghiên cứu cho rằng loại người dùng này đã tồn tại đủ lâu trong hệ thống để có thể xây dựng được một lớp cá nhân hóa cho chính nó. Khi làm việc với loại dữ liệu trên máy khách của họ, hệ thống biết chính xác nên sử dụng bộ trọng số nào là phù hợp nhất.

Đối với người dùng mới, nghiên cứu giả định họ là những người vừa tham gia vào hệ thống. Do đó, hệ thống không thể biết được nên sử dụng trọng số nào để làm việc với phân phối dữ liệu của họ. Chính vì vậy, nghiên cứu đề xuất thực hiện ensemble test trên loại người dùng này.

Dựa trên việc phân chia dữ liệu theo hướng ML, chúng tôi không đồng ý với cách tiếp cận ensemble test trên người dùng mới vì hai lý do. Thứ nhất, khả năng thích ứng trên tập dữ liệu mới của các lớp phần riêng của hệ thống bị triệt tiêu, thay vào đó là tính cá nhân hóa cho từng tập dữ liệu cục bộ mà hệ thống đã làm việc trước đó. Đứng trước một tập dữ liệu mới, các lớp phần riêng này hầu như không đạt được hiệu suất cao, dẫn đến kết quả của ensemble test không được như kỳ vọng. Thứ hai, cách tổ chức dữ liệu của hệ thống FL theo hướng ML yêu cầu chia tập dữ liệu cục bộ ra thành hai tập con (tập support và query). Mô hình học sẽ được thích ứng với dữ liệu kiểm tra thông qua một vài bước tinh chỉnh (fine-tune) trên tập support. Vì vậy, không những có thể chọn ra lớp phần riêng phù hợp nhất trong quá trình tinh chỉnh, lớp phần riêng được chọn còn có khả năng thích ứng nhanh trên tập dữ liệu mới do đã được huấn luyện bằng các thuật toán ML.

3.3 Thuật toán đề xuất: $FedMeta - Per$

3.3.1 Cấu trúc hệ thống

Theo hướng tiếp cận PL, chúng tôi đề xuất chia mạng học sâu ra thành hai phần. Phần chung bao gồm các lớp rút trích đặc trưng của mạng, được hợp tác huấn luyện bởi các máy khách và tổng hợp bởi máy chủ hệ thống. Phần riêng bao gồm các lớp tuyến tính còn lại, được duy trì tại mỗi máy khách, giúp tăng tính cá nhân hóa mô hình cho tập dữ liệu biên.

Điểm khác biệt giữa hệ thống này và hệ thống `FedPer` nằm ở chỗ các

mạng học sâu trong phần chung được huấn luyện theo hướng ML. Do đó, chúng tôi đặt tên cho hệ thống của mình là FedMeta-Per - là sự kết hợp giữa các thuật toán FedMeta và thuật toán FedPer.

Trong ba phần dưới đây, chúng tôi mô tả về cách thức mà hệ thống hoạt động trong quá trình huấn luyện (bao gồm huấn luyện cục bộ và tổng hợp toàn cục) và các kịch bản kiểm thử thuật toán.

3.3.2 Huấn luyện cục bộ

Giai đoạn huấn luyện cục bộ bằng cách sử dụng thuật toán MAML và Meta-SGD được trình bày trong thuật toán 5 và 6.

Đối với các máy khách huấn luyện theo thuật toán MAML, tại bước huấn luyện toàn cục thứ t, một máy khách c_i ban đầu sẽ nhận được trọng số khởi tạo w_B^t của phần chung do máy chủ gửi đến. Dưới hình thức huấn luyện của ML, c_i cần phải chuẩn bị hai tập dữ liệu $\mathcal{D}_{train(i)}^{support}$ và $\mathcal{D}_{train(i)}^{query}$. Tiếp đến, chúng tôi tiến hành hợp nhất trọng số phần chung w_B^t với trọng số phần riêng $w_{P(i)}^t$ (được khởi tạo ngẫu nhiên trong bước huấn luyện toàn cục đầu tiên và được tải lại trong các bước huấn luyện sau) để thu được trọng số của mô hình hoàn chỉnh w_i^t. w_i^t sau đó được dùng để thực thi các huấn luyện như sau:

$$\text{Train: } \hat{w}_i^{t+1} \leftarrow w_i^t - \alpha \nabla_{w_i^t} f_{local}(w_i^t, \mathcal{D}_{train(i)}^{support}) \tag{3.26}$$

$$\text{Meta-train: } w_i^{t+1} \leftarrow w_i^t - \beta \nabla_{w_i^t} f_{local}(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query}) \tag{3.27}$$

Đối với các máy khách sử dụng thuật toán Meta-SGD trong huấn luyện, phần chung của mạng học sâu bao gồm hai tham số: trọng số huấn luyện chung w_B^t và siêu tham số huấn luyện chung α_B^t; phần riêng của mạng bao gồm hai tham số $w_{P(i)}^t$ và $\alpha_{P(i)}^t$. Quá trình hợp nhất tham số cũng được diễn ra giữa các tham số phần chung và phần riêng để tạo thành bộ tham số hoàn chỉnh w_i^t và α_i^t. Hai tham số này sau đó cũng tham gia vào quá trình huấn luyện giống như MAML:

$$\text{Train: } \hat{w}_i^{t+1} \leftarrow w_i^t - \alpha_i^t \circ \nabla_{w_i^t} f_{local}(w_i^t, \mathcal{D}_{train(i)}^{support}) \tag{3.28}$$

$$\text{Meta-train: } (w_i^{t+1}, \alpha_i^{t+1}) \leftarrow (w_i^t, \alpha_i^t) - \beta \nabla_{(w_i^t, \alpha_i^t)} f_{local}(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query}) \tag{3.29}$$

Kết thúc quá trình huấn luyện cục bộ, trọng số mô hình mới w_i^{t+1} được phân giải thành trọng số phần chung mới $w_{B(i)}^{t+1}$ và trọng số phần riêng mới $w_{P(i)}^{t+1}$. $w_{B(i)}^{t+1}$ được gửi về máy chủ để tổng hợp w_B^{t+1} còn $w_{P(i)}^{t+1}$ được lưu lại tại bộ nhớ của máy khách. Việc phân giải, gửi về máy chủ và lưu trữ tại máy khách cũng diễn ra tương tự với siêu tham số α_i^t.

Algorithm 5 FedMeta-Per (MAML Client)

ModelTrainingMAML(c_i, w_B^t)**:**
Chọn tập support $\mathcal{D}_{train(i)}^{support}$ và tập query $\mathcal{D}_{train(i)}^{query}$
if $t = 0$ **then**
    Khởi tạo $w_{P(i)}^t$
else
    Tải lại $w_{P(i)}^t$ đã được lưu trữ trước đó
$w_i^t \leftarrow w_B^t \bigoplus w_{P(i)}^t$ ▷ Hợp nhất w_B^t và $w_{P(i)}^t$ để tạo thành w_i^t
Tính toán:

$$\hat{w}_i^{t+1} \leftarrow w_i^t - \alpha \nabla_{w_i^t} f_{local}(w_i^t, \mathcal{D}_{train(i)}^{support})$$

$$w_i^{t+1} \leftarrow w_i^t - \beta \nabla_{w_i^t} f_{local}(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query})$$

$w_{B(i)}^{t+1}, w_{P(i)}^{t+1} \leftarrow w_i^{t+1}$ ▷ Phân giải w_i^{t+1} để tạo thành $w_{B(i)}^{t+1}$ và $w_{P(i)}^{t+1}$
Gửi $w_{B(i)}^{t+1}$ về máy chủ và lưu trữ $w_{P(i)}^{t+1}$

Algorithm 6 FedMeta-Per (Meta-SGD Client)

ModelTrainingMetaSGD(c_i, w_G^t, α_B^t)**:**
Chọn tập support $\mathcal{D}_{train(i)}^{support}$ và tập query $\mathcal{D}_{train(i)}^{query}$
if $t = 0$ **then**
    Khởi tạo $(w_{P(i)}^t, \alpha_{P(i)}^t)$
else
    Tải lại $(w_{P(i)}^t, \alpha_{P(i)}^t)$ đã được lưu trữ trước đó
$w_i^t \leftarrow w_B^t \bigoplus w_{P(i)}^t$ ▷ Hợp nhất w_B^t và $w_{P(i)}^t$ để tạo thành w_i^t
$\alpha_i^t \leftarrow \alpha_B^t \bigoplus \alpha_{P(i)}^t$ ▷ Hợp nhất α_B^t và $\alpha_{P(i)}^t$ để tạo thành α_i^t
Tính toán:

$$\hat{w}_i^{t+1} \leftarrow w_i^t - \alpha_i^t \circ \nabla_{w_i^t} f_{local}(w_i^t, \mathcal{D}_{train(i)}^{support})$$

$$(w_i^{t+1}, \alpha_i^{t+1}) \leftarrow (w_i^t, \alpha_i^t) - \beta \nabla_{(w_i^t, \alpha_i^t)} f_{local}(\hat{w}_i^{t+1}, \mathcal{D}_{train(i)}^{query})$$

$w_{B(i)}^{t+1}, w_{P(i)}^{t+1} \leftarrow w_i^{t+1}$ ▷ Phân giải w_i^{t+1} để tạo thành $w_{B(i)}^{t+1}$ và $w_{P(i)}^{t+1}$
$\alpha_{B(i)}^{t+1}, \alpha_{P(i)}^{t+1} \leftarrow \alpha_i^{t+1}$ ▷ Phân giải α_i^{t+1} để tạo thành $\alpha_{B(i)}^{t+1}$ và $\alpha_{P(i)}^{t+1}$
Gửi $(w_{B(i)}^{t+1}, \alpha_B^{t+1})$ về máy chủ và lưu trữ $(w_{P(i)}^{t+1}, \alpha_{P(i)}^{t+1})$

3.3.3 Tổng hợp toàn cục

Máy chủ thi triển thuật toán 7 để tổng hợp trọng số mới toàn cục của hệ thống. Tại đây diễn ra ba quá trình cơ bản của một máy chủ FL: gửi tham số toàn cục, nhận tham số cập nhật cục bộ, tổng hợp tham số toàn cục mới (bảng 3.2).

Bảng 3.2: Bảng các tham số tại máy chủ hệ thống FedMeta-Per

	Thuật toán	
	MAML	Meta-SGD
Gửi tham số	w_B^t	(w_B^t, α_B^t)
Nhận các tham số	Các trọng số $w_{B(i)}^{t+1}$	Các tham số $(w_B(i)^t + 1, \alpha_B(i)^t + 1)$
Tổng hợp tham số	w_B^{t+1}	$(w_B^t + 1, \alpha_B^t + 1)$

Máy chủ tổng hợp các tham số nhận về bằng phương pháp lấy trung bình tham số. Theo đó, các tham số toàn cục mới của hệ thống là:

$$\text{MAML: } w_B^{t+1} \leftarrow \sum_{i=0}^{m} \frac{n_i}{N_m} w_{B(i)}^{t+1} \tag{3.30}$$

$$\text{Meta-SGD: } (w_B^{t+1}, \alpha_B^{t+1}) \leftarrow \sum_{i=0}^{m} \frac{n_i}{N_m} (w_i^{t+1}, \alpha_{B(i)}^{t+1}) \tag{3.31}$$

Algorithm 7 FedMeta-Per (Server)

Server:
Khởi tạo w_B^0 cho MAML hoặc (w_B^0, α_B^0) cho Meta-SGD.
for $t = 0, 1, 2, ...$ **do**
    Chọn một tập C_t gồm m máy khách
    for $c_i \in C_t$ **do**
        Tính toán $w_{B(i)}^{t+1} \leftarrow \text{ModelTrainingMAML}(c_i, w_B^t)$ cho MAML
        Tính toán $(w_{B(i)}^{t+1}, \alpha_{B(i)}^{t+1}) \leftarrow \text{ModelTrainingMetaSGD}(c_i, w_B^t, \alpha_B^t)$ cho Meta-SGD

    Tính toán $n_i = \left|\mathcal{D}_{train(i)}^{query}\right|, N_m = \sum_{i=0}^{m} n_i$
    Cập nhật $w_B^{t+1} \leftarrow \sum_{i=0}^{m} \frac{n_i}{N_m} w_{B(i)}^{t+1}$ cho MAML
    Cập nhật $(w_B^{t+1}, \alpha_B^{t+1}) \leftarrow \sum_{i=0}^{m} \frac{n_i}{N_m} (w_i^{t+1}, \alpha_{B(i)}^{t+1})$ cho Meta-SGD

3.3.4 Giai đoạn kiểm thử

Dựa theo quá trình kiểm thử của nghiên cứu [16], chúng tôi cũng chia ra hai loại người dùng: người dùng cũ và người dùng mới.

Đối với người dùng cũ, chúng tôi sử dụng lại các lớp học sâu thuộc phần chung để đạt được độ mức cá nhân hóa cao.

Đối với người dùng mới, chúng tôi không sử dụng kỹ thuật ensemble test vì các lý do đã nêu tại phần 3.2.2, mà cho từng lớp phần riêng đã được xây dựng trước đó kết hợp với lớp phần chung để hoạt động trên bộ dự liệu mới. Tại quá trình tinh chỉnh, chúng tôi sẽ chọn được bộ tham số cho độ lỗi nhỏ nhất. Từ đó sử dụng bộ tham số này để xử lý dữ liệu kiểm thử.

Chúng tôi nhận thấy, các thuật toán `FedMeta-Per` được phép tinh chỉnh trên tập support của máy khách trong giai đoạn kiểm thử nhưng thuật toán `FedPer` thì không. Để công bằng trong đánh giá, chúng tôi đề xuất thuật toán `FedPerMeta`. Thuật toán này cho phép mô hình huấn

luyện theo FedPer được phép chạy một hoặc một vài bước huấn luyện trên tập support của máy khách trong lúc đánh giá.

Chương 4

Cài đặt thực nghiệm

4.1 Mô tả dữ liệu

Tập dữ liệu CIFAR-10 [6] là tập dữ liệu hình ảnh được sủ dụng phổ biến trong việc huấn luyện các mô hình máy học hay các thuật toán thị giác máy tính. Đây là một trong các tập dữ liệu được dùng nhiều nhất trong quá trình nghiên cứu máy học. Tập dữ liệu bao gồm 60,000 ảnh màu kích thước 32×32 thuộc 10 phân lớp khác nhau.

Tập dữ liệu MNIST [22] là tập dữ liệu hình ảnh được sử dụng trong việc huấn luyện các hệ thống xử lý ảnh. Tập dữ liệu này cũng được sử dụng rộng rãi trong lĩnh vực học máy. Tập dữ liệu có tổng cộng 70,000 ảnh đen trắng các chữ số viết tay được viết bởi nhiều người.

Chúng tôi sử dụng hai tập dữ liệu MNIST và CIFAR-10 để đánh giá thuật toán đề xuất và các thuật toán được khảo sát. Bằng các đặc tính của hệ thống Horizontal FL và dữ liệu Non-IID, chúng tôi cấu hình mỗi máy khách chỉ gồm 2/10 phân lớp dữ liệu, số lượng nhãn giữa các lớp và số lượng dữ liệu giữa các máy khách là không đồng đều. Thống kê dữ liệu Non-IID được trình bày trong bảng 4.1.

Bảng 4.1: Thống kê trên hai tập dữ liệu MNIST và CIFAR-10 (dữ liệu Non-IID)

Dataset	#clients	#samples	#classes	#samples/client				#classes/client
				min	mean	std	max	
MNIST	50	69,909	10	135	1,398	1,424	5,201	2
CIFAR-10	50	52,497	10	506	1,049	250	1,986	2

Chúng tôi cũng tiến hành các thí nghiệm của mình trên kịch bản dữ liệu IID, nơi các máy khách có phân phối giống nhau và chứa đủ dữ liệu của 10 phân lớp. Chi tiết thống kê được trình bày trong bảng 4.2.

Bảng 4.2: Thống kê trên hai tập dữ liệu MNIST và CIFAR-10 (dữ liệu IID)

Dataset	#clients	#samples	#classes	#samples/client				#classes/client
				min	mean	std	max	
MNIST	50	70,000	10	1,395	1,400	35	1,645	10
CIFAR-10	50	60,000	10	1,200	1,200	0	1,200	10

Dữ liệu trên mỗi máy khách được chia làm hai tập: tập huấn luyện chiếm 75% tổng số điểm dữ liệu và tập kiểm tra chiếm 25% tổng số điểm dữ liệu. Theo hướng ML, chúng tôi tiếp tục chia nhỏ dữ liệu trong tập huấn luyện và tập kiểm tra thành hai tập: tập support chiếm 20% dữ liệu và tập query chiếm 80% dữ liệu. Như vậy, thực chất mô hình được huấn luyện trên 75% tổng số điểm dữ liệu (tập huấn luyện), tinh chỉnh trên 5% dữ liệu (tập support của dữ liệu kiểm tra) và kiểm thử trên 20% tổng số điểm dữ liệu (tập query của dữ liệu kiểm tra).

Trong quá trình kiểm thử, dữ liệu kiểm thử chứa trong 50 máy khách được cấu hình để tạo ra hai loại người dùng: người dùng cũ và người dùng mới. Dữ liệu của người dùng cũ được chia như đã trình bày ở trên. Đối với người dùng mới, chúng tôi tiến hành chia lại từ 25% dữ liệu tập kiểm tra sao cho phân phối của những người dùng này khác hoàn toàn với các phân phối đã tồn tại trước đó trong hệ thống.

Tóm lại, ký hiệu $C_{train} = \{c_1^{train}, ..., c_{50}^{train}\}$ là tập máy khách dùng trong huấn luyện, $C_{test} = \{c_1^{test}, ..., c_{50}^{test}\}$ là tập máy khách dùng trong kiểm thử, N là tổng số điểm dữ liệu, ta có số lượng dữ liệu huấn luyện và kiểm tra lần lượt là:

$$N_{train} = \sum_{i=1}^{50} \left|c_i^{train}\right| = 0.75N$$

$$N_{test} = \sum_{i=1}^{50} \left|c_i^{test}\right| = 0.25N$$

Trong cài đặt ML, ký hiệu $c_i^{train} = \{\mathcal{D}_{train(i)}^{support}, \mathcal{D}_{train(i)}^{query}\}$, $c_i^{test} = \{\mathcal{D}_{test(i)}^{support}, \mathcal{D}_{test(i)}^{query}\}$. Ta có số lượng dữ liệu chứa trong tập support và query của tất cả các máy khách lần lượt là:

$$N^{support}_{train/test(i)} = \left|\mathcal{D}^{support}_{train/test(i)}\right| = 0.2\left|c_i^{train/test}\right|$$

$$N^{query}_{train/test(i)} = \left|\mathcal{D}^{query}_{train/test(i)}\right| = 0.8\left|c_i^{train/test}\right|$$

Người dùng $c_j^{test} \in C_{test}$ được gọi là người dùng cũ nếu tồn tại người dùng $c_i^{train} \in C_{train}$ sao cho $p\left((x,y) \in c_j^{test}\right) = p\left((x,y) \in c_i^{train}\right)$. Ngược lại, c_j^{test} là người dùng mới nếu $p\left((x,y) \in c_j^{test}\right) \neq p\left((x,y) \in c_i^{train}\right)$ với mọi $c_i^{train} \in C_{train}$.

4.2 Phương pháp đánh giá

Sau quá trình huấn luyện mô hình toàn cục sử dụng dữ liệu trong tập C_{train}, chúng tôi thực hiện đánh giá mô hình này trên dữ liệu của tập C_{test} bằng cách ghi nhận lại hai thông tin [5]: (1) - Độ chính xác trong tương quan với tất cả các điểm dữ liệu, (2) - Độ chính xác trong tương quan với tất cả các máy khách.

Độ chính xác trong tương quan với tất cả các điểm dữ liệu được tính toán bằng cách duyệt qua tất cả các máy khách để thống kê số lượng mẫu dữ liệu được phân lớp đúng và tổng số mẫu dữ liệu, sau đó lấy thương của hai đại lượng này. Gọi r_i^t, n_i lần lượt là số lượng mẫu được phân lớp đúng tại bước huấn luyện thứ t, tổng số mẫu dữ liệu trên tập dữ liệu của người dùng c_i^{test} và n là số người dùng tham gia kiểm thử. Độ đo này tại bước huấn luyện toàn cục thứ t được tính như sau:

$$AccDataPoint^t = \frac{\sum_{i=1}^{n} r_i^t}{\sum_{i=1}^{n} n_i} \tag{4.1}$$

Độ chính xác đặt trong tương quan với tất cả các máy khách được tính bằng cách lấy trung bình cộng độ chính xác trên toàn bộ máy khách tham gia kiểm thử. Với a_i^t là độ chính xác của mô hình chạy trên máy khách c_i^{test}, tại bước huấn luyện toàn cục thứ t, ta tính toán thông tin về độ chính xác và độ lệch chuẩn của n người dùng như sau:

$$AccClient^t = \frac{1}{n}\sum_{i=1}^{n} a_i^t \tag{4.2}$$

$$\sigma^t = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i^t - AccClient^t)^2} \tag{4.3}$$

Trong đó, σ^t dùng để biểu thị mức độ phân tán trên độ chính xác của người dùng kiểm thử tại bước huấn luyện thứ t.

Các mô hình trong hệ thống của chúng tôi được đánh giá bằng độ chính xác trong tương quan với các điểm dữ liệu sau mỗi 20 bước huấn luyện toàn cục. Độ đánh giá trong tương quan với các máy khách được tính một lần duy nhất, khi mô hình toàn cục được huấn luyện xong.

4.3 Mô tả thực nghiệm

4.3.1 Kiến trúc mô hình

Chúng tôi sử dụng hai mô hình để rút trích đặc trưng và phân lớp dữ liệu cho tập dữ liệu CIFAR-10 và MNIST.

CIFAR-10. Mô hình nhận các ảnh đầu vào có kích thước $(32 \times 32 \times 3)$. Hai lớp tích chập (kernel có kích thước (5×5), số chanel lần lượt là 6 và 16) được sử dụng để rút trích đặc trưng. Theo sau mỗi lớp tích chập là một lớp `MaxPooing` có kích thước (2×2). Phần phân lớp gồm ba lớp tuyến tính có đầu ra lần lượt là 120, 84 và 10. Các hàm kích hoạt được sử dụng là `ReLU` và `Softmax`.

MNIST. Mô hình nhận các ảnh đầu vào đã được làm phẳng có kích thước (1×784). Sử dụng hai lớp tuyến tính có đầu ra lần lượt là 100 và 10. Các hàm kích hoạt được sử dụng là `ReLU` và `Softmax`.

4.3.2 Huấn luyện tập trung

Chúng tôi huấn luyện tập trung dựa theo định nghĩa về hệ thống FL trong nghiên cứu [25]. Theo đó, cần cấu hình tập dữ liệu $\mathcal{D}_{train}$ chứa 80% tổng dữ liệu. Kiến trúc mạng học sâu mô tả trong phần 4.3.1 sẽ được huấn luyện trên tập dữ liệu này. Mô hình sau khi huấn luyện được thực thi trên tập dữ liệu $\mathcal{D}_{test}$ chứa 20% tổng dữ liệu. Các tập $\mathcal{D}_{train}, \mathcal{D}_{test}$ có dữ liệu tuân theo phân phối đều. Kết quả về độ chính xác sau khi kiểm thử gọi là kết quả huấn luyện tập trung.

4.3.3 Huấn luyện phân tán

Trước hết chúng tôi cài đặt và huấn luyện hệ thống FL bằng thuật toán FedAvg để có kết quả đối sánh chính. Thuật toán này được huấn luyện trên toàn bộ dữ liệu của tập C_{train} và thực hiện kiểm thử trên các tập $\mathcal{D}_{test(i)}^{query}$ của từng người dùng trong tập C_{test}.

Khi thuật toán FedAvg cập nhật xong mô hình toàn cục, trong lúc kiểm thử, chúng tôi cho phép mô hình này thực hiện tinh chỉnh một hoặc một vài bước huấn luyện trên tập dữ liệu $\mathcal{D}_{test(i)}^{support}$ của những người dùng trong tập C_{test}. Đây chính là ý tưởng của thuật toán FedAvgMeta, thuật toán sinh ra nhằm so sánh công bằng với các thuật toán FedMeta.

Các thuật toán FedMeta và FedMeta-Per tiến hành huấn luyện như đã trình bày tại chương 3.

Để dễ dàng trong việc tinh chỉnh các mô hình học cũng như phù hợp với phần cứng thấp của máy khách trong hệ thống Horizontal FL, chúng tôi cố định một vài tham số: mỗi bước huấn luyện toàn cục sẽ sử dụng 5 máy khách, lượng dữ liệu trong một lô (batch) là 32, số bước huấn luyện cục bộ là 1 bước, số lượng bước huấn luyện cục bộ là 300 đối với tập dữ liệu MNIST và 600 đối với tập dữ liệu CIFAR-10.

Các thuật toán được huấn luyện và kiểm thử trên dữ liệu Non-IID với hai kịch bản kiểm thử: người dùng mới và người dùng cũ. Riêng thuật toán FedAvg, FedAvgMeta được chạy trên cả dữ liệu IID lẫn Non-IID để minh họa tác động của dữ liệu Non-IID đối với hệ thống.

Các thuật toán FedMeta được cài đặt và sử dụng trong huấn luyện mô hình để kiểm tra khả năng thích ứng nhanh trên tập dữ liệu mới của ML khi được tích hợp vào hệ thống FL. Ngoài ra, việc này còn dùng để lấy dữ liệu so sánh với thuật toán FedMeta-Per.

Thuật toán FedPer được cài đặt sử dụng trong huấn luyện mô hình để chứng minh việc các lớp phần chung của hệ thống chưa đạt hiệu quả cao khi làm việc trên tập dữ liệu của người dùng mới. Kết quả này cũng được đem so sánh với kết của của FedMeta-Per.

Thuật toán FedMeta-Per được cài đặt sử dụng trong huấn luyện mô hình để kiểm tra khả năng thích ứng nhanh trên tập dữ liệu mới của các lớp phần chung và khả năng cá nhân hóa dựa trên từng tập dữ liệu của

các lớp phần riêng. Ngoài ra, cần kiểm chứng độ chính xác của thuật toán này so với các thuật toán FedMeta, FedPer và FedAvg.

Tất cả các thí nghiệm của chúng tôi được giả lập bằng framework Flower 0.17.0 [3] trên một máy chủ duy nhất. Theo đó, máy chủ và các máy khách trong hệ thống đều sử dụng chung các tài nguyên tính toán (CPU và GPU) và giao tiếp với nhau thông qua giao thức gRPC.

Tại máy chủ, một tiến trình được đặt ra để "lắng nghe" kết nối từ các máy khách tại một cổng (port) cố định. Khi các máy khách được khởi tạo xong, máy chủ tiến hành khởi tạo tham số toàn cục bằng cách chọn ngẫu nhiên bộ tham số từ một máy khách. Tại một bước huấn luyện toàn cục, máy chủ chọn ngẫu nhiên một tập con các máy khách để gửi thông tin tới máy khách thông qua giao thức gRPC. Thông tin này giúp máy khách biết mình cần thực hiện huấn luyện hay kiểm thử mô hình. Thông tin huấn luyện bao gồm tham số toàn cục, siêu tham số huấn luyện, số bước huấn luyện cục bộ, lượng dữ liệu trong một lô dữ liệu (batch). Thông tin kiểm thử bao gồm tham số toàn cục, số bước tinh chỉnh cục bộ (nếu có), siêu tham số huấn luyện (nếu có), lượng dữ liệu trong một lô dữ liệu (nếu có). Sau khi thực hiện xong yêu cầu của máy chủ, máy khách sẽ gửi thông tin về. Trong trường hợp máy chủ yêu cầu máy khách thực hiện huấn luyện, thông tin máy chủ nhận về là các tham số cục bộ và số điểm dữ liệu tham gia huấn luyện ra tham số cục bộ đó. Máy chủ tiến hành tổng hợp tham số toàn cục từ các thông tin này. Đối với yêu cầu kiểm thử mô hình, máy chủ sẽ nhận được thông tin về độ chính xác, độ lỗi, số điểm dữ liệu được kiểm thử trên các mô hình cục bộ.

Tại máy khách, sau khi khởi tạo, máy khách kết nối đến máy chủ thông qua cổng được chỉ định trước và đợi các thông tin gửi đến từ máy chủ. Dựa trên thông tin này, máy khách sẽ thực hiện huấn luyện hay kiểm thử cục bộ. Sau đó gửi các thông tin được yêu cầu về máy chủ.

Chương 5

Kết quả & Thảo luận

5.1 Kết quả huấn luyện tập trung

5.2 *FedAvg* trên dữ liệu Non-IID

Hệ thống FL do Google đề xuất được giới thiệu là có thể thực thi tốt không những trên dữ liệu IID mà còn trên cả dữ liệu Non-IID [18]. Tuy nhiên, nhiều nghiên cứu khác ([5, 27, 28, 23]) không đồng ý với quan điểm này. Chúng tôi cấu hình hai môi trường dữ liệu IID và Non-IID làm dữ liệu đầu vào cho thuật toán `FedAvg` và thu được kết quả như bảng 5.1. Quan sát bảng kết quả, ta có thể nhận thấy độ chính xác của `FedAvg` giảm khoảng 6% trên tập dữ liệu MNIST và giảm lên đến 32% trên tập dữ liệu CIFAR-10 khi phân phối dữ liệu chuyển từ IID sang Non-IID. Mô hình trên tập dữ liệu CIFAR-10 thậm chí còn không đạt hội tụ.

Kỹ thuật tinh chỉnh cục bộ được thêm vào hệ thống FL dưới dạng đơn giản giúp cải thiện một phần độ chính xác của mô hình. Thật vậy, khi cho phép mô hình học tinh chỉnh bằng dữ liệu của tập support trên máy khách trong lúc kiểm thử, độ chính xác thu được so với trước khi áp dụng kỹ thuật này tăng nhẹ trên tập dữ liệu MNIST và tăng hơn gấp đôi trên tập dữ liệu CIFAR-10. Tuy nhiên, điều đáng buồn ở đây là mô hình trên tập dữ liệu CIFAR-10 vẫn không đạt hội tụ. Độ chính xác thu được không ổn định `#todo: chèn con hình acc k hội tụ`.

Nguyên nhân dẫn đến việc hiệu suất của hệ thống bị giảm nghiêm trọng là vì mô hình toàn cục không được huấn luyện trên một phân phối đều, dẫn đến việc bị bias trên dữ liệu huấn luyện. Trong quá trình kiểm thử, khi đứng trước một phân phối dữ liệu lạ, mô hình toàn cục không thể nào nắm bắt được đặc trưng dữ liệu một cách hiệu quả dẫn đến việc phân lớp không chính xác. Một cách dễ hiểu, dữ liệu kiểm thử trên các máy khách

có tính cá nhân hóa cao và rất khác với những gì mô hình đã được huấn luyện trước đó nên mô hình hoạt động không tốt.

Nắm bắt được nguyên nhân trên, người ta đề xuất duy trì một phần của mạng học sâu tại máy khách. Đây chính là kỹ thuật sử dụng lớp cá nhân hóa. Chúng tôi cài đặt lại thuật toán FedPer để chạy trên dữ liệu của mình. Tuy nhiên, kết quả thu được không tốt lắm (bảng 5.1). Trên tập dữ liệu CIFAR-10, độ chính xác không những không được cải thiện mà còn tệ hơn thuật toán FedAvg. Trên tập dữ liệu MNIST, mô hình học bị quá khớp dẫn đến việc giảm hiệu suất từ bước huấn luyện toàn cục thứ 180 từ khoảng 84% xuống còn như kết quả trong bảng.

Trong khi đó, như thông tin về độ chính xác được trình bày trong bảng 3.1, độ chính xác của thuật toán FedPer đạt 83.39±0.47% (hệ thống gồm 50 máy khách trên dữ liệu CIFAR-10 Non-IID). Chúng tôi tìm hiểu và nhận thấy mạng học sâu mà FedPer sử dụng để tạo ra kết quả trên là mạng MobileNet-v1 [13], một mạng học sâu phức tạp hơn kiến trúc của chúng tôi rất nhiều lần. Nhờ vào MobileNet-v1, hệ thống FL rút trích được nhiều đặc trưng hơn và đạt kết quả tốt hơn. Tuy nhiên, chi phí phần cứng cho việc duy trì các lớp phần riêng tại máy khách cũng tăng lên. Trái lại, mạng học sâu chúng tôi sử dụng rất đơn giản, giúp làm giảm đáng kể chi phí phần cứng nhưng hiệu quả đem lại rất cao #todo: chèn ref tới phần chứng minh cái này.

Bảng 5.1: Bảng độ chính xác (%) của thuật toán FedAvg, FedAvgMeta, FedPerMeta tính trên điểm dữ liệu (dữ liệu IID và Non-IID)

	CIFAR-10			MNIST		
	IID	Non-IID		IID	Non-IID	
		local client	new client		local client	new client
FedAvg	**53.37**	22.25	21.38	**91.31**	85.49	85.69
FedAvgMeta	-	40.14	46.82	-	85.99	85.66
FedPer	-	15.73	12.54	-	64.01	58.68

5.3 *FedMeta* trên dữ liệu Non-IID

Nhằm kiểm chứng khả năng thích ứng nhanh trên tập dữ liệu mới của ML, cũng như hiệu suất của hệ thống FL có tích hợp ML, chúng tôi thí

nghiệm trên các thuật toán FedMeta và thu được kết quả như bảng 5.2. Theo đó, ML giúp cải thiện độ chính xác của hệ FL từ 10% đến 53% so với thuật toán FedAvg và từ 10% đến 35% so với thuật toán FedAvgMeta, một phiên bản được đưa ra nhằm so sánh công bằng với các thuật toán FedMeta.

Bảng 5.2: Bảng độ chính xác (%) của thuật toán FedAvg và các thuật toán FedMeta tính trên điểm dữ liệu (dữ liệu Non-IID)

	CIFAR-10		MNIST	
	local client	new client	local client	new client
FedAvg	22.25	21.38	85.49	85.69
FedAvgMeta	40.14	46.82	85.99	85.66
FedMeta(MAML)	66.58	64.86	93.09	92.55
FedMeta(Meta-SGD)	**75.75**	**70.02**	**97.36**	**95.89**

Để có cái nhìn sâu hơn về hệ thống này, chúng tôi tiến hành so sánh và phân tích khả năng hội tụ của chúng với thuật toán FedAvg trên hai loại máy khách tham gia kiểm thử (hình 5.1). Một điểm chung rất dễ nhận thấy khi quan sát quá trình hội tụ của các thuật toán FedMeta và FedAvg là các thuật toán FedMeta hội tụ nhanh hơn và đạt độ chính xác cao hơn. Đối với tập dữ liệu CIFAR-10, chỉ trong vòng 25 bước (đối với người dùng mới) và 100 bước (đối với người dùng cũ) huấn luyện cục bộ đầu tiên, độ chính xác đạt được đã tiệm cận ngưỡng hội tụ trong khi FedAvg và FedAvgMeta không ổn định và không đạt hội tụ sau 600 bước huấn luyện. Tập dữ liệu MNIST với dữ liệu ảnh đen trắng khiến việc huấn luyện trở nên dễ dàng hơn: chỉ sau 50 bước huấn luyện (trên cả người dùng mới lẫn người dùng cũ), các thuật toán đã đều gần chạm ngưỡng hội tụ của mình. Tuy nhiên, mức hội tụ của các thuật toán FedMeta vẫn tỏ ra nổi trội khi bỏ xa hai thuật toán còn lại 20% chỉ trong 50 bước huấn luyện đầu tiên.

Việc hội tụ này càng đặc biệt hơn khi mô hình toàn cục thể hiện sự thích ứng với tập dữ liệu mới rất nhanh khi chỉ thi triển một bước huấn luyện trên 20% dữ liệu kiểm tra tại máy khách. Điều này đã chứng minh được khả năng thích ứng nhanh trên tập dữ liệu mới của các thuật toán ML. Chúng tôi kỳ vọng khả năng này vẫn được duy trì trong thuật toán đề xuất, nơi chúng tôi sử dụng ML để tối ưu các lớp phần chung.

Tuy nhiên, điều khiến chúng tôi chưa vừa ý với hệ thống này nằm ở chỗ, khả năng cá nhân hóa cho từng mô hình học là chưa cao. Thật vậy, các thuật toán theo hướng PL được trình bày trong bảng 3.1 có độ lệch chuẩn nhỏ hơn rất nhiều lần so với kết quả của các thuật toán FedMeta (bảng 5.3). Trong khi đó, các thuật toán PL không thực hiện tinh chỉnh mô hình của mình trên tập dữ liệu kiểm thử. Việc này chứng tỏ hai điều: (1) - Khả năng cá nhân hóa của các thuật toán PL đến từ các lớp phần riêng được đặt tại máy khách, (2) - Việc duy trì các lớp phần riêng trên máy khách cho khả năng cá nhân hóa cao hơn việc tinh chỉnh mô hình toàn cục trên tập support. Trong nghiên cứu của mình, chúng tôi vừa cho phép mô hình toàn cục tinh chỉnh trên tập dữ liệu support của máy khách trước khi kiểm thử, vừa duy trì các lớp phần riêng của mạng học sâu trên các máy khách. Việc này được kỳ vọng giúp làm tăng tính cá nhân hóa hơn nữa cho hệ thống FL.

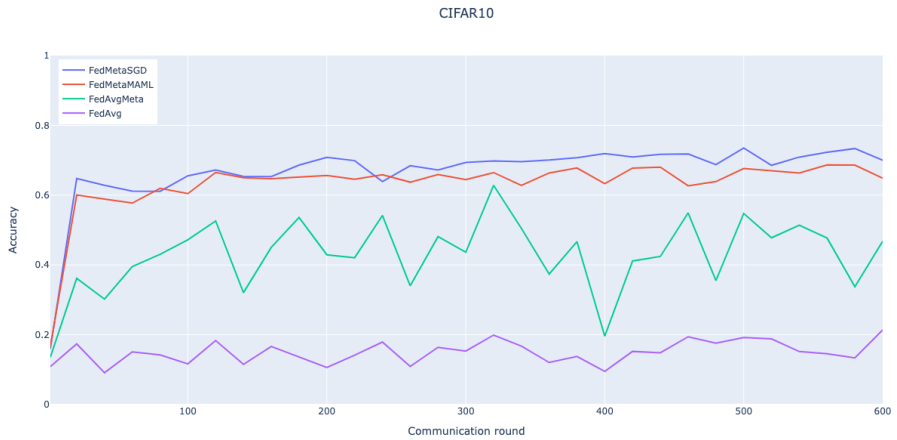
Bảng 5.3: Bảng độ chính xác (%) của thuật toán FedAvg và các thuật toán FedMeta tính trên máy khách (dữ liệu Non-IID)

	CIFAR-10		MNIST	
	local client	new client	local client	new client
FedAvg	22.10±21.57	23.23±17.02	81.23±16.95	82.59±18.63
FedAvgMeta	38.32±32.38	40.18±22.57	81.67±14.48	83.15±16.28
FedMeta(MAML)	66.31±13.72	61.34±19.99	**92.63±5.04**	**92.79±5.29**
FedMeta(Meta-SGD)	**76.01±11.98**	**67.98±15.28**	92.21±10.19	90.45±11.18

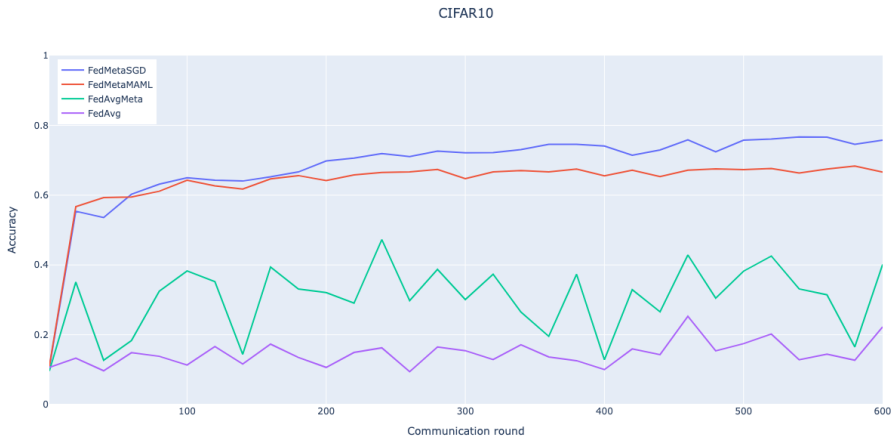
5.4 *FedMeta – Per* trên dữ liệu Non-IID

Với hai kỳ vọng vừa trình bày về khả năng thích ứng nhanh trên dữ liệu mới và khả năng cá nhân hóa mô hình học cho từng người dùng, chúng tôi thử nghiệm thuật toán đề xuất trên hai tập dữ liệu CIFAR-10, MNIST và đạt được kết quả như bảng 5.4. Theo đó, các thuật toán FedMeta-Per luôn cho kết quả cao nhất và cao hơn các thuật toán FedPer và FedPerMeta từ 2 đến 6 lần trên cả hai tập dữ liệu CIFAR-10 và MNIST.

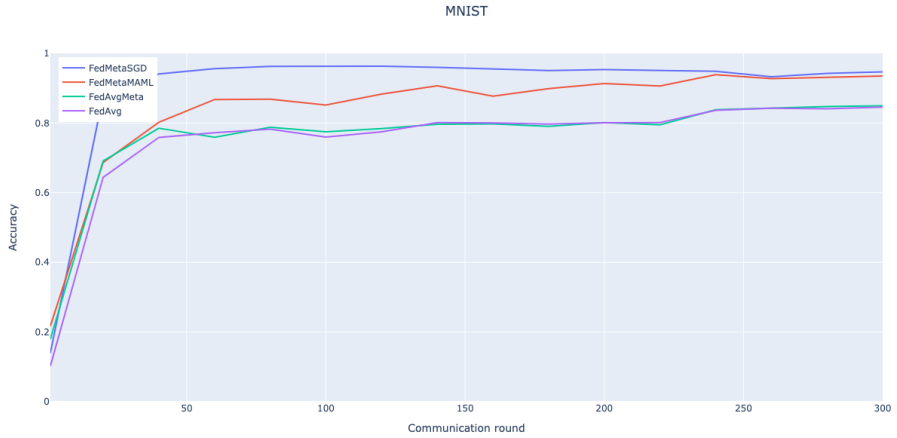
Xét về khả năng thích ứng nhanh trên tập dữ liệu mới, FedMeta-Per cho kết quả tốt hơn hẳn so với FedPerMeta khi chỉ cần thực hiện một bước huấn luyện trên tập dữ liệu support (chỉ chiếm 20% dữ liệu) của máy



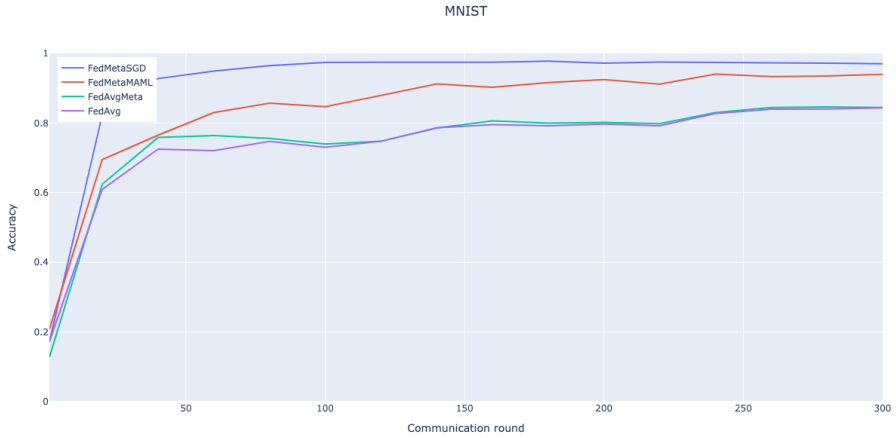
(a) CIFAR-10, new clients



(b) CIFAR-10, local clients



(c) MNIST, new clients



(d) MNIST, local clients

Hình 5.1: Kết quả chạy thuật toán FedAvg và các thuật toán FedMeta trên các tập dữ liệu và máy khách kiểm thử tương ứng

Bảng 5.4: Bảng độ chính xác (%) của thuật toán FedPer và các thuật toán FedMeta-Per tính trên điểm dữ liệu (dữ liệu Non-IID)

	CIFAR-10		MNIST	
	local client	new client	local client	new client
FedPer	15.73	12.54	64.01	58.68
FedPerMeta	18.13	25.60	52.71	63.29
FedMeta-Per(MAML)	85.73	66.19	**99.35**	93.21
FedMeta-Per(Meta-SGD)	**86.20**	**72.99**	99.00	**96.63**

Bảng 5.5: Bảng độ chính xác (%) của thuật toán FedPer và các thuật toán FedMeta-Per tính trên máy khách (dữ liệu Non-IID)

	CIFAR-10		MNIST	
	local client	new client	local client	new client
FedPer	15.41±20.90	13.91±20.40	63.67±20.04	55.06±26.09
FedPerMeta	17.46±23.15	20.99±22.61	54.94±23.14	59.43±27.31
FedMeta-Per(MAML)	85.58±7.40	61.81±21.63	**99.33±0.89**	94.06±7.73
FedMeta-Per(Meta-SGD)	**86.10±6.42**	**67.21±16.57**	98.69±1.41	**95.33±6.59**

khách trong lúc kiểm thử, đã có thể nắm bắt được các đặc tính dữ liệu trong máy khách đó và cho kết quả dự đoán phân lớp với độ chính xác rất cao. Từ hình 5.2, có thể quan sát thấy việc hội tụ của các thuật toán này cũng xảy ra nhanh hơn rất nhiều với hai thuật toán còn lại. Kết quả này tương đồng với kết quả thí nghiệm của các thuật toán `FedMeta`. Từ những ý phân tích vừa nêu, chúng tôi kết luận rằng, các thuật toán `FedMeta-Per` đã thừa hưởng thành công khả năng thích ứng nhanh trên tập dữ liệu mới của các thuật toán ML như chúng tôi đã kỳ vọng trước đó.

Xét về khả năng cá nhân hóa mô hình cho từng người dùng, chúng tôi dựa vào độ chính xác trong tương quan với máy khách được trình bày trong bảng 5.5. Bảy trong số tám các trường hợp các thuật toán `FedMeta-Per` có độ lệch chuẩn nhỏ hơn thuật toán `FedPer` (trừ trường hợp kiểm định thuật toán `FedMeta-Per(MAML)` trên người dùng mới của tập CIFAR-10). Điều này chứng tỏ rằng khả năng cá nhân hóa của thuật toán đề xuất tốt hơn so với thuật toán `FedPer` và `FedPerMeta`.

Mặt khác, chúng tôi so sánh các thuật toán `FedMeta` với các thuật toán `FedMeta-Per` và tổng hợp kết quả trong hai bảng 5.6 và 5.7. Theo đó, độ chính xác của `FedMeta-Per` luôn cao hơn `FedMeta` từ 1% đến 19% khi

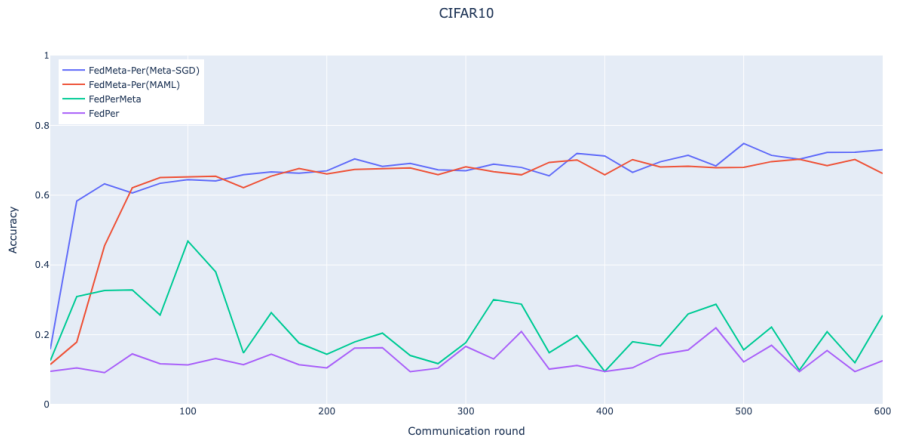
Bảng 5.6: Bảng so sánh độ chính xác (%) giữa FedMeta và FedMeta-Per tính trên điểm dữ liệu (dữ liệu Non-IID)

	CIFAR-10		MNIST	
	local client	new client	local client	new client
FedMeta(MAML)	66.58	64.86	93.09	92.55
FedMeta(Meta-SGD)	75.75	70.02	97.36	95.89
FedMeta-Per(MAML)	85.73	66.19	**99.35**	93.21
FedMeta-Per(Meta-SGD)	**86.20**	**72.99**	99.00	**96.63**

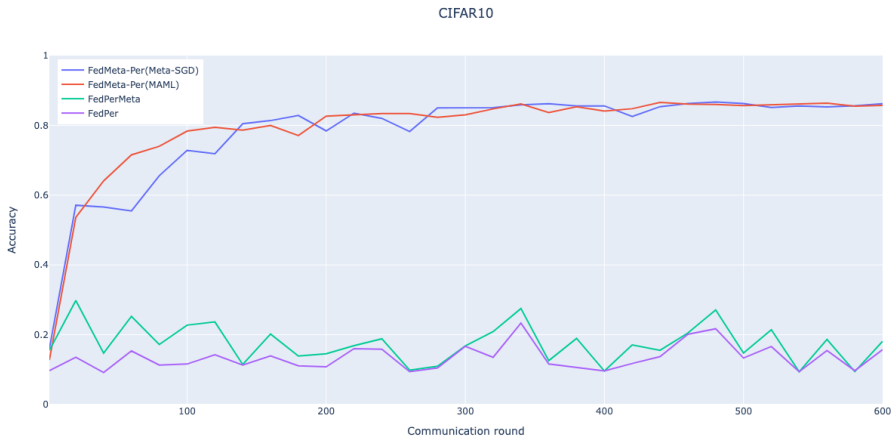
Bảng 5.7: Bảng so sánh độ chính xác (%) giữa FedMeta và FedMeta-Per tính trên máy khách (dữ liệu Non-IID)

	CIFAR-10		MNIST	
	local client	new client	local client	new client
FedMeta(MAML)	66.31±13.72	61.34±19.99	92.63±5.04	92.79±5.29
FedMeta(Meta-SGD)	76.01±11.98	**67.98±15.28**	92.21±10.19	90.45±11.18
FedMeta-Per(MAML)	85.58±7.40	61.81±21.63	**99.33±0.89**	94.06±7.73
FedMeta-Per(Meta-SGD)	**86.10±6.42**	67.21±16.57	98.69±1.41	**95.33±6.59**

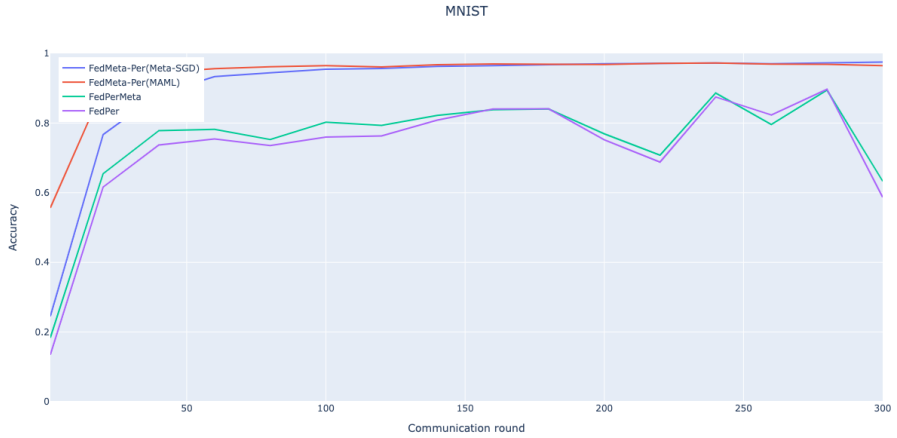
so sánh trên cùng thuật toán huấn luyện cục bộ. Tính cá nhân hóa của `FedMeta-Per` cũng cao hơn đáng kể so với `FedMeta` khi thuật toán đề xuất của chúng tôi, trong đa số các trường hợp, đã làm giảm độ lệch chuẩn của độ chính xác xuống từ 2 đến 10 lần xét trên cùng thuật toán huấn luyện cục bộ.



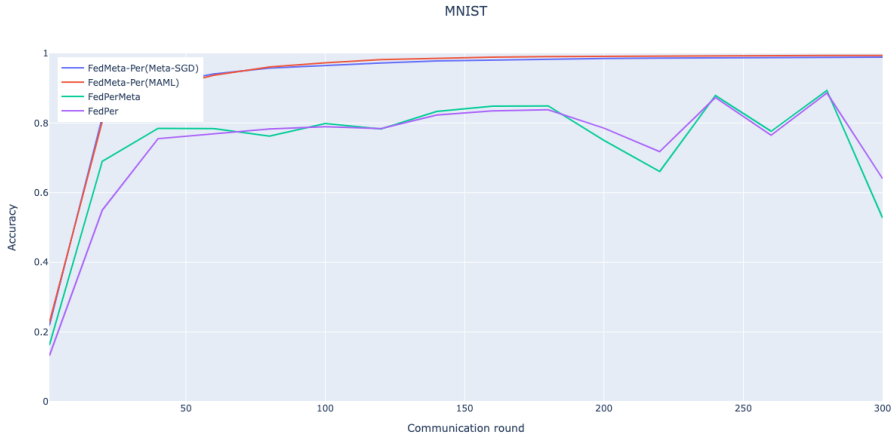
(a) CIFAR-10, new clients



(b) CIFAR-10, local clients



(c) MNIST, new clients



(d) MNIST, local clients

Hình 5.2: Kết quả chạy thuật toán FedPer và các thuật toán FedMeta-Per trên các tập dữ liệu và máy khách kiểm thử tương ứng

Chương 6

Kết luận

Tài liệu tham khảo

References

[1] Yoshinori Aono et al. "Privacy-preserving deep learning via additively homomorphic encryption". In: *IEEE Transactions on Information Forensics and Security* 13.5 (2017), pp. 1333–1345.

[2] Manoj Ghuhan Arivazhagan et al. "Federated learning with personalization layers". In: *arXiv preprint arXiv:1912.00818* (2019).

[3] Daniel J Beutel et al. "Flower: A Friendly Federated Learning Research Framework". In: *arXiv preprint arXiv:2007.14390* (2020).

[4] Kapil Chandorikar. *Introduction to Federated Learning and Privacy Preservation*. 2020. URL: `https://towardsdatascience.com/introduction-to-federated-learning-and-privacy-preservation-75644686b559`.

[5] Fei Chen et al. "Federated meta-learning with fast convergence and efficient communication". In: *arXiv preprint arXiv:1802.07876* (2018).

[6] *CIFAR-10 and CIFAR-100 datasets*. 2022. URL: `https://www.cs.toronto.edu/~kriz/cifar.html`.

[7] Moming Duan et al. "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications". In: *2019 IEEE 37th international conference on computer design (ICCD)*. IEEE. 2019, pp. 246–254.

[8] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. "Personalized federated learning: A meta-learning approach". In: *arXiv preprint arXiv:2002.07948* (2020).

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1126–1135.

[10] Avishek Ghosh et al. “An efficient framework for clustered federated learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19586–19597.

[11] Harry F Harlow. “The formation of learning sets.” In: *Psychological review* 56.1 (1949), p. 51.

[12] Timothy Hospedales et al. “Meta-learning in neural networks: A survey”. In: *arXiv preprint arXiv:2004.05439* (2020).

[13] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[14] Qinbin Li et al. “A survey on federated learning systems: vision, hype and reality for data privacy and protection”. In: *IEEE Transactions on Knowledge and Data Engineering* (2021).

[15] Zhenguo Li et al. “Meta-sgd: Learning to learn quickly for few-shot learning”. In: *arXiv preprint arXiv:1707.09835* (2017).

[16] Paul Pu Liang et al. “Think locally, act globally: Federated learning with local and global representations”. In: *arXiv preprint arXiv:2001.01523* (2020).

[17] Wei Yang Bryan Lim et al. “Federated learning in mobile edge networks: A comprehensive survey”. In: *IEEE Communications Surveys & Tutorials* 22.3 (2020), pp. 2031–2063.

[18] Brendan McMahan et al. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[19] H Brendan McMahan et al. “Learning differentially private recurrent language models”. In: *arXiv preprint arXiv:1710.06963* (2017).

[20] Aviv Shamsian et al. “Personalized federated learning using hypernetworks”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9489–9502.

[21] Martin A Tanner and Wing Hung Wong. "The calculation of posterior distributions by data augmentation". In: *Journal of the American statistical Association* 82.398 (1987), pp. 528–540.

[22] *The mnist database*. 2022. URL: `http://yann.lecun.com/exdb/mnist/`.

[23] Kangkang Wang et al. "Federated evaluation of on-device personalization". In: *arXiv preprint arXiv:1910.10252* (2019).

[24] Qiang Yang et al. "Federated machine learning: Concept and applications". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19.

[25] Xuefei Yin, Yanming Zhu, and Jiankun Hu. "A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions". In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–36.

[26] Tehrim Yoon et al. "Fedmix: Approximation of mixup under mean augmented federated learning". In: *arXiv preprint arXiv:2107.00233* (2021).

[27] Yue Zhao et al. "Federated learning with non-iid data". In: *arXiv preprint arXiv:1806.00582* (2018).

[28] Hangyu Zhu et al. "Federated Learning on Non-IID Data: A Survey". In: *arXiv preprint arXiv:2106.06843* (2021).