Federated Learning & Meta Learning

Đặt vấn đề

- FL là phương pháp huấn luyện, nhằm huấn luyện mô hình máy học trực tiếp trên các tập dữ liệu riêng biệt (phân bố trên thiết bị của người dùng). So với phương pháp huấn luyện truyền thống, FL giúp đảm bảo quyền riêng tư dữ liệu của người dùng.
- Tuy nhiên, phương pháp này cũng gặp rất nhiều thách thức: Các client không đồng nhất về dữ liệu, phần cứng, không đảm bảo bảo mật, chi phí truyền tin cao. Tại đây giải quyết vấn đề dữ liệu không đồng nhất (dữ liệu Non-IID) trên các client.
- Ngoài ra, nhóm còn nghiên cứu và đề xuất

Sơ lược về các dạng Non-IID data

- Dữ liệu Non-IID là dữ liệu mà ở đó, phân phối dữ liệu giữa các client là hoàn toàn khác nhau.
- Các dạng Non-IID bao gồm:
  - Non-IID về thuộc tính, thời gian: Không thuộc phạm vi giải quyết của khóa luận.
  - Non-IID về nhãn dữ liệu:
    - **Phân phối số lượng nhãn giữa các client khác nhau**: Với mọi client i, j: $P_i(y) \neq P_j(y)$ và $P_i(y|x) = P_j(y|x)$ (Ví dụ: Với cùng 1 ảnh x, các client sẽ gắn cho nó cùng 1 nhãn). Đây là loại Non-IID được xét đến trong khóa luận, điển hình cho một hệ thống Horizontal FL.
    - $P_i(y|x) \neq P_j(y|x)$: Với cùng một ảnh x, client i có thể gắn nhãn khác với client j. Ví dụ, cùng một ảnh con mèo nhưng sẽ có người thích người không thích. Không được xét đến trong khóa luận.

Khảo sát hướng tối ưu hệ thống FL cho Non-IID data

- Thuật toán FedAvg sử dụng một server (đóng vai trò là đối tượng duy trì global model) để phân phối/tổng hợp global model tới/từ các client (đóng vai trò là đối tượng lưu trữ dữ liệu). Kiến trúc này đã được chứng minh là không thể trụ vững trước dữ liệu Non-IID.
- Các hướng tối ưu đã được đề xuất như sau:
  - Xử lý về data:
    - **Tăng cường dữ liệu**: Sinh ra ảnh mới mang một hoặc một số nhãn nhất định để giải quyết vấn đề mất cân bằng nhãn trên tập dữ liệu bên.

        - **Chia sẻ dữ liệu**: Các client chia sẻ một phần dữ liệu nhỏ cho server để huấn luyện global model, việc này có thể làm tăng accuracy lên thêm 30% bằng cách chia sẻ chỉ 5% dữ liệu người dùng. Vi phạm mục đích ban đầu về bảo mật của FL.
    - Xử lý thuật toán:
        - **Meta learning, Local finetune**: Các thuật toán Meta learning được biết đến với khả năng thích ứng tốt trên tập dữ liệu mới. Tại đây, hệ thống FL coi mỗi client là một task cần xử lý trong bài toán meta learning.
        - **Personalized layer**: Mỗi mô hình biên được chia làm 2 phần, cả 2 phần này đều được huấn luyện tại biên nhưng phần based layer do server tổng hợp và phân phối, phần personalized layer được các client duy trì. Điều này giúp cho các client thích ứng trực tiếp với dữ liệu cục bộ.
        - **Transfer learning, Multi-task learning, Life-long learning, Server optimization**: Chưa khảo sát.
    - Xử lý hệ thống:
        - **Gom cụm người dùng**: Phân client vào các cụm khác nhau. Mỗi cụm có một global model riêng. Yếu tố để phân cụm client là weight hoặc loss của client gửi về server. Tuy nhiên chi phí giao tiếp của hệ thống sẽ lớn.
        - **Thích ứng tại thiết bị biên**: Chưa khảo sát.

Tích hợp ML vào hệ thống FL (FedMeta - thuật toán của paper)

Tóm tắt thuật toán

- Kết hợp các thuật toán MAML và Meta-SGD vào hệ thống FL.
- Ký hiệu:
    - θ là tham số mô hình.
    - α, β là siêu tham số.
    - $D_S^u = \{(x, y)\}, D_Q^u = \{(x', y')\}$ lần lượt là tập *support* và *query* của người dùng.
    - l là hàm lỗi.
    - m là số client tham gia huấn luyện trong 1 round.
- Mã giả của thuật toán được minh họa như hình dưới đây [3]:

Algorithm 1: `FedMeta` with MAML and Meta-SGD

// Run on the server
AlgorithmUpdate:
Initialize θ for MAML, or initialize (θ, α) for Meta-SGD.
for each episode $t = 1, 2, ...$ **do**
  Sample a set U_t of m clients, and distribute θ (for MAML) or (θ, α) (for Meta-SGD) to the sampled clients.
  for each client $u \in U_t$ in parallel **do**
    Get test loss $g_u \leftarrow \text{ModelTrainingMAML}(\theta)$ or
    $g_u \leftarrow \text{ModelTrainingMetaSGD}(\theta, \alpha)$
  end
  Update algorithm paramters $\theta \leftarrow \theta - \frac{\beta}{m} \sum_{u \in U_t} g_u$ for MAML or
  $(\theta, \alpha) \leftarrow (\theta, \alpha) - \frac{\beta}{m} \sum_{u \in U_t} g_u$ for Meta-SGD.
end

// Run on client u

ModelTrainingMAML(θ):
Sample support set D_S^u and query set D_Q^u
$\mathcal{L}_{D_S^u}(\theta) \leftarrow \frac{1}{|D_S^u|} \sum_{(x,y) \in D_S^u} \ell(f_\theta(x), y)$
$\theta_u \leftarrow \theta - \alpha \nabla \mathcal{L}_{D_S^u}(\theta)$
$\mathcal{L}_{D_Q^u}(\theta_u) \leftarrow \frac{1}{|D_Q^u|} \sum_{(x',y') \in D_Q^u} \ell(f_{\theta_u}(x'), y')$
$g_u \leftarrow \nabla_\theta \mathcal{L}_{D_Q^u}(\theta_u)$
Return g_u to server

ModelTrainingMetaSGD(θ, α):
Sample support set D_S^u and query set D_Q^u
$\mathcal{L}_{D_S^u}(\theta) \leftarrow \frac{1}{|D_S^u|} \sum_{(x,y) \in D_S^u} \ell(f_\theta(x), y)$
$\theta_u \leftarrow \theta - \alpha \circ \nabla \mathcal{L}_{D_S^u}(\theta)$
$\mathcal{L}_{D_Q^u}(\theta_u) \leftarrow \frac{1}{|D_Q^u|} \sum_{(x',y') \in D_Q^u} \ell(f_{\theta_u}(x'), y')$
$g_u \leftarrow \nabla_{(\theta,\alpha)} \mathcal{L}_{D_Q^u}(\theta_u)$
Return g_u to server

Thí nghiệm

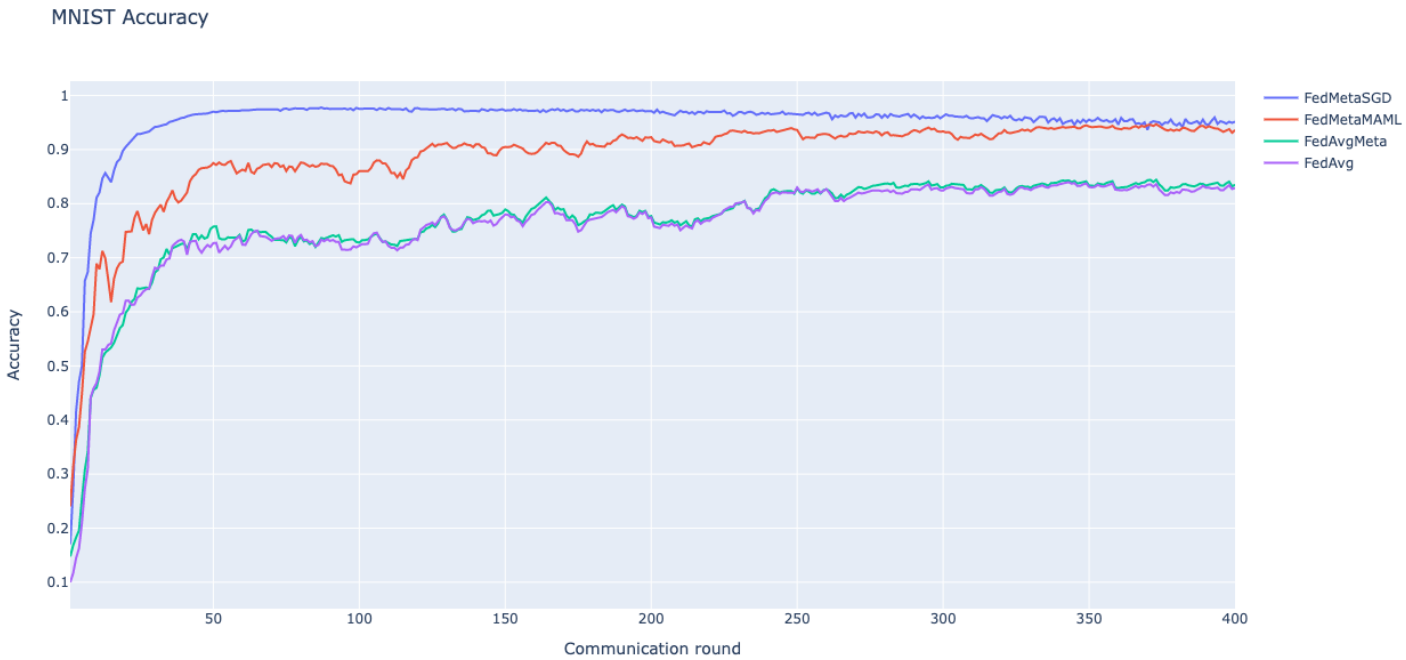
- Thông tin tập dữ liệu:

	Cifar10	**MNIST**
Số mẫu	60,000	70,000
Số client	50	50
Số mẫu lớn nhất trên 1 client	1,500	3,900

- Phân chia dữ liệu thí nghiệm: Mỗi client (mỗi episode) chỉ chứa 2 nhãn dữ liệu, được chia thành tập *support* và *query*. Trong đó, tập *support* chiếm 20% dữ liệu của client. Tập huấn luyện bao gồm 50 client, chiếm 80% tổng số mẫu dữ liệu. Tập kiểm tra bao gồm 50 client, chiếm 20% tổng số mẫu dữ liệu.
- Mô hình huấn luyện:
  - Cifar: Nhận vào ảnh có kích thước $(32 \times 32 \times 3)$. Sử dụng 2 layer `Convolution` để rút trích đặc trưng (kernel có kích thước (5×5), chanel lần lượt là 6 và 16); 2 layer `MaxPooling` kích thước (2×2); 3 layer tuyến tính có đầu ra lần lượt là $120, 84$ và 10. Tất cả đều đi kèm hàm kích hoạt `ReLU` ngoại trừ layer cuối có hàm kích hoạt là `Softmax`.

    - MNIST: Nhận vào ảnh có kích thước (28×28). Sử dụng 2 layer tuyến tính có đầu ra lần lượt là 100 và 10. Theo sau layer đầu tiên là hàm kích hoạt `ReLU`, theo sau layer thứ hai là hàm `Softmax`.
- Tổ chức thí nghiệm: Cả 2 tập dữ liệu đều có chung các tham số: $client/round = 5, epochs = 1, batchSize = 32$. Sau mỗi round huấn luyện, tiến hành kiểm tra mô hình trên tập kiểm tra.
    - Cifar: Chạy 600 rounds, fine-tune các tham số học.
    - MNIST: Chạy 400 rounds, fine-tune các tham số học.
    - Đối với các thuật toán: Sử dụng 4 thuật toán: `FedAvg` (baseline), `FedAvgMeta` (huấn luyện như `FedAvg` nhưng khi test thì được phép fine-tune trên tập *query* của client), `FedMetaMAML`, `FedMetaSGD`.

Kết quả & nhận xét



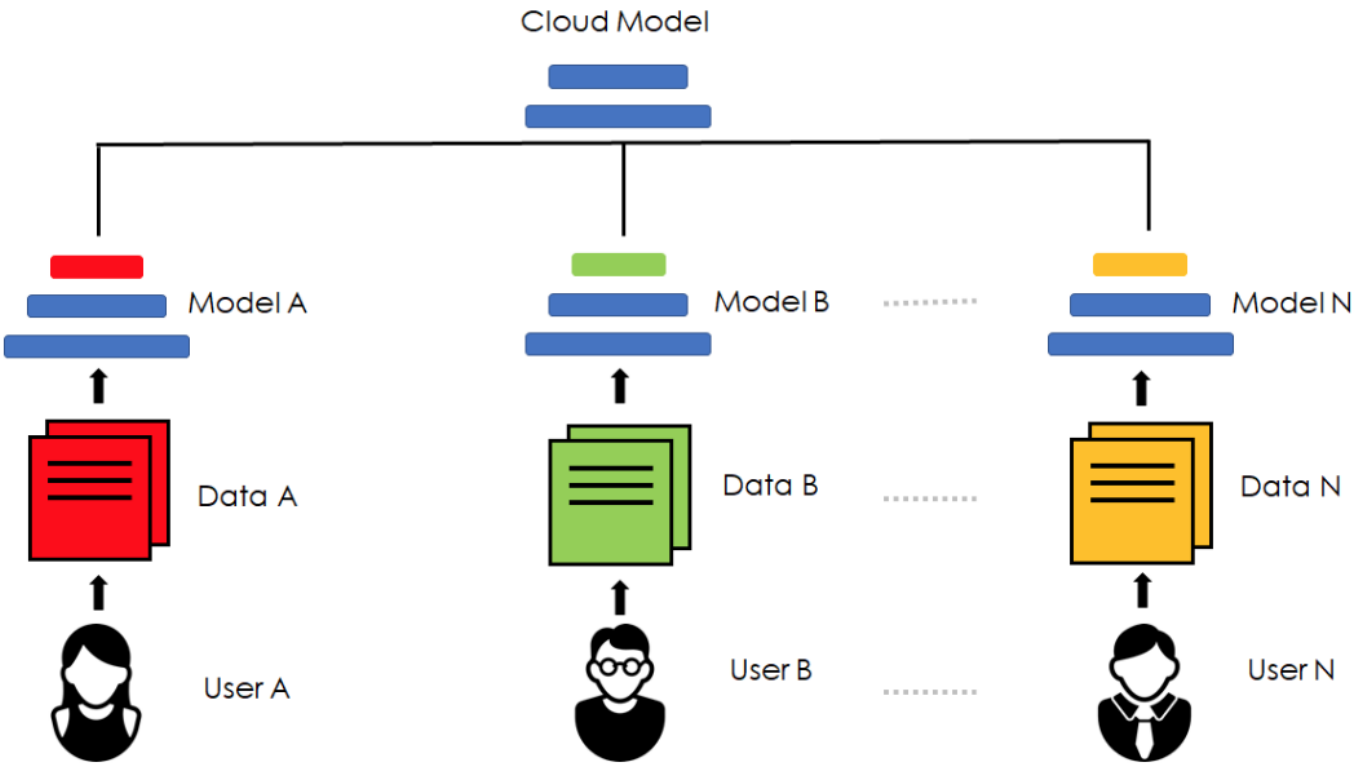
- Bảng kết kết quả:

	FedAvg	FedAvgMeta	FedMetaMAML	FedMetaSGD
Cifar10	16.62%	30.79%	75.52%	**76.54%**
MNIST	82.9%	83.57%	93.46%	**95.03%**

- Nhận xét: Thuật toán `FedAvg` và `FedAvgMeta` khi chạy trên tập MNIST, đều cho khả năng hội tụ tương đương nhau. Tuy nhiên, khi chạy trên tập Cifar10, chúng không thể hội tụ được. Trái lại, các thuật toán Meta không những chạy tốt trên cả 2 tập dữ liệu (đạt độ chính xác cao) mà còn cho khả năng hội tụ nhanh hơn so với `FedAvg` và `FedAvgMeta` .

Personalized layer

Tóm tắt thuật toán



- Hướng tiếp cận của thuật toán nằm ở chỗ, nó coi mạng neural bao gồm 2 thành phần: phần base layer và phần personalization layer. Base layer được huấn luyện như thuật toán FedAvg. Phần personalized layer chỉ được huấn luyện tại các thiết bị cuối. Việc này giúp chống lại các phân phối thống kê không đồng nhất trong dữ liệu non-IID.
- Ký hiệu: n_j là số mẫu tham gia huấn luyện của client j, W_B là tham số của base layer, $W_{P,j}$ là tham số lớp personalization của client j.
- Mã giả của thuật toán FedPer được đề xuất như sau [2]:

Algorithm 1 FedPer-Client(j)

Require: $f(\cdot;\cdot,\cdot), e, b, \{(\boldsymbol{x}_{j,i}, y_{j,i}) \mid i \in \{1,\dots,n_j\}\}$
Require: $\eta_j^{(k)}$ for $k \in \mathbb{Z}_+$
1: Initialize $\mathbf{W}_{P_j}^{(0)}$ at random
2: Send n_j to server
3: **for** $k = 1, 2, \dots$ **do**
4: Receive $\mathbf{W}_B^{(k-1)}$ from server
5: $\left(\mathbf{W}_{B,j}^{(k)}, \mathbf{W}_{P_j}^{(k)}\right) \leftarrow \mathrm{SGD}_j\left(\mathbf{W}_B^{(k-1)}, \mathbf{W}_{P_j}^{(k-1)}, \eta_j^{(k)}\right)$
6: Send $\mathbf{W}_{B,j}^{(k)}$ to server
7: **end for**

Algorithm 2 FedPer-Server

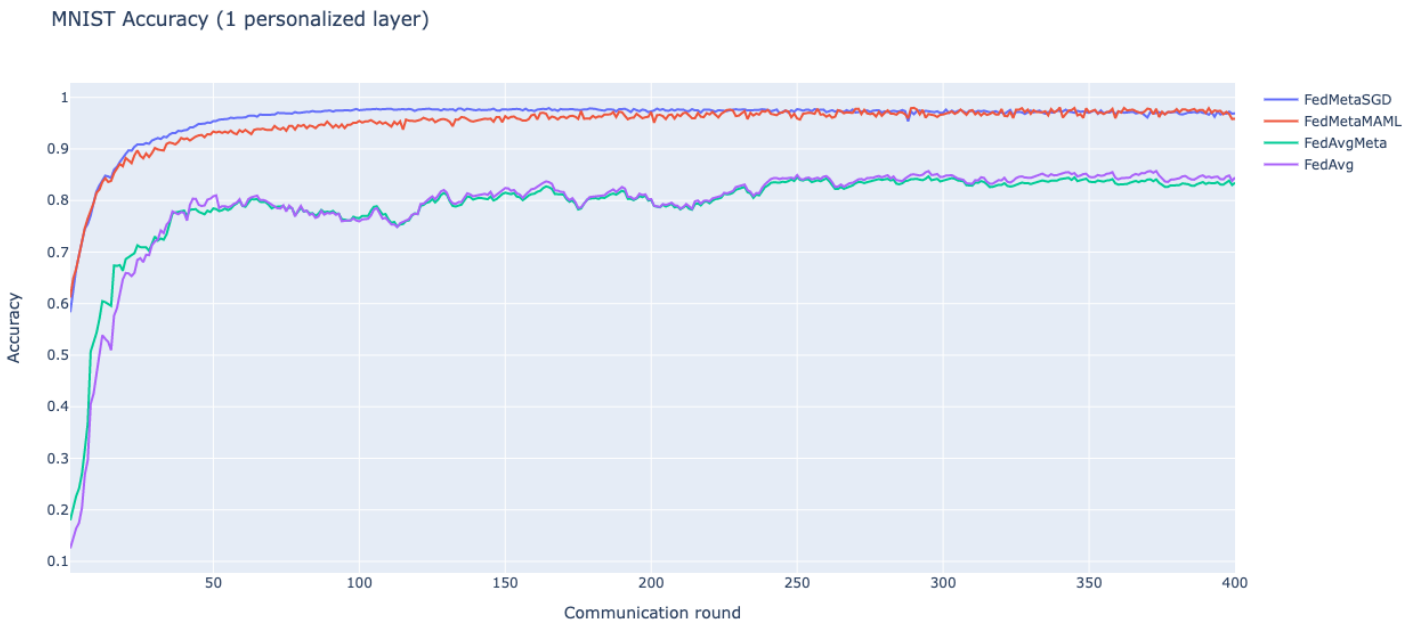
1: Initialize $\mathbf{W}_B^{(0)}$ at random
2: Receive n_j from each client $j \in \{1,\dots,N\}$ and compute $\gamma_j = n_j / \sum_{j=1}^N n_j$
3: Send $\mathbf{W}_B^{(0)}$ to each client
4: **for** $k = 1, 2, \dots$ **do**
5: Receive $\mathbf{W}_{B,j}^{(k)}$ from each client $j \in \{1,\dots,N\}$
6: Aggregate $\mathbf{W}_B^{(k)} \leftarrow \sum_{j=1}^N \gamma_j \mathbf{W}_{B,j}^{(k)}$
7: Send $\mathbf{W}_B^{(k)}$ to each client
8: **end for**

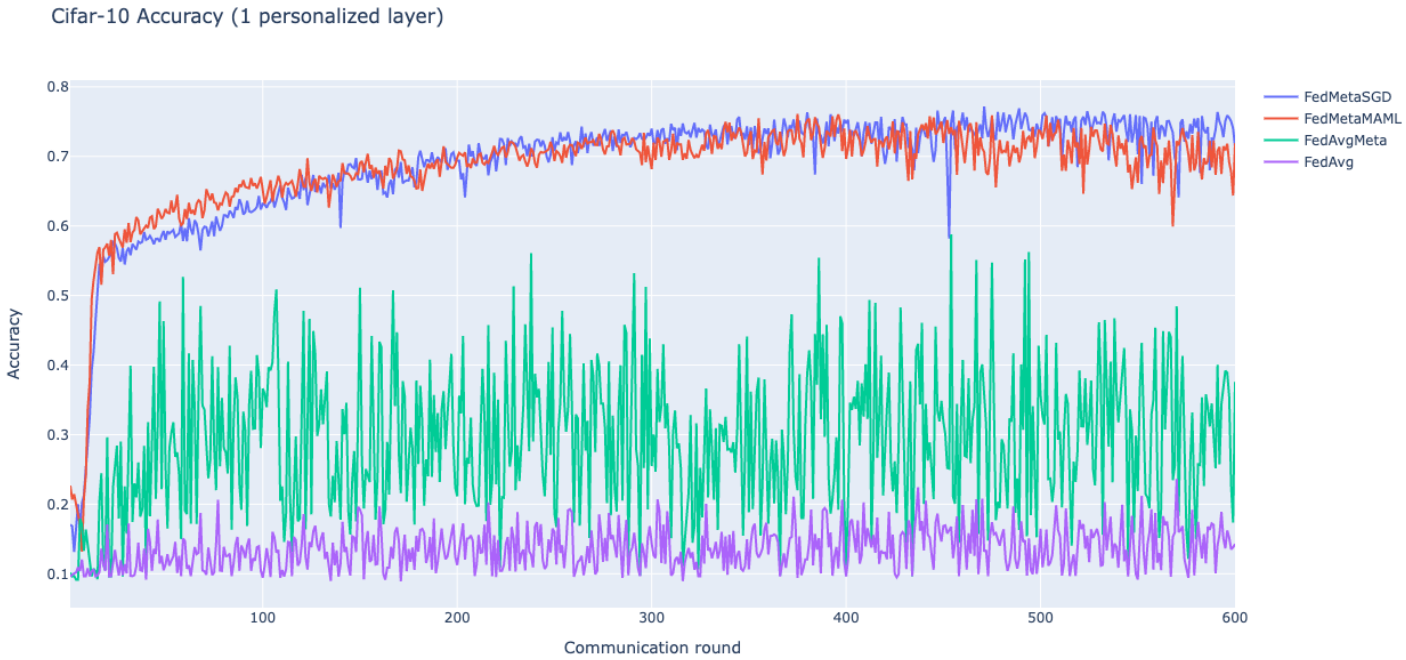
- **Đề xuất**: Kết hợp ý tưởng của FedPer với FedMeta bằng cách huấn luyện mạng neural theo hướng meta learning đồng thời giữ lại một phần mạng neural để huấn luyện trực tiếp tại thiết bị biên.

Thí nghiệm

- Thông tin tập dữ liệu và mô hình huấn luyện: Như đã nêu ở thí nghiệm trên.
- Tổ chức thí nghiệm: Cả 2 tập dữ liệu đều có chung các tham số: $client/round = 5, epochs = 1, batchSize = 32$. Sau mỗi round huấn luyện, tiến hành kiểm tra mô hình trên tập kiểm tra.
  - Cifar: Chạy 600 round, dùng lại các siêu tham số như ở thí nghiệm trước, fine-tune số lớp dùng làm personalized layer.
  - MNIST: Chạy 400 round, dùng lại các siêu tham số như ở thí nghiệm trước, số lớp personalized layer là 1.

Kết quả & Nhận xét





- Bảng kết kết quả:

	FedAvg	FedAvgMeta	FedMetaMAML	FedMetaSGD
Cifar10	14.6%	30.4%	68.99%	**74,37%**
MNIST	84.37%	83.57%	96.47%	**96.95%**

- Nhận xét: Khi kết hợp Personalized layer cho các thuật toán FedAvg, FedAvgMeta, FedMetaMAML, FedMetaSGD, độ chính xác thu được gần như tương tự như trước khi kết hợp.

Tài liệu tham khảo

[1]: Federated Learning on Non-IID Data: A Survey, URL: https://arxiv.org/pdf/2106.06843.pdf

[2]: Federated Learning with Personalization Layers, URL: https://arxiv.org/pdf/1912.00818.pdf

[3]: Federated Meta-Learning with Fast Convergence and Efficient Communication, URL: https://arxiv.org/pdf/1802.07876.pdf