Decentralized Automatic Modulation Classification Method Based on Lightweight Neural Network

Biao Dong[†], Guozhen Xu[‡], Xue Fu[†], Heng Dong[†], and Guan Gui[†], Haris Gacanin[††] and Fumiyuki Adachi[*]

[†]College of Telecommunications and Information Engineering, NJUPT, Nanjing, China
[‡] College of Electronic countermeasure, National University of Defense Technology, Hefei, China
[††]Faculty of Electrical Engineering and Information Technology, RWTH Aachen University, Aachen, Germany
[*]Research Organization of Electrical Communication, Tohoku University, Sendai, Japan

***Abstract*—Due to the computing capability and memory limitations, it is difficult to apply the traditional deep learning (DL) models to the edge devices (EDs) for realizing automatic modulation classification (AMC). In this paper, a lightweight neural network for decentralized learning-based automatic modulation classification (DecentAMC) method is proposed. Specifically, group convolutional neural network (GCNN) is designed by replacing the standard convolution layer with the group convolution layer, replacing the flatten layer with the global average pooling (GAP) layer and removing part of fully connected layers. DecentAMC method is achieved by the cooperation in which multiple EDs update and upload the model weight to a central device (CD) for model aggregation to avoid the data privacy disclosure. Experimental results show that the proposed GCNN-based DecentAMC method can improve training efficiency to about 4 times and 57 times than that of GCNN-based centralized AMC (CentAMC) and CNN-based DecentAMC respectively. GCNN-based DecentAMC method can effectively reduce the communication cost and save storage of EDs when compared with CNN-based DecentAMC. Meanwhile, the time complexity and the space complexity of GCNN is significantly decreased when compared with CNN and SCNN, which is suitable to be deployed in EDs.**

***Index Terms*—Automatic modulation classification, decentralized learning, lightweight neural network.**

I. Introduction

Automatic modulation classification (AMC) is a promising technique that can be applied at the receiver to distinguish the different types of modulated signals [1]. Recently, AMC has played a significant role in both military and civilian communications, such as cognitive radio and link adaptation. In the past, typical methods of AMC included likelihood-based AMC and feature-based AMC [2]. Traditional machine learning (ML), such as decision tree and k-nearest neighbor (KNN), was also widely used in feature-based AMC as a classifier. However, these traditional methods need to extract the features manually.

In recent years, deep learning (DL) has made great achievements in many fields, such as wireless communications [3]–[6] and cyber security [7]–[9]. Hence, more and more scholars tried to apply DL to the feature-based AMC for achieving better performance. F. Meng *et al*. proposed a convolution neural network based AMC (CNN-AMC) [10], which can outperform the feature-based AMC and has a faster computing speed with parallel computation. P. Qi *et al*. addressed methods to realize deep residual networks (ResNet)-based AMC [11], which can efficiently distinguish among sixteen modulated signals. However, existing DL-based AMC methods generally use the algorithm of local learning (LocalAMC) or centralized learning (CentAMC) with CNN or ResNet [10], [11], [13]. The LocalAMC means that each edge device (ED) relies on a limited local dataset for training without using datasets from other EDs, which leads to limited performance. The CentAMC means that multiple EDs upload the local datasets to a central device (CD) for training, which has a better performance because training the model with multiple datasets. However, CentAMC challenges the computing capability and storage of CD and threats the privacy security of data during the process of dataset sharing.

To solve the problems of LocalAMC and CentAMC, a distributed learning-based AMC (DistAMC) method based on CNN was proposed in [12], which can realize decentralized training of data by the way of multiple devices uploading the model weight rather than datasets to a CD. However, the model size of CNN adopted in DistAMC is large and the model weight of CNN needs to be updated frequently, which lead to that the communication cost and the training efficiency of DistAMC method based on CNN is intolerable. In addition, CNN has a high complexity and thus is hard to deploy to EDs. Hence, we attempted to deploy lightweight network in the decentralized learning-based (DecentAMC) method. Many scholars have adopted different ways to realize lightweight network. Y. Lin *et al*. [13] proposed an improved lightweight AMC method by considering the pruning technology to reduce the size of neural networks for promising edge applications. A lightweight neural network based on separable convolution (SCNN) for AMC was proposed in [14] in which the model complexity of SCNN was decreased by about 94% when compared with CNN.

In this paper, a more lightweight neural network based on group CNN (GCNN) is proposed for DecentAMC method. The main contributions of the proposed GCNN-based DecentAMC method include:

- The proposed GCNN occupy less memory and computing capability of IoT devices because the space complexity

is decreased by at least 98.30% and the time complexity is decreased by at least 99.58%, compared with CNN.

- The proposed GCNN-based DecentAMC method can increase the training efficiency to at least 3.99 times than that of GCNN-based CentAMC with the cooperation in which multiple EDs update and upload more light model weight to a CD for model aggregation.
- The proposed GCNN-based DecentAMC method can effectively reduce the communication cost when compared with CNN-based DecentAMC.

II. Problem Formulation

A. Signal Model

We assume that the model of the unknown single-carrier (SC) modulated signals as

$$u(k) = \lambda e^{j(2\pi f_0 k/K+\theta)} \sum_{l=0}^{L-1} h[l]q[k-l-\tau]_{\mathrm{mod}K} + \sigma(k),\ k \in \{0,1,\cdots,K-1\}, \tag{1}$$

where $u(k)$ represents the unknown modulated signals received by receiver, and λ represents the gain of the channel, and f_0 represents the frequency offset, and θ represents the phase offset, and τ represents the timing offset, and $h[l : l = 0, 1, \cdots, L-1]$ represents the channel impulse response (CIR) of Rayleigh fading channel, L is the length of the CIR, and $q(k)$ is the baseband signal sequence, and $\sigma(k)$ is additive white Gaussian noise, and K is the number of sampled points from the signals.

The in-phase component (I) and quadrature component (Q) of $u(k)$ called IQ signals are input to neural network for modulation classification and expressed as: $I = \{\mathrm{real}[u(k)]\}_{k=0}^{K-1}$, and $Q = \{\mathrm{imag}[u(k)]\}_{k=0}^{K-1}$.

B. The DL-based AMC Methods

DL-based AMC method belongs to feature based AMC method, which can complete feature extraction and classification simultaneously. The modulation type of the received modulated signal belongs to the set $D = \{d_j, j = 0, 1, \cdots, J-1\}$, where J represents the number of modulation types. DL-based AMC can be expressed as: $d_j = F_{d_j \in D}([I;Q], P)$, where $F(\cdot)$ is the function of DL-based AMC classifier, and P represents the parameter of the network. The location of DL-based AMC method in communication system is shown in Fig. 1

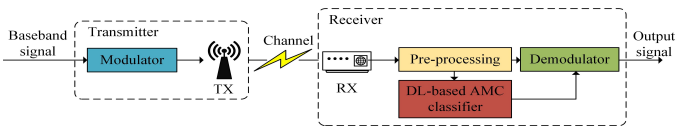


Fig. 1. The location of DL-based AMC method in communication system.

We adopt cross entropy (CE) loss function with ℓ_2 regularization to compile the model, which is expressed as

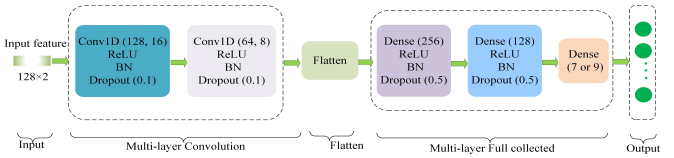
$$L_{CE} = -\frac{1}{R}\sum_{r=1}^{R} y_r log[F([I;Q],P)] + \lambda Q(F([I;Q],P)), \tag{2}$$

where R represents the size of the training samples, and y_r is the true label, and $Q(\cdot)$ is a penalty function to avoid overfitting, and λ is to balance the penalty function.

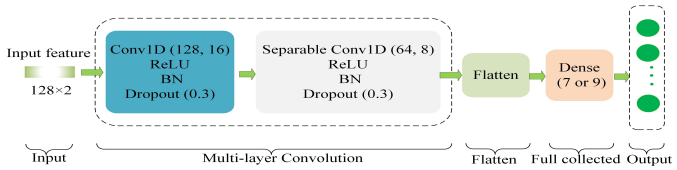
III. The Proposed AMC Method

A. The Proposed Lightweight Network

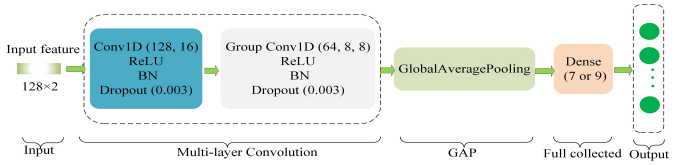
To design lightweight network for AMC, we start with standard CNN [12] and lightweight network named SCNN [14] for AMC, and then a more lightweight network called GCNN is proposed for AMC.



(a) CNN



(b) SCNN



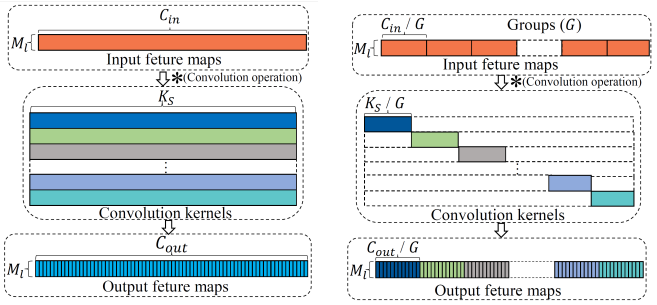
(c) GCNN

Fig. 2. The structures of CNN, SCNN and GCNN.

1) CNN for AMC: The structure of standard CNN for AMC is shown in Fig. 2(a). Two standard convolutional layers (Conv1D) are used to extract signal features. One flatten layer transforms multidimensional inputs into one-dimensional outputs and is usually located between convolutional layer and the fully connected (FC) layer. Three FC layers realize the classification of the final modulated signal.

2) SCNN for AMC: The structure of lightweight network called SCNN for AMC is shown in Fig. 2(b). The SCNN is an improvement of lightweight on CNN. Specifically, separable convolutional layer (Separable Conv1D) is adopted to replace the second standard convolutional layer of CNN and part of FC layers of CNN are removed.

3) The Proposed GCNN for AMC: Depends on the basis of the research on CNN and SCNN, a new lightweight network called GCNN is designed. The structure of GCNN is shown in Fig. 2(c). The complexity of the neural network includes time complexity and space complexity. The time complexity is described by floating point operations (FLOPs), while the space complexity is represented by parameters and output feature maps. The complexity of CNN can be further reduced by replacing the second standard convolutional layer and the flatten layer with the group convolutional layer (Group Conv1D) and the global average pooling (GAP) layer respectively and by removing part of FC layers. Group convolution was first applied in AlexNet for training on multiple GPUs because of the limitation of single GPU [15].



(a) Standard convolution (b) Group convolution

Fig. 3. Comparison of standard convolution and group convolution. C_{in}, C_{out}, M_l and K_s respectively represent the number of input channels, and the number of output channels, and the length of out feature maps of convolutions, and the size of convolution kernel. G is the number of groups for group convolution. (a) standard convolution: the convolution kernels are conducted on all of the input channels. (b) group convolution: the input channels and the convolution kernels are equally divided into G groups when the number of groups is not 1.

The structure comparison of group convolution and standard convolution is shown in Fig. 3. The model complexity comparison of group convolution and standard convolution [16] can be calculated by

$$\frac{S^g_{group}}{S^g_{sta}} = \frac{M_l \cdot C_{out}}{M_l \cdot C_{out}} = 1, \tag{3}$$

$$\frac{S^p_{group}}{S^p_{sta}} = \frac{K_s \cdot C_{in} \cdot C_{out} \cdot \frac{1}{G}}{K_s \cdot C_{in} \cdot C_{out}} = \frac{1}{G}, \tag{4}$$

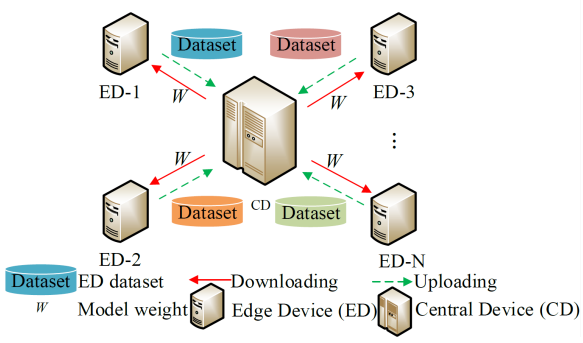
$$\frac{F_{group}}{F_{sta}} = \frac{M_l \cdot K_s \cdot C_{in} \cdot C_{out} \cdot \frac{1}{G}}{M_l \cdot K_s \cdot C_{in} \cdot C_{out}} = \frac{1}{G}, \tag{5}$$

where S^p_{sta}, S^g_{sta}, and F_{sta} respectively represent parameters, out feature maps, and FLOPs of standard convolution; S^p_{group}, S^g_{group}, and F_{group} respectively represent parameters, out feature maps, and FLOPs of group convolution.

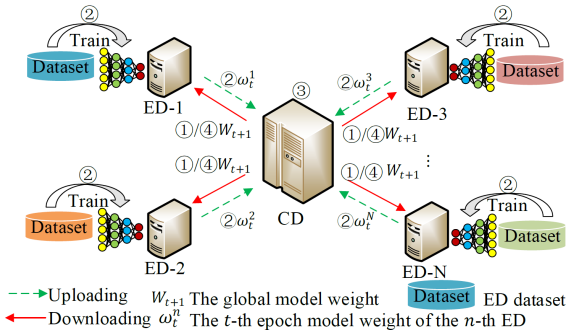
It is obvious that the FLOPs and parameters of standard convolution are G times than that of group convolution, and the out feature maps are the same with group convolution by observing Eqs. (3)∼(5). It means that group convolution can generate the same size out feature maps with a smaller number of parameters and computation. This is the theoretical basis for designing lightweight network using group convolution to take place of standard convolution. In addition, to further lighten the designed network, we replace the flatten layer with GAP to achieve feature compression and to avoid overfitting, and remove the first two FC layers. As is shown in Fig. 2(c), the final GCNN has only four layers, where ReLU activation function is applied in the first convolution layer and the second group convolution layer, and the last layer uses softmax as the activation function. Each layer is added batch normalization (BN) and dropout to avoid overfitting except for the last FC layer, meanwhile the BN can accelerate training of the model.

B. The Proposed GCNN-based DecentAMC Method

Traditional IoT devices generally adopt CentAMC method, as is shown in Fig. 4(a). Multiple EDs upload the local datasets to a CD, and then the CD trains the model based on the aggregated datasets. When the training is finished, the model weight W of the CD is downloaded to the EDs for testing. As shown in Fig. 4(b), compared with CentAMC method, each ED uploads the model weight instead of the dataset to the CD in DecentAMC method. It needs to be pointed out that the DecentAMC method falls with Federated Learning (FL) [17], which applies the idea of FL to model training process for AMC. The GCNN-based DecentAMC method consists of the following 4 steps, which are shown in **Algorithm 1**.



(a) CentAMC



(b) DecentAMC

Fig. 4. The structures of CentAMC and DecentAMC.

1) Model initialization and parameter broadcasting: The CD builds a lightweight model (GCNN model) and initialization parameters, such as the batch size, model weight,

learning rate and so on. Then, the CD broadcasts the built model and initialized parameter to the EDs.

2) The model weight of the EDs updating and uploading: Each ED downloads the built model and initialized parameter from the CD, and then updates the local model weight based on the local datasets by Adam. When the updating is finished, each ED uploads their model weight to the CD. Here, we assume that the EDs upload the model weight ω_t^n once every training epoch, where the t-th epoch model weight of the n-th ED as ω_t^n.

3) The EDs model weight aggregation by CD: The trained model weight is uploaded from each ED to the CD and the CD averages the received model weights. Model weight averaging is used as: $W_{t+1} = \frac{1}{N}\sum_{n=1}^{N}\omega_t^n$, where W_{t+1} is the t-th epoch global model weight.

Algorithm 1: The GCNN-based DecentAMC Method.

Input: IQ samples and corresponding labels in n-th ED dataset.
Output: W_T

```
CD initializes a GCNN model and broadcasts the model and initial parameters to EDs.
foreach ED in parallel do
    ED downloads the built model.
    ED initializes parameter.
end
for n = 1, ..., N do
    Each ED downloads the global model weight.
    for t = 0, ..., T do
        Each ED updates the local model weight ω_t^n.
        Each ED uploads their updated model weight.
    end
    CD receives the weight and averages them and obtains updated global model weight
    W_{t+1} = (1/N) Σ_{n=1}^{N} ω_t^n.
end
return W_T
```

4) The global model weight updating: After the CD gets the global model weight, each ED downloads the global model weight from the CD and replaces the original weight with the global model weight, i.e., $\omega_{t+1}^n = W_{t+1}, n = [1, N]$ and then repeat 2) $\sim$ 4) until the loss convergence.

IV. Experimental Results

To test the generalization classification performance of the proposed AMC method, we use two different datasets for training, named D_1: {BPSK, QPSK, 8PSK, 2FSK, 4FSK, 8FSK, 16QAM}, D_2: {BPSK, QPSK, 8PSK, 2FSK, 4FSK, 8FSK, 16QAM, 128QAM, 256QAM}. We assume that there are 12 EDs and 1 CD. Hence, 24 different ED datasets are used to simulate 12 different EDs. The size of training samples, validating samples and testing samples of each ED are $6,000$, $1,000$ and $10,000$ respectively. The signal to noise ratio (SNR) of ranging from -10 dB to 20 dB with 2 dB as interval is used. The number of the training epochs is set as 500. The parameters $\{\alpha, \beta_1, \beta_2\}$ for Adam are pre-set values of Keras, i.e., $\{0.001, 0.9, 0.999\}$. The balance factor λ is set as 0.005. The experiment platform is GTX 2080Ti and the DL framework is Tensorflow 1.10.0 with Keras 2.2.4.

A. The Setting of K and G

TABLE I
THE ACCURACY, PARAMETERS, FLOPS AND TEST TIME OF GCNN-BASED DECENTAMC IN D_2 UNDER DIFFERENT K.

Metric	K = 64	K = 128	K = 256	K = 512
Accuracy	63.31%	70.58%	74.42%	77.63%
Parameter	7,049	13,385	26,057	51,401
FLOPs	144,299	269,611	520,235	1,021,483
Test time (ms/sample)	0.220	0.231	1.159	1.291

As is shown in TABLE I, with the increasing the number of sampling points K, the classification performance of GCNN-based DecentAMC is improved obviously. Meanwhile, the time and space complexity of the model become higher and the test time of each samples is longer. Therefore, we adopted a compromise between classification performance and model lightweight and set K as 128.

TABLE II
THE ACCURACY, PARAMETERS, FLOPS AND TEST TIME OF GCNN-BASED DECENTAMC IN D_2 UNDER DIFFERENT G.

Metric	G = 2	G = 4	G = 8	G = 16
Accuracy	69.75%	70.16%	70.58%	69.61%
Parameter	37,961	21,577	13,385	8,289
FLOPs	736,525	425,239	269,611	191,827
Test time (ms/sample)	0.359	0.262	0.231	0.206

The group number G is a key hyper-parameter, which influences the time and space complexity and classification performance of the model. As is shown in TABLE II, G needs to be set as a divisor of the size of convolution kernel (K_s) and we set different group numbers to make a balance between model complexity and classification performance. Finally, it is appropriate to set G as 8.

B. Loss Evaluation & Classification Performance

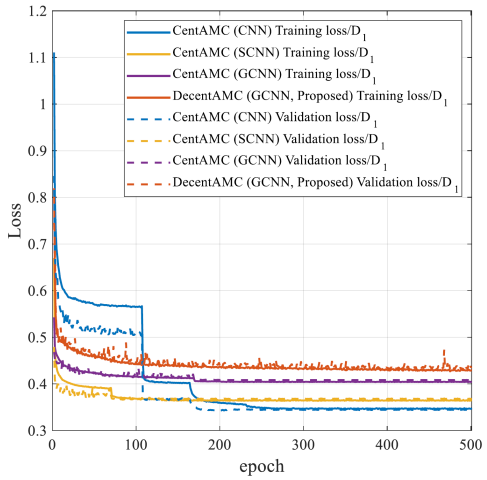
The training loss and the validation loss are given in Fig. 5. The validation loss is almost consistent with the training loss, which indicates that the algorithm is perfectly trained. The loss gap is nearly 0.1 in D_1 and 0.15 in D_2 between GCNN-based on DecentAMC and CNN-based on CentAMC, which means that GCNN-based on DecentAMC has similar classification performance with CNN-based on CentAMC.

The correct classification probability (PCC) is generally adopted to describe the recognition and classification performance, which can be written as $P_{cc}^i = \frac{N_{correct}^i}{N_{test}} \times 100\%$, where P_{cc}^i is the correct classification probability at SNR$= i$ dB, and $N_{correct}^i$ is the number of corrected classification samples at SNR$= i$ dB, and N_{test} is the number of the testing samples. The average value of P_{cc}^i is P_{cc}.

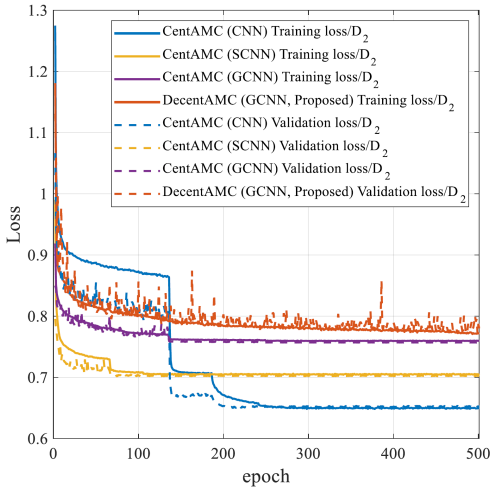
TABLE III
THE FLOPS, PARAMETERS, OUTPUT FEATURE MAPS, P_{CC}, T_{train}^{Cent} AND $T_{train}^{Decent}(+T_w)$ OF THE THREE NETWORKS.

Network/Dataset	FLOPs	Parameters	Output feature maps	P_{cc}	$T_{train}^{Cent}(s)$	$T_{train}^{Decent}(+T_w)(s)$
CNN/D_1	63,007,759	2,202,183	33,159	84.71%	103.88	7.41 (+1059.90)
SCNN/D_1	1,214,955	71,239	49,161	84.19%	56.23	5.08 (+36.15)
GCNN/D_1	267,153 (99.58%↓)	13,255 (99.40%↓)	24,647 (25.67%↓)	82.75% (2.31%↓)	60.72	5.08 (+10.12)
CNN/D_2	63,012,137	2,202,441	33,161	72.43%	139.76	8.35 (+1060.32)
SCNN/D_2	1,493,509	87,625	49,163	71.63%	97.49	6.41 (+44.04)
GCNN/D_2	231,712 (99.63%↓)	13,385 (99.40%↓)	24,649 (25.67%↓)	70.68% (2.42%↓)	82.98	8.61 (+10.17)

Note: The red numbers represent the model complexity, P_{CC} and the time to transfer weight gap between GCNN and CNN.

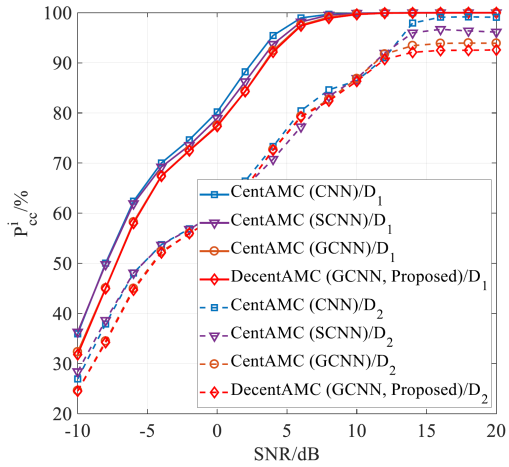


(a)

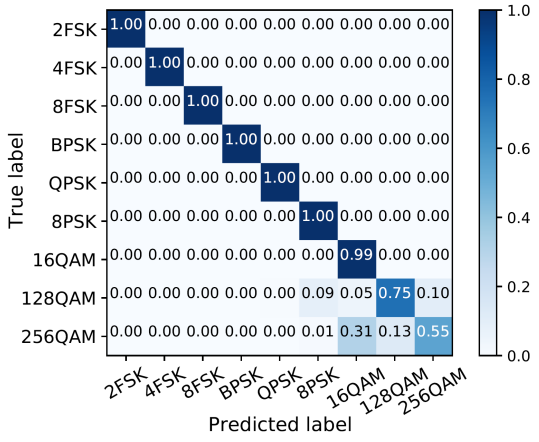


(b)

Fig. 5. The training loss (a) and the validation loss (b) of the CentAMC and the DecentAMC based on CNN, SCNN, and GCNN in D_1 and D_2 datasets.



(a)



(b)

Fig. 6. Classification performance. (a) The P_{cc}^{i} of CNN-based CentAMC, SCNN-based CentAMC, and GCNN-based DecentAMC in D_1 and D_2. (b) the confusion matrix of GCNN-based DecentAMC in D_2 when SNR = 15 dB.

The curves of P_{cc}^{i} is shown in Fig. 6(a). The experiment results show that there is a little classification performance loss between GCNN-based DecentAMC and CNN-based CentAMC, where the maximum classification performance gap is nearly 5% in D_1 and 7% in D_2. The average classification performance P_{CC} gap is nearly 2% either in D_1 or D_2, as is shown in TABLE III. In addition, GCNN-based DecentAMC has a limited classification performance for high-order modulation schemes such as 128QAM and 256QAM when observing Fig. 6(b).

C. Other Performance Analysis

1) Training Efficiency: The time to transfer weight can be calculated as: $T_w = \frac{W_m}{v_{up}} + \frac{W_m}{v_{down}}$, where W_m represents

the model weight, and v_{up} and v_{down} respectively represent downlink and uplink rate. And hence, training efficiency can be defined as: $E_f = \frac{1}{T_w+T_{train}}$, where T_{train} represents average training time per epoch. According to TABLE III, T_{train}^{Cent} and T_{train}^{Decent} represent average training time per epoch when adopted CentAMC and DecentAMC respectively. It is clear that the training efficiency of DecentAMC is higher than that of CentAMC when adopted lightweight networks (SCNN and GCNN). In addition, the training efficiency of GCNN is higher than that of GCNN-based CentAMC and CNN-based DecentAMC. Specifically, the training efficiency of GCNN-based DecentAMC is 3.99 times than that of GCNN-based CentAMC in D_1 and 4.42 times in D_2, and 70.22 times than that of CNN-based DecentAMC in D_1 and 56.90 times in D_2.

2) Communication Cost: The communication cost of CentAMC is defined as: $C_{Cent} = N(W_m + W_d)$, where C_{Cent} represents the communication cost of CentAMC, and W_d represents dataset size and N is the number of EDs. Similarly, the communication cost of DecentAMC can be defined as: $C_{Decent} = 2NW_mT$, where C_{Decent} represents the communication cost of DecentAMC, and T represents the number of average training epochs of DecentAMC and we assumed that T equals 250 epochs to calculate the communication cost. As is shown in TABLE IV, the DecentAMC consumes more communication overhead than CentAMC and lightweight network can reduce the gap of communication overhead between CentAMC and DecentAMC. In addition, the communication cost of GCNN-based DecentAMC is less than CNN-based DecentAMC and SCNN-based DecentAMC because GCNN has more lightweight global weight that need to be updated between EDs and CD.

TABLE IV
THE COMMUNICATION COST AND MODEL SIZE OF THE THREE NETWORKS BASED CENTAMC AND DECENTAMC.

Model/Dataset	C_{Cent}	C_{Decent}	**Model size**
CNN/D_1	1.28 GB	155.28 GB	8.86 MB
SCNN/D_1	0.98 GB	5.15 GB	315.99 KB
GCNN/D_1	0.97 GB	1.44 GB	122.92 KB
CNN/D_2	1.55 GB	155.30 GB	8.86 MB
SCNN/D_2	1.26 GB	6.42 GB	381.53 KB
GCNN/D_2	1.25 GB	1.45 GB	123.18 KB

3) Model Complexity: As is shown in TABLE III. Compared with CNN, the time complexity of GCNN is decreased by 99.58% in D_1 and 99.63% in D_2, and the space complexity of GCNN is decreased by nearly 98.30% in D_1 and D_2, while the P_{cc} is decreased by less than 3%. In addition, the model size of GCNN takes up less storage according to TABLE IV, which means that it is appropriate to deploy GCNN to EDs. Compared with SCNN, the time complexity of GCNN is decreased by 80.64% in D_1 and 83.39% in D_2, and the space complexity of GCNN is decreased by 68.30% in D_1 and 73.53% in D_2, while the P_{cc} is decreased by less than 0.86%.

V. CONCLUSIONS

In this paper, we proposed a GCNN-based DecentAMC method to realize AMC. Compared with CNN, the model complexity of the proposed GCNN is decreased significantly with a limited P_{cc} loss. In addition, the proposed GCNN-based DecentAMC method can increase the training efficiency whether compared with GCNN-based CentAMC or CNN-based DecentAMC. Meanwhile, the communication cost of GCNN-based DecentAMC can effectively reduce the communication cost and save storage of EDs when compared with CNN-based DecentAMC.

REFERENCES

[1] Z. Qin, X. Zhou, L. Zhang, Y. Gao, Y. Liang, and G.-Y. Li, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 6–20, Mar. 2020.

[2] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IET Commun.*, vol. 1, no. 2, pp. 137–156, Feb. 2007.

[3] J. Wang, G. Gui, T. Ohtsuki, B. Adebisi, H. Gacanin, and H. Sari, "Compressive sampled CSI feedback method based on deep learning for FDD massive MIMO systems," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5873–5885, Sept. 2021.

[4] Y. Wang, G. Gui, H. Gacanin, T. Ohtsuki, O. A. Dobre, and H. V. Poor, "An efficient specific emitter identification method based on complex-valued neural networks and network compression," *IEEE J. Sel. Areas. Commun.*, vol. 39, no. 8, pp. 2305–2317, Aug. 2021.

[5] Y. Tu, Y. Lin, *et al.*, "Large-scale real-world radio signal recognition with deep learning," *Chinese Journal of Aeronautics*, doi: 10.1016/j.cja.2021.08.016.

[6] Y. Wang, Z. Su, T. H Luan, R. Li, and K. Zhang, "Federated learning with fair incentives and robust aggregation for UAV-aided crowdsensing," *IEEE Trans. Netw. Sci.*, early access, doi: 10.1109/TNSE.2021.3138928

[7] S. Huang, C. Lin, W. Xu, Y. Gao, Z. Feng, and F. Zhu, "Identification of active attacks in internet of things: joint model- and data-driven automatic modulation classification approach," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 2051–2065, Mar. 2021.

[8] Y. Liu, Z. Su, and Y. Wang, "Energy-efficient and physical-layer secure computation offloading in blockchain-empowered internet of things," *IEEE Internet Things J.*, doi: 10.1109/JIOT.2022.3159248.

[9] R. Zhao, *et al.*, "A novel intrusion detection method based on lightweight neural network for internet of things," *IEEE Internet Things J.*, early access, doi: 10.1109/JIOT.2021.3119055

[10] F. Meng, P. Chen, L. Wu and X. Wang, "Automatic modulation classification: a deep learning enabled approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10760–10772, Nov. 2018.

[11] P. Qi, X. Zhou, S. Zheng, and Z. Li, "Automatic modulation classification based on deep residual networks with multimodal information," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 21–33, Mar. 2021.

[12] Y. Wang, L. Guo, Y. Zhao, J. Yang, B. Adebisi, H. Gacanin, and G. Gui, "Distributed learning for automatic modulation classification in edge devices," *IEEE Wireless Commun. Lett.*, vol. 9, no. 12, pp. 2177–2181, Dec. 2020.

[13] Y. Lin, Y. Tu, and Z. Dou, "An improved neural network pruning technology for automatic modulation classification in edge device," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5703–5706, May 2020.

[14] X. Fu, *et al.*, "Lightweight automatic modulation classification based on decentralized learning," *IEEE Trans. Cogn. Commun. Netw.*, early access, doi: 10.1109/TCCN.2021.3089178

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, no. 2, pp. 1097–1105, 2012.

[16] H. Wei, Z. Wang and G. Hua, "Dynamically Mixed Group Convolution to Lighten Convolution Operation," *ICAIBD*, vol. 25, pp. 203–206, 2021.

[17] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, pp. 1–11, Apr. 2016