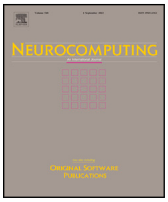
Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



FedBoosting: Federated learning with gradient protected boosting for text recognition

Hanchi Ren [a], Jingjing Deng [b,*], Xianghua Xie [a], Xiaoke Ma [c], Yichuan Wang [d]

[a] *Department of Computer Science, Swansea University, United Kingdom*
[b] *Department of Computer Science, Durham University, United Kingdom*
[c] *Department of Computer Science, Xidian University, China*
[d] *Department of Computer Science, Xi'an University of Technology, China*

ARTICLE INFO

Communicated by Z. Le

MSC:
00-01
99-00

Keywords:
Deep learning
Federated learning
Privacy preserving

ABSTRACT

Conventional machine learning methodologies require the centralization of data for model training, which may be infeasible in situations where data sharing limitations are imposed due to concerns such as privacy and gradient protection. The Federated Learning (FL) framework enables the collaborative learning of a shared model without necessitating the centralization or sharing of data among the data proprietors. Nonetheless, in this paper, we demonstrate that the generalization capability of the joint model is suboptimal for Non-Independent and Non-Identically Distributed (Non-IID) data, particularly when employing the Federated Averaging (FedAvg) strategy as a result of the weight divergence phenomenon. Consequently, we present a novel boosting algorithm for FL to address both the generalization and gradient leakage challenges, as well as to facilitate accelerated convergence in gradient-based optimization. Furthermore, we introduce a secure gradient sharing protocol that incorporates Homomorphic Encryption (HE) and Differential Privacy (DP) to safeguard against gradient leakage attacks. Our empirical evaluation demonstrates that the proposed Federated Boosting (FedBoosting) technique yields significant enhancements in both prediction accuracy and computational efficiency in the visual text recognition task on publicly available benchmarks.

1. Introduction

Personal data protection and privacy-preserved issues have particularly attracted researchers' attention [1–7]. Typical machine learning approaches that require centralized data for model training may not be possible as restrictions on data sharing are in place. Therefore, decentralized data-training approaches are more attractive since they offer desired benefits in privacy preserving and data security protection. FL [8,9] was proposed to address such concerns that allows individual data providers to collaboratively train a shared global model without aggregating the data centrally. McMahan et al. [9] presented a practical decentralized training method for deep networks based on averaging aggregation. Experimental studies were carried out on various datasets and architectures, which demonstrated the robustness of FL on unbalanced and Independent and Identically Distributed (IID) data. Frequent updating approach can generally lead to higher prediction performance whereas the communication cost increases sharply, especially for the large datasets [8–12]. Konečnỳ et al. [8] focused on addressing the efficiency issue and proposed two weight updating methods, namely structured updates and sketched updates approaches based on FedAvg to reduce the up-link communication costs of transmitting the gradients from the local machine to the centralized server.

Prediction performance and data privacy are two major challenges in FL research. On one hand, the accuracy of FL decreases significantly on Non-Independent and Non-Identically Distributed (Non-IID) data [13–21]. Zhao et al. [13] showed the weight divergence can be measured quantitatively using Earth Mover's Distance (EMD) between the distributions over classes on each local machine and the global population distribution. Hence, they proposed to share a small subset of data among all the edge devices to improve model generalization on Non-IID data. However, such a strategy is infeasible when restrictions on data sharing are in place which usually leads to privacy breaching. Li et al. [22] studied the convergence properties of FedAvg and concluded a trade-off between its communication efficiency and convergence rate is existed. They argued that the model converges slowly on heterogeneous datasets. Based on our empirical study in this paper, we confirm that given Non-IID datasets, the training needs far more iterations to reach an optimal solution and often fails to converge, especially when the local models are trained on large-scale datasets with a small number

[*] Corresponding author.
E-mail address: jingjing.deng@durham.ac.uk (J. Deng).

https://doi.org/10.1016/j.neucom.2023.127126
Received 1 May 2023; Received in revised form 9 August 2023; Accepted 6 December 2023
Available online 12 December 2023

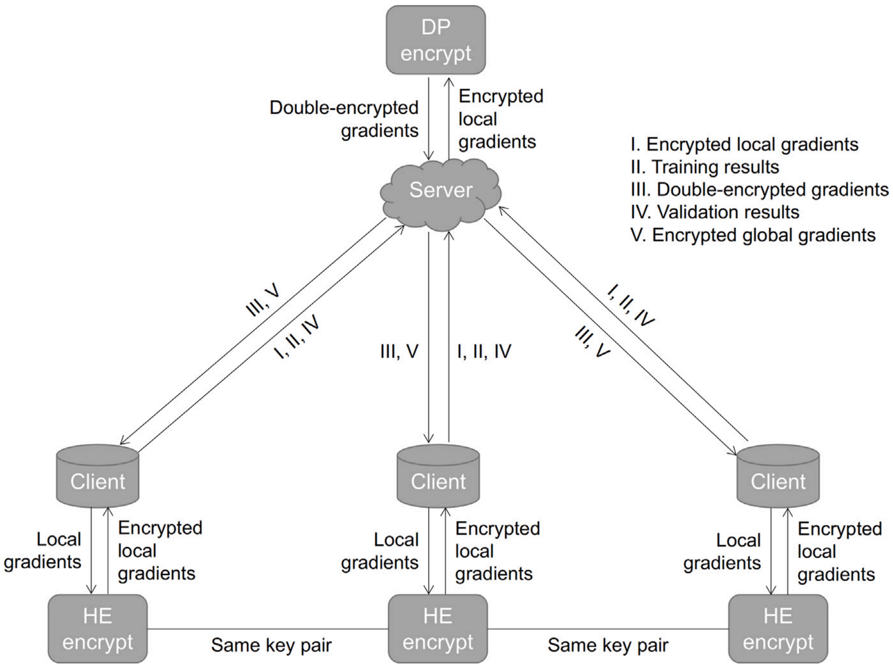


Fig. 1. The schematic diagram of proposed FedBoosting and encryption protocol. There are two clients used for demonstration purpose, whereas the proposed method can work with arbitrary number of local clients.

of batch size or the global model are aggregated after a large number of epochs. The feasibility of adaptive aggregation through the utilization of training and testing outcomes has been established [23,24]. On the other hand, model gradient is generally considered to be safe to share in the FL system for model aggregation. However, some studies have shown that it is feasible to recover training data information from model gradients. For example, Fredrikson et al. [25] and Melis et al. [26] reported two methods that can identify a sample with certain properties is in the training batch. Hitaj et al. [27] proposed a Generative Adversarial Networks (GANs) model as an adversarial client to estimate the distribution of the data from the output of other clients without knowing their training data. Zhu et al. [28] and Zhao et al. [29] demonstrated data recovery can be formulated as a gradient regression problem assuming the gradient from a targeted client is available, which is a largely valid assumption in most FL systems. Furthermore, Generative Regression Neural Network (GRNN) proposed by Ren et al. [30] consists of two branches of generative models, one is based on GAN for generating fake training data and the other one is based on fully-connected layer for generating corresponding labels. The training data is revealed by regressing the true gradient and the fake gradient generated by the fake data and relevant label.

In this paper, we propose FedBoosting method to address the weight divergence and gradient leakage issues in general FL framework. Instead of treating individual local models equally when the global model is aggregated, we consider the data diversity of local clients in terms of the status of convergence and the ability of generalization. To address the potential risk of data leakage via shared gradients, a DP based linear aggregation method is proposed using HE [31] to encrypt the gradients which provides two layers of protection. The proposed encryption scheme only leads to a negligible increase in computational cost.

The proposed method is evaluated using a text recognition task on public benchmarks, as well as a binary classification task on two datasets, which demonstrates its superiority in terms of convergence speed, prediction accuracy and security. The performance reduction due to encryption is also evaluated. Our contributions are four-fold:

- We propose a novel aggregation strategy namely FedBoosting for FL to address the weight divergence and gradient leakage issues. We empirically demonstrate that FedBoosting converges significantly faster than FedAvg while the communication cost is identical to traditional approaches. Especially when the local models are trained with small batch size and the global model are aggregated after a large number of epochs, our approach can still converge to a reasonable optimum whereas FedAvg often fails in such case.
- We introduce a dual layer protection scheme using HE and DP to encrypt gradients flowing between server and clients, which protect the data privacy from gradient leakage attack.
- We show the feasibility of our method on two datasets by evaluating the decision boundaries visually. Furthermore, we also demonstrate its superior performance in a visual text recognition task on multiple large-scale Non-IID datasets compared to centralized approach and FedAvg. The experimental results confirm that our approach outperforms FedAvg in terms of convergence speed and prediction accuracy. It suggests FedBoosting strategy can be integrated with other Deep Learning (DL) models in the privacy-preserving scenarios.
- Our implementation of proposed FedBoosting is publicly available to ensure reproducibility. It can also be run in a distributed multiple Graphics Processing Units (GPUs) setup.[1]

The rest of the paper is organized as follows: In Section 2 the related work on encryption method, collaborative learning and gradient leakage are presented. The proposed method FedBoosting and relevant encryption method are described in Section 3 and evaluated on a text recognition task and a binary classification task. The details of experiments and discussions on the results, as well as a performance comparison, are provided in Section 4, followed by the conclusions in Section 5.

[1] https://github.com/Rand2AI/FedBoosting.

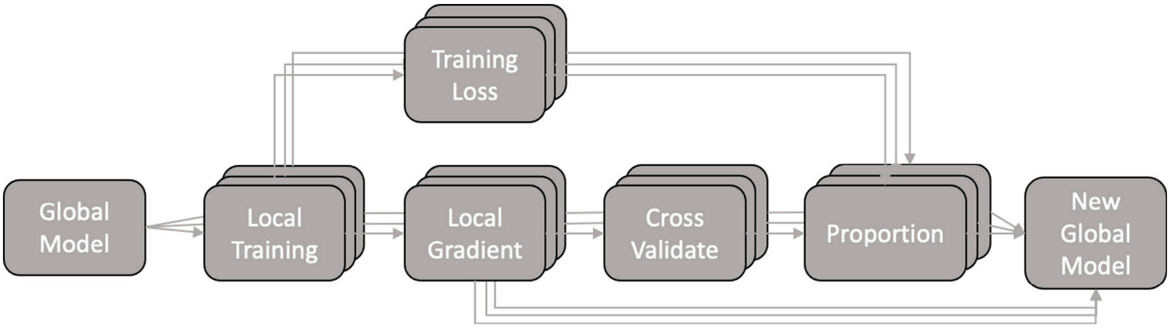


Fig. 2. The illustration for the process of FedBoosting.

2. Related work

FL for privacy-preserving machine learning is proposed for training a model across multiple decentralized edge devices or clients holding local data samples [8,9,32–34]. More specifically, FL framework keeps the raw data to the owners and trains the model locally at client nodes individually, whereas gradients of those models are exchanged and aggregated instead of data. Compared to Secure Multi-Party Computation (MPC) [35,36] which ensures a high level of security at the price of expensive cryptography operations, FL has loosened security requirements that enables more efficient implementation and lower running cost. Since there is no explicit data exchange, FL does not require adding noises to the data as DP [37–41], nor encrypting data into homomorphic phase to fit a homomorphic operation as HE [1,42–44]. Gradient aggregation from local models is one of the core research problems in FL. Mcmahan et al. [9] introduced the FedAvg method for training deep neural networks over multiple parties, where the global model takes the average of gradients from local models, *i.e.* $\omega' = \sum_i^N \frac{1}{N}\omega_i$, where ω' and ω_i are the gradients of global model and the i_{th} local model, and N is the total number of clients. The method is evaluated on MNIST benchmark and demonstrates its feasibility on the classic image classification task using Convolutional Neural Networks (CNNs) as the learning model. Although the experimental results show that FedAvg is suitable for both IID and Non-IID data, it is still a statistical challenge of FL when a local model is trained on large-scale Non-IID data. In this paper, our experimental results also support such an argument, where the prediction accuracy and convergence rate drop significantly with large-scale Non-IID data on FedAvg.

In the work [45], the authors had a comprehensive discussion on adaptive weights aggregation for FL and proposed a flexible framework, termed FedOPT, which is capable for many optimization algorithms. They specialized the FedOPT to FedAdam, FedAdagrad and FedYOGI by using three example optimization algorithms, Adam [46], Adagrad [47] and YOGI [48]. This work mirrors the process of FedAvg closely, diverging only at the final stage of weight aggregation. Upon obtaining the averaged local gradients $\hat{g}$, the first-order matrices of momentum m are computed for FedAdam, FedAdagrad, and FedYOGI, as detailed in Eq. (1). However, the derivation of the second-order matrices of variance v is distinguished by algorithm. FedAdam utilizes Eq. (2), while FedAdagrad and FedYOGI adopt Eqs. (3) and (4) respectively to derive their second-order matrices.

$$m_r = \beta_1 m_{r-1} + (1-\beta_1)\hat{g}_r \tag{1}$$

$$v_r = \beta_2 v_{r-1} + (1-\beta_2)\hat{g}_r^2 \tag{2}$$

$$v_r = v_{r-1} + \hat{g}_r^2 \tag{3}$$

$$v_r = \beta_2 v_{r-1} + (1-\beta_2)\hat{g}_r^2 sign(v_{r-1} - \hat{g}_r^2) \tag{4}$$

where, r is the training round, β_1 and β_2 are two momentum parameters, $sign()$ is the symbolic functions. In the end, all those three methods employ Eq. (5) for weights aggregation.

$$w_r = w_{r-1} + \eta_t \frac{m_r}{\sqrt{v_r} + \epsilon} \tag{5}$$

where, η_t is the adaptive learning rate, calculated by:

$$\eta_t = \eta_0 \frac{\sqrt{1-\beta_2^r}}{1-\beta_1^r} \tag{6}$$

where η_0 is the initial learning rate, β_1^r and β_2^r are the rth power of the parameter β_1 and β_2.

The work in [49] aims to solve the high communication cost challenge in FL. They proposed a novel communication-efficient adaptive federated learning method called FedCAMS. A theoretically analysis is also provided for the guarantee of the model convergence. The authors first improved FedAdam by using AMSGrad [50] with max stabilization. Two optional methods are provided and the authors claimed that a non-decreasing variance is indeed necessary to obtain the complete proof with a positive β_1. On the other hand, an error feedback compression strategy is employed for reducing the communication cost. While in FedOPT and FedCAMS, they both gather the local updates g_i and average then to have the $\hat{g}$, then conduct the global model aggregation work. That is to say, they both treated each client equally to have the global updates $\hat{g}$. However, the basic opinion of the adaptive optimization tends to treat each individual weight independently. Although the work in FedOPT and FedCAMS utilized the adaptive way for global model aggregation, the averaging processing does not distinguish the contribution of local models trained on different dataset fairly. That is because different datasets cause different convergence levels. Different proportions should be given to each client.

FL is designed for privacy protected training as the data is kept and processed locally. However, it has been highlighted in multiple studies, *e.g.* [27,28,30], FL suffers from the so-called gradient leakage problem that is the private training data can be recovered from the publicly shared gradients with significantly high success rate. Hitaj et al. [27] proposed a training data recovery approach from FL system using GAN. It aims to generate similar training samples given one specific class rather than recover the original training data directly. First, the global FL model is trained as usual for several iterations to achieve a relatively high accuracy. They assume that the malicious participant can obtain one of client model and used as discriminator. Then an image generator is updated based on the output of the discriminator given a targeted image class. Finally, the well-trained generator can produce image samples that are similar to the training data given the specified image class. Zhu et al. [28] formulated the data recovery task as a gradient regression problem, where the pixel values of input image are treated as random variables that are optimized using back-propagation while the shared model parameters are fixed. The object function measures the Euclidean distance between the shared gradient in FL and the gradient given by the random image input, which is minimized during the training phase. They hypothesized that the optimized input when the model converges is as similar as the original training image that is stored on local client alone. The experimental results on public benchmark datasets proves the hypothesis is valid, which in turn indicates that gradient sharing could lead to privacy data leakage. Our previous work GRNN [30] further improves the success rate of leakage attack using generative model for data recovery, particularly a large batch size is used in training.

Algorithm 1 FedBoosting with HE and DP: Server

```
build model and initialize weights $\omega_0$;
for each round $r = 1, 2, ..., R$ do
    for each client $i \in C_N$ do
        if $r == 1$ then
            $g_r^{*i}, T_r^i \leftarrow Train(r, i, \omega_{r-1})$ via Algorithm 2.a;
        else
            $g_r^{*i}, T_r^i \leftarrow Train(r, i, G_{r-1}^*)$ via Algorithm 2.a;
        end if
    end for
    for each client $i \in C_N$ do
        generate $\hat{G}_r^{*i}$ via Equ.(12);
        $V_r^i \leftarrow \sum_j^N Evaluate(j, \hat{G}_r^{*i})$ via Algorithm 2.b;
    end for
    generate $p_r^i$ via Equ.(8)&(10);
    generate $G_r^*$ via Equ.(11);
    if $r == R$ then
        $\omega_r \leftarrow Decrypt(G_r^*)$ via Algorithm 2.c
    end if
end for
```

3. Proposed method

Algorithm 2 FedBoosting with HE and DP: Client

```
a. Train($r, i, \omega_{r-1} \mid\mid G_{r-1}^*$):
if $i == 1$ then
    generate key pair and sent to other clients
else
    wait for key pair from $C_1$
end if
if $r == 1$ then
    load $TrnD^i, ValD^i$
else
    decrypt $G_{r-1}^*$ to $G_{r-1}$ by secret keys
    $\omega_{r-1} = \omega_{r-2} + G_{r-1}$
end if
for each epoch $e = 1, 2, ..., E$ do
    for each batch $b = 1, 2, ..., B$ do
        $\omega_r \leftarrow \omega_{r-1} - \eta \cdot \nabla f(TrnD^{i,b}, \omega_{r-1})$
    end for
end for
$G_r^i = \omega_r - \omega_{r-1}$
$g_r^i = \lfloor (G_r^i * 1e32)/P \rceil$ and generate $g_r^{*i}$ by public keys
$T_r^i \leftarrow f(TrnD^i | \omega_r)$
return $g_r^{*i}, T_r^i$ to server
b. Evaluate($j, \hat{G}_r^{*i}$):
decrypt $\hat{G}_r^{*i}$ to $\hat{G}_r^i$ by secret keys
$\hat{\omega}_r^i = \omega_{r-1} + \hat{G}_r^i$
$V_r^{i,j} \leftarrow f(ValD^j | \hat{\omega}_r^i))$
return $V_r^{i,j}$ to server
c. Decrypt($G_r^*$):
decrypt $G_r^*$ to $G_r$ by secret keys
$\omega_r = \omega_{r-1} + G_r$
return $\omega_r$ to server
```

3.1. FedBoosting framework

The algorithm known as FedAvg [9] facilitates the production of a consolidated model by computing the mean of the gradients originating from distinct local clients. This approach simplifies the centralization of distributed learning but exhibits certain caveats. In scenarios involving Non-IID data, the heterogeneous nature of data distribution across different clients leads to distinct convergence directions of the local model weights. This inconsistency, a natural consequence of the Non-IID characteristics, introduces significant challenges to the application of the FedAvg algorithm. When the local datasets are Non-IID, the variances in data distribution can cause the weights of the local models to diverge significantly. A simple averaging scheme, as utilized in FedAvg, may thus perform suboptimally, especially in cases where pronounced biases and extreme outliers are present in the data. In these situations, the straightforward approach of averaging the gradients can lead to a loss of valuable information and may skew the overall model in a way that does not truly represent the underlying data. This has led to the study of more sophisticated strategies that go beyond averaging and take into account the uniqueness of each local dataset. For instance, weighted averaging or other adaptive schemes might be employed, where the contributions from each local model are balanced based on factors such as data distribution, quality, or relevance to the overall learning task [13,22,51]. These advanced techniques can lead to a more robust and accurate global model that can handle the complex nuances of Non-IID data distribution, thus overcoming some of the inherent limitations of the traditional FedAvg method. Therefore, we propose using boosting scheme, namely FedBoosting, to adaptively merge local models according to the generalization performance on different local validation datasets. Meanwhile, in order to preserve data privacy, information exchanges among decentralized clients and server are prohibited. Hence, instead of exchanging data between clients, encrypted local models are exchanged via the centralized server and validated on each client independently. More details are shown in Fig. 1.

Compared to FedAvg, our proposed FedBoosting strategy evaluates the fitness and general performance of each client model, and correspondingly adjusts the global model by utilizing varying weights from all client models. This is accomplished by generating three distinct sets of information from each client: local gradients G_r^i, training loss T_r^i, and cross-validation loss $V_r^{i,j}$. Here, G_r^i and T_r^i represent the local gradients and training loss of the ith local model during the rth training round, while $V_r^{i,j}$ is the cross-validation loss of the ith local model on the jth local validation dataset in the rth training round. The local gradients G_r^i are subsequently shared with all other clients through the centralized server. In addition, the cross-validated loss $V_r^{i,j}$ (where $i \neq j$) can be obtained from all other clients. Both training and cross-validation losses serve as performance measures for local models. The model's large training loss can imply inadequate convergence and a lack of generalization ability, yet it may also mean the model's gradient holds ample information for training. Conversely, a low training loss does not assure good generalization ability (over-fitting) and may imply less training data. Consequently, validation losses are also a factor to be considered. The aggregation weight of a local model's contribution to the global model is determined by these two kinds of losses, as demonstrated in Eq. (8). On the server side, all validation results for the ith model are aggregated, denoted as V_r^i, representing the ith model's generalization ability. To facilitate convergence, a *softmax* layer is utilized, which takes T_r as input. The outputs along with V_r^i are then used to calculate the aggregation weight p_r^i. During the current round of aggregation, the updated global gradients G_r are derived by combining all the local gradients G_r^i according to their corresponding weight p_r^i. Please see Fig. 2 for more details.

$$G_r = \sum_i^N p_r^i G_r^i; \forall p_r^i \in [0, 1], \tag{7}$$

$$p_r^i = softmax(softmax(T_r^i) \cdot \sum_{j \neq i}^N V_r^{i,j}), \tag{8}$$

$$softmax(T_r^i) = \frac{exp(T_r^i)}{\sum_{j=1}^N exp(T_r^j)}, \tag{9}$$

$$V_r^{i,j} = \begin{pmatrix} V_r^{1,1} & V_r^{1,2} & \cdots & V_r^{1,j} \\ V_r^{2,1} & V_r^{2,2} & \cdots & V_r^{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ V_r^{i,1} & V_r^{i,2} & \cdots & V_r^{i,j} \end{pmatrix} \tag{10}$$

where T_r^i and p_r^i are training loss and mixture coefficient for the ith local model.

In addition, the proposed FedBoosting scheme is resilient to some malicious attacks, such as data poisoning. For instance, when a malicious client injects poisoned data into the training set and the contaminated local model is aggregated with the same weight as other clean models, our method can mitigate as the validation scores of the toxic model on other clients will be significantly lower, which in turn leads to a significantly lower aggregation weight.

3.2. HE aggregation with quantized gradient

HE ensures that the computation can be carried out on the encrypted data as $Enc(A) \bullet Enc(B) = Enc(A * B)$, where "$\bullet$" stands for operation on encrypted data and "$*$" is operation on plain data. Since the FedBoosting involves computing the global model based on local gradient, the HE is used in our method to ensure the aggregation and gradient exchange among clients and server are secure. In FedBoosting, local models are trained on each client and then local gradients are transmitted to the server, where all local gradients are integrated to build global gradients in every round of aggregation. To preserve gradient information, FedBoosting utilizes the HE method, *Paillier* [31]. Once the training starts, a pair of HE keys are shared among clients. The public key is used to encrypt gradients and the secret key is for decryption. After a round of local training, local gradients can be calculated by $G_r^i = \omega_r^i - \omega_{r-1}$, where ω_{r-1} is the global weight from last round and ω_r^i is the weight after training at current round.

It is infeasible to encrypt G_r^i and transmit it to the server directly, as *Paillier* can only process integer value. To address this issue, we propose to convert G_r^i into scaled integer form, denoted as $G_r^{'i}$ by multiplying with $1e^{32}$. As the weighting scheme at the server side will break the integer-only constrain for homomorphic computation, to ensure the correctness of aggregation, we divide $G_r^{'i}$ into P pieces and then round to an integer according to $g_r^i = \lfloor G_r^{'i}/P \rceil$. There is a negligible precision loss as only the last few bits are dropped. For example, in the case of $P = 10$, the value loss is only at the 32th bit, similarly, for $P = 100$, the loss is at the 31th and 32th bits. Finally, g_r^i is encrypted using *Paillier* and the encrypted g_r^{*i} is transmitted to the server. On the contrary, the client weight p_r is converted to an integer by multiplying P followed by a rounding operation. In FedBoosting, the aggregation weight is computed with respect to Eqs. (8) & (10). The final encrypted global gradients G_r^* can be computed by merging all gradients from clients (see Eq. (11)). The final encrypted global gradients G_r^* is then transmitted back to each client to be decrypted and generate global weights by $\omega_r = \omega_{r-1} + G_r$, where G_r is decrypted from G_r^*. The proposed secure aggregation approach using HE with quantized gradient is generalizable that can also be used for FedAvg.

$$G_r^* = \sum_i^N \lfloor p_r^i \cdot P \rceil \cdot g_r^{*i}; \forall p_r^i \in [0,1] \tag{11}$$

3.3. DP fusion for local model protection

Local gradient between client and server is protected by HE as aforementioned. While in our proposed FedBoosting mechanism, local models are shared with all the other clients for cross-validation, and all clients have the same key pair. FedAvg shows that the uniformly combined global model is capable of performing similarly as any local model. Therefore, to protect gradient privacy among clients, inspired by DP, we propose to perturb individual local models using a linearly combination of with HE-encrypted local models, where the target model takes dominant proposition giving highest weight. Only the perturbed local models are shared among clients for cross-validation. Empirically, the reconstructed model performs similarly to the local model. Once the server receives all the encrypted local gradients piece $g_r^{*i}, \forall i = 1, 2, \dots, N$, the server randomly generates N sets of private fusing weights within which the corresponding local model always takes the largest proposition. Then, the server computes N reconstructed local model using the HE according to the N sets of weights (see Eq. (12)).

$$\hat{G}_r^{*i} = \lfloor \hat{p} \cdot P \rceil \cdot g_r^{*i} + \sum_{j:j\neq i}^{N} \lfloor \frac{(1-\hat{p}) \cdot P}{N-1} \rceil \cdot g_r^{*j} \tag{12}$$

where $\hat{G}_r^{*i}$ is the ith dual-encrypted whole gradient in round r. Finally, the server distributes the reconstructed local models to all clients for cross-validation. As DP is used on the server for linear combination and HE is used on the client, the model is restrictively protected during the exchanging process among server and clients. The performance of local model might drop due to either precision loss using quantized HE and linear reconstruction for cross-validation. However, in Fig. 6, our experimental results show that there is no significant loss in testing accuracy. Therefore, the privacy budget ϵ can be expressed as the logarithm of the performance drop d induced by the linear reconstruction outlined as $\epsilon = \ln d$.

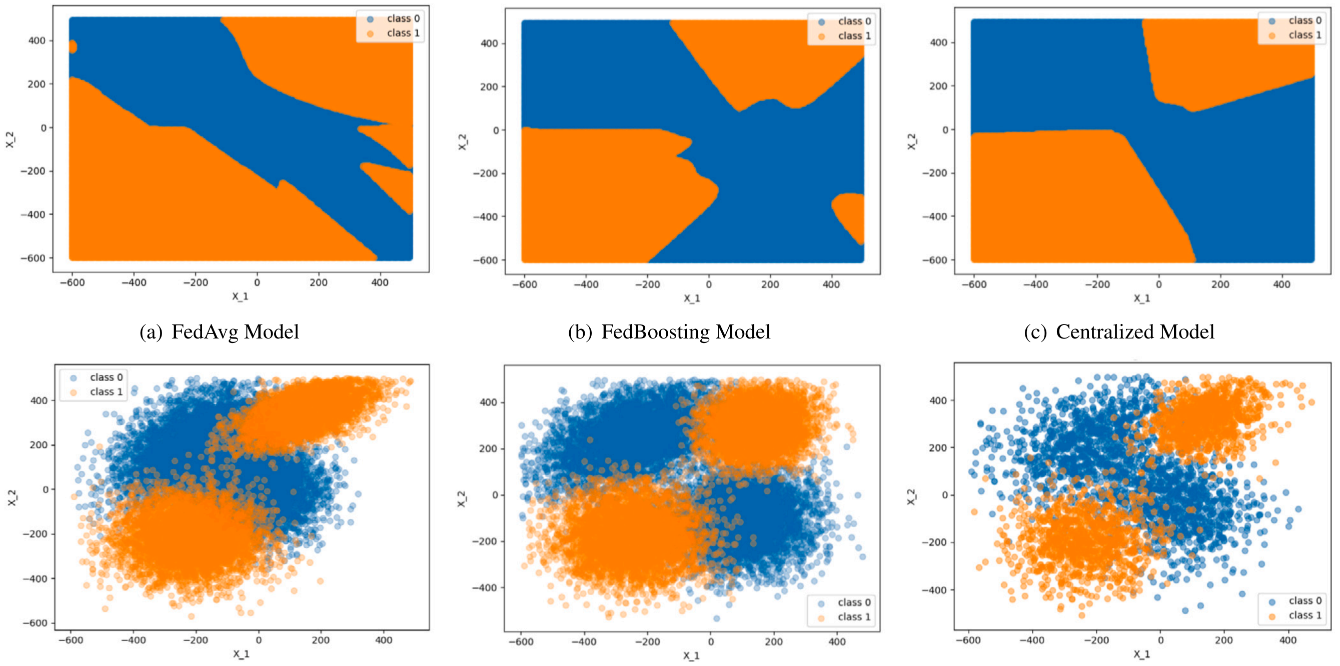
4. Experiments

4.1. Decision boundary comparison using synthetic dataset

We firstly conducted the evaluations using two datasets to compare the decision boundary between FedAvg and FedBoosting. The task is a binary classification problem with 2D feature in order to provide a visible visualization for the decision boundary. We assume the data is subjected to a 2D Gaussian distribution and two datasets was randomly sampled with different mean centers and stand deviations in order to simulate the Non-IID scenario. Individual dataset was used for training on a client and the global model were aggregated using FedAvg and proposed FedBoosting, where each of them contains 40 000 samples and was split into a training set and a testing set by a proportion of 9:1. Fig. 3(d), (e), and (f) show those two training datasets and the combined testing dataset respectively. A simple neural network was adopted that contains 2 fully-connected layers following by a *Sigmoid* activation layer and a *Softmax* layer respectively. The first fully-connected layer has 8 hidden nodes and the second one has 2 hidden nodes. The optimizer is *Adam* whose learning rate is 0.003. All the models trained by FedBoosting outperform those trained by FedAvg with batch size of 8 and epoch of 1. Fig. 3(a) and (b) present the visualizations of decision boundary of global models trained using FedAvg, FedBoosting and a centralized training scheme respectively. It can be concluded that the proposed FedBoosting can form a significantly smoother decision boundary compared to FedAvg approach. In addition, the decision boundary of our method is much closer to the model that was trained using centralized scheme, where both two datasets are used together. This study shows that our method is more generalizable in principle.

4.2. Evaluation on text recognition task

We adopt CRNN [52] as the local neural network model for text recognition mission. CRNN uses *VGGNet* [53] as the backbone network for feature extraction, where the fully-connected layers are removed and the feature maps are unrolled along the horizontal axis. To model the sequential representation, a multi-layer Bidirectional Long-Short Term Memory (BiLSTM) network [54] is placed on the top of the convolutional layers that take the unrolled visual features as input and models the long-term dependencies within the sequence in both directions. The outputs of BiLSTM are fed into a *Softmax* layer, and each element unrolled sequence is projected to the probability distribution of possible characters. The character with the highest *Softmax* score is treated as an intermediate prediction. Connectionist Temporal Classification (CTC) [55] decoder is utilized to merge intermediate prediction to produce the final output text. For more details of CRNN model, the reader can refer to its original publication [52].



(a) FedAvg Model (b) FedBoosting Model (c) Centralized Model

(d) Training Dataset One (e) Training Dataset Two (f) Testing Dataset

Fig. 3. The figures (a), (b) and (c) in the first row show the decision boundaries of global models trained using FedAvg, FedBoosting and non FL method (centralized training scheme). The figures (d), (e) and (f) in second row show two training datasets for two clients in a FL setting and the testing dataset. The model obtained in figure (c) is trained using all datasets including (d) and (e).

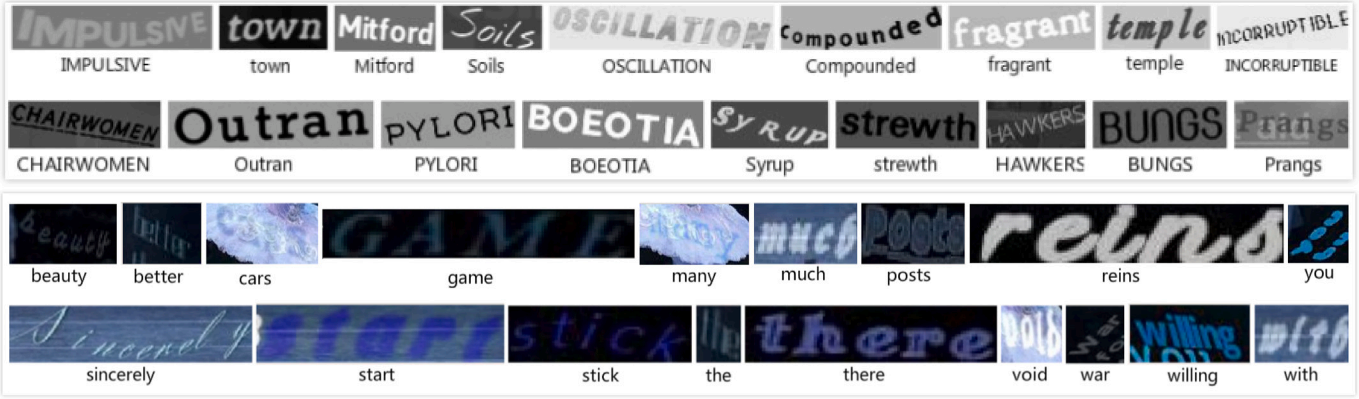


Fig. 4. Visual example of training images taken from *Synth90K* (top two rows) and *SynthText* (bottom two rows) datasets.

4.2.1. Experimental setting

The proposed model is trained on two large-scale synthetic datasets, *Synthetic Text 90k* and *Synthetic Text*, without fine-tuning on other datasets. Models are tested on other four standard datasets to evaluate their general recognition performances. For all experiments, the prediction performance is measured using word-level accuracy.

Synthetic Text 90k [56] (Synth90K) is one of two training datasets for all the experiments in this paper. The dataset contains about 7.2 million images and their corresponding ground truth words. We split the images into two sets for FedBoosting, the first one containing 6.5 million images is for training, the rest is for validation which contains 0.7 million images.

Synthetic Text [56] (SynthText) is the second training dataset we used. The dataset has about 85K natural images containing many synthetic texts. We cropped all the texts via labeled text bounding boxes to build a new dataset of 5.3 million text images. We split it into training dataset of 4.8 million images and validation dataset of 0.5 million images for FedBoosting.

IIIT 5K-Words [57] (IIIT5K) is collected from the internet containing 3000 cropped word images in its testing set. Each image contains a ground truth word.

Street View Text [58] (SVT) is collected from the *Google Street View*, consists of 647 word images in its testing set. Many images are severely corrupted by noise and blur, or have very low resolutions.

SCUT-FORU [59] (SCUT) consists of 813 training images and 349 testing images. The background and illumination vary in large scales in the dataset.

ICDAR 2015 [60] (IC15) contains 2077 cropped images with relatively low resolutions and multi-oriented texts. For fair comparison, we discard the images that contain non-alphanumeric characters, which results in 1811 images.

Fig. 4 shows some visual examples from *Synth90K* and *SynthText*, where large variation of backgrounds and texts can be observed in the images between two datasets. Therefore, we can conclude that two datasets are of Non-IID. All the training and validation images are scaled to the size of 100 ∗ 32 in order to fit the model using mini-batch and accelerate the training process. Testing images in Tables 1

Table 1
Recognition accuracies (%) on four testing datasets. “90K” and “ST” stand for *Synth90K* and *SynthText* datasets respectively. The results of the first row (CRNN*) and the second row (CRNN) are produced by the normal CRNN model without using FL framework, where CRNN* corresponds to accuracies reported in [52] and CRNN corresponds to the results reproduced using our implementation.

Method	Dataset	#Batch	#Epoch	IIIT5K	SVT	SCUT	IC15
CRNN*	90K	–	–	81.20	82.70	–	–
CRNN	ST	512	–	76.07	77.60	89.38	55.92
	ST	800	–	73.69	78.00	86.94	58.88
	90K	512	–	80.95	86.40	87.49	67.43
	90K	800	–	80.71	83.60	87.80	62.50
	ST & 90K	512	–	83.81	90.40	93.08	71.71
	ST & 90K	800	–	85.48	88.00	93.78	72.70
FedAvg	ST & 90K	512	1	85.48	87.60	93.31	73.36
	ST & 90K	512	3	80.83	87.60	91.11	64.80
	ST & 90K	800	1	86.67	89.60	94.49	72.37
	ST & 90K	800	3	81.82	88.00	93.47	70.07
FedAdam	ST & 90K	256	1	84.56	89.20	93.65	70.87
	ST & 90K	512	1	85.33	88.40	93.21	73.40
	ST & 90K	800	1	87.30	89.60	93.94	73.58
Ours (two clients)	ST & 90K	256	1	85.83	89.20	94.26	72.37
	ST & 90K	256	3	84.88	91.20	94.65	70.07
	ST & 90K	512	1	85.12	88.00	93.39	74.34
	ST & 90K	512	3	87.38	90.80	93.86	70.39
	ST & 90K	800	1	87.62	89.20	94.18	**75.99**
	ST & 90K	800	3	85.60	**91.60**	94.65	70.72
Ours (four clients)	ST & 90K	256	1	85.93	89.60	94.48	72.25
	ST & 90K	512	1	86.38	89.20	92.90	74.37
	ST & 90K	800	1	**87.92**	90.40	**94.78**	75.79

and 3 are scaled proportionally to match with the height of 32 pixels. While in Figs. 5, 6 and 7, testing images are processed in the same way with what training and validation images do aforementioned. The testing images whose label length are less than 3 or more than 25 characters are dropped out due to the limitation of CTC. In one case, we deploy the *Synth90K* and *SynthText* datasets on two clients each. In the other case, we split the two *Synth90K* and *SynthText* datasets into four subsets and spread them across four clients. On the local training nodes, *AdaDelta* is used for back-propagation optimization and the initial learning rate is set to 0.05. Regarding to HE we set 128 as key size in bits. The whole gradient are split into 100 pieces and $\hat{p} = 0.9$. Our method is implemented using *Keras* and *Tensorflow*, and the source code is publicly available to ensure reproducibility, which can be run in a distributed multiple GPUs setup.

4.2.2. Classification results

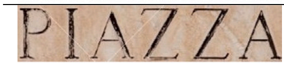
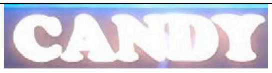
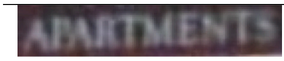
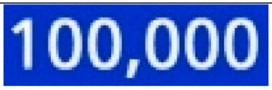
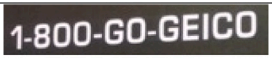
Table 1 shows the comparison results on testing datasets with different training hyper-parameters including batch size, number of epoch and client number. The results of the first row (CRNN*) and the second row (CRNN) are produced by the original CRNN model without using FL framework, where CRNN* corresponds to accuracies reported by its authors in [52] and CRNN corresponds to the results reproduced by us. Compared to the original CRNN model, FedAvg shows a decent amount of improvement. For example, the FedAvg model with batch size of 800 and epoch of 1 reports 86.67% on *IIIT5k* dataset, where an improvement of 1.19% is achieved compared to CRNN that is of 85.48% using the same setting. An improvement of 1.65% of FedAvg with the batch size of 512 and epoch of 1 can be observed on *IC15* dataset. FedAdam [45] exhibits a slight advantage over FedAvg in many cases, but the difference is not substantial. This is because FedAdam, like FedAvg, averages all the local models, introducing local model divergence into the global model. Despite the application of *Adam* following the averaging process in FedAdam, the negative consequences of local model divergence on the global model’s performance remain significant. More importantly, the proposed FedBoosting achieves the highest accuracy across all the four testing datasets, where 87.92%, 91.60%, 94.78% and 75.99% are reported on *IIIT5k*, *SVT*, *SCUT* and *IC15* respectively. Our method outperforms both FedAvg and non-FL methods by significant margins. In addition, we investigated the impact of the client count on performance by examining two distinct configurations: two clients and four clients. Generally, it is found that the models trained using four clients tend to outperform those trained with two clients. This can be attributed to the division of the two large-scale datasets into four subsets, thereby reducing the training iteration count in each epoch. As previously mentioned, fewer iterations result in a reduction in model divergence. More qualitative results are shown in Table 2.

It can be observed that the FedAvg models with bigger batch size and smaller epoch have better performance. In other words, the models perform better when model integration occurs more frequently, which however will increase the communication cost. In Table 1, the model with 256 batch size and 1 epoch even produces no result due to model divergence after a few rounds of integration. The potential reason could be the extreme differences in parameters that are learned on each local machine. Fig. 5 shows the comparison of convergence curves between FedAvg and proposed FedBoosting. The convergence curves of FedAvg model with smaller batch size or larger epoch are always lower than the curves of larger batch size or smaller number of epoch. For example, by the strategy of FedAvg, the model with batch size of 800 and epoch of 1 (iterated 6689 and 9030 times per epoch on *SynthText* and *Synth90K* datasets respectively) performs significant better than the model with batch size of 512 and epoch of 3 (iterated 10,452 and 14,110 times per epoch on *SynthText* and *Synth90K* datasets respectively) on *IIIT5K* testing dataset. On the other hand, the accuracy curves of FedBoosting models (see the second row in Fig. 5), do not have such issue. Therefore, we can conclude that the boosting strategy we propose can overcome the model collapse issue of FedAvg to a great extent.

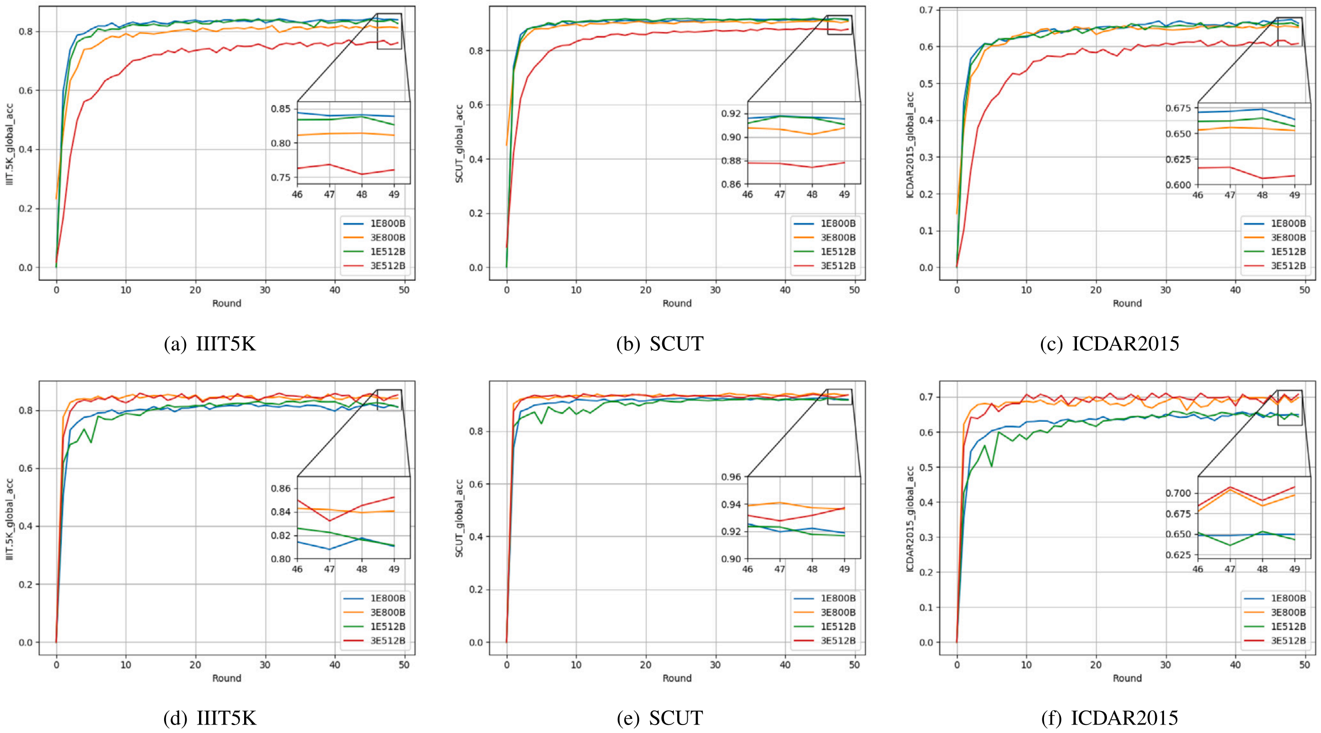
4.2.3. Encryption impact

Table 3 provides the comparison results on three testing datasets (*IIIT5K*, *SVT* and *IC15*) with different FL gradients merging methods and encryption modes under the hyper-parameters of 800 batch size and 1 epoch. The results of FedAvg illustrate that by using HE, although it has slight precision loss in processing of dividing the whole gradient into many pieces, accuracy has nearly no reduction on testing datasets. Even it has accuracy raising on *IC15* dataset from 72.37% to 73.00%. On the other two testing datasets, the losses of accuracy are 0.27% and 1.6% separately. Same for FedBoosting models, testing results show

Table 2
Visual example of testing results using FedAvg and FedBoosting models with a batch size of 512 and epochs of 3. Incorrectly predicted characters are highlighted in red, and green characters in brackets are the ones missing from the predictions.

Samples	FedAvg	FedBoosting
	MOUNTRIN	MOUNTAIN
	STATIONNN	STATION
	MANHATITAN	MANHATTAN
	PLAZZA	PIAZZA
	VIRG(I)N	VIRGIN
	MAXIVUS	MAXIMUS
	UGREENN	GREEN
	JECOIL	ECOIL
	DIR(E)CION	DIRECTORY
	CENIRAL	CENTRAL
	CANSY	CANDY
	ABARTMENTS	APARTMENTS
	WORKSHOPH(E)	WORKSHOPE
	TISC	TSC
	CRIAVEN	CRAVEN
	20MBIES	ZOMBIES
	344	844
	17040	1040
	1001000	100000
	1800GOGEICO	1800G0GE1CO
	YMCA	IMCA

a slight accuracy reduction which can be tolerant when only use HE. While adding DP into FedBoosting with HE, it has nearly no affection on the accuracy of global model. As DP encryption is only used to encrypt local gradients between clients for evaluation and get the results on all clients' validation datasets. Table 4 illustrates the influence of DP on the local model's performance. Validation experiments employing DP are executed on separate clients, necessitating the involvement of HE. We posited that HE exerts negligible effects on validation accuracy, given that it only alters the final two digits out of 32 decimal places. The perturbation to p_r is relatively minimal referring to Eq. (8), leading to the conclusion that DP has a limited impact on the generation of global gradients. However, testing results have a fall down between common FedBoosting and encrypted FedBoosting models, *e.g.* accuracy reduced from 87.62% to about 85% on *IIIT5K* and also have an approximately



(a) IIIT5K

(b) SCUT

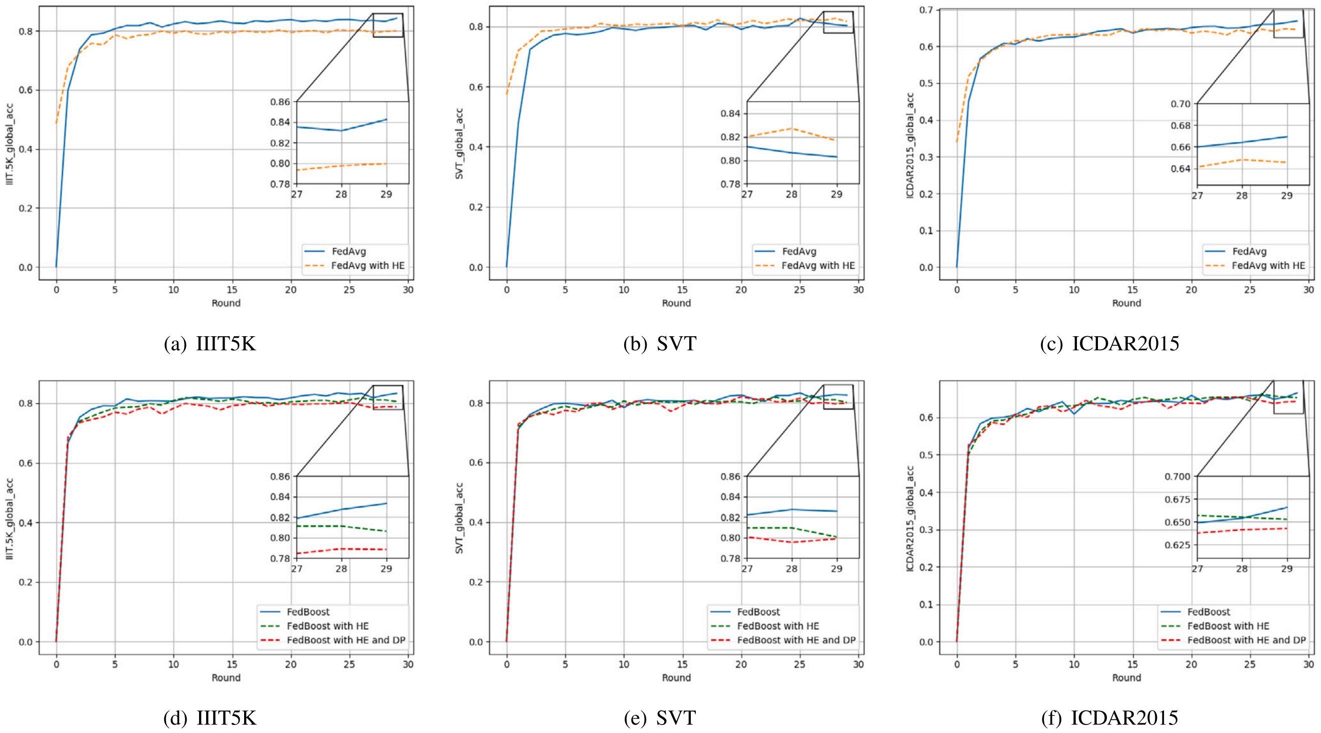
(c) ICDAR2015

(d) IIIT5K

(e) SCUT

(f) ICDAR2015

Fig. 5. Testing accuracy of FedAvg (first row) and FedBoosting (second row) models over rounds with datasets of *IIIT5K*, *SCUT* and *IC15*. "1E800B" means the model is trained on client with 800 batch size and 1 epoch. All samples in these testing parts are resized to 100 ∗ 32, which is different with the processing in Table 1.



(a) IIIT5K

(b) SVT

(c) ICDAR2015

(d) IIIT5K

(e) SVT

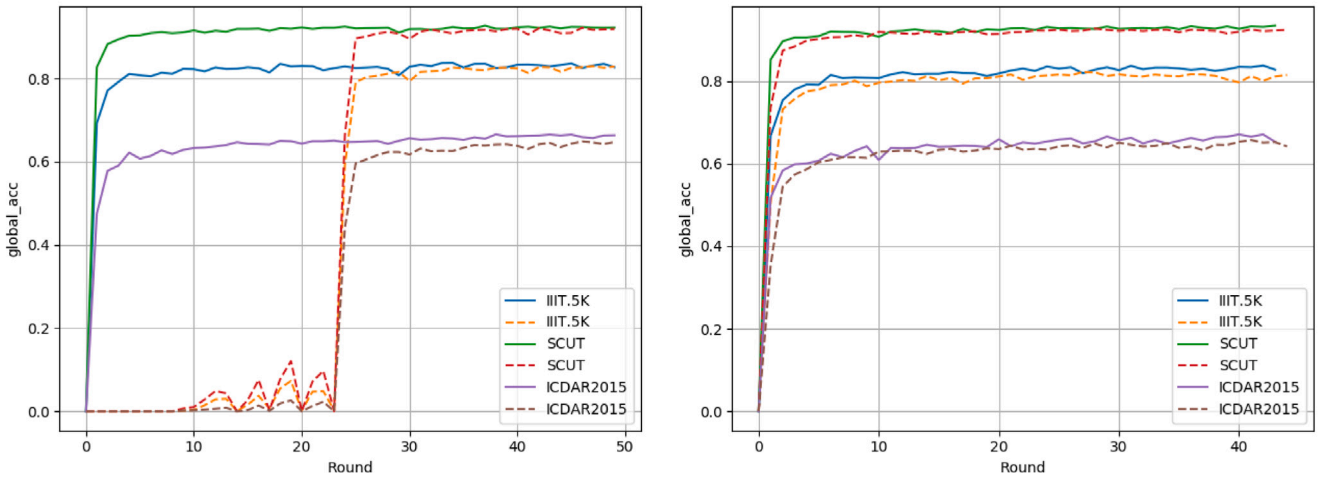
(f) ICDAR2015

Fig. 6. Testing accuracy of FedAvg (first row) and FedBoosting (second row) models with and without using encryption protocol over training rounds on *IIIT5K*, *SVT* and *IC15* datasets.

3% on *IC15*. That is normal fluctuate for training DL models. Although all three testing accuracies have different degrees of reduction, from the curve lines in Fig. 6, accuracy climbing trend presents the differential under different encryption modes. It can be observed that differentials on most testing datasets are rather small.

4.2.4. *Discussion on training and validation results*

We then consider that the reason for the divergence issue is the quality of datasets, see Fig. 4. That is to say, each local model trained on different private datasets has surely different generalization abilities. In our experiments, aggregating the global model rudely by averaging



(a) 256 batch size and 1 epoch

(b) 800 batch size and 1 epoch

Fig. 7. Performance comparison on how the training results affect the model performance. Solid lines refer to global models aggregated with training and validation results, while dash lines refer to global models aggregated only with validation results.

Table 3
Recognition accuracies (%) on three testing datasets. All experiments are using batch size of 800 and epoch of 1.

Method	Encryption	IIIT5K	SVT	IC15
FedAvg	N/A	86.67	**89.60**	72.37
	HE	86.40	88.00	73.00
FedBoosting	N/A	**87.62**	89.20	**75.99**
	HE	85.12	88.40	72.70
	HE+DP	85.00	88.80	72.37

Table 4
Validation accuracies from local models without and with linear reconstruction.

Client	#Batch/#Epoch	w/o DP	w/ DP	Gap
1	256/1	94.34%	92.23%	−2.11%
2		90.32%	86.83%	−3.49%
1	256/3	95.81%	92.82%	−2.99%
2		90.26%	86.92%	−3.34%
1	512/1	94.09%	92.31%	−1.78%
2		88.34%	85.29%	−3.05%
1	512/3	96.46%	92.89%	−3.57%
2		91.84%	86.71%	−5.13%
1	800/1	94.71%	92.52%	−2.19%
2		89.44%	87.19%	−2.25%
1	800/3	97.08%	92.74%	−4.34%
2		93.45%	87.91%	−5.54%

all the weights of local models may cause the decreasing of generalization ability, especially when the local updating iteration number is large (i.e. small batch size or large epoch number). So the proposed FedBoosting prefers to give a more fair weight instead of a mean value by trading off the training and validation performance of a local model. Following this thought FedBoosting first considers each model's validation results on every client's validation dataset, then collaborates with training results to compute weights of local models. The reason we think over training results is that usually a local model trained on high quality dataset has a nice fitness, while which may perform badly on a poor quality validation datasets. It is unfair to say this model has a poor generalization ability only considering its validation performance on different quality datasets. Inversely, a model that is trained on a poor quality dataset may perform very well on a high quality validation dataset as well, but we do not want this kind of local model to occupy too much among the global model.

Furthermore, if only use the training results to show the importance of the validation results, it is theoretically unsound for several reasons: (a) If the client training dataset is of good quality, it will invariably yield a lower loss. This can cause the client model to disproportionately influence the global model than other local models with poor quality dataset. (b) Some datasets will come with some bad samples, which can unexpectedly enhance the model's generalization and make it perform better than a good quality dataset on the test dataset. (c) If a local model is overfitted, this can lead to impaired performance of the global model. (d) If a particular client dataset is compromised with issues, such as messed up data and labels, then using only the training results will corrupt the global model.

To this end, to leverage the performance of a local model, we first sum the validation results as a reference representing the local model's generalization ability. Furthermore, training results are taken into account to rectify the reference to obtain the final weights for each local model. It is observed in our experiments that the weights are about 55% for the local model trained on *Synth90K* dataset and 45% on *SynthText* dataset, which is reasonable cause we can see the accuracy results in Table 1 that CRNN models trained on *Synth90K* dataset always obtain better performance comparing with whose trained on *SynthText* dataset. While if we get rid of training results, the weight for the local model trained on *Synth90K* dataset would be smaller than which on *SynthText* dataset.

To prove the above idea in FedBoosting, a performance comparison is given here. It is commonly accepted that generalization ability is a good metrics of judging a model's performance, whereas only considering generalization ability is not feasible for our proposed method FedBoosting. Otherwise, it is impossible as well to deploy our method only considering training results and get rid of validation results, which may lead to an extremely unfair situation that local model trained on *Synth90K* may take a weight up to about 80% for the global model. So the following content is mainly talking about how training results work in FedBoosting. We trained a global model with 256 batch size and 1 epoch under the strategy of FedBoosting without considering training results. As described above, the reason of thinking over training results is to rectify the weight for local model. From Fig. 7(a), we can see that the global model without taking training results gains a delay convergence at round 24. While in other experiments, models all converge quickly and properly under the supervision of training results. In the meantime, the performance of global model with training results is always better than that without training results. As a supplement, we visualize the global testing accuracy of two models with 800 batch size and 1 epoch in Fig. 7(b), one uses training results and the other one does not. Two models converge normally in this case, but the model performance of using training results outperforms all the time. From all above, we consider that using training results to supervise the local weight is essential in our scenario. To clarify, all testing images during training are resized to 100 ∗ 32, which is different to individual testing experiments where testing images are resized to W ∗ 32, where W is the proportionally scaled with heights, but at least 100 pixels. That is why accuracies in Fig. 7 are lower than those in Table 1. Please refer to our codes for more details.

5. Conclusion

In this study, the paper presents a novel boosting methodology called FedBoosting, specially tailored for the FL framework. This approach was developed to overcome the constraints encountered when using FedAvg on Non-IID datasets. The inherent limitations of FedAvg become more pronounced when dealing with Non-IID data, which led to the necessity of devising an alternative strategy. To ensure the security against gradient leakage attacks, a well-structured gradient sharing protocol was put into practice. This protocol leveraged advanced cryptographic techniques involving both HE and DP to create a secure environment for collaboration. The application of these cryptographic methods ensures that the integrity of the local data is maintained, and privacy is protected. An extensive comparison study was undertaken, encompassing both synthetic datasets and widely-accepted public text recognition benchmarks. The results of these comparative analyses clearly illustrate the superior performance of FedBoosting over the conventional FedAvg scheme on Non-IID datasets. Several metrics were examined, and various experimental setups were explored to substantiate the findings. A thorough theoretical investigation was carried out. This study focused on diverse areas such as model convergence in FL scenarios, potential privacy leakage from gradients, and the development of a more efficient method for gradient quantization. These areas were identified which hold great promise for future advancements.

On the other hand, we acknowledge the limitations inherent in our FedBoosting, particularly in relation to the dual-layer encryption component. This limitation arises from the substantial assumption necessitated by the protocol, which presupposes that no client will engage in a conspiracy with the server to disclose the private key. Such an assumption, while theoretically possible, represents a point of vulnerability, as it does not account for the potential threats posed by rogue clients or server. Thus, the reliability of the FedBoosting's encryption layers fundamentally depends on this strong assumption, underscoring the need for further investigation into more robust, adversarial-resilient methods within the context of FL.

As we look to the future, our work will concentrate on refining the existing system through the incorporation of adaptive aggregation techniques. By moving beyond simple weighted averaging, we aim to develop an intelligent system that dynamically adjusts according to the data and the learning scenario. This transition will undoubtedly render the system more efficient and sophisticated.

CRediT authorship contribution statement

Hanchi Ren: Conceptualization, Methodology, Software, Investigation, Validation, Visualization, Writing – original draft. **Jingjing Deng:** Conceptualization, Methodology, Investigation, Writing – review & editing, Supervision. **Xianghua Xie:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration, Resources, Funding acquisition. **Xiaoke Ma:** Writing – review & editing, Supervision, Funding acquisition. **Yichuan Wang:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The datasets are public available.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) UK, Data Release - Trust, Identity, Privacy and Security [Grant Number: EP/N028139/1] and Key Research and Development Program of Shaanxi China (Program No. 2021ZDLGY02-02).

References

[1] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al., Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (5) (2017) 1333–1345.

[2] E. Hesamifard, H. Takabi, M. Ghasemi, R.N. Wright, Privacy-preserving machine learning as a service, Proc. Priv. Enhanc. Technol. 2018 (3) (2018) 123–142.

[3] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, J. Passerat-Palmbach, A generic framework for privacy preserving deep learning, 2018, arXiv preprint arXiv:1811.04017.

[4] M. Al-Rubaie, J.M. Chang, Privacy-preserving machine learning: Threats and solutions, IEEE Secur. Priv. 17 (2) (2019) 49–58.

[5] J. Liu, X. Meng, Survey on privacy-preserving machine learning, J. Comput. Res. Dev. 57 (2) (2020) 346.

[6] H.C. Tanuwidjaja, R. Choi, S. Baek, K. Kim, Privacy-preserving deep learning on machine learning as a service—A comprehensive survey, IEEE Access 8 (2020) 167425–167447.

[7] N. Koti, M. Pancholi, A. Patra, A. Suresh, {SWIFT}: Super-fast and robust privacy-preserving machine learning, in: Security Symposium, 2021.

[8] J. Konecný, H.B. McMahan, F.X. Yu, P. Richtárik, A.T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, 2016, ArXiv abs/1610.05492.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: AIS, PMLR, 2017, pp. 1273–1282.

[10] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Found. Trends® Mach. Learn. 14 (1–2) (2021) 1–210.

[11] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proc. Mach. Learn. Syst. 2 (2020) 429–450.

[12] V. Smith, C.-K. Chiang, M. Sanjabi, A.S. Talwalkar, Federated multi-task learning, in: NIPS, 2017, pp. 4424–4434.

[13] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-IID data, 2018, ArXiv abs/1806.00582.

[14] M.G. Arivazhagan, V. Aggarwal, A.K. Singh, S. Choudhary, Federated learning with personalization layers, 2019, ArXiv abs/1912.00818.

[15] P.P. Liang, T. Liu, L. Ziyin, R. Salakhutdinov, L.-P. Morency, Think locally, act globally: Federated learning with local and global representations, 2020, ArXiv abs/2001.01523.

[16] T. Lin, L. Kong, S.U. Stich, M. Jaggi, Ensemble distillation for robust model fusion in federated learning, Neural Inf. Process. Syst. 33 (2020) 2351–2363.

[17] C. Li, G. Li, P.K. Varshney, Decentralized federated learning via mutual knowledge transfer, IEEE Internet Things 9 (2) (2021) 1136–1147.

[18] D. Li, J. Wang, Fedmd: Heterogenous federated learning via model distillation, 2019, arXiv preprint arXiv:1910.03581.

[19] X. Peng, Z. Huang, Y. Zhu, K. Saenko, Federated adversarial domain adaptation, in: International Conference on Learning Representations, 2019.

[20] A. Ghosh, J. Chung, D. Yin, K. Ramchandran, An efficient framework for clustered federated learning, Neural Inf. Process. Syst. 33 (2020) 19586–19597.

[21] C. Briggs, Z. Fan, P. Andras, Federated learning with hierarchical clustering of local updates to improve training on non-IID data, in: IJCNN, IEEE, 2020, pp. 1–9.

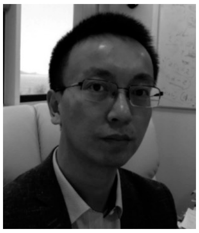
[22] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of FedAvg on non-IID data, in: International Conference on Learning Representations, 2019.

[23] Y. Chen, X. Yang, X. Qin, H. Yu, P. Chan, Z. Shen, Dealing with label quality disparity in federated learning, in: Federated Learning: Privacy and Incentive, Springer, 2020, pp. 108–121.

[24] B. Pejó, G. Biczók, Quality inference in federated learning with secure aggregation, IEEE Trans. Big Data (2023).
[25] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: ACM SIGSAC CCCS, 2015, pp. 1322–1333.
[26] L. Melis, C. Song, E. De Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: IEEE S&P, IEEE, 2019, pp. 691–706.
[27] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the GAN: information leakage from collaborative deep learning, in: ACM SIGSAC CCCS, 2017, pp. 603–618.
[28] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, in: NIPS, 2019, pp. 14774–14784.
[29] B. Zhao, K.R. Mopuri, H. Bilen, idlg: Improved deep leakage from gradients, 2020, arXiv preprint arXiv:2001.02610.
[30] H. Ren, J. Deng, X. Xie, GRNN: Generative regression neural network–a data leakage attack for federated learning, ACM Trans. Intell. Syst. Technol. (2021).
[31] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: ICTACT, Springer, 1999, pp. 223–238.
[32] B. McMahan, D. Ramage, Federated Learning: Collaborative Machine Learning Without Centralized Training Data, Vol. 3, Google Research Blog, 2017.
[33] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, H. Yu, Federated learning, Synth. Lect. Artif. Intell. Mach. Learn. 13 (3) (2019) 1–207.
[34] W. Zhang, Y. Qiu, S. Bai, R. Zhang, X. Wei, X. Bai, Fedocr: Communication-efficient federated learning for scene text recognition, 2020, arXiv preprint arXiv:2007.11462.
[35] A.C.-C. Yao, How to generate and exchange secrets, in: SFCS, IEEE, 1986, pp. 162–167.
[36] O. Goldreich, Secure multi-party computation, 1998, Manuscript. Preliminary version 78.
[37] M. Hao, H. Li, G. Xu, S. Liu, H. Yang, Towards efficient and privacy-preserving federated deep learning, in: ICC, IEEE, 2019, pp. 1–6.
[38] C. Xu, J. Ren, D. Zhang, Y. Zhang, Z. Qin, K. Ren, GANobfuscator: Mitigating information leakage under GAN via differential privacy, IEEE Trans. Inf. Forensics Secur. 14 (9) (2019) 2358–2371.
[39] J. Zhao, Y. Chen, W. Zhang, Differential privacy preservation in deep learning: Challenges, opportunities and solutions, IEEE Access 7 (2019) 48901–48911.
[40] D. Byrd, A. Polychroniadou, Differentially private secure multi-party computation for federated learning in financial applications, in: Proceedings of the First ACM International Conference on AI in Finance, 2020, pp. 1–9.
[41] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou, A hybrid approach to privacy-preserving federated learning, in: ACM AIS, 2019, pp. 1–11.
[42] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, B. Thorne, Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption, 2017, arXiv preprint arXiv:1711.10677.
[43] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, Y. Liu, Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning, in: 2020 Annual Technical Conference, 2020, pp. 493–506.
[44] H. Fang, Q. Qian, Privacy preserving machine learning with homomorphic encryption and federated learning, Future Internet 13 (4) (2021) 94.
[45] S.J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, H.B. McMahan, Adaptive federated optimization, in: International Conference on Learning Representations, 2020.
[46] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[47] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (7) (2011).
[48] M. Zaheer, S. Reddi, D. Sachan, S. Kale, S. Kumar, Adaptive methods for nonconvex optimization, Adv. Neural Inf. Process. Syst. 31 (2018).
[49] Y. Wang, L. Lin, J. Chen, Communication-efficient adaptive federated learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 22802–22838.
[50] S.J. Reddi, S. Kale, S. Kumar, On the convergence of adam and beyond, in: International Conference on Learning Representations, 2018.
[51] C. Xu, Z. Hong, M. Huang, T. Jiang, Acceleration of federated learning with alleviated forgetting in local training, in: International Conference on Learning Representations, 2021.
[52] B. Shi, X. Bai, C. Yao, An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, IEEE Trans. Pattern Anal. Mach. Intell. 39 (11) (2016) 2298–2304.
[53] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations (ICLR 2015), Computational and Biological Learning Society, 2015.
[54] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2008) 855–868.
[55] A. Graves, S. Fernández, F. Gomez, J. Schmidhuber, Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, in: ICML, ACM, 2006, pp. 369–376.
[56] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Synthetic data and artificial neural networks for natural scene text recognition, in: NIPS Deep Learning Workshop, Neural Information Processing Systems, 2014.
[57] A. Mishra, K. Alahari, C. Jawahar, Scene text recognition using higher order language priors, 2012.
[58] K. Wang, B. Babenko, S. Belongie, End-to-end scene text recognition, in: ICCV, IEEE, 2011, pp. 1457–1464.
[59] S. Zhang, M. Lin, T. Chen, L. Jin, L. Lin, Character proposal network for robust text extraction, in: ICASSP, IEEE, 2016, pp. 2633–2637.
[60] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V.R. Chandrasekhar, S. Lu, et al., ICDAR 2015 competition on robust reading, in: ICDAR, IEEE, 2015, pp. 1156–1160.

Hanchi Ren received the M.Sc. and Ph.D. degree in computer science from Swansea University, U.K., in 2016 and 2023. He is currently an academic tutor in the Computer Vision and Machine Learning Laboratory, Department of Computer Science, Swansea University. His research subject is on privacy-preserving federated learning and machine learning and artificial intelligence in general and applications in computer vision and medical image analysis.

Jingjing Deng received his Ph.D. in Visual Computing from Swansea University, U.K., in 2017. He holds the position of Assistant Professor in the VIViD Group (Vision, Imaging, and Visualization in Durham), Department of Computer Science, Durham University, U.K. His research covers a wide range of topics in computer vision and machine learning. He found the Rand2AI Lab at Durham in 2022, and the team focuses on developing computational models that can cultivate and generalize intelligence from and for the complex world.



Xianghua Xie received the M.Sc. and Ph.D. degrees in computer science from the University of Bristol, U.K., in 2002 and 2006, respectively. He is currently a Full Professor with the Department of Computer Science, Swansea University, UK, and is leading the Computer Vision and Machine Leaning Laboratory. He has published more than 160 refereed conference and journal publications and (co-)edited several conference proceedings. His research interests include various aspects of pattern recognition and machine intelligence and their applications to real-world problems. He is a member of BMVA. He is an Associate Editor of several journals, including Pattern Recognition and IET Computer Vision.

Xiaoke Ma received the Ph.D. degree in computer science from Xidian University, China, in 2012. He was a Postdoctoral Fellow with the University of Iowa, USA, from 2012 to 2015. He is a Full Professor with the School of Computer Science and Technology, Xidian University. Dr. Ma is an Ad Hoc Reviewer for many international journals and publishes about 100 papers in the peer-reviewed international journals, such as IEEE TKDE, IEEE Transactions On Cybernetics, ACM TKDD, Pattern Recognition, Information Sciences, Bioinformatics, PLoS Computational Biology, Nuclear Acids Research, IEEE TCBB, and IEEE TNB. His research interests include machine learning, data mining, and bioinformatics.

Yichuan Wang received his Ph.D. degrees in computer system architecture from Xidian University of China in 2014. He is an ACM member and a CCF member. Now he is a Lecturer in Xi'an University of Technology and with Shaanxi Key Laboratory of Network Computing and Security Technology. His research areas include cloud computing and networks security.