Lead federated neuromorphic learning for edge artificial intelligence

Helin Yang (✉ hyang013@e.ntu.edu.sg)

Nanyang Technological University <https://orcid.org/0000-0001-9697-7470>

Kwok-Yan Lam

Nanyang Technological University

Liang Xiao

Xiamen University

Zehui Xiong

Singapore University of Technology and Design

Hao Hu

Nanyang Technological University <https://orcid.org/0000-0003-2892-1971>

Dusit Niyato

Nanyang Technological University

Vincent Poor

Princeton University

Article

Keywords:

Posted Date: January 21st, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1270383/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Version of Record: A version of this preprint was published at Nature Communications on July 25th, 2022. See the published version at <https://doi.org/10.1038/s41467-022-32020-w>.

Lead federated neuromorphic learning for edge artificial intelligence

Helin Yang^{1✉}, Kwok-Yan Lam^{1,2✉}, Liang Xiao³, Zehui Xiong⁴, Hao Hu⁵,
Dusit Niyato² and H. Vincent Poor⁶

¹Strategic Centre for Research in Privacy-Preserving Technologies and Systems, Nanyang Technological University, Singapore.

²School of Computer Science and Engineering, Nanyang Technological University, Singapore.

³Department of Informatics, Xiamen University, Xiamen, China.

⁴Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore.

⁵Department of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

⁶Department of Electrical and Computer Engineering, Princeton University, Princeton, USA.

✉Corresponding author. E-mail: hyang013@e.ntu.edu.sg, kwokyan.lam@ntu.edu.sg

Abstract

Despite the great potential of edge artificial intelligence (AI) which is the convergence of edge computing and AI, it acquires sufficiently large/diverse datasets and requires high energy consumption for model training on resource-constrained edge devices, hence hindering the application of edge AI at edge devices. This paper proposes a lead federated neuromorphic learning (LFNL) technique, which is a decentralized energy-efficient brain-inspired computing method, enabling edge devices to collaboratively train a global model while preserving privacy. Experimental results validate that LFNL substantially reduces the data traffic by $>3.5\times$ and computational latency by $>2.0\times$ compared to centralized learning, with a comparable classification accuracy, as well as significantly outperforms local learning with uneven dataset distribution among edge devices. Meanwhile, LFNL significantly reduces the energy consumption by $>4.5\times$ compared to standard federated learning with a slight accuracy loss up to 1.5%. Therefore, the newly proposed LFNL can facilitate the development of brain-inspired computing and edge AI.

Introduction

In recent years, with the rapid development of mobile computing and the Internet of Things (IoT), billions of devices, sensors, tablets, robots, and machines are connected to the Internet, generating very large amounts of data at the ends of computer networks¹. Driven by this trend, a powerful technique termed edge artificial intelligence (AI) amalgamating edge computing and AI²⁻⁶, has been proposed to enable edge devices to locally analyze and process data without offloading collected data to the center. Such a unique ability can ensure data privacy preservation, and reduce data traffic and network latency. Moreover, record accuracies have been achieved by deep learning of neural networks employed for speech recognition, image and video classification, and object detection in edge AI³⁻⁶. Despite these benefits, edge AI also faces the following two fundamental challenges. Firstly, modern AI-based algorithms depend intrinsically on sophisticated learning methods⁷, and more importantly on sufficiently rich training datasets^{8,9}. Thus, the limited local datasets of edge devices make training reliable learning model almost impossible^{8,9}. Secondly, running AI-based algorithms is generally energy-hungry, which inhibits energy-constrained edge devices from training/analyzing data locally^{2,3,10,11}.

One potential technique to address the first challenge is federated learning (FL)^{12,13}. In FL, as reported^{2,4,14-16}, multiple collaborative devices locally train an AI model (i.e., each on its own data, in parallel) without uploading raw data. In this context, devices only upload model parameters to a central server for global model aggregation. Then, the updated model parameters are sent back to devices for the next training epoch, repeating the process until convergence. FL not only enables edge AI to achieve a comparable model quality to centralized learning, but also reduces data traffic and preserves data privacy. For these reasons, FL has recently been applied in privacy-sensitive medical^{9,17-19}, e.g., medical image classification¹⁷. Considering the central coordinator in FL, all clients/devices are required to trust the central server and the training speed is limited by the heterogeneity of edge devices²⁰. To address this issue, decentralized FL has been presented¹⁹⁻²⁴, where model parameters are exchanged only between interconnected devices without using a central server. Even so, cycling repeatedly model aggregation among devices results in increased training latency²⁵⁻²⁷. Furthermore, even if centralized or decentralized FL provides a solution for privacy-enhancing and reliable model training under insufficient datasets at edge devices, model training based on deep learning can consume significant amounts of energy, further hindering application of decentralized FL in energy-constrained edge devices.

As noted above, standard deep learning algorithms, e.g., multi-layer artificial neural networks (ANNs) and convolutional neural networks (CNNs), are generally power-hungry²⁸⁻³¹. To address this challenge, inspired by biological neurons, spiking neural networks (SNNs)^{32,33} have been proposed explored as a promising neuromorphic computing solution for the implementation of AI algorithms in edge devices due to their low energy consumption. SNNs simulate the electrical activity of human-brain systems and

operates with continuous spatio-temporal dynamics and discrete spike events using Integrate-and-Fire (IF) or Leaky IF (LIF) neuron units³⁴. Owing to the inherent parallelism of binary spike-based sparse computing over time steps, SNNs promise fast, sparse, and energy-efficient information processing³⁵⁻⁴⁰. Furthermore, several attempts have been made to combine SNNs with FL^{41,42} to improve both learning capability and energy efficiency, but model parameters are still aggregated by a central server.

In this article, we propose lead federated neuromorphic learning (LFNL), a decentralized brain-inspired computing method based on SNNs, enabling multiple edge devices to collaboratively train a global neuromorphic model without a central coordinator. In particular, we present a leader election scheme to elect one device with high capability (e.g., computation and communication capabilities) as a leader to manage the model aggregation, which effectively accelerates the federated learning speed and defends against model poisoning attacks (Supplementary Fig. 2 and Fig. 4). Experimental results demonstrate that LFNL achieves high classification accuracies of 94.3%, 95.6% and 94.7% on audio, visual and radar signal recognition tasks with uneven dataset distribution among devices. Such high accuracy is approximately equivalent to that of centralizing learning and significantly outperforms local learning. LFNL also substantially reduces data traffic (Fig. 2m and Fig. 6e) and computational latency (Fig. 2n, Fig. 3j, etc.) compared to centralizing learning. The results further verify that LFNL yields approximately state-of-the-art accuracy (up to 1.5% loss) with significant energy consumption reduction ($\sim 4.5\times$) compared to standard federated learning methods. LFNL promises several important benefits for edge AI compared to existing computing paradigms, including privacy enhancement, low computational latency, data traffic reduction, energy efficiency, and robustness. As such, LFNL is envisioned to significantly boost the development of brain-inspired computing and edge AI.

Results

Construction of lead federated neuromorphic learning. In order to enable edge devices to perform computing with low energy consumption, low latency, high-accuracy recognition with privacy-preservation, we developed a lead federated neuromorphic learning (LFNL) system, as shown in Fig. 1. Figure 1a illustrates the schematic diagram of a collaborative human social system. Each human being uses five general sensory organs to observe analog stimulus from outside environment, and then transforms the stimulus into spike signal using specialized neurons before processing them by the human brain nervous system. Each human being builds its corresponding knowledge model, and then shares the model with others to create an optimized knowledge model for better recognition. Inspired by this, a federated neuromorphic learning system is introduced for edge AI (Fig. 1b), where each edge device is equipped with camera sensors (vision), audio detectors (hearing), object sensing (radar), pressure sensors

Fig. 1: Schematic diagram of the lead federated neuromorphic learning system. **a**, Schematic of a social leaning network, where each human being uses five sensory organs to interact the outside environment via neural networks. Human beings in a group exchange learning knowledge with each other for better recognition. **b**, Inspired by the collaborative human learning system (Fig.1 **a**), a federated neuromorphic learning system is introduced to perform model aggregation from edge devices in a group. The integration between devices and outside environment is handled by using sensors (camera, sound collector, radar, touch sensor, etc.). **c**, The structure of an SNN which is adopted to perform neuromorphic computing for edge devices. **d**, Principle of LFNL without a central server, where one device is selected as a leader to manage model aggregation in a group. **e**, Simulated situation of multiple human beings crossing a vehicular road, where multiple edge devices can observe, hear, and sense traffic objects. **f**, Illustration of LFNL-based traffic recognition. The leader (device) leads other followers (devices) to train their own local neuromorphic models independently, and it collects local model parameters ($\mathbf{w}_2, \mathbf{w}_3, \dots, \mathbf{w}_K$) to perform model aggregation before broadcasting the global parameters $\mathbf{w}$ to followers for next local training. The exchange of local and global parameters repeats until the convergence.

(touch), and signal detector (wireless communication). These sensors adopt SNNs as a neuromorphic processor to convert detected information into spike signals. The structure of SNNs with Meta-Dynamic Neurons (MDNs)³¹ is illustrated in Fig. 1c, and the inputs of SNNs are discrete spikes which are encoded from object analog signals (vision, audio, radar, etc.)³¹. The signals in input, hidden and output layers of SNNs are all spike trains (see Methods).

LFNL is implemented with a leader and a number of followers in a group (Fig. 1d), the learning model parameters are shared and exchanged via distributed networks with each device training its model independently on local data. Note that the device with high computation, communication, and energy supply capabilities is elected as the leader to effectively manage model aggregation and accelerate the federated learning speed (leader election protocol and performance evaluation can be seen in Supplementary Fig. 1,2). To better illustrate the concept of LFNL for edge AI, we consider several edge crossing a vehicular road as an example (Fig. 1e). In social networks, human beings share their learning knowledge with each other to provide better object recognition, and one of them with a rich experience acts as a leader to lead the group members to better explore, learn, and adapt to the physical world. Inspired by the human-like learning functionalities, LFNL realizes object recognition by training or evaluating spike signals from auditory, vision, and radar systems using neuromorphic learning (Fig. 1e). The object of the leader is to aggregate the uploaded local neuromorphic model parameters ($\mathbf{w}_2, \mathbf{w}_3, \dots, \mathbf{w}_K$) from followers. All followers only need to send their local model parameters to the leader instead of uploading the raw data. After aggregating the model parameters at each global epoch, the updated global parameter $\mathbf{w}$ will be sent to followers for the next training epoch. More details of the LFNL can be found in Methods.

Application to audio recognition. We first tested the audio recognition capability of LFNL, and selected traffic sound dataset⁴³ for performance evaluation. At first, we introduce four benchmark methods based on SNNs. Local neuromorphic learning (LNL) enables each device to locally train its model without sharing raw data with each other (Fig. 2a). Centralized neuromorphic learning (CNL) uses a central server to collect datasets of all devices for global model training (Fig. 2b). Centralized federated neuromorphic learning (CFNL) can keep the raw data on the device sides (Fig. 2c), and only the local model parameters need to be uploaded to the central server for model aggregation which enhances global accuracy, but it relies on a centralized structure. Transfer neuromorphic learning (TNL) keeps data on the device sides (Fig. 2d), and each device trains its model and then passes it to the next device for training, repeating the process cyclically. However, the devices train in sequence rather than in parallel, leading to a longer training latency^{26,27}.

Fig. 2: LFNL for audio recognition in a vehicular road. **a**, Principle of local neuromorphic learning (LNL) with data and model on the device sides. **b**, Principle of centralized neuromorphic learning (CNL) with data and model being stored at the central server. **c**, Principle of centralized federated neuromorphic learning (CFNL) with data being kept on the device sides, and local model parameters are uploaded to the central server for model aggregation. **d**, Principle of transfer neuromorphic learning (TNL) with data being kept on the device side, and each device trains its model and then passes it to the next device for training, cyclically repeat the process. **e**, An example of audio recognition in a vehicular road, including firetruck, ambulance and general traffic sounds. **f**, **g**, Validation loss curves for three locally training devices and LFNL. **h-j**, Box plots show test accuracy performed for three locally training devices and LFNL with uneven distributions of training dataset (F: firetruck sound class, A: ambulance sound class, T: general traffic sound class). The training dataset distributions of three sound classes for three devices are shown at the top of these figures. **k**, Confusion matrix for the test set in LFNL after training. **l-n**, Test accuracy, data traffic, and training latency comparisons for different learning methods, respectively.

We implemented the experiments on Raspberry PI 4B, Raspberry PI 3B+ and one laptop (see Methods). In the benchmark of LFNL, an SNN has 128-2000-3 neurons (input-hidden-label layer size). For traffic sound dataset, in total, 600 sound samples were used with three classes, including firetruck, ambulance, and general traffic sound samples with each having 200 samples. 80% of the sound samples are used for training, and the remaining samples (20%) are adopted for validation and testing.

As the validation loss is widely used to directly reflect the quality of training capability, we therefore visualized the validation loss values via training epochs for three locally training devices and LFNL. Since device 3 has insufficient training samples (Fig. 2f), it has a higher validation loss than those of the other two devices with more training samples, leading to a lower test accuracy of 84.2% (Fig. 2h). Both device 1 and device 2 achieve considerable high validation loss performance (Fig. 2f), and obtain the test accuracy of 91.3% and 91.4% (Fig. 2h), respectively. However, by applying LFNL, the system achieves a faster training convergence speed (Fig. 2g) and higher test accuracy of 94.3% (Fig. 2h) than those of the three locally training devices. Generally speaking, AI performs well when the training data is sufficient, for example, the training and test accuracy of device 1 and device 2 are considerably good. We further examined the stability and robustness of LFNL with extremely uneven dataset distribution of three classes at three devices. As depicted in Fig. 2i, as each device has very insufficient training samples on one class dataset, the overall test accuracy of the three locally training devices decreases substantially compared to Fig. 2h. Similarly, the phenomenon also happens with the uneven distribution of datasets on device 2 and device 3 (Fig. 2j). However, LFNL still maintains a high test accuracy of 95% and significantly outperforms locally training devices. Moreover, the LFNL results do not deteriorate when we divided the training samples into six smaller parts for six devices (Supplementary Fig.3). In addition, LFNL can effectively defenses the positioning model attack (Supplementary Fig.4).

We further compared the performance of LFNL with other learning methods (Fig. 2l-n). As depicted in Fig. 2l, LFNL not only achieves a similar test accuracy to CNL, but significantly reduces both data traffic size by $>3.5\times$ (Fig. 2m) and training latency by $\sim 2.0\times$ (Fig. 2n). Both CFNL and LFNL train the learning model in parallel, and have approximately similar test accuracy (Fig. 2l) and training latency (Fig. 2n), but the former needs a central server for model aggregation and it also has higher data traffic (Fig. 2m). Unlike LFNL, TNL trains sequentially rather than in parallel and has lower data traffic than that of LFNL (Fig. 2m), but it requires much longer computational latency (Fig. 2n). The results shown in Fig. 2l-n strongly support the conclusion that LFNL is more suitable for edge AI by considering recognition accuracy, data traffic, latency and privacy-preserving factors.

Application to visual recognition. Next, we applied LFNL to implement the visual recognition, as illustrated in Fig. 3a. In the benchmark LFNL, an SNN has 1728-2500-3 neurons (input-hidden-label

layer size). For the traffic image dataset^{44,45}, in total, 872 images are used with three classes, including 160 bicycle images, 205 car images, and 507 traffic light images. 80% of the images are used for training, and the remaining images (20%) are adopted for validation and testing.

Fig. 3: LFNL for visual recognition in a vehicular road. **a**, An example of visual recognition (image classification) in a vehicular road, including bicycle, car, and traffic light image classes. **b**, Validation loss curves for three locally training devices and LFNL. **c**, Confusion matrix for the test set in LFNL after training. **d-f**, Box plots show test accuracy performed for three locally training devices and LFNL with uneven distributions of training dataset (B: bicycle image class, C: car image class, L: traffic light image class). **g**, Illustration of box plots showing test accuracy performed for six locally training devices and LFNL, where the training dataset is divided into six smaller parts for six devices. **h-j**, Validation loss curves, test accuracy and training latency performance comparisons for different learning methods, respectively.

Figure 3b illustrates the validation loss curves of three locally training devices and LFNL. The training dataset distributions of three traffic types (i.e., bicycle (B), car (C) and traffic light (L)) for three devices are shown at the top of Fig. 3d. Due to the insufficient training images, both device 2 and device 3 overfit quickly and result in an unstable training (Fig. 3b), achieving low test accuracies of 88.0% and 78.5% (Fig. 3d), respectively. Compared with the three locally training devices, LFNL overcomes the local overfitting issue, and significantly obtains a smoother and faster convergence speed (Fig. 3b), and achieves a higher test accuracy of 95.6% (Fig. 3d). We further examined the stability and robustness of LFNL with extremely uneven and insufficient dataset distribution. From Fig. 3e-g, we find that the overall test accuracy of locally training devices with uneven and insufficient dataset declines significantly, whereas the LFNL results do not deteriorate. Furthermore, LFNL still robustly achieves a high classification accuracy under the random image rotation angles (Supplementary Fig. 5).

We also demonstrated the performance comparisons for different learning methods (Fig. 3h-j). As shown in Fig. 3h, LFNL achieves the similar training convergence and validation loss values to CNL, CFNL and TNL. In addition, it also obtains a comparable test accuracy to other learning methods (Fig. 3i). However, using our LFNL method, the training latency is significantly reduced in comparison to CNL and TNL (Fig. 3i). We note that both CNL and CFNL require a central structure which increases the data traffic size for edge AI. In these results (Fig. 3b-j), LFNL significantly outperforms individual devices regardless of how uneven the data distributions are, and its recognition capability is close to that of the centralized learning method.

Application to radar signal recognition. This chapter tests the radar signal recognition capability of LFNL, where we simulate a situation (Fig. 4a) that devices (e.g., vehicles) use radar systems to recognize the human being gestures in a vehicular road, such as gesture recognition when persons cross a road, call a taxi, and call for a stop of a bus. In total, 1695 five-class radar gesture samples³⁰ are adopted for classification evaluation, where 80% and 20% of the samples are used for training and testing, respectively. In the benchmark LFNL, an SNN has 4800-1000-5 neurons (input-hidden-label layer size). Figure 4b represents the validation loss curves for three locally training devices and LFNL, where the dataset distribution for the three devices are set as 39.5%, 27.8% and 12.7%, respectively. Similarly, due to the insufficient training samples, device 3 achieves higher validation loss values than those of the other two devices, leading to a lower test accuracy of 78.3% (Fig. 4c). Fortunately, this key challenge can be addressed by using LFNL, as illustrated in Fig. 4c,d, the accuracy of the three locally training devices are obviously improved from 93.2%, 91.1% and 78.3% to 94.7%, respectively. Here, we divided the training samples into six parts for six devices with each being with smaller training dataset, the dataset distribution at six devices are set as 17.7%, 17.7%, 14.8%, 11.8%, 8.9%, and 9.1% (running at the laptop),

respectively. As shown in Fig. 4e, the overall test accuracy of the six locally training devices significantly reduces, especially the performance of device 5 and device 6, because they have very small training samples, whereas the LFNL results do not deteriorate. The confusion matrix of five-label recognition is provided in Fig. 4f. For each gesture class, only very few samples are miss-classified into other classes. Similar to the results (Fig. 2 and Fig. 3), the test accuracy of LFNL still is still equivalent to other learning methods (e.g., CNL and TNL), as shown in Fig. 4g. However, LFNL significantly reduces the training latency compared with CNL and TNL (Fig. 4h), and it does not need a central server compared with CNL and CFNL.

Fig. 4: LFNL for radar signal recognition in a vehicular road. **a**, An example of radar signal recognition in a vehicular road, which has five radar gesture classes. **b**, Validation loss curves for three locally training devices and LFNL. **c**, **d**, Box plots show test accuracy and histogram plot illustrate average test accuracy performed for three locally training devices and LFNL after training. **e**, Illustration of box plots showing test accuracy performed for six locally training devices and LFNL, where the training dataset is divided into six smaller parts for them. **f**, Confusion matrix for the test set in LFNL after training. **g**, **h**, Test accuracy and training latency performance comparisons for different methods, respectively.

Analysis of recognition accuracy and energy consumption. It is also necessary to analyze the reasons why we proposed LFNL-SNN for edge AI instead of using standard AI algorithms. This chapter compares both the accuracy and energy consumption performances (Fig. 5) between LFNL-SNN and standard lead federated learning based ANN (LFL-ANN), in order to reflect the energy efficiency of LFNL. Note that both SNNs and ANNs have the same learning structure for fair comparison. The details of energy consumption analysis can be found in Methods.

Fig. 5: Recognition accuracy and computation energy comparisons between ANNs and SNNs trained on the audio, vision and radar datasets. **a**, Audio classification accuracy comparison between LFNL-SNN and LFL-ANN versus the spike train duration T of the operation of SNNs on traffic sound dataset. Both SNNs and ANNs have the same learning structure with 128-500-3 neurons (input-hidden-label layer size). **b**, **c**, Average test accuracy and energy consumption comparisons when the spike train duration T is 15. **d**, Visual classification accuracy comparison between LFNL-SNN and LFL-ANN versus T of SNNs on traffic image dataset. Both SNNs and ANNs have the same learning structure with 1728-2500-3 neurons. **e**, **f**, Average test accuracy and energy consumption comparisons when the spike train duration T is 15. **g**, Radar gesture classification accuracy comparison between LFNL-SNN and LFL-ANN versus T of SNNs on radar gesture dataset. Both SNNs and ANNs have the same structure with 4800-1000-5 neurons. **h**, **i**, Average test accuracy and energy consumption comparisons when the spike train duration T is 15.

In Fig. 5a,d,g, we find that the accuracy of LFNL-SNN suffers a slight loss (up to 1.5% loss) compared to LFL-ANN. In ANN, neurons characteristic with each other adopting activation coded in high-precision as well as continuous values, and only propagate signals in the spatial region. Different from ANNs, SNNs show signal in spike trains coded in binary events instead of continue activation and each spiking neuron goes through rich dynamic behaviors. Thus, the presented LFNL-SNN generally has more temporal versatility but slightly lower recognition accuracy (Fig. 5a,d,g) compared to LFL-ANN mainly with spatial propagation and continuous activation³⁵⁻⁴². For example, as illustrated in Fig. 5e, the visual recognition accuracy of LFNL-SNN is 94.3%, only slightly lower than that of LFL-ANN with the accuracy of 95.8% in this scenario.

Figure 5c,f,h show the estimated energy consumption of ANNs and SNNs models trained on the audio, vision and radar datasets for three devices, respectively. From these figures, compared with LFL-ANN, the significant larger energy saving gains achieved can be attributed to the sparsity obtained with event-driven spike trains in SNNs. For example, for visual recognition illustrated in Fig. 5f, the energy consumption by LFL-ANN is 13.85 μ J, whereas that of LFNL-based SNNs is 2.92 μ J which is 4.75 $\times$ more efficient. Using LFNL-SNN, the computation energy is substantially reserved and recognition accuracy is also guaranteed for edge AI.

Application in large and high-dimensional datasets. In addition to evaluating performance on audio, visual, and radar signal recognition tasks, LFNL is expected to be effective and robust in a larger and more high-dimensional multi-class classification. Therefore, we further applied LFNL to the classification of MNIST handwritten digits⁴⁶ and speech TIDIGITS^{47,48}.

Figure 6 a-e shows the experiment results for different learning methods on the MNIST dataset with 784-500-10 neurons (input-hidden-label layer size). As observed in Fig. 6a,b, the test accuracies of the three locally training devices are more than 92%, because the three devices have sufficient training samples (25000, 25000 and 5000 samples). By using LFNL, the test accuracy of device 3 can be significantly improved from 92.3% to 97.5%. The confusion matrix of FLNL for the test data set after training is depicted in Fig. 6c, in which the classification accuracy of each class is more than 96%. Figure 6d,e captures the test accuracy and data traffic comparisons. The test accuracy of CNL is slightly higher than that of other three methods (Fig. 6d), but the local dataset of each device is required to share which needs greater data traffic (Fig. 6e) and data sharing leaks the private information. Although TNL has lower traffic size than that of FLNL (Fig. 6e), it needs longer training latency as it is not parallel training.

We further evaluated the classification capability of LFNL on eleven-class speech TIDIGITS dataset, where Fig. 6f-j indicates the results for learning networks with 1640-1000-11 neurons (input-hidden-label layer size). Owing to sufficient training samples at three locally training devices, an approximate test

Fig. 6: Classification evaluation on MNIST and TIDIGITS datasets. **a**, 35 independent test accuracy evaluations for three locally training devices and LFNL after training on MNIST dataset. The dataset distributions for the three devices are accordingly set as 38.8%, 38.5% and 7.7%, respectively. **b**, Box plots show test accuracy for 35 independent test evaluations of Fig. 6a. **c**, Confusion matrix for the test set in LFNL after training. **d**, **e**, Test accuracy and data traffic size performance comparisons for different methods on MNIST dataset, respectively. **f**, **g**, Test accuracy evaluation and box plots show test accuracy for three locally training devices and LFNL on TIDIGITS dataset after 35 independent experiment runs. The dataset distribution for the three devices are accordingly set as 37.7%, 37.7% and 6.2%, respectively. **h**, Confusion matrix for the test set in LFNL after training. **i**, **j**, Test accuracy and training latency performance comparisons for different methods on TIDIGITS dataset, respectively.

accuracy of 95% at three devices can be achieved (Fig. 6f,g), but the performance can be enhanced to 97.1% by using LFNL. Notably, as shown in Fig. 6i, both CNL and TNL obtain slightly higher test accuracy than that of CFNL and LFNL. However, CFNL and LFNL require $2.5\times$ longer training latency than that of CNL and TNL (Fig. 6j) as they train sequentially rather than in parallel, which may not be suitable for real-time edge AI.

We note that the overall accuracy of local training devices substantially declines (Supplementary Fig. 6) when we divide MNIST and TIDIGITS datasets for more small devices, whereas the LFNL results still do not deteriorate.

Discussion and conclusion

In this paper, we have proposed a lead federated neuromorphic learning method for edge AI, namely LFNL, integrating brain-inspired neuromorphic computing and federated learning in the domain of human-like machine intelligence. LFNL enables edge devices to collaboratively train a global reliable model while preserving privacy without a central server, in the presence of uneven and insufficient training data on edge devices. Owing to the decentralized federated learning and parallel training structures of LFNL, it directly replaces the centralized data sharing paradigm across edge device without any central server, and thus significantly reduces the heavy data traffic, enforces data privacy and decreases training latency compared to existing centralized learning methods. Moreover, with the implementation of spike-based processing features in LFNL, our platform is highly economic in energy, which makes LFNL available to energy-constrained edge devices.

The advantages of LFNL were experimentally demonstrated in a series of benchmark comparisons on audio, visual and radar signal recognition tasks under uneven dataset distribution. LFNL achieves an inference accuracy of more than 94% for each task, and it significantly outperforms the locally training method and obtains a comparable recognition accuracy to centralized learning without sharing heavy data traffic, as shown in Fig. 2-4. Due to the spike activation driven in LFNL, the method requires a finite number of training time steps T to optimize LFNL-SNN and obtains slight lower classification accuracy than that of the standard federated learning-based ANNs, but it can significantly reduce the energy consumption for energy-constrained devices (Fig. 5). Due the scalability of LFNL, such a high classification accuracy (up to 97.5%) is still observed, even when we extend it to the classification on larger and more high-dimensional datasets (MNIST and TIDIGITS) (Fig. 6).

In brief, LFNL offers a unique but powerful route to democratize the use of neuromorphic learning in the domain of human-like machine intelligence. Owing to the aforementioned benefits and advantages, LFNL can effectively deploy deep learning of neural networks for resource-constrained edge devices

with various practical applications, such as speech recognition, image and video classification, smart sensing, health monitoring and multiple object detection in edge AI. It also can deploy deep learning on large-scale scientific/industrial systems, for example, autonomous instruments, autonomous vehicles and mission critical diagnostics. This gives us confidence that LFNL will greatly contribute to the development of brain-inspired computing and edge AI.

Methods

SNNs model. In order to enable SNNs to achieve a better learning efficiency and generalization for object recognition/classification in LFNL, SNNs combined with MDN architecture³¹ is used in our work. The MDN is designed with meta neurons including the first-order and second-order dynamics of membrane potentials, as well as the spatial and temporal meta types supported by hyper-parameters³¹.

Fig. 1c shows a typical SNNs architecture with LIF neurons. The spiking neurons in SNNs communicate with each other by spike trains coded in binary events (1: spike, 0: no spike) in a temporal domain over a given number of time steps T , referred to as spike train duration. In this work³⁴, LIF³⁴ is applied to perform standard first-order dynamic spike neurons which includes only up to an attractor. The dynamic behavior of the i -th spike neuron using LIF is characterized by

$$\tau \frac{dU_i(t)}{dt} = \beta U_i(t) + C(t), \quad (1)$$

where $U_i(t)$ denotes the the membrane potential and τ is the time constant for $U_i(t)$. $C(t)$ is the input synaptic current (the weighted summation of pre-spike) at time t can be defined by

$$C(t) = \sum_{i=1}^N w_{i,j} \sum_n V_i(t - t_n), \quad (2)$$

where $V_i(t - t_n)$ is the spike event from the current neuron i to its pre-neuron j , $w_{i,j}$ is the synaptic weight between them and N denotes the number of neurons. In this context, the dynamic behavior of spike neurons in the first-order is given by

$$\begin{cases} \tau \frac{dU_i(t)}{dt} = \beta U_i(t) + \sum_{i=1}^N w_{i,j} \sum_n V_i(t - t_n), \\ S_i(t) = 1 \ \& \ U_i(t) = U_{\text{re}}, \text{ if } U_i(t) \geq U_{\text{th}}, \\ S_i(t) = 0, \text{ if } U_i(t) < U_{\text{th}}, \end{cases} \quad (3)$$

where $S_i(t)$ is the output of the i -th neuron at time t , U_{re} and U_{th} are the reset potential and firing threshold (resting potential), respectively. In equation (3), the spike event $S_i(t)$ can be obtained based on the value of membrane potential $U_i(t)$, e.g., $S_i(t) = 1$ when $U_i(t) \geq U_{\text{th}}$, otherwise, $S_i(t) = 0$.

The dynamic behavior of the i -th second-order Izhikevich neuron with MDNs can be expressed by³¹

$$\begin{cases}
\tau \frac{dU_i(t)}{dt} = U_i^2(t) - U_i(t) - H_i(t) + \sum_{i=1}^N w_{i,j} \sum_n V_i(t - t_n), \\
\frac{dO_i(t)}{dt} = \eta_a (\eta_b U_i(t) - H_i(t)), \\
S_i(t) = 1, \quad U_i(t) = \eta_c \ \& \ H_i(t) = H_i(t) + \eta_d \text{ if } U_i(t) \geq U_{th}, \\
S_i(t) = 0, \text{ if } U_i(t) < U_{th},
\end{cases} \quad (4)$$

where $H_i(t)$ denotes a resistance value simulating hyperpolarization which is tapped to charge the activation and inactivation of currents, η_a , η_b , η_c , and η_d are the dynamic parameters which are used to distinguish the different second-order dynamics of membrane potential in SNN. From equation (4), we can observe that the attractor of $U_i(t)$ is determined by $H_i(t)$ and $C(t) = \sum_{i=1}^N w_{i,j} \sum_n V_i(t - t_n)$. Note that a sigmoid function is utilized to limit the range of input currents $C(t)$ ³¹.

In SNN, the loss function is used to evaluate the mean square error between output firerates and labels Y_i , where i denotes the index of neurons, which is given by

$$Loss = \sum_{i=1}^N \left(\frac{1}{T} \sum_{t=1}^T S_i(t) - Y_i \right)^2. \quad (5)$$

Considering the fact that the widely-used gradient back propagation (BP) is not biologically-plausible and SNNs has non-differential characteristic, and thus it is hard to directly adopt BP to train the SNNs model in LFNL. Thus, an approximate BP trick technique^{31,35} is used to train SNNs by setting a pseudo differential gradient with lower and upper bounds for infinite differential gradients by a spiking time window. The pseudo differential gradient is expressed by

$$G = \begin{cases} 1, & \text{if } |U_i(t) - U_{th}| < U_{tar}, \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

where G is the gradient which is used to dynamically update the synaptic values in SNNs, and U_{tar} is the range of membrane potential between input information. In SNNs, the learning structure is trained on a sequence of binary spike events over a given number of time steps T , more detail can be found in the study³¹.

LFNL model. Federated learning has become an important paradigm aiming to train a collaborative AI model while keeping all the training data localized^{12,13} and privacy-preserved. Thus, federated learning is recently applied to hold great promises on edge data analytics¹⁴⁻¹⁶, which enables edge devices to train their AI models locally without sharing sensitive private data to each other.

To perform the privacy-preserving model aggregation, in the system, a set of edge devices $K = \{1, \dots, K\}$ participate in a global neuromorphic model training (i.e., training federation for object recognition) with a

leader to performance model aggregation. Each device k adopts its local database D_k to train its local neuromorphic model parameters $\mathbf{w}_k$ without sharing local data with the leader. In the federated learning scenario, the loss function $f(\mathbf{w}_k; \mathbf{x}_{k,j}, \mathbf{y}_{k,j})$ is introduced to quantify the federated performance error over the input data sample vector $\mathbf{x}_{k,j}$ on the training model $\mathbf{w}_k$ and the desired output scale vector $\mathbf{y}_{k,j}$ for each input sample j at the k -th device. Accordingly, the local loss function on the training set D_k at the k -th device can be expressed by

$$F(\mathbf{w}_k) = \frac{1}{|D_k|} \sum_{j \in D_k} f(\mathbf{w}_k; \mathbf{x}_{k,j}, \mathbf{y}_{k,j}), \quad (7)$$

where $|D_k|$ denotes the cardinality of the set D_k . At the leader side, the global loss function with the local datasets of participating device can be expressed by

$$F(\mathbf{w}) \triangleq \sum_{k \in K} \frac{|D_k|}{|D|} F(\mathbf{w}_k) = \frac{1}{|D|} \sum_{k \in K} \sum_{j \in D_k} f(\mathbf{w}; \mathbf{x}_{k,j}, \mathbf{y}_{k,j}), \quad (8)$$

where $\mathbf{w}$ denotes the global model parameters at the leader and D is the sum data samples from all participating devices. The objective of the federated task is to find an optimal model parameter $\mathbf{w}^*$ by minimizing the global loss function expressed by

$$\mathbf{w}^* = \arg \min F(\mathbf{w}). \quad (9)$$

The leader iteratively updates the aggregated model through the local training procedure across edge devices in a group until the model converges to a certain learning accuracy target. The leader election protocol and process can be found in Supplementary Fig. 1b,c.

Energy consumption analysis. Similar to the work⁴⁹, the energy consumption can be calculated based on the number of floating point operations (FLOPS) of ANNs or SNNs networks which are approximately equivalent to the number of multiply-and-accumulate (MAC) operations. In the case of ANNs, FLOPS mainly includes the MAC operations of convolutional and linear layers. On the contrary, for SNNs, as it performs training over binary spike signals, only accumulate (AC) operations are needed to handle the dot operations, except the first input layer. For each convolutional layer of ANN or SNN, with I input channels, O output channels, $M \times M$ input feature map size, and $Q \times Q$ output size, and its FLOPS for ANNs and SNNs are respectively calculated by⁴⁹

$$F^{ANN} = Q^2 \times I \times p^2 \times O, \quad (12)$$

$$F^{SNN} = Q^2 \times I \times p^2 \times O \times R, \quad (13)$$

where R is the net spiking rate across training latency steps in SNNs, and we know $R < 1$ due to the sparse event-driven activity. We note that equation (13) is calculated over one time-step in SNN.

After providing the operation calculation for ANN and SNN in equations (12) and (13), we specify the energy consumption of each MAC or AC operation on 45nm CMOS processor with 32-bit integer arithmetic⁵⁰. Each MAC operation consumes 3.2 pJ while each AC operation needs only 0.1 pJ in the 45nm CMOS processor. Hence, the total energy consumption for an ANN (E_{ANN}) and an SNN (E_{SNN}) can be calculated by⁴⁹

$$E_{ANN} = \left(\sum_{l=1}^L F_l^{ANN} \right) \times E_{MAC}, \quad (14)$$

$$E_{SNN} = F_1^{SNN} \times E_{MAC} + \left(\sum_{l=2}^L F_l^{SNN} \right) \times E_{AC} \times T, \quad (15)$$

where L is the number of layers of an ANN or SNN. E_{MAC} and E_{AC} are the energy consumption of one MAC or AC operation, respectively. In the case of SNNs, as shown in equation (15), the total energy consumption requires considering the total number of AC operations over T time training steps. In addition, as the first layer (input layer) in SNNs needs to process the analog input into binary spike events, and thus MAC operations are used in this layer. Note that the energy calculation in equations (14) and (15) are approximate estimations which do not take into account the memory and any hardware circuit energy consumption.

Experiment hardware and software. We implement the experiments on two Raspberry PI 4B (CPU clock: 1.5 GHz, RAM: LPDDR4 8 GB), one Raspberry PI 3B+ (CPU clock: 1.4 GHz, RAM: LPDDR4 1 GB) and one laptop (CPU: 1.60GHz, RAM: 8 GB). Note that when we implement the experiment with more devices setting, we run the results on the laptop. For software, the experiments of the study are performed by using PyTorch via Python 3.0. The network and training parameters in our experiments are shown in Supplementary Table 1.

Data availability

The sound or speech databases can be accessed at <https://www.kaggle.com/vishnu0399/emergency-vehicle-siren-sounds>⁴³, the TIDIGITS database⁴⁷ (<https://catalog.ldc.upenn.edu/LDC93S10>) and the work⁴⁸. The image databases are collected from <https://github.com/nikhilpatil99/Smart-Traffic-Management-Using-Deep-Learning>⁴⁴, <https://www.kaggle.com/hj23hw/pedestrian-augmented-traffic-light-dataset>⁴⁵, and the MNIST database⁴⁶ (<http://yann.lecun.com/exdb/mnist/>). The radar gesture database is collected from the study³⁰.

References

1. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K. & Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **107**, 1738–1762 (2019).
2. Li, B. et al. Random sketch learning for deep neural networks in edge computing. *Nat. Comput. Sci.* **1**, 221–228 (2021).
3. Chakraborty, I. Constructing energy-efficient mixed-precision neural networks through principal component analysis for edge intelligence. *Nat. Mach. Intell.* **2**, 43–55 (2020).
4. Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X. & Chen, M. In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network* **33**, 156–165 (2019).
5. Wu, Y., et al. Brain-inspired global-local learning incorporated with neuromorphic computing. *Nat. Commun.* **13**, 65 (2022).
6. Vaughan, O. Working on the edge. *Nat. Electron.* **2**, 2–3 (2019).
7. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S. & Zomaya, A. Y. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet Things J.* **7**, 7457–7469 (2020).
8. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
9. Kaissis, G. A., Makowski, M. R., Rückert, D. & Braren, R. F. Secure, privacy-preserving and federated machine learning in medical imaging. *Nat. Mach. Intell.* **2**, 305–311 (2020).
10. Han, T., et al. Breaking medical data sharing boundaries by using synthesized radiographs. *Sci. Adv.* **6**, eabb7973 (2020).
11. Loukides, M. TinyML: The challenges and opportunities of low-power ML applications. O’ Reilly <https://www.oreilly.com/radar/tinyml-the-challenges-and-opportunities-of-low-power-ml-applications/> (2019).
12. Reddi, V. J. Enabling ultra-low power machine learning at the edge. In TinyML Summit 2020 (TinyML, 2020), <https://cms.tinyml.org/wp-content/uploads/summit2020/tinyMLSummit2020-4-4-JanapaReddi.pdf>.
13. Konečný, J. et al. Federated learning: strategies for improving communication efficiency. Preprint at <https://arxiv.org/abs/1610.05492.pdf> (2016).
14. McMahan, B. & Ramage, D. Federated learning: collaborative machine learning without centralized training Data. Google AI Blog (2017).
15. Wang, S. et al. Adaptive federated learning in resource constrained edge computing systems. *IEEE J. Sel. Areas Commun.* **37**, 1205–1221 (2019).
16. Lim, W. Y. B. et al. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Commun. Surveys Tuts.* **22**, 2031–2063 (2020).
17. Chen, M., Shlezinger, N., Poor, H. V., Eldar, Y. C. & Cui, S. Communication efficient federated learning. *Proc. Natl. Acad. Sci.* **118**, e2017318118 (2021).
18. Kaissis, G. et al. End-to-end privacy preserving deep learning on multi-institutional medical imaging. *Nat. Mach. Intell.* **3**, 473–484 (2021).

19. Froelicher, D. et al. Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption. *Nat. Commun.* **12**, 5910 (2021).
20. Warnat-Herresthal, S. et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature* **594**, 265–270 (2021).
21. Lim, W. Y. B. et al. Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning. *IEEE Trans. Parallel Distrib. Syst.* **33**, 536–550 (2022).
22. Li, Y., Chen, C., Liu, N., Huang, H., Zheng Z., & Yan, Q. A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network* **35**, 234–241 (2021).
23. Sun, Y. et al. Semi-decentralized federated edge learning for fast convergence on Non-IID data. Preprint at <https://arxiv.org/pdf/2104.12678.pdf> (2021).
24. Che, C., Li, X., Chen, C., He, X. & Zheng, Z. A decentralized federated learning framework via committee mechanism with convergence guarantee. Preprint at <https://arxiv.org/pdf/2108.00365.pdf> (2021).
25. Yang, H., Zhao, J., Xiong, Z., Lam, K. -Y., Sun, S., & Xiao, L. Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management. *IEEE J. Sel. Areas Commun.* **39**, 3144–3159 (2021).
26. French, R. M. Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.* **3**, 128–135. [https://doi.org/10.1016/S1364-6613\(99\)01294-2](https://doi.org/10.1016/S1364-6613(99)01294-2) (1999).
27. Chang, K. et al. Distributed deep learning networks among institutions for medical imaging. *J. Am. Med. Inform. Assoc.* **25**, 945–954. <https://doi.org/10.1093/jamia/ocy017> (2018).
28. Sheller, M.J. et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Sci. Rep.* **10**, 12598 (2020).
29. García-Martín, E., Rodrigues, C. F., Riley, G. & Grahm, H. Estimation of energy consumption in machine learning. *J. Parallel Distrib. Comput.* **134**, 75–88 (2019).
30. Li, D., Chen, X., Becchi, M. & Zong, Z. Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs. In *Proc. 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloudSocialCom-SustainCom)*, 477–484 (IEEE 2016).
31. Stuijt, J., Sifalakis, M., Yousefzadeh, A. & Corradi, F. (2021). μ Brain: An event-driven and fully synthesizable architecture for spiking neural networks. *Front. Neurosci.* **15**, 538 (2021)
32. Cheng, X., Zhang, T., Jia, S. & Xu, B. Finite meta-dynamic neurons in spiking neural networks for spatio-temporal learning. Preprint at <https://arxiv.org/pdf/2010.03140.pdf> (2021)
33. Maass, W. Networks of spiking neurons: the third generation of neural network models. *Neural Netw.* **10**, 1659–1671 (1997).
34. Schranghamer, T.F., Oberoi, A. & Das, S. Graphene memristive synapses for high precision neuromorphic computing. *Nat. Commun.* **11**, 5474 (2020).
35. Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. Neuronal dynamics: From single neurons to networks and models of cognition. *Cambridge University Press*. (2014)
36. Subbulakshmi Radhakrishnan, S., Sebastian, A., Oberoi, A., Sarbashis, D., & Saptarshi D., A biomimetic neural encoder for spiking neural network. *Nat. Commun.* **12**, 2143 (2021).

37. Indiveri, G. & Horiuchi, T. Frontiers in neuromorphic engineering. *Front. Neurosci.* **5**, 118 (2011).
38. Stöckl, C. & Maass, W. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nat. Mach. Intell.* **3**, 230–238 (2021).
39. Zhang, Y. et al. A system hierarchy for brain-inspired computing. *Nature* **586**, 378–384 (2020).
40. Kundu, S., Datta, G., Pedram, M. & Beerel, P. A. Towards low-latency energy-efficient deep SNNs via attention-guided compression. Preprint at <https://arxiv.org/pdf/2107.12445.pdf> (2021).
41. Göltz, J. et al. Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nat. Mach. Intell.* **3**, 823–835 (2021).
42. Skatchkovsky, N., Jang, H. & Simeone, O. Federated neuromorphic learning of spiking neural networks for low-power edge intelligence. In *Proc. 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 8524–8528 (IEEE, 2020)
43. Venkatesha, Y., Kim, Y., Tassiulas, L. & Panda, P. Federated learning with spiking neural networks. Preprint at <https://arxiv.org/abs/2106.06579.pdf> (2021).
44. <https://www.kaggle.com/vishnu0399/emergency-vehicle-siren-sounds>.
45. Chintamani, N., Yash, M., Nikhil, P., Vivek, P. & Sandeep, P. Smart traffic control using deep learning. Preprint at <https://github.com/nikhilpatil99/Smart-Traffic-Management-Using-Deep-Learning> (2019).
46. <https://www.kaggle.com/hj23hw/pedestrian-augmented-traffic-light-dataset>.
47. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
48. Leonard, R. G. & Doddington, G. TIDIGITS LDC93S10. *Web Download. Philadelphia: Linguistic Data Consortium* (1993).
49. Pan, Z., Chua, Y., Wu, J., Zhang, M. & Li, H. & Ambikairajah E. An efficient and perceptually motivated auditory neural encoding and decoding algorithm for spiking neural networks. *Front. Neurosci.* **13**, 1420 (2020).
50. Horowitz, M. Computing’s energy problem (and what we can do about it). In *Proc. 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 10–14 (IEEE, 2014).
51. Kundu, S., Datta, G., Pedram, M. & Beerel, P. A. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proc. 2021 the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 3953–3962 (IEEE, 2021).

Code availability

The program codes presented in the research were developed using the Python and Matlab platform. To assist researchers in reproducing the experimental results, we will share the codes by Kaggle or GitHub recently.

Acknowledgements

This work is support by Nanyang Technological University (NTU) Startup Grant, Singapore Ministry of Education Academic Research Fund, Singapore National Research Foundation under its Strategic Capability Research Centres Funding Initiative: Strategic Centre for Research in Privacy-Preserving Technologies & Systems.

Author contributions

H.L.Y., K.-Y. L., and L. X. conceived the ideal. H.L.Y. wrote the paper, collected data, and designed the experiment. Z.H.X., H.H., N.D., and H.V.P. assisted to analyze the experiment results. All authors contributed to revising the manuscript.

Competing interests

The authors declare no competing interests.

Supplementary information (SI)

SI.1: Leader election for federated learning

The leader election plays an important role of federated model aggregation performance for edge AI. The leader with high computation and communication capabilities can speed up the federated aggregation process and reduce the overall training latency, whereas the federated aggregation latency will be negatively impacted if the network elects a leader with low computation and communication capabilities. For example, as illustrated in Supplementary Fig. 1a, there are three leader election scenarios in terms of communication capability evaluation. We consider the communication capability as an example for leader election, and assume that other metrics (e.g., computation and energy supply capabilities) are equal for all devices. Note that with the increase of communication distance, the communication data rate decreases¹⁻³, thus leading to the increase of data packet transmission latency. For Scenario 1 or Scenario 2, as the system does not take the communication distance (wireless communication link quality) into account but randomly elects one of devices as the leader to perform model aggregation. In this case, as the elected leader (device 5 or device 6) locates at the edge, the followers (e.g., device 2) located at the opposite corner have a greater communication distance, resulting in a limited communication data rate and increasing the data packet transmission latency. The leader needs to wait the last follower to upload the local model parameters before performing model aggregation, which directly increases the overall processing delay. On the contrary, this issue can be effectively addressed by considering the communication capability into the leader election, as shown in Scenario 3 (Supplementary Fig. 1a). The reason is that the elected leader locates at the center of the edge devices, and the communication distance (wireless communication link quality) from followers to the leader are relatively balanced, and thus there does not exist the extreme situation where the communication distance between any edge follower and the leader is extremely far. In this context, taking the device's metrics into account for the leader election can greatly improve the federated model aggregation in terms of lower latency.

The flow chart for leader election is shown in Supplementary Fig. 1b. Firstly, each participant edge device calculates its weighted score based on the three metrics, i.e., computation, communication and energy supply capabilities, where the weighted score can be expressed by

$$\text{Weighted Score} = \text{Communication capability} + \text{Computation capability} + \text{Energy supply capability}. \quad (1)$$

Then, each candidate advertises its score for leader election. The device with the highest score is elected as the leader to perform federated model aggregation.

Supplementary Fig. 1: Leader election and federated model aggregation process. **a**, An example of the leader election in three scenarios in terms of communication capability. **b**, Flow diagram of proposed leader election protocol. **c**, Algorithm of the LFNL enabled federated model aggregation process.

Once the leader is elected, the federated learning can be implemented which is shown in Supplementary Fig. 1c. Firstly, the leader broadcasts the initialized model $\mathbf{w}$ to all followers. Each device trains its own local model independently and in parallel based on its dataset, and then uploads its local model parameters to $\mathbf{w}_k$ to the leader instead of uploading the raw data. The leader collects local model parameters ($\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \dots, \mathbf{w}_K$) to perform model aggregation before broadcasting the updated global parameters $\mathbf{w}$ to followers for next round of local training. The exchange of local and global parameters repeats until the convergence. Here, the federated averaging method is used for model aggregation¹.

As mentioned earlier, leader election aims to accelerating the federated training process. Thus, both the computation and communication times are also essential metrics to be optimized. Before evaluating the performance of the leader election scheme, we introduce the definitions of the computation and communication times in federated learning systems as follows^{2,3}.

The computation time T_k of the k -th device mainly depends on its training dataset size $|D_k|$ and the computation capability. In each global epoch, the local computation time of device k is calculated by

$$T_k^{\text{local}} = E_k C_k |D_k| / f_k, \quad (2)$$

where C_k (cycles/bit) denotes the number of Central Processing Unit (CPU) cycles needed for computing a sample data, E_k is the number of local training epochs and f_k is the CPU frequency of device k . In LFNL, as a device is elected to perform model aggregation and model quality evaluation. In this case, the computation time on testing dataset per global epoch is calculated by

$$T_{\text{test}} = C_{\text{leader}} |D_{\text{test}}| / f_{\text{leader}}, \quad (3)$$

where C_{test} is the number of CPU cycles needed for computing a sample data, $|D_{\text{test}}|$ is the testing dataset size of its set D_{test} , and f_{leader} is the CPU frequency of the elected leader.

In each global epoch, each device needs to upload its local training model parameters to the leader for model aggregation, and the leader also broadcasts the updated global model parameters to all participating followers for the next training process. During this process, the communication time of one model upload or broadcast between the leader and the k -th follower can be expressed by

$$T_k^{\text{com}} = |D_{\text{model}}| / R_k, \quad (4)$$

where $|D_{\text{model}}|$ is the model parameters size of its set D_{model} , and R_k is the the achievable data rate between the leader and follower k , which is given by

$$R_k = B \log_2(1 + \frac{Ph_k}{\delta^2}), \quad (5)$$

where B is the wireless transmission bandwidth, P parameters is the transmission power, h_k is the channel gain between the leader and device k , and δ^2 is the background Gaussian noise power. We assume that all devices have the same bandwidth and transmission power.

Overall, the total training time depends on communication together with computation time over a number of global epochs I . Here, we use the synchronized updates property in the federated systems⁴, where the leader begins to aggregate the global model until all followers upload their local models. Thus, the overall training time defined as

$$T = I \left(\max_k T_k^{\text{local}} + \max_k (2 \times T_k^{\text{com}}) + T_{\text{test}} \right). \quad (6)$$

Note that in equation (6), as model parameters are required to be uploaded and downloaded, and thus the communication time is $2 \times T_k^{\text{com}}$ per global epoch.

We consider a single-cell network with a radius of 60 meter. Six devices are randomly located over the network. The path loss between one device and another device is $h = d^{-4}$ ^{5,6} with d being the communication distance in metre. We set that each device has $B=0.5$ MHz bandwidth for uploading and downloading model parameters. The transmission power P and background Gaussian noise power δ^2 are set as 50 mW and -100 dBm, respectively. The computation frequency of devices is uniformly set from the set [0.5, 1.0, 1.5] GHz. Parameter C is uniformly distributed in [50, 100] cycles/bit for all devices. Each device has $|D| = 1553$ Kbits training samples, $|D_{\text{model}}| = 2296$ Kbits model parameters, and the testing dataset size is $|D_{\text{test}}| = 1170$ Kbits. In the benchmark SNNs, the number of nodes of the input layer is 1728, the number of neuron nodes of one hidden layer is 300, and the number of nodes of the output layer is 3. The traffic image dataset^{7,8} is used for performance elevation, in total, 872 images are used with three classes, including 160 bicycle images, 205 car images, and 507 traffic light images. 80% of the images are used for training, and the remaining images (20%) are adopted for validation and testing. The training dataset is divided into six small parts for six devices, and we run the experiments at a laptop.

Here we provide the performance comparisons between the proposed leader election scheme and the random leader election scheme. Supplementary Fig.2 shows the obtained convergence speed and test accuracy throughout the federated training time for the two schemes. Overall, the proposed LFNL method considers computation and communications aspects into the leader election that results in the convergence speed enhancements supporting real-time edge computing deployments. In particular, we find that LFNL obtains the faster convergence speed while maintaining the higher test accuracy via training time slots. For example, as illustrated in Supplementary Fig.2a, when the validation loss is 0.025, the training completion times of the leader election scheme and the random leader election scheme are 20.6s and 34.3s, respectively, significantly reducing the overall training time by nearly 66.5%. The reason lies in the fact that electing the leader with high computation and communication capabilities to perform federated model aggregation, which substantially reduces computation and communication times. Hence, LFNL with leader election is valuable in accelerating the federated learning process on edge devices.

Supplementary Fig. 2: Performance evaluation of LFNL-based leader election on traffic image dataset^{7,8}. **a**, Validation loss curves via training time for the leader election and random leader election schemes. **b**, Test accuracy via training time for the leader election and random leader election schemes.

SI.2: Learning with more edge devices on traffic sound dataset

Supplementary Fig. 3: LFNL for audio recognition with scenario that dividing the training dataset for six edge devices with each being with small dataset size. **a**, 35 independent test accuracy evaluations for six locally training devices and LFNL after training. The dataset distributions for the six devices are equally set as 13.3%. **b**, Box plots show test accuracy for 35 independent evaluations of Supplementary Fig. 3a.

As described in the main manuscript, AI performs well when the training data is sufficient^{9,10}, for example, the test accuracy values of device 1 and device 2 (Fig. 2h in the main manuscript) are considerably good. To demonstrate the scalability and robustness of LFNL, we divide the training samples into six parts for six devices with each being with smaller training dataset, the dataset distribution at the six devices is equally set as 13.3%. As depicted in Supplementary Fig. 3a,b, not only the test accuracy values of six locally training devices are all significant lower than that of LFNL, but also their test accuracy values fluctuate more frequently due to insufficient training samples. This is because the insufficient training sound samples enable local learning to overfit quickly and lead to an unstable training model. In this scenario, LFNL still achieves the comparable test accuracy of 92.4% to the centralized learning method (shown in Fig. 2l with yellow color box in the main manuscript).

SL3: Learning under poisoning model attack on traffic sound dataset

In the federated learning system that operates on large scales, it may give rise to performance degradation caused by malicious or low-quality participants¹¹, where poisoning or low-quality model parameters hamper the global accuracy of jointly trained model by uploading malicious or low-quality inputs during model aggregation process. Fortunately, as reported¹²⁻¹⁴, various poisoning model detection algorithms have been proposed to defense poisoning attack, such as generative adversarial networks-based FL¹² and participants selection³. To demonstrate the robustness of LFNL, we also examine it in the presence of poisoning or low-quality model.

We set up an experimental scenario (Supplementary Fig. 4a) that the traffic sound dataset¹⁵ is divided into six small parts for six edge devices, where one of six devices acts as a malicious participant with poisoning or low-quality model and others are normal. Here, we compared the following learning methods: 1. The normal federated learning system with no poisoning model attack and all edge devices are normal, denoted by *no poisoning model attack*. 2. One malicious edge device joins the federated learning system and sends its poisoning model parameters to degrade the global training accuracy. All devices are decentralized edge devices and no leader detects the poisoning model and selects the normal participant devices for model aggregation, denoted by *decentralized FNL(DFNL) without selection*. 3. The leader detects the poisoning model and selects the normal devices for model aggregation, denoted by *LFNL with selection*.

In the experiments presented in Supplementary Fig. 4a, due to the poisoning model attack during federated model aggregation process, the validation loss curve of the method (DFNL without selection) has an unstable training. Interestingly, its validation loss values increase as the increasing number of

Supplementary Fig. 4: Performance elevation under positioning model attack on traffic sound dataset. **a**, Scenario that one of six edge devices is a malicious participant with poisoning or low-quality model to attack the federated learning process, while other devices are normal. The dataset distributions for the six devices are equally set as 13.3%. **b**, Validation loss curves for three learning methods. **c**, 35 independent test accuracy evaluations for three learning methods after training. **d**, Box plots show test accuracy for 35 independent test evaluations of Supplementary Fig. 4c.

training epoch, leading to a low test accuracy of 80.3% (Supplementary Fig. 4b,c). On the contrary, as indicated by a lower loss and higher test accuracy in Supplementary Fig. 4 (red color), LFNL with selection defense the poisoning model attack and significantly outperforms DFNL without selection. The reason is that the proposed method (LFNL with selection) enables the system to select a leader to detect the malicious device, and schedules normal devices to participate the federated model aggregation. Thus, its loss curve is smoother and its classification accuracy is higher than those of DFNL without selection. In these results, we also find that the performance of LFNL with selection is approximately equivalent to the system with no poisoning model attack. This gives us confidence that it can effectively defense the positioning model attack during federated model aggregation for edge AI.

SI.4: Learning against image rotation on traffic image dataset

In the real-world scenario, the collected images maybe not in the right orientation and they are sometimes rotate with random angles. In this case, we demonstrate the classification robustness of LFNL on traffic image dataset^{7,8} in the presence of image rotation (Supplementary Fig. 5a). In the benchmark LFNL, an SNN is with 1728-2500-3 neurons (input-hidden-label layer size). For the traffic image dataset^{7,8}, in total, 872 images were used with three classes, including 160 bicycle images, 205 car images, and 507 traffic light images. 80% of the images are used for training, and the remaining images (20%) are adopted for validation and testing.

a

b

c

d

● — Normal state ▲ — Rotation by 90°, 180° and 270° ■ — Random rotation

Supplementary Fig. 5: Classification evaluation of LFNL on traffic image dataset under image rotation. **a**, Three scenarios that traffic images are rotated, i.e., normal state with no image rotation, image rotation by 90°, 180°, and 270°, as well as image rotation with random angles. **b**, Validation loss curves for the three scenarios of Supplementary Fig. 5a. **c**, 35 independent test accuracy evaluations for three learning methods after training. **d**, Box plots show test accuracy for 35 independent evaluations of Supplementary Fig. 5c.

As illustrated in Supplementary Fig. 5b, LFNL with image rotation (90°, 180°, and 270°) achieves a similar test accuracy to the normal state with no rotation under different number of global training epochs, and the mean classification accuracy is reached 96.20% for these two scenarios (Supplementary Fig. 5c,d). The overall test accuracy of LFNL with random image rotation is just slightly lower than that of the normal state with no rotation via each global training epoch (Supplementary Fig. 5b), still achieving a test accuracy of 93.40% (Supplementary Fig. 5c,d). These results support the conclusion that the proposed LFNL method can robustly recognize/classify images even though the images are placed in different rotation angles.

SI.5: Learning with more edge devices on MNIST and TIDIGITS datasets

To showcase LFNL’s scalability and flexibility in classifying image and speech signal on larger and more high-dimensional datasets, we provide an additional case investigation where MNIST¹⁶ and TIDIGITS¹⁷ datasets are divided into more parts for small edge devices. Similarity, the dataset distributions for the six devices are equally set as 13.3% with each being with a small dataset.

As plotted in Supplementary Fig. 6a,b, we find that the overall test accuracy of the six locally training devices can maintain a high level (around 95%), which is slight lower than that of three locally training devices (Fig. 6a,b in the main manuscript). The reason lies in the fact that even though the 60000 training image of MNIST dataset is equally divided into six small parts for six devices, each locally training device still has sufficient samples (each has 10000 samples) to train a reliable classifier.

However, when we test the classification performance on TIDIGITS dataset by dividing it into six small parts for six edge devices, the overall test accuracy values of the six locally training devices substantially decline (Supplementary Fig. 6c,d). Moreover, the test accuracy values of the six locally training devices fluctuate more frequently than that of LFNL. The reason is that the number of training samples of each device is 660, which is significant less than that of MNIST dataset. Therefore, the local learning with insufficient training samples overfit quickly and led to an unstable training model. On the contrary, LFNL overcomes this local overfitting issue and significantly outperforms the locally training devices (Supplementary Fig. 6c-e). It is worth noting that even if the training dataset is divided into more smaller parts for edges devices, the LFNL results still does not deteriorate (Supplementary Fig. 6c-e).

Supplementary Fig. 6: Scenario that dividing the MNIST and TIDIGITS datasets for six edge devices with each being with small dataset size. a, Evaluation of test accuracy for six edge devices and LFNL over 35 independent runs on MNIST dataset. The dataset distributions for the six devices are equally set as 13.3%. **b,** Box plots show test accuracy for 35 independent test evaluations of Supplementary Fig. 6a. **c,** Evaluation of test accuracy for six edge devices and LFNL over 35 independent runs on TIDIGITS dataset. The dataset distributions for the six devices are equally set as 13.3%. **d,** Box plots show test accuracy for 35 independent evaluations of Supplementary Fig. 6c. **e,** Box plots show test accuracy for twenty small edge devices and LFNL over 35 independent runs on TIDIGITS dataset. The dataset distributions for the twenty devices are equally set as 4%.

Supplementary Table 1: Network and training parameters for training on LFNL used to produce the experimental results in this work.

Parameters	Traffic sound	Traffic image	Radar gesture	MNIST image	TIDIGITS sound
Input size	128	1728	4800	784	1640
Hidden size	2000	2500	2000	500	1000
Output size	3	3	5	10	11
Batch size	16	16	32	100	64
Optimizer	Adam	Adam	Adam	Adam	Adam
Adam parameter betas	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Adam parameter eps	10^{-6}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
lr_scheduler	StepLR	StepLR	StepLR	StepLR	StepLR
lr_scheduler step size	50	15	15	15	15
lr_scheduler gamma	0.1	0.9	0.9	0.9	0.9
Learning rate	0.01	0.005	0.02	0.005	0.005

References

1. Sjöberg A., Gustavsson E., Koppisetty A.C., & Jirstrand M. Federated learning of deep neural decision forests. machine learning, optimization, and data science. *Lecture Notes in Computer Science*, 11943 (2019).
2. Yang, Z., Chen, M., Saad, W., Hong C. & Shikh-Bahaei, M. Energy efficient federated learning over wireless communication networks. *IEEE Trans. Wireless Commun.* **20**, 1935–1949 (2021).
3. Yang, H., Zhao, J., Xiong, Z., Lam, K. -Y., Sun, S., & Xiao, L. Privacy-preserving federated learning for UAV-enabled networks: Learning-based joint scheduling and resource management. *IEEE J. Sel. Areas Commun.* **39**, 3144–3159 (2021).
4. Wen, D., Bennis, M., & Huang, K., Joint parameter-and-bandwidth allocation for improving the efficiency of partitioned edge learning. *IEEE Trans. Wireless Commun.* **68**, 2128–2142 (2020).
5. 3GPP TS 36.213, Evolved universal terrestrial radio access (EUTRA) physical layer procedures (Release 12), Sep. 2014.
6. Wu, F., Zhang, H., Di, B., Wu J. & Song, L. Device-to-device communications underlaying cellular networks: To use unlicensed spectrum or not? *IEEE Trans. Wireless Commun.* **67**, 6598–6611 (2019).
7. Chintamani, N., Yash, M., Nikhil, P., Vivek, P. & Sandeep, P. Smart traffic control using deep learning. Preprint at <https://github.com/nikhilpatil99/Smart-Traffic-Management-Using-Deep-Learning> (2019).

- 853 8. <https://www.kaggle.com/hj23hw/pedestrian-augmented-traffic-light-dataset>.
854 9. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
855 10. Kaissis, G. A., Makowski, M. R., Rückert, D. & Braren, R. F. Secure, privacy-preserving and federated machine
856 learning in medical imaging. *Nat. Mach. Intell.* **2**, 305–311 (2020).
857 11. Finlayson, S. G. et al. Adversarial attacks on medical machine learning. *Science* **363**, 1287–1289 (2019).
858 12. Han, T., et al. Breaking medical data sharing boundaries by using synthesized radiographs. *Sci. Adv.* **6** eabb7973
859 (2020).
860 13. Ma, Z., Ma, J., Miao, Y., Liu, X., Choo, K. -K. & Deng, R. Pocket diagnosis: Secure federated learning against
861 poisoning attack in the cloud. Preprint at <https://arxiv.org/pdf/2009.10918.pdf> (2009).
862 14. Cao, D., Chang, S., Lin, Z., Liu G. and Sun, D. Understanding distributed poisoning attack in federated learning.
863 In *Proc. 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, 233–239
864 (IEEE 2016).
865 15. <https://www.kaggle.com/vishnu0399/emergency-vehicle-siren-sounds>.
866 16. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc.*
867 *IEEE* **86**, 2278–2324 (1998).
868 17. Leonard, R. G. & Doddington, G. TIDIGITS LDC93S10. Web Download. Philadelphia: Linguistic Data
869 Consortium (1993).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [SupplementaryInformation.pdf](#)