# HỌC LIÊN KẾT (FEDERATE LEARNING) VÀ BẢO MẬT TẠI SERVER

## Báo cáo thực hành lab 02

Nguyễn Minh Vũ  Phùng Hoài Thi

Đại học Khoa học Tự nhiên, ĐHQG-HCM

Trí tuệ nhân tạo cho an ninh thông tin
Ngày 31 tháng 10 năm 2024

# Tổng quan

# Đánh giá công việc

| Yêu cầu | Phụ trách | Mức độ hoàn thành |
|---|---|---|
| Đọc dữ liệu | Vũ | 100% |
| Xử lý dữ liệu | Vũ | 100% |
| Cài đặt mô hình hồi quy logistic | Vũ | 100% |
| Cài đặt mô hình 1 server, 3 clients | Vũ, Thi | 100% |
| Báo cáo về mô hình tấn công Adversarisal Examples | Thi | 100% |
| Viết báo cáo | Vũ, Thi | 100% |
| Làm slide | Vũ, Thi | 100% |

Bảng 1: Bảng đánh giá công việc

# Bộ dữ liệu CIFAR10

CIFAR10 là tập dữ liệu hình ảnh được sử dụng phổ biến trong các mô hình học máy và thị giác máy tính. Trong đó, CIFAR10 chứa $60,000$ ảnh màu với kích thước $32 \times 32$ chia đều cho 10 lớp và tỉ lệ train:test là $5:1$.
Tập dữ liệu được phân thành 5 train_batch và 1 test_batch, mỗi batch gồm $10,000$ ảnh.
Hướng dẫn cài đặt dữ liệu ở local:

- Cài đặt thủ công: tải từ trang cs.toronto.edu
- Sử dụng Torchvision: dùng lệnh "*torchvision.datasets.CIFAR10*"
- Sử dụng Keras: dùng lệnh "*keras.datasets.cifar10.load_data()*"

# Đọc dữ liệu trong Python


Hình 1: Dữ liệu CIFAR10

# Vấn đề của Hồi quy Logistic

Mô hình Hồi quy Logistic là bộ phân loại tuyến tính thường được dùng rộng rãi với bài toàn phân lớp nhị phân (2-lớp), nhưng không được hiệu quả đối với dạng dữ liệu phức tạp như CIFAR10 ($32 \times 32 \times 3$ và có tính phi tuyến cao).
Một biến thể khác của Logistic Regression là Softmax Regression, khi này output của mô hình sẽ là xác suất các lớp của dữ liệu đầu vào, và dữ liệu sẽ được phân vào lớp các xác suất cao nhất.

# Rút trích đặc trưng

Tuy Softmax Regression có thể hoạt động trên bài toán phân loại đa lớp nhưng dạng dữ liệu thô như CIFAR10 khi huấn luyện với tính toán độ lỗi SparseCategoricalEntropy sẽ cao dẫn đến mô hình học kém hiệu quả.
Vì vậy một cách tiếp cận khác đó là áp dụng phương pháp **Rút trích đặc trưng (Feature Extraction)**. Phương pháp này có thể chia thành 2 nhóm chính là thủ công (SIFT, HoG,...) và áp dụng Deep Learning (mạng tích chập CNN,...).

# Histogram of Oriented Gradient

Histogram of Oriented Gradients (HoG) [3] là một phương pháp trích xuất đặc trưng mô tả cấu trúc và hình dạng của đối tượng trong ảnh, tập trung vào việc tính toán gradient và hướng cạnh tại từng vùng của ảnh.

# Histogram of Oriented Gradient

## Cài đặt HoG

```
normalize = True
block_norm = 'L2-Hys'
orientations = 9
pixels_per_cell = [8, 8]
cells_per_block = [2, 2]
def extractFeature(img, vis=False):
    from skimage.feature import hog
    return hog(img, orientations, pixels_per_cell,
               cells_per_block, block_norm,
               visualize = vis,
               transform_sqrt=normalize)
```

# Mạng tích chập CNN

Mạng tích chập CNN cho phép trích xuất đặc trưng từ dữ liệu thô mà không cần định nghĩa cụ thể trước. Một trong những cải tiến của CNN là **ResNet (Residual Network)**.
Trong thư viện Keras, ta có thể load model và sử dụng cho rút trich đặc trưng như sau:

```
def preprocess_image_input(input_images):
  input_images = input_images.astype('float32')
  input_images = tf.keras.layers.UpSampling2D(size=
  (sampling, sampling))(input_images)
  output_ims = preprocess_input(input_images)
  return output_ims
fe_model = ResNet50(weights='imagenet',
    include_top=False, input_shape=(image_size[0]
    * sampling, image_size[1] * sampling,
    image_size[2]))
```

# Logistic Regression

Mô hình hồi quy Logistic là một loại mô hình phân lớp dùng để dự đoán xác suất của điểm dữ liệu nhị phân (ví dụ như “có” hoặc “không”, “chó” và “meo”). Mô hình này sử dụng hàm sigmoid để biến đổi giá trị đầu ra thành xác suất từ 0 đến 1. Một biến thể khác của Logistic Regression là Softmax Regression, khi này xác suất sẽ được trải đều cho nhiều lớp (lớn hơn 2) do đó Softmax được dùng cho bài toán phân lớp dữ liệu CIFAR10 sẽ phù hợp hơn.

# Logistic Regression

Cài đặt mô hình cho bài toán phân lớp CIFAR10 với Keras:

```
def create_model(n_features):
    """Creates a Logistic Regression model."""
    model = Sequential([
        Input(shape=(n_features)),
        Flatten(),
        Dense(10, activation='softmax')
    ])
    model.compile(optimizer=Adam(learning_rate=LR),
    loss=SparseCategoricalCrossentropy(from_logits
    =False), metrics=['accuracy'])
    return model
```

# Logistic Regression

Trong đó, các thông số huấn luyện mô hình bao gồm:

- Sử dụng phương pháp tối ưu Adam.
- Chọn hàm mất mát SparseCategoricalCrossentropy.
- Áp dụng chỉ số lường độ chinh xác Accuracy.

# Huấn luyện mô hình

Áp dụng các phương pháp rút trích đặc trưng đã giới thiệu ở phần trước và huấn luyện mô hình, lưu kết quả tốt nhất để đánh giá.

```
    checkpoint = ModelCheckpoint(
    'model/model.keras',
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)
model.fit(X_training, y_training, epochs=num_epochs,
batch_size=batch_size, validation_data=
(X_testing, y_testing, callbacks=[checkpoint])
```

# Kết quả đánh giá - Không trích xuất đặc trưng

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Airplane | 0.2406 | 0.1470 | 0.1825 | 1000 |
| Automobile | 0.6081 | 0.0900 | 0.1568 | 1000 |
| Bird | 0.4375 | 0.0140 | 0.0271 | 1000 |
| Cat | 0.2209 | 0.3130 | 0.2590 | 1000 |
| Deer | 1.0000 | 0.0030 | 0.0060 | 1000 |
| Dog | 0.3125 | 0.0250 | 0.0463 | 1000 |
| Frog | 0.3538 | 0.4490 | 0.3958 | 1000 |
| Horse | 0.4610 | 0.3190 | 0.3771 | 1000 |
| Ship | 0.2062 | 0.8590 | 0.3326 | 1000 |
| Truck | 0.3190 | 0.5050 | 0.3910 | 1000 |
| **Accuracy** | | 0.2724 | | 10000 |
| **Macro Avg** | 0.4160 | 0.2724 | 0.2174 | 10000 |
| **Weighted Avg** | 0.4160 | 0.2724 | 0.2174 | 10000 |

Bảng 2: Bảng đánh giá không dùng đặc trưng

# Kết quả đánh giá - Đặc trưng HoG

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Airplane | 0.5961 | 0.5800 | 0.5879 | 1000 |
| Automobile | 0.5981 | 0.6190 | 0.6084 | 1000 |
| Bird | 0.4433 | 0.3870 | 0.4132 | 1000 |
| Cat | 0.4365 | 0.2820 | 0.3426 | 1000 |
| Deer | 0.4752 | 0.3920 | 0.4296 | 1000 |
| Dog | 0.4210 | 0.4530 | 0.4364 | 1000 |
| Frog | 0.4677 | 0.6650 | 0.5491 | 1000 |
| Horse | 0.5255 | 0.6080 | 0.5637 | 1000 |
| Ship | 0.5921 | 0.5850 | 0.5885 | 1000 |
| Truck | 0.6289 | 0.6320 | 0.6304 | 1000 |
| **Accuracy** | | 0.5203 | | 10000 |
| **Macro Avg** | 0.5184 | 0.5203 | 0.5150 | 10000 |
| **Weighted Avg** | 0.5184 | 0.5203 | 0.5150 | 10000 |

Bảng 3: Bảng đánh giá với đặc trưng HoG

# Kết quả đánh giá - Đặc trưng ResNet50

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Airplane | 0.7610 | 0.8470 | 0.8017 | 1000 |
| Automobile | 0.8526 | 0.8910 | 0.8714 | 1000 |
| Bird | 0.7417 | 0.7380 | 0.7398 | 1000 |
| Cat | 0.6647 | 0.5750 | 0.6166 | 1000 |
| Deer | 0.7546 | 0.6580 | 0.7030 | 1000 |
| Dog | 0.7453 | 0.7110 | 0.7277 | 1000 |
| Frog | 0.7581 | 0.8870 | 0.8175 | 1000 |
| Horse | 0.7323 | 0.8700 | 0.7952 | 1000 |
| Ship | 0.8734 | 0.8420 | 0.8574 | 1000 |
| Truck | 0.9269 | 0.7730 | 0.8430 | 1000 |
| **Accuracy** | | 0.7792 | | 10000 |
| **Macro Avg** | 0.7811 | 0.7792 | 0.7773 | 10000 |
| **Weighted Avg** | 0.7811 | 0.7792 | 0.7773 | 10000 |

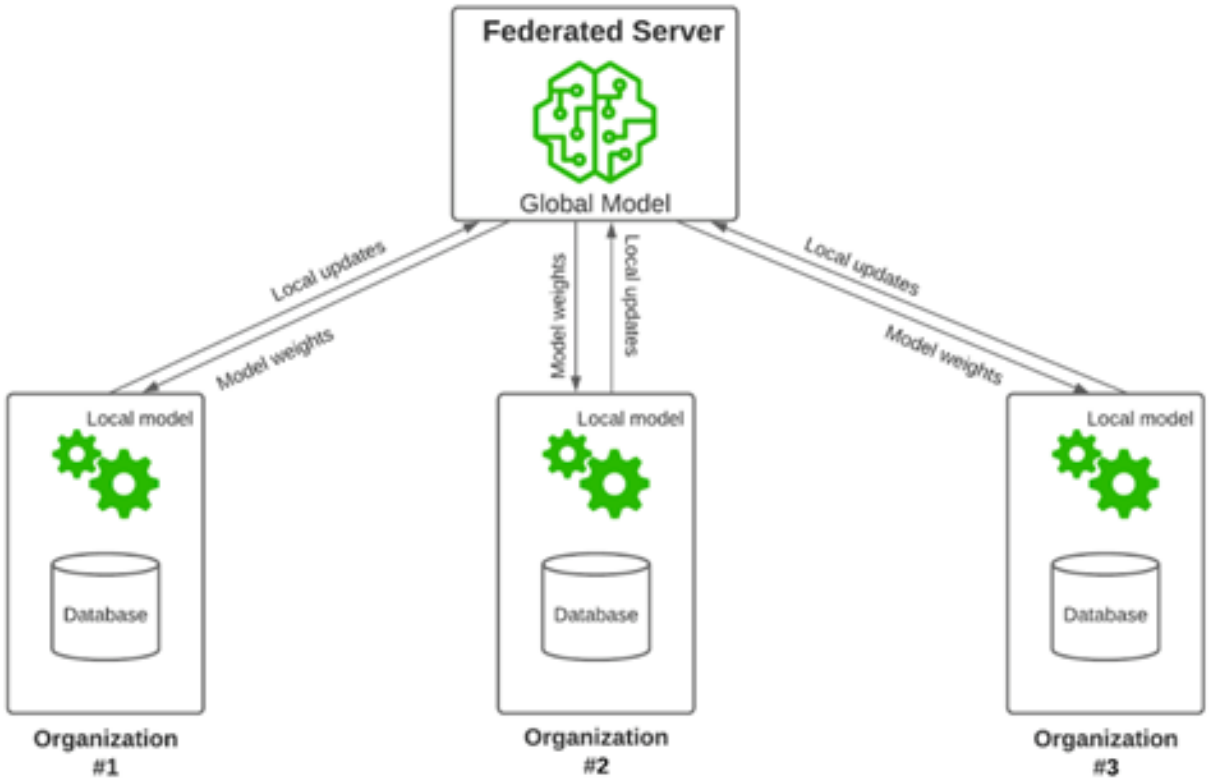
Bảng 4: Bảng đánh giá với đặc trưng Resnet50

# Kết luận

Kết quả đánh giá cho thấy việc xử lý đặc trưng từ dữ liệu gốc có thể tăng độ hiệu quả cho mô hình. Phần trên là các kết quả đã được chạy test và có thể tối ưu hơn để đạt độ chính xác cao hơn. Tuy nhiên để máy local có thể hoạt động tác vụ **Học liên kết (Federate Learning)**, nhóm đã ràng buộc một vài điều kiện của mô hình rút trích đặc trưng.

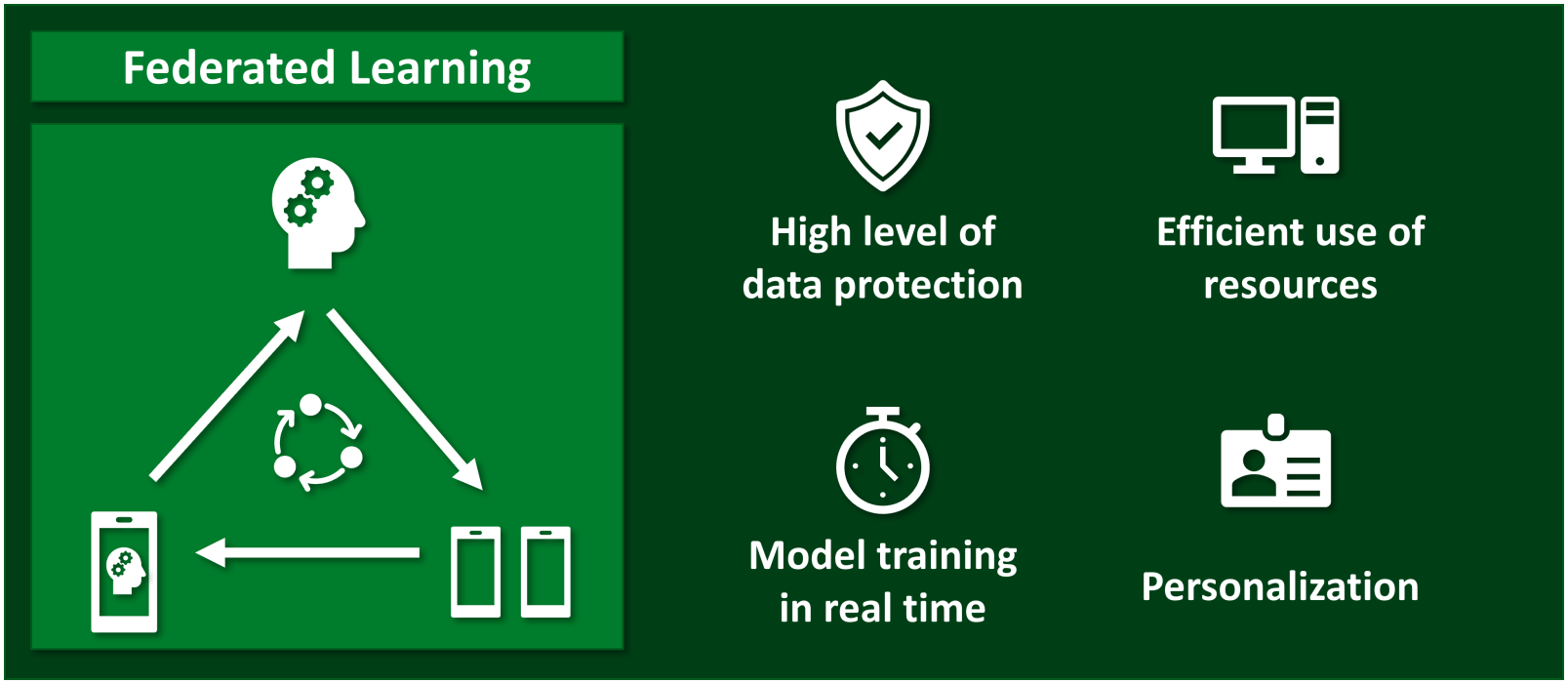
# Federate Learning là gì?

Federated Learning (FL) là phương pháp học máy phân tán, trong đó mô hình được đào tạo trên các thiết bị cá nhân và chỉ gửi các cập nhật mô hình về máy chủ trung tâm để tổng hợp.



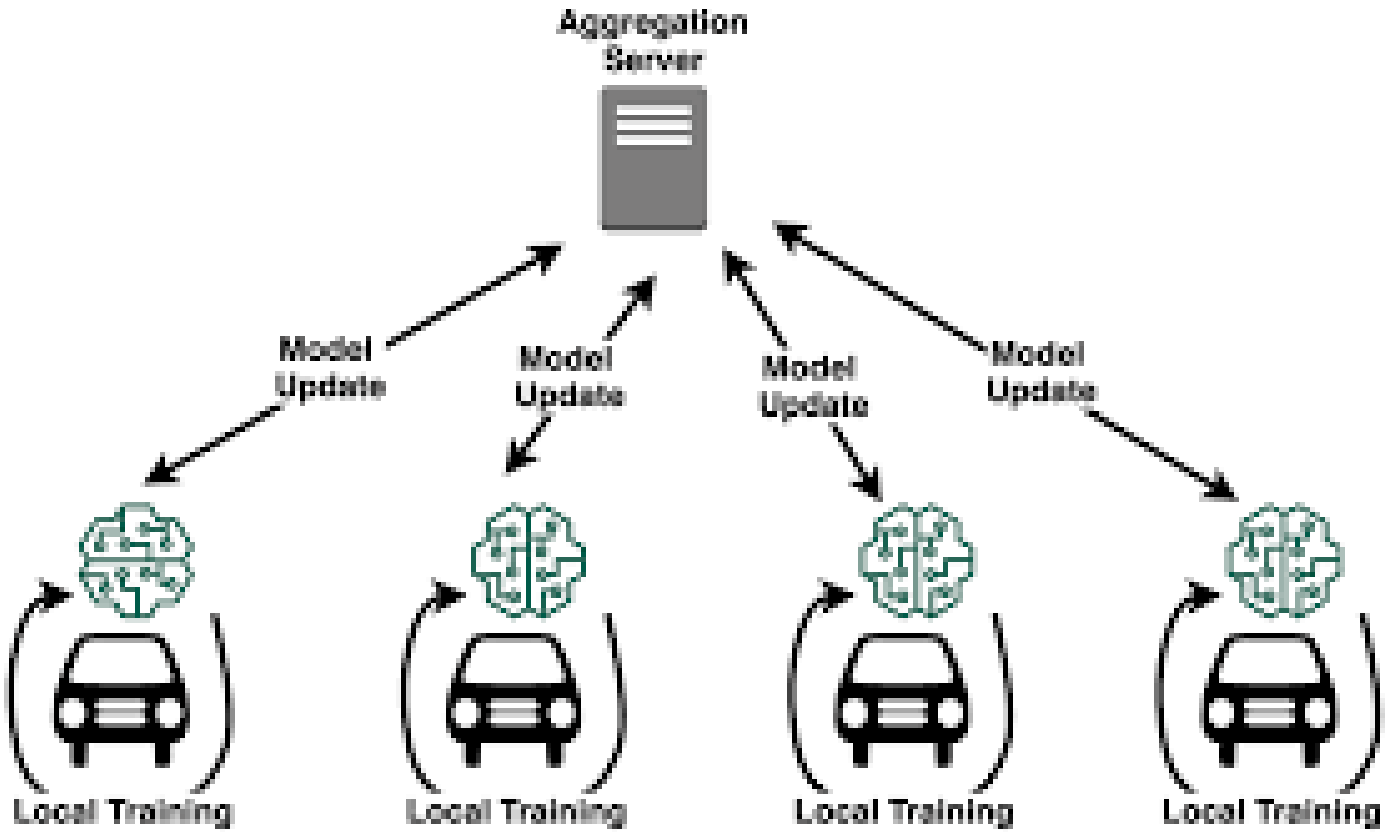

Hình 2: Mô tả FL

# Tại sao cần FL?



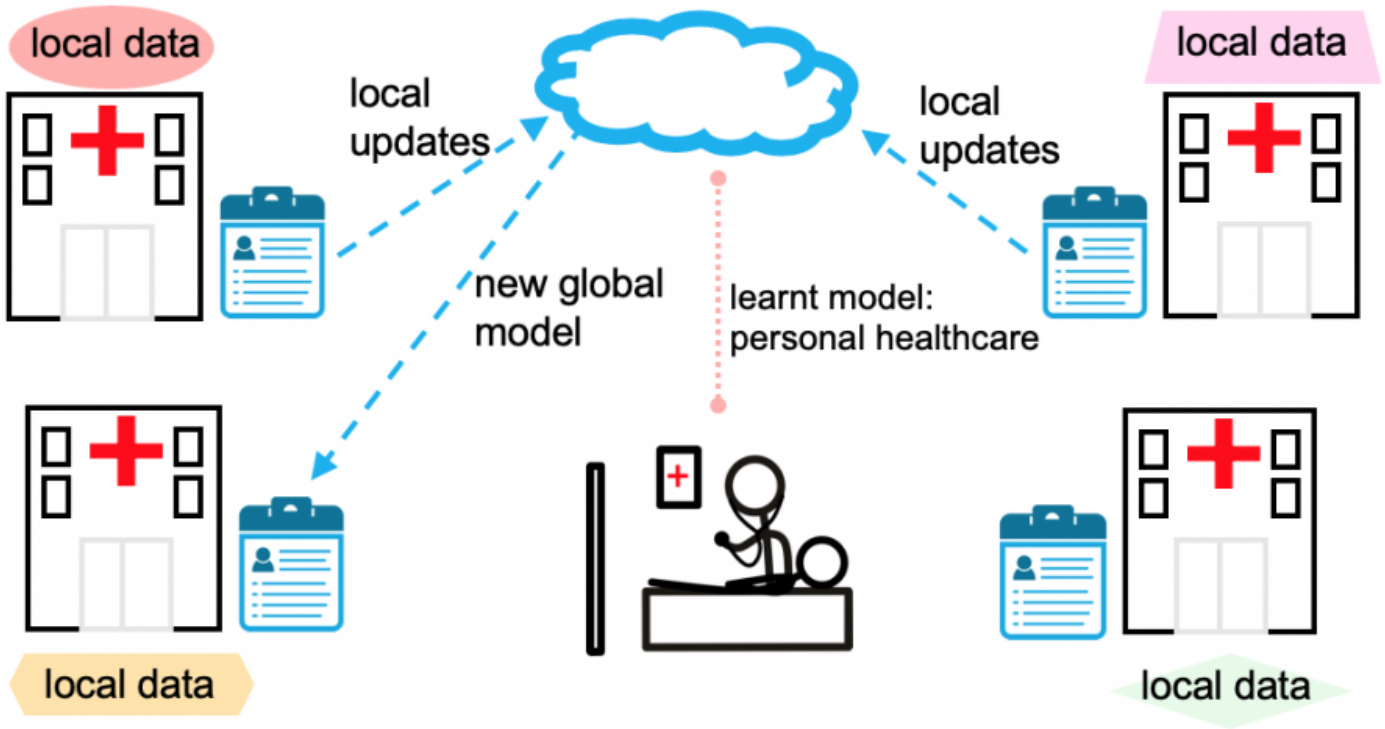

Hình 3: Lý do nên triển khai FL

# Ô tô tự lái




Hình 4: Triển khai trên xe điện

# Ứng dụng trong y tế




Hình 5: Triển khai trong y tế

# Rủi ro trong FL

- Rủi ro tấn công và rò rỉ thông tin
- Chi phí tính toán cao
- Thách thức trong việc tổng hợp

# Học đồng bộ

Trong học đồng bộ, server sẽ đợi tất cả các client hoàn thành một số batch huấn luyện nhất định trước khi thực hiện tổng hợp trọng số.

**Algorithm 1** Quy trình Học Đồng Bộ trong Federated Learning

1: **Server:** Khởi tạo mô hình toàn cục $\mathbf{W}_0$ và gửi đến tất cả các client.
2: **for** each round $t = 1, 2, \ldots, T$ **do**
3: **Server:** Gửi bộ trọng số mới nhất cho các client.
4: **Client:** Nhận yêu cầu và tải mô hình $\mathbf{W}_t$ từ server.
5: **Client:** Huấn luyện mô hình $\mathbf{W}_t$ trên dữ liệu cục bộ trong N batch, kết quả là mô hình cục bộ $\mathbf{W}_t^i$.
6: **Client:** Gửi $\mathbf{W}_t^i$ về server.
7: **Server:** Đợi cho đến khi nhận đủ trọng số từ tất cả các client.
8: **Server:** Tổng hợp các trọng số để cập nhật mô hình toàn cục:

$$\mathbf{W}_{t+1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{W}_t^i \tag{1}$$

9: **Server:** Lặp lại quá trình từ bước 2.

Hình 6:

# Học bất đồng bộ

Trong học bất đồng bộ, server không cần đợi tất cả các client hoàn thành huấn luyện. Khi bất kỳ client nào hoàn thành quá trình huấn luyện cục bộ, trọng số sẽ được gửi về server để tổng hợp ngay lập tức.

**Algorithm 1** Quy trình Học Bất Đồng Bộ trong Federated Learning

1: **Server:** Khởi tạo mô hình toàn cục $\mathbf{W}_0$ và gửi đến các client.
2: **Server:** Gửi yêu cầu huấn luyện đến tất cả client.
3: **loop**
4: **Client:** Nhận yêu cầu và tải mô hình $\mathbf{W}_t$ từ server.
5: **Client:** Huấn luyện mô hình $\mathbf{W}_t$ trên dữ liệu cục bộ trong N batch, tạo ra mô hình cục bộ $\mathbf{W}_t^i$.
6: **Client:** Gửi $\mathbf{W}_t^i$ về server ngay khi hoàn thành huấn luyện.
7: **Server:** Nhận trọng số từ các client và cập nhật mô hình toàn cục ngay lập tức:

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \eta \cdot (\mathbf{W}_t^i - \mathbf{W}_t) \tag{1}$$

8: **Server:** Tiếp tục lặp khi có client mới gửi trọng số về.

Hình 7:

# FedAvg

Phương pháp này thực hiện tổng hợp bằng cách tính trung bình có trọng số các mô hình từ các client.

$$w_{\text{glob}}^{t+1} = \sum_{k \in S_t} \frac{n_k}{n} w_k^{t+1}$$

Trong đó:

- $w_{\text{glob}}^{t+1}$ là mô hình toàn cục sau khi cập nhật.
- $S_t$ là tập hợp các client được chọn tại vòng lặp $t$.
- $n_k$ là số lượng dữ liệu của client $k$.
- $n$ là tổng số dữ liệu của tất cả các client.

# FedProx

FedProx là một biến thể của FedAvg, nhằm giải quyết vấn đề tối ưu hóa cục bộ và sự không đồng nhất giữa các client.

$$\mathcal{L}_{\text{prox}} = \mathcal{L}(w_k) + \frac{\mu}{2}\|w_k^t - w_{\text{glob}}^t\|^2$$

Trong đó:

- $\mathcal{L}(w_k)$ là hàm mất mát của client $k$.
- $\mu$ là một hằng số điều chỉnh, cho phép kiểm soát độ lớn của thuật ngữ điều chỉnh.
- $w_k^t$ là mô hình cục bộ của client $k$ tại vòng lặp $t$.
- $w_{\text{glob}}^t$ là mô hình toàn cục tại vòng lặp $t$.

# FedNova

FedNova là một cải tiến của FedAvg, trong đó các cập nhật từ mỗi client được chuẩn hóa và điều chỉnh dựa trên số vòng lặp địa phương mà mỗi client thực hiện.

$$w^{t+1} = w_{\text{glob}} - \tau_{\text{eff}} \sum_{k \in S_t} \frac{n_k}{n} \cdot \eta d_k^t$$

Trong đó:

- $\tau_{\text{eff}}$ là bước lặp hiệu quả, thể hiện sự ảnh hưởng của các cập nhật.
- $\eta$ là tỷ lệ học (learning rate).
- $d_k^t$ là gradient sau khi được chuẩn hóa của client $k$ tại vòng lặp $t$.

# Scaffold

Scaffold là một phương pháp tổng hợp sử dụng biến kiểm soát để giảm thiểu sai lệch do dữ liệu không đồng nhất giữa các client

$$c_k^+ = c_k - c + \frac{1}{\eta I}(w_{\text{glob}}^t - w_k^t)$$

Trong đó:

- $c_k$ là biến kiểm soát của client $k$.
- $c$ là biến kiểm soát trung bình.
- $\eta$ là tỷ lệ học.
- $I$ là số lần lặp (iterations) của client.

# Scaffold

Scaffold là một phương pháp tổng hợp sử dụng biến kiểm soát để giảm thiểu sai lệch do dữ liệu không đồng nhất giữa các client

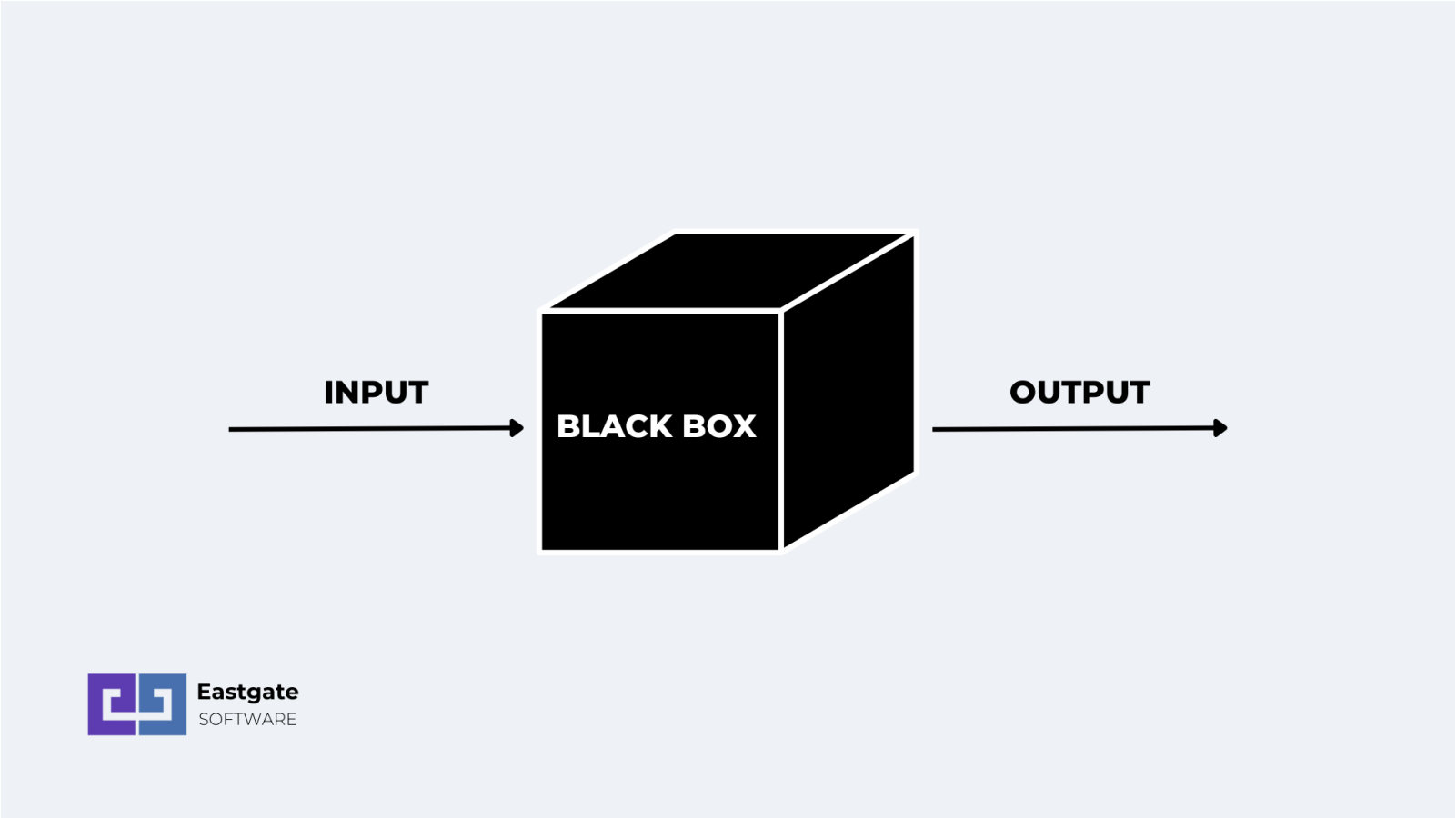
$$c_k^+ = c_k - c + \frac{1}{\eta l}(w_{\text{glob}}^t - w_k^t)$$

Trong đó:

- $c_k$ là biến kiểm soát của client $k$.
- $c$ là biến kiểm soát trung bình.
- $\eta$ là tỷ lệ học.
- $l$ là số lần lặp (iterations) của client.

# Black-box Attacks

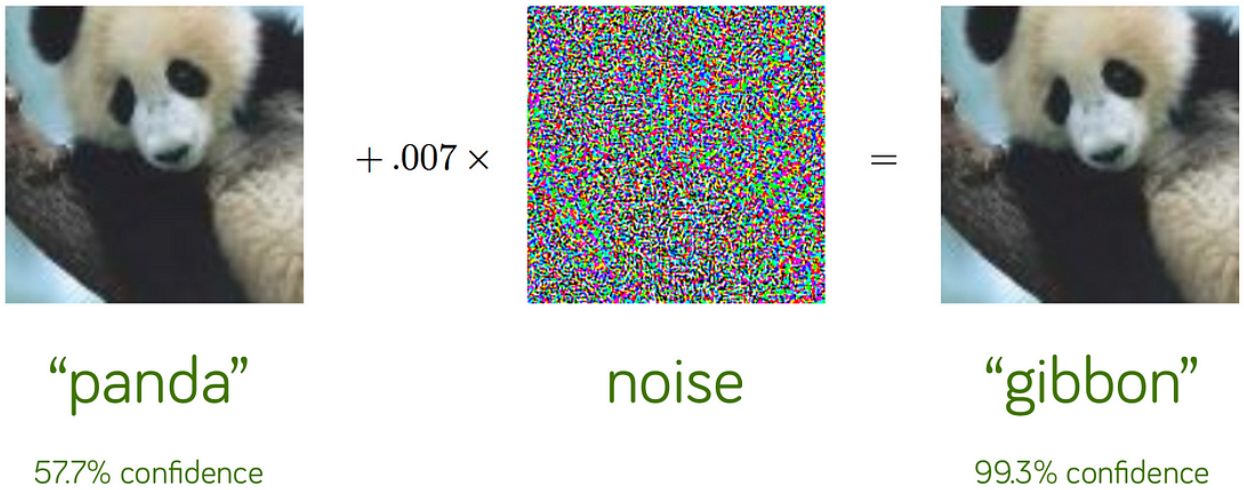
Black-box attacks là một tình huống tấn công trong đó kẻ tấn công chỉ tiếp cận được input và output và khai thác các tài nguyên thông qua chúng.




Hình 8: Minh họa Blackbox Attacks

# Adversarial Examples

Đây là những đầu vào được thiết kế tinh vi để khiến mô hình phân loại sai, dù với con người, chúng trông gần như không khác gì so với các đầu vào hợp lệ.




Hình 9: Minh họa Adversarial Examples

# Thuật toán

Mục tiêu của tấn công là tạo ra một nhiễu $\delta$ sao cho đầu vào bị nhiễu $x^{adv} = x + \delta$ khiến mô hình phân loại sai. Điều này được thực hiện bằng cách tối đa hóa hàm mất mát $J(\theta, x, y)$ thông qua thuật toán Gradient Ascent

**Algorithm 3** Tạo mẫu đối kháng bằng thuật toán Gradient Ascent

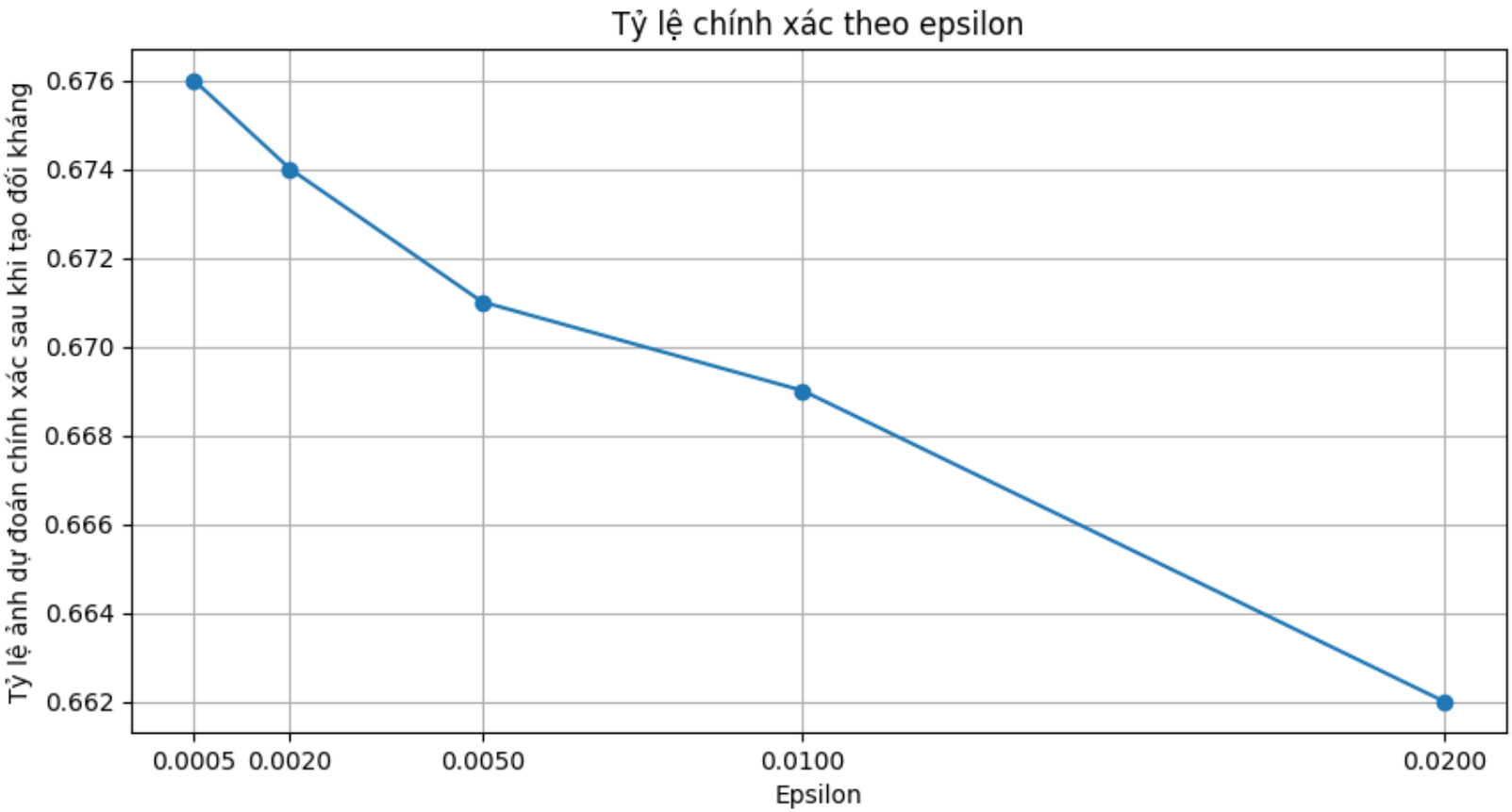
1: **Khởi tạo:** Đầu vào $x$, nhãn $y$, hệ số học $\alpha$, số lần lặp $N$.
2: Khởi tạo nhiễu $\delta \leftarrow 0$.
3: **for** $i = 1$ **to** $N$ **do**
4: Tính gradient: $g \leftarrow \nabla_x J(\theta, x + \delta, y)$ ▷ Tính gradient hàm mất mát
5: Cập nhật nhiễu: $\delta \leftarrow \delta + \alpha \cdot SIGN(g)$ ▷ Cập nhật nhiễu theo hướng gradient ascent
6: Chuẩn hóa $\delta$ để đảm bảo $x + \delta$ hợp lệ.
7: $x^{adv} \leftarrow x + \delta$ ▷ Mẫu đối kháng
8: **return** $x^{adv}$

Hình 10: Thuật toán Adversarial Examples

# Mô phỏng tấn công

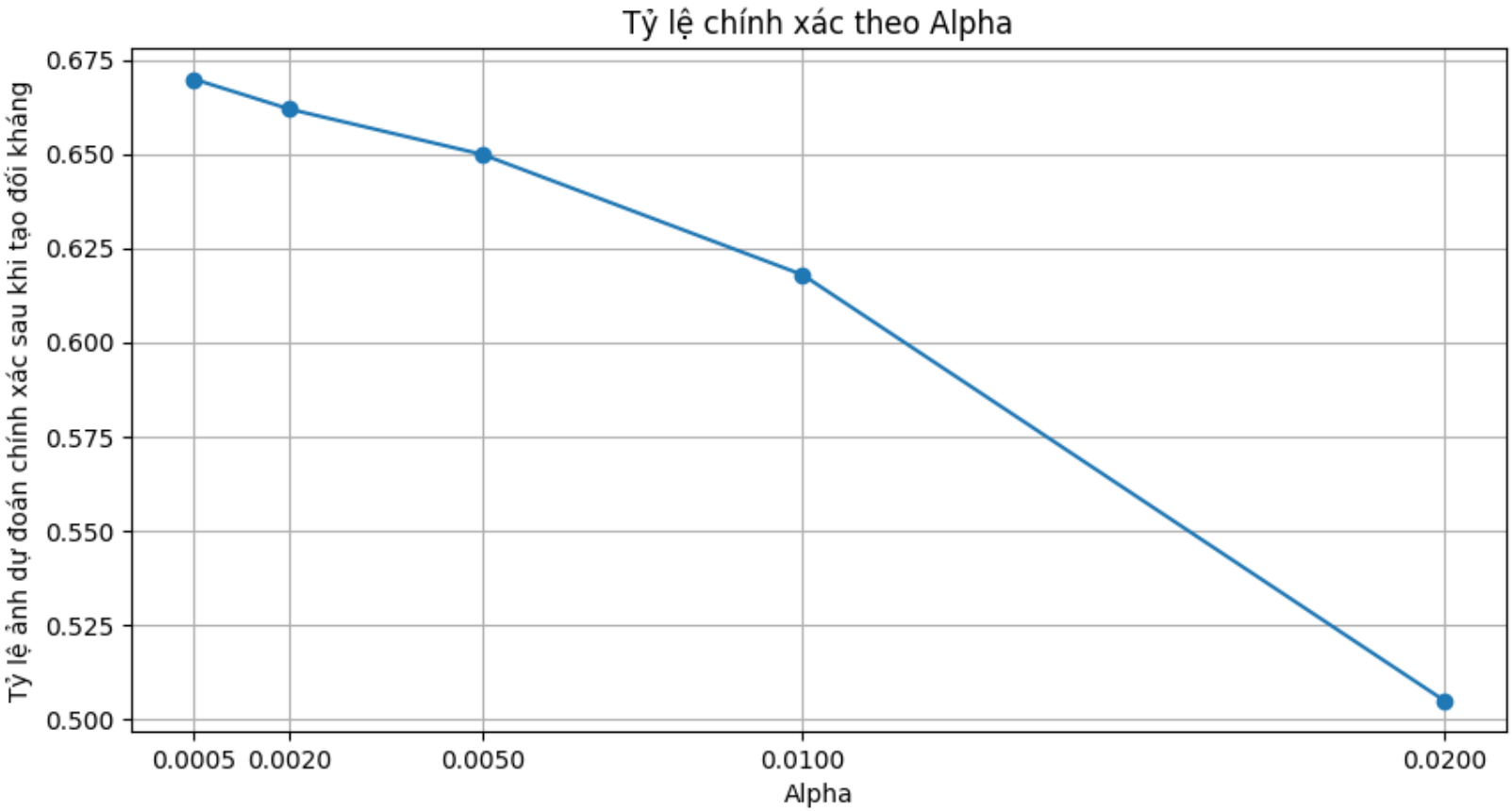
Để tấn công vào mô hình mục tiêu ResNet + Softmax, giả sử ta là user và khám phá được mô hình của dịch vụ dùng deep learning để xử lý đặc trưng và softmax để phân loại. Ta xây dựng một mô hình tương đương là Dense + Softmax để mô phỏng hành vi, từ đó ước tính gradient $\nabla_x J(\theta, x, y)$ và thực hiện cập nhật nhiễu.

# Kết quả mô phỏng tấn công




Hình 11: LGSM-Epsilon

# Kết quả mô phỏng tấn công




Hình 12: LGSM-Alpha

# Tài liệu tham khảo I


- Vangelis Metsis, Ion Androutsopoulos, and Geogios P. *Spam filtering with naive bayes-which naive bayes?*. Third conference on email and anti-spam (CEAS), 2006.
- Vũ Hữu Tiệp. *Machine learning cơ bản*. Nhà xuất bản Khoa học và Kỹ thuật, 2018.
- Dalal, Navneet and Triggs, Bill. *Histograms of oriented gradients for human detection*, 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)
- Pian Qi and Diletta Chiaro and Antonella Guzzo and Michele Ianni and Giancarlo Fortino and Francesco Piccialli *Model aggregation techniques in federated learning: A comprehensive survey* Future Generation Computer Systems
- Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E *Imagenet classification with deep convolutional neural networks* Advances in neural information processing systems
- LeCun, Yann and Boser, Bernhard and Denker, John S and Henderson, Donnie and Howard, Richard E and Hubbard, Wayne and Jackel, Lawrence D *Backpropagation applied to handwritten zip code recognition* Neural computation

# Tài liệu tham khảo II


He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian *Deep residual learning for image recognition*, Proceedings of the IEEE conference on computer vision and pattern recognition

Sai Praneeth Karimireddy and Satyen Kale and Mehryar Mohri and Sashank J. Reddi and Sebastian U. Stich and Ananda Theertha Suresh *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning*

# The End