Over-the-Air Split Machine Learning in Wireless MIMO Networks

Yuzhi Yang, Student Member, IEEE, Zhaoyang Zhang, Senior Member, IEEE, Yuqing Tian, Student Member, IEEE, Zhaohui Yang, Member, IEEE, Chongwen Huang, Member, IEEE, Caijun Zhong, Senior Member, IEEE, and Kai-Kit Wong, Fellow, IEEE

Abstract— In split machine learning (ML), different partitions of a neural network (NN) are executed by different computing nodes, requiring a large amount of communication cost. As over-the-air computation (OAC) can efficiently implement all or part of the computation at the same time of communication, thus by substituting the wireless transmission in the traditional split ML framework with OAC, the communication load can be eased. In this paper, we propose to deploy split ML in a wireless multiple-input multiple-output (MIMO) communication network utilizing the intricate interplay between MIMO-based OAC and NN. The basic procedure of the OAC split ML system is first provided, and we show that the inter-layer connection in a NN of any size can be mathematically decomposed into a set of linear precoding and combining transformations over a MIMO channel carrying out multi-stream analog communication. The precoding and combining matrices which are regarded as trainable parameters, and the MIMO channel matrix, which are regarded as unknown (implicit) parameters, jointly serve as a fully connected layer of the NN. Most interestingly, the channel estimation procedure can be eliminated by exploiting the MIMO channel reciprocity of the forward and backward propagation, thus greatly saving the system costs and/or further improving its overall efficiency. The generalization of the proposed scheme to the conventional NNS is also introduced, i.e., the widely used convolutional NNs. We demonstrate its effectiveness under both the static and quasi-static memory channel conditions with comprehensive simulations.

Index Terms— Over-the-air computing (OAC), multiple-input multiple-output (MIMO), split machine learning, neural network.

I. INTRODUCTION

A. Motivation

In future sixth-generation (6G) wireless communication systems, human-like intelligence will be brought everywhere [1]. The rapid development of artificial intelligence leads to booming mobile machine learning (ML) applications and requires vast data interaction. The integration of ML and wireless communications leads to an emerging area called split ML, which distributes a neural network (NN) to several edge devices, thus reducing the system computation burden. In a split ML system, each device first proceeds forward computation on the allocated partial NN and then transmits calculated intermediate results to the next device for further computation. The backpropagation process is conducted in a way similar to forward transmission but in a backward order. Through cooperative computation, split ML can be applied to solve deep learning-based joint source-channel coding (JSCC) problem [2] and other ML problems in cloud networks and the Internet of things. However, split ML requires frequent information exchange among devices, leveraging a heavy communication burden on wireless networks. Thus, deploying split ML over wireless networks calls for the design of new wireless techniques based on a communication-and-computation integration approach.

Recently, over-the-air computation (OAC) has emerged as a meaningful approach to innovate the traditional wireless digital communication framework [3]. Through using the intrinsic linear superposition property of wireless channels, OAC enables wireless communication systems to compute some large-scale, but simple calculation tasks [4]. Previous work [5] shows that if a group of devices simultaneously transmits analog modulated signals, the receiver obtains an aggregated signal, which can be directly applied to the typical federated learning (FL) network. However, traditional OAC works only consider the weighted sum of the edge users’ messages within a time-synchronized multiple-access framework and has not been well suited to the split ML system. In a typical wireless system with multiple-input multiple-output (MIMO) antenna arrays, a group of antennas at the transmitter sends different
signals simultaneously, while another group of antennas at the receiver collects the transmitted signal. On the receiver side, each antenna can individually receive the aggregated signal from multiple antennas of the transmitter. Intuitively, the MIMO channel can be viewed as a multiplication-and-addition procedure on the transmitted analog signals, which is widely used in NNs, and thus is of potential to be applied to the split ML in general wireless systems.

There is an intricate interplay between MIMO-based OAC and NN. A MIMO channel can provide a full connection between the inputs and the outputs and can thus be viewed as a weighted sum calculator. By different channel gains, each antenna on the receiver side conducts calculations with different equivalent weights. Unlike fully connected layers in NNs, which can be viewed as controlled weighted sums, the equivalent weights in MIMO systems are determined by the channel matrices, which are determined by the environment and hence uncontrollable. To control the equivalent weight in such systems, the typical precoding and combining operations in a MIMO system can be properly exploited. Both procedures are controllable linear transformations on the signals and are inherent in MIMO systems. As a result, we can control the parameters in the equivalent weighted sum of the overall system by controlling the precoding and combining matrices.

However, implementing the MIMO OAC in split ML still faces two fundamental issues: the forward and backward channels are different and may not be accurately known, and the analog transmission in OAC results in unavoidable noise. To deal with the uncertainty issue of the MIMO channel, we find that the forward-backward propagation of a NN and the channel reciprocity of a wireless channel are mathematically related (see Section II-B for details). This can be exploited to deploy a NN through MIMO-based OAC, which can still lead to correct gradients even without any prior knowledge about the MIMO channel as long as the channel reciprocity and quasi-stability are assumed. Moreover, most NNs can work well after proper training, even when the intermediate results cannot be fully and accurately interpreted. Such unexplainably in NNs inspires us that casting a deterministic linear transformation, i.e., multiplying an implicit matrix determined by the MIMO channel on the intermediate results of a NN through the whole training and test process, may not intensively deteriorate the performance.

The other issue about the noise in wireless communication is not fatal in NNs. From the perspective of information theory, both digital and analog communication can be optimal in wireless communications such as sensor networks [6]. However, the digital communication system can reduce transmission errors by employing error-detecting and correcting codes, whereas transmission errors can only be restricted but never eliminated in analog communication. Moreover, it is found that noise is tolerable and sometimes even becomes a training trick in NNs [7], which can also be viewed as implicit dropout [8]. Since the devices transmit unexplainable, intermediate results of the NN in split ML systems, the advantage of progressive error-free is insignificant in such applications. For instance, Jankowski et al. [2] show that analog communication performs better than digital communication in the deep learning-based JSCC problem, which is a particular case of split ML.

### B. Related Works

Split ML is a method where multiple computation nodes cooperatively execute an ML application. In such a system, the ML model is split into multiple parts allocated to different computation nodes. Each node executes a part of the ML model in order and transmits the intermediate results to the next one. When training the NN, the nodes also execute backward propagation in backward order. Most of the existing works [9], [10], [11] on split ML focus on how to distribute the model on the nodes in a way to minimize the total communication and computation delay. There are also some other works bringing up specialized NN structures for split ML [12], [13], [14]. Recent work also tries to deploy a proper NN architecture on a given communication network [15]. Their framework utilizes the neural architecture search method to meet latency and accurate requirements. However, the above works [9], [10], [11], [12], [13], [14], [15] all consider ideal communication among the computing nodes, which ignores the communication scheme design.

On the other hand, in communication systems, OAC is usually deployed in multiple access systems to compute the weighted sum or some easy mathematical operations such as geometric mean, polynomial, and Euclidean norm [4]. Hence most OAC works are mainly used in FL, where the weighted sum is widely deployed. For example, the authors in [5] optimize the number of simultaneous accesses, which may improve the efficiency of FL. Besides, Zhu et al. apply broadband analog aggregation to improve OAC in FL with multiple bands [16], while Shao et al. consider FL with misaligned OAC [17].

OAC for multiple access systems still has significant drawbacks. Firstly, OAC requires strict synchronization among all transmitters, which is hard to realize. Moreover, OAC does not support backward communication, which is rarely considered in previous works as far as we know. Furthermore, the above works [4], [5], [16], [17] do not consider the MIMO system, which is widely used in practical scenarios. The multiple antennas of the transmitter in a MIMO system can be viewed as a group of transmitters in the multiple access scheme, which overcomes the drawback of synchronization. The authors of [18] apply MIMO OAC to multimodal sensing. However, in [18], the scenario is still a multiple access system where all channels are MIMO channels, and the task of OAC is still the weighted sum, which can be regarded as a direct expansion of previous designs in [4], [5], [16], and [17]. Besides, the implementation of [18] is also strictly limited to FL applications, which is not suitable for split ML.

In other fields, OAC has also provided an alternative to traditional NNs by realizing parts of NNs with acoustic [19], optical [20], and radio frequency [21] signals. The authors in [19], [20], and [21] use the characteristics of the target systems similar to NNs, and employ the target systems as part of a NN. OAC systems calculate aggregated results of multiple inputs from different transmitters or time slots by moderating the environments or some parts of the system.
Recently, in wireless communications, Sanchez et al. also realize NNs with the help of multiple paths, and reconfigurable intelligent surfaces [22]. Their system transmits the intermediate output of NNs via time-sequential signals and uses the delay of multiple paths to realize one-dimensional convolution. However, in this paper, we use MIMO channels to realize fully connected layers through multiplexed signals.

C. Contributions

The main contributions of this paper are summarized as follows:

- A split ML framework is proposed for wireless MIMO networks by exploiting the MIMO’s OAC capability, which not only enables high-throughput and efficient wireless transmission but also reduces the overall computation load by synergistically incorporating the split ML process with the wireless transmission procedure rather than just taking it as a bit pipe.
- We show that the inter-layer connection in a NN of any size can be mathematically decomposed into a set of linear precoding and combining transformations over the MIMO channels. Therefore, the precoding matrix at the transmitter and the combining matrix at the receiver of each MIMO link, as well as the channel matrix itself, can jointly serve as a fully connected layer of a NN.
- By exploiting the reciprocity of the MIMO channel in the forward and backward propagation procedures in the proposed framework, we find it unnecessary to conduct explicit channel estimation as otherwise indispensable in conventional communication systems, thus further improving the overall communication and computation efficiency.
- We also provide some design rules for the proposed system so as to apply it to a fully connected layer of any size in a fully connected NN or a convolutional layer of any size in a convolutional NN. Simulation results show that the proposed scheme is efficient under both static and slowly-varying memory channel conditions.

The remainder of the paper is organized as follows. We first introduce the proposed system in Section II. We then mathematically provide some principles and propose a training algorithm in Section III. We extend the proposed system to convolutional NNs and compare different implementations of the system in Section IV. Numerical results are provided in Section V. Section VI concludes the paper and provides future directions.

D. Notations

In this paper, we use bold italic lower-case letters for vectors and bold letters for matrices. All the vectors and matrices are assumed to be complex. The meanings of frequently used notations are summarized in detail in Table I.

II. BASIC OAC UNIT

In this section, we first briefly introduce the system model and then propose a MIMO OAC-based approach to accelerate communication in split ML.

| Notation | Meaning |
|----------|---------|
| $N_t, N_r$ | The number of antennas on the transmitter and the receiver. |
| $H_{t, r}^{N_t, N_{s}}$ | The channel matrix. |
| $\mathbf{P}_k, \mathbf{C}_k$ | The precoding and combining matrices of transmission $k$. |
| $\mathbf{w}$ | The noise vector. |
| $r$ | The roughly estimated rank of $H$. |
| $\mathbf{W}$ | A trainable channel in a NN. |
| $K$ | A trainable convolutional kernel in a NN. |

TABLE I
NOTATIONS USED IN THIS PAPER

Consider a MIMO OAC-based split ML system with multiple base stations, as shown in Fig. 1. Each base station is equipped with multiple antennas, and the MIMO channels among base stations are quasi-stable. A deep NN is split into several segmented NN parts, each of which is deployed on one specific base station. The base stations process the forward computation and backward propagation of the assigned NN fragment. For simplicity, unless otherwise stated, we consider the split ML system with only one splitting point since it can be easily generalized to multiple-split conditions by conducting a set of single-split systems. To simplify the description, we use “transmit-ter” and “receiver” to refer to the transmitter and receiver with $N_t$ and $N_r$ antennas in the forward transmission, respectively.

Assuming that the channel between the transmitter and the receiver is a quasi-stable MIMO channel with reciprocity, i.e., the forward channel is $H \in \mathbb{C}^{N_r \times N_t}$, and the backward channel is $H^T$, we do not make any further priori hypotheses of the channel as the split ML applications may be deployed in different scenarios. Under the system setting, we assume that the rank of channel $H$ is known to be $r$, which determines the maximum amount of dataflows transmitted simultaneously under $H$. We note that the rank $r$ is mainly determined by the number of paths, which can be approximately known from the channel model and the number of scatterers. Therefore, although obtaining the accurate value of $r$ is impossible without channel estimation, we can get $r$ roughly. Some small singular values below a certain threshold can be regarded as zeros in the aspect of engineering. Hence, the rank $r$ can be underestimated in some cases. When designing the system, we assume that $r$ is known exactly, and we later show the cases where there exists a mismatch numerically in Section V. Besides, to better realize OAC, we may apply other techniques, such as applying the orthogonal frequency division multiplexing technique to realize multiple transmissions simultaneously. However, we do not consider such techniques since they do not affect the principles we use in the paper and can be easily combined with the proposed system.
B. OAC for Split ML

We consider a two-node split ML system, where the original NN is split into two parts, corresponding to the first several layers and the other layers, which are respectively deployed on the transmitter and the receiver. A fully connected layer between the transmitter and the receiver is realized through the OAC technique. We consider complex NNs [23], which have shown effectiveness in graph classification tasks in the proposed system. Since most data in wireless communication is complex, complex NNs are also potential in native wireless communication learning tasks. In complex NNs, the parameters, gradients, as well as intermediate results are complex numbers, and the backpropagation is almost the same as that of traditional real NNs. The transmitter can be regarded as a federation of several synchronized single-antenna transmitters in MIMO systems. Hence OAC still works similarly with multiple access systems. We will later discuss how to extend the proposed system to convolutional layers in Section IV-A. In particular, a fully connected layer in a NN can be represented by

\[ x_{\text{next}} = \phi(Wx + b), \]

(1)

where \( x \in \mathbb{C}^{N_i \times 1} \) and \( x_{\text{next}} \) is the input and the output of the considered layer, respectively, \( W \in \mathbb{C}^{N_o \times N_i} \) and \( b \in \mathbb{C}^{N_o \times 1} \) are the network parameters, and \( \phi(\cdot) \) is the element-wise activation function.

Considering the procedure of OAC, the transmitted and received signals may be amplified. Rather than using amplified parameters, we adopt a batch normalization (BN) layer to remain other parameters in a reasonable range and accelerate training [24]. A BN layer casts an element-wise linear transformation to the forward propagated data and normalizes them into a predetermined mean and variation distribution. The BN layer normalizes the intermediate output calculated by OAC to a reasonable range and can counteract the widely existing amplifications in communication systems, making it suitable for the considered system. With the BN layer, the output becomes

\[ x_{\text{next}} = \phi(\text{BN}(y + b)) = \phi(\text{BN}(Wx + b)), \]

(2)

where \( y = Wx \) is calculated through OAC and \( x_{\text{next}} \) is calculated on the receiver by the received \( y \). In the remaining parts of this paper, we only consider the OAC part of the system, i.e., how to calculate \( y \) from \( x \), since other parts have been well studied in traditional NNs. We discuss the mathematical similarity between the channel reciprocity and the forward-backward propagation below, which is also shown in Fig. 2.

1) Forward Computation: For each intermediate result, since its size is usually far larger than the maximum number of simultaneous transfers in a single OAC, we need to conduct \( K \) times of OAC for each forward computing, where \( K \) is the minimum number of times of progressing the intermediate results and its value is discussed in the following. In the \( k \)-th OAC, the precoding, transmitting, and combining matrices are denoted by \( P_k \in \mathbb{C}^{r \times N_i} \) and \( C_k \in \mathbb{C}^{r \times N_o} \), respectively. Hence, the precoding, transmitting, and combining procedures of the \( k \)-th OAC can be described respectively as

\[ x_{t,k} = P_k x, \]

(3)

\[ y_{r,k} = H x_{t,k} + n_k, \]

(4)

\[ y_k = C_k^H y_{r,k}, \]

(5)

where \( n_k \sim \mathcal{C}\mathcal{N}(0, \sigma^2 I) \) is the Gaussian white noise vector of the forward transmission with the variance \( \sigma^2 \). The receiver then combines all the \( K \) received signals, and the output of
the receiver or the transmitter except for (8). Equation (8)
received signal at the transmitter becomes
\[ y = \sum_{k=1}^{K} y_k = \sum_{k=1}^{K} C_k H P_k x + n_k. \]  
(6)

2) Backward Propagation: Here, we consider the gradient
backward propagation in our system. Let the gradient of \( y \)
be \( g_y = \frac{\partial \text{loss}}{\partial y} \). According to (3)-(5) and the principle
of gradient backward propagation, we can easily obtain the following equations:

\[ g_{C_k} = y_{t,k} g_y^H, \]  
(7)
\[ g_{x_{t,k}} = H^H C_k g_y, \]  
(8)
\[ g_{P_k} = x g_{x_{t,k}}^H, \]  
(9)
\[ g_x = \sum_{k=1}^{K} P_k^H g_{x_{t,k}}. \]  
(10)

All the above equations can be easily calculated by either
the receiver or the transmitter except for (8). Equation (8)
can be calculated with noise through backward OAC, where
the receiver acts as a transmitter in OAC and transmits \( g_y \)
with precoding matrix \( C_k \). For the backward propagation, the
received signal at the transmitter becomes
\[ \tilde{g}_{x_{t,k}} = H^T C_k g_y^* + \tilde{n}_k = g_{x_{t,k}}^* + \tilde{n}_k, \]  
(11)
where \( \tilde{n}_k \sim \mathcal{CN}(0, \sigma^2 I) \) is the Gaussian white noise vector in
the backward transmission. Equation (11) reveals that channel
reciprocity ensures that the backward OAC is identical to
backward propagation in NNs if we transmit the conjugation
of the gradient and ignore the noise.

III. SYSTEM IMPLEMENTATION AND ALGORITHM DESIGN

In this section, we first provide theoretical findings for split
ML based on fully connected layers over wireless networks.
We then formulate the optimization problem and propose the
training algorithm for the system.

A. Theoretical Findings in the Proposed System

Here, we only consider the matrix multiplication in the fully
connected layer, i.e., \( y = W x \). To ensure that the split ML
system works well, we must guarantee that the OAC system
can be equivalent to any parameter \( W \) of the fully connected
layer. Hence, we have the following feasible condition for
noiseless channels.

To realize the function of a fully connected layer, condition
(12) should have at least a feasible solution \( W \). Hence, we have the following feasible condition for
noiseless channels.

From feasible condition (12), we can obtain the following
lemma.

Lemma 1: If the rank of \( G = (C_1^H, \cdots, C_K^H) \) is at
least \( N_o \), then \( W \) can take any values from \( \mathcal{C}_{N_o \times N_i} \),
then feasible condition (12) always holds.

Proof: We first prove the first part. Since \( G = (C_1^H, \cdots, C_K^H) \),
we have \( \sum_{k=1}^{K} C_k^H P_k = G \). Due to the fact that the rank of \( G \)
is at least \( N_o \), we can easily observe that for arbitrary
\( W \in \mathcal{C}_{N_o \times N_i} \), let \( P_i, k = 1, \cdots, K \) be the submatrix
consisting the \((kr-k+1)\)-th to the \((kr)\)-th rows of \( G^{-1} W \).
Then, \( W = \sum_{k=1}^{K} C_k^H P_k \), which completes the first part
of Lemma 1. The second part of Lemma 1 is symmetric to
the first part, hence Lemma 1 is proved.

Based on Lemma 1, we further obtain the requirement for the
time \( K \) of OAC and the rank \( r \) of \( H \) that can satisfy (12).

Theorem 1: Under the condition that \( P_k \) and \( C_k \) can take
any values from \( \mathcal{C}_{N_i \times N_t} \) and \( \mathcal{C}_{N_r \times N_o} \) respectively, feasible
condition (12) holds if and only if \( Kr \geq \min\{N_i, N_o\} \).
Proof: We first provide the necessary condition. Since \( \text{rank}(C_k^H H P_k) \leq \text{rank}(H) = r \), we have
\[
\text{rank} \left( \sum_{k=1}^K C_k^H H P_k \right) \leq Kr. \tag{13}
\]
Hence if \( Kr < \min \{ N_i, N_o \} \), for any \( W \) with rank \( \min \{ N_i, N_o \} \), feasible condition (12) does not hold.

Then, we provide the sufficient condition. We know \( \text{rank}(I_K \otimes H) = Kr \), where \( \otimes \) denotes the Kronecker product and \( I_K \) denotes the \( K \times K \) identity matrix. If \( N_i \geq N_o \), we can easily know that there exist some \( C_k, k = 1 \cdots K \), such that \( \text{rank}(G) = Kr > N_o \), where \( G \) is defined in Lemma 1. From the first part of Lemma 1, feasible condition (12) holds. Similarly, if \( N_i < N_o \), we can still find some \( P_k, k = 1 \cdots K \), which satisfy the condition of the second part of Lemma 1, and hence feasible condition (12) also holds.

As a result, Theorem 1 is proved.

Lemma 1 and Theorem 1 give the sufficient and necessary conditions to feasible condition (12). Theorem 1 shows that we should at least conduct \( \min \{ N_i, N_o \}/r \) times of transmissions to satisfy feasible condition (12). The proof to Theorem 1 also guides us that we must follow the conditions in Lemma 1 to reach the bound. We will later show how to design the system with minimum transmission times following the conditions above.

In practical communication systems, there exist noise in both forward and backward communications, and the maximum transmitting powers for both transmissions are limited. However, the influences of the noise on the forward and backward channels are not the same. In the forward channel, the noise directly takes part in the forward computation and results in worse performance on the task. However, the noise only affects the gradient calculation in the backward channel. The well-known stochastic gradient descent (SGD) [25] algorithm uses gradients on random parts of the total dataset instead of the entire dataset and still works well since the expectation of the stochastic gradient equals the actual gradient. Since the expectation of the gradient added with zero-mean noise still equals the real gradient, the noise in the backward channel may not destroy the learning performance.

Besides, we note that all NNs consist of only two components, i.e., linear layers and nonlinear activation functions. Since the activation functions are always unary, the backward propagation procedures of both types of components are linear. Hence the backward propagation from the output of the NN to any parameter is linear. The only nonlinear part in the NNs training process may lie in the loss function at the output. Hence if the noise at the backward receiving antenna is Gaussian, as shown in Fig. 3, compared with the corresponding centralized NN, the gradient for any parameter should be either noiseless or with Gaussian noise.

We note that the stochastic gradient Langevin dynamics (SGLD) [26], [27], [28] also introduces noise to SGD. SGLD is proposed to accelerate the training of minibatch SGD, and the core inspiration of it is to introduce a carefully designed Gaussian noise to overcome the randomness brought by the sampling of minibatches. However, in the proposed system, the noise is brought by the wireless channel, whose distribution is hard to control. As a result, the proposed system works similarly to the SLGD, but the origins of the noises are different. Besides SLGD, we also give another way regarding this problem as follows.

Inspired by SGD, we know that the white noise in the backward channel does not have obvious significance in convergence, which can also be theoretically shown in the following Theorem 2. Before introducing Theorem 2, we define the following notations. Denote \( h(x; \theta) \) as the model, where \( x \) represents the input and \( \theta \) stands for the parameters. With SGD, the loss of the \( t \)-th batch can be written by
\[
\ell_t(\theta) = \sum_{b=1}^B \ell(h(x_{b,t}; \theta), y_{b,t}), \tag{14}
\]
where \( B \) represents the batch size, \( \ell(\cdot, \cdot) \) denotes the loss function, \( x_{b,t} \) and \( y_{b,t} \) stand for the \( b \)-th input data and the corresponding label of the \( t \)-th batch, respectively. Now, it is ready to provide the following theorem about convergence with noise in the backward channel.

**Theorem 2:** If all loss functions \( f_t(\theta), t = 1, 2, \cdots \) are convex, all the parameters and the gradients are bounded, i.e., \( \| f_t(\theta) - f_t(\theta') \| \leq D \) and \( \| g_t \| \leq G \) for any \( \theta \) and \( \theta' \), where \( g_t = \nabla f_t(\theta) \), the SGD algorithm converges in \( O(\sigma^2 T^{-1/2}) \) even with noise \( n \). However, the coefficient in the convergence time increases linearly with noise power.

**Proof:** Please refer to Appendix.

Theorem 2 indicates that the introduction of Gaussian white noise in the backward propagation will not affect the order in the convergence rate of SGD. Besides, the noise power only linearly changes the coefficient of the convergence rate.

Another problem comes from the amplification. In wireless communication systems, due to the effect of noise and the limited budget of the transmitting power, we must apply amplification for both forward and backward transmissions to match the transmit power, which may influence the performance of NNs. Hence, in the following, we discuss how to realize arbitrary amplification in the forward computation and backward propagation without sacrificing the performance of NNs significantly.

In the forward computation, we can apply a simple power normalization before transmitting, i.e., transmit \( \overline{P}_k x / A \) instead of \( P_k x \), where \( A = \sqrt{\mathbb{E}(x^H P_k^H P_k x)} \), as shown in Fig. 3. Since the normalization layer amplifies all the transmitted signals and there is a BN layer at the receiver to offset the effect of power amplification, it does not significantly affect the NN’s result.

In backpropagation, normalizing the power of the gradient of the backward propagation only amplifies the gradient of the subsequent calculation on the same scale. To overcome this problem, we can use optimizers insensitive to the magnitude of the gradient, such as the Adam optimizer [29].

**B. The Implementations for Fully Connected NNs**

Since the channel does not change over time, the optimum beamforming matrix remains the same among the transmissions. Hence we can always construct a receiving combining...
matrix $\mathbf{C} \in \mathbb{C}^{N_r \times r}$ such that matrix $\mathbf{H}^* \mathbf{C}$ is of rank $r$. As a result, the receiver can simultaneously decode $r$ independent data streams, i.e.,

$$\mathbf{C}_k = (\mathbf{0}_{N_r \times (k-1)r}, \mathbf{C}, \mathbf{0}_{N_r \times (K-k)r}), \quad k = 1, \ldots, K. \tag{15}$$

According to Lemma 1 and Theorem 1, the $\mathbf{C}_k$ above satisfies feasible condition (12) with the least number of transmissions $K$ when $N_t \geq N_o$. The procedure above is symmetric between $\mathbf{P}_k$ and $\mathbf{C}_k$ that we can also use the same precoding matrix $\mathbf{P} \in \mathbb{C}^{N_t \times r}$ such that matrix $\mathbf{H}^* \mathbf{P}$ is of rank $r$, and $\mathbf{P}_k = (\mathbf{0}_{N_t \times (k-1)r}, \mathbf{P}, \mathbf{0}_{N_t \times (K-k)r})$ for $k = 1, \ldots, K$ to satisfy feasible condition (12) with the least $K$ when $N_t \leq N_o$. We refer to the former ones as receiver parameterized designs and the latter ones as transmitter parameterized designs, respectively. Based on symmetry, we can only discuss the transmitter parameterized designs in this section.

According to Lemma 1, we obtain a straightforward implementation by directly setting all the $\mathbf{P}_k$s to be trainable parameters as shown in Fig. 3. However, the straightforward implementation does not take advantage of the fact that the structure of the precoding matrices should also be determined by the channel $\mathbf{H}$ to a certain degree. It is more desirable to separate the precoding matrices by two parts, i.e.,

$$\mathbf{P}_k = \mathbf{W}_0(\mathbf{0}_{N_r \times (k-1)r}, \mathbf{P}, \mathbf{0}_{N_r \times (K-k)r}), \quad k = 1, \ldots, K, \tag{16}$$

where $\mathbf{P}$ plays a similar role in the precoding matrices compared to $\mathbf{C}$ in the receiving combining matrices, and $\mathbf{W}_0$ is corresponding to the remaining NN parameters. The system is also illustrated in Fig. 3. In this paper, as shown in Fig. 3, we refer to the former implementation as the combined design and the latter as the separated design, respectively. The detailed comparisons among the implementations are provided in Section IV-B.

C. Problem Formulation

We first analyze the transmitter parameterized and separated design for full-connected layers. From equations (6), (15), and (16), the sum of the received signals of the forward transmissions $\mathbf{y}$ can be represented by

$$\mathbf{y} = \begin{pmatrix} \mathbf{C}^H \mathbf{P} \mathbf{x}_1 + \mathbf{C}^H \mathbf{n}_1 \\ \vdots \\ \mathbf{C}^H \mathbf{P} \mathbf{x}_K + \mathbf{C}^H \mathbf{n}_K \end{pmatrix}, \tag{17}$$

$$\begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_K \end{pmatrix} = \mathbf{W}_0 \mathbf{x} / A, \tag{18}$$

where $A$ denotes the power normalization parameter, i.e., $A = \| \mathbf{W}_0(\mathbf{0}_{(k-1)r : kr-1}) \mathbf{x} \|_2$. Hence the signal-noise ratio (SNR) of the $(k-1)r + t$-th element in $\mathbf{y}$ can be represented as

$$\text{SNR}_{f,(k-1)r+t} = \mathbb{E}_x(\| \mathbf{C}^H \mathbf{P} \mathbf{x}_k \|_2^2) N_r / P_n, \tag{19}$$

where $\mathbf{C}^H_k$ denotes the $t$-th row of $\mathbf{C}^H$ and $P_n$ is the total power of noise.

Symmetrically, the SNR of the backward transmission can be written by

$$\text{SNR}_{b,(k-1)r+t} = \mathbb{E}_x(\| \mathbf{P}^T (t,:) \mathbf{H}^T \mathbf{C}^* \mathbf{g}_{y,k} \|_2^2) N_t / (\bar{A}P_n), \tag{20}$$
where $\bar{A}$ is the power normalization parameter of backward communication.

Then, we briefly show the relationship among different implementations of our system. The transmitter parameterized designs are symmetric to the corresponding receiver parameterized designs, and the only difference is that the preceding matrices or the combining matrices are more complicated for combined designs. From the definition of SNR and the structure, SNR$_f$ and SNR$_b$ can be obtained similarly following the procedure above.

Naturally, the goal of MIMO communication is to maximize the SNR of the worst-used equivalent subchannel, i.e.,

$$\arg \max_{P,C} \min_{k,t} \min_{r \in [1:t]} \text{SNR}_f(k-1)r+t, \min_{k,t} \text{SNR}_b(k-1)r+t.$$  \hspace{1cm} \text{(21)}

In the optimal solution to (21), each row of $P$ and $C$ is the singular vector corresponding to the largest singular value of channel matrix $H$, i.e., both transmitter and receiver allocate full power on the optimal beamforming vector. Considering that our system is a joint design of communication and split ML, we should give a thought to the constraints of split ML, i.e., the conditions of Lemma 1 and Theorem 1 hold. However, the communication goal and the computation goal (the goal of NN) is coupled in the proposed system. We do not need to distinctly express the constraints of split ML in the overall loss of the NN since the computation goal of the system can implicitly ensure them. Based on this fact, we show how to translate problem (21) to the loss of NN in the following.

D. Training Method of the Proposed System

In this section, we introduce a method to translate (21) to the loss of NN. As explained above, simply considering the goal in (21) leads to an obviously unreasonable solution. Moreover, the goal in (21) is not derivable, making it impossible to directly applied in NNs. However, it inspires us that in the optimal solution to (21), each row of $H$ is coupled in the proposed system. From the definition of SNR and the singular vector corresponding to the largest singular value of $H$ in (21), we can find conditions for suboptimal solutions of (21) that

$$H = U\Sigma V^H,$$ \hspace{1cm} \text{(22)}

$$PV_{t-r:} = 0,$$ \hspace{1cm} \text{(23)}

$$CU_{t-r:} = 0,$$ \hspace{1cm} \text{(24)}

where (22) is the SVD decomposition of $H$, and $V_{t-r:}$ denotes the matrix consisting of the last $r$ columns of $V$, i.e., the singular vectors corresponding to the smallest $r$ singular values.

The conditions mainly come from Lemma 1 that $r$ is the minimum value that ensures the conditions of Lemma 1 and the conditions above could be satisfied simultaneously. Under the conditions above, the columns of the matrices $P_k$ and $C_k$ are constrained to linear spaces of dimension $r$, respectively. According to the basic linear algebra theory, there exist $P_{ks}$ and $C_{ks}$ satisfying both constraints. We note that in the final trained NNs, the ranks $P_k$ and $C_k$ may not always be $r$.

**Algorithm 1 Training Split ML via OAC**

```plaintext
for n = 1 to N do
  if n = 1 then
    Sample a batch of input data x
    Begin forward computation
    y0 ← x
  else
    Receive signal y by antennas from multiple transmissions
    Calculate the covariance matrix:
    $R_{f,n-1} ← E(yy^H)$
    Update the time-averaged covariance matrix:
    $R_{f,n-1} ← αR_{f,n-1} + (1-α)R_{f,n-1}$
    Obtain $y_{n-1}$ with the combining matrices as shown in Fig. 3
    Conduct local forward computation with $y_{n-1}$, and outputs $x_n$ for transmitting
    if n = N then
      Output $x_n$ as the final prediction
    else
      Transmit $x_n$ with the precoding matrices as shown in Fig. 3
  Calculate the loss $ℓ$ of the main task.
for n = N to 1 do
  if n ≠ N then
    Receive signal $g_x$ by antennas from multiple transmissions
    Begin backward propagation
    Calculate the covariance matrix:
    $R_{b,n} ← E(g_xg_x^H)$
    Update the time averaged covariance matrix:
    $R_{b,n} ← αR_{b,n} + (1-α)R_{b,n}$
    Cast SVD decomposition on $R_{b,n}$:
    $R_{b,n} = U\Sigma V^H$
    Add the goal of communication:
    $g_x ← g_x + (1/2B^2)\partial\|x_n^H V(:, N_t - r + 1 : N_t)\|^2/\partial x$
    Obtain $g_{x,n}$ with the preceding matrices as shown in Fig. 3
    Employ backward propagation from $g_{x,n}$, and obtain $g_{y,n-1}$
  else
    Employ backward propagation from $ℓ$, and obtain $g_{y,n-1}$
  if n ≠ 1 then
    SVD decomposition on $R_{f,n-1}$:
    $R_{f,n-1} = U\Sigma V^H$
    Transmit $g_y$ with the combining matrices as shown in Fig. 3
    Add the goal of communication:
    $g_C ← g_C + \partial\|CU(:, N_r - r + 1 : N_r)\|^2/\partial C$
```

Each node updates local parameters based on the calculated gradients.
However, it is caused by the sparsity of NN parameters [30], which does not mean a mistake in our design.

In real systems, we cannot directly obtain \( H \) directly. However, we can still realize the constraints in a similar way with self-adaptive beamforming. In simple self-adaptive beamforming algorithms, we usually use the covariance matrix of the received signal to estimate the singular vectors of the channel \( H \) and hence obtain the optimal beamforming vector. Similarly, we can use the singular vectors of the covariance matrix of the received signal to estimate the singular vectors and then compute the loss by (23) and (24). Combining (23), (24), and the training algorithm of centralized NNs, we finally obtain the detailed training algorithm for each batch of input data as shown in Alg. 1.

IV. GENERALIZATION AND COMPARISON

In this section, we generalize the proposed MIMO OAC-based split ML system to convolutional NNs. We also analyze and compare the different implementations of the proposed system.

A. Extension to Convolutional NNs

For the considered convolutional layer, we denote the size of the input and the output images as \( N_{wi} \times N_{hi} \) and \( N_{wo} \times N_{ho} \) with \( N_{ci} \) and \( N_{co} \) channels, respectively. The size of the convolutional kernel is \( N_k \times N_k \).

Here, we show how to extend the implementations to convolutional layers. The calculation in a convolutional layer can be described as:

\[
\begin{pmatrix}
Y_1 \\
\vdots \\
Y_{N_{co}}
\end{pmatrix} = \begin{pmatrix}
K_1 & \cdots & K_1 \\
\vdots & \ddots & \vdots \\
K_{N_{co}} & \cdots & K_{N_{co}}
\end{pmatrix} \ast \begin{pmatrix}
X_1 \\
\vdots \\
X_{N_{co}}
\end{pmatrix},
\]

(25)

where \( X_n, Y_n, K_n \) represent the \( n \)-th input channel, the \( n \)-th output channel, and the \( n \)-th convolutional kernel, respectively. The symbol \( \ast \) represents convolution operation, and the calculation rules between matrices are the same as matrix multiplication, except that convolution replaces multiplication.

Since the convolution operation can be seen as a particular multiplication with specific rules, it is still a linear operation. Hence it still satisfies the associative laws the. Introducing an arbitrary matrix \( W \in C_{N_{ci} \times N_{co}} \), we have

\[
(W \otimes I_{N_k \times N_k}) \begin{pmatrix}
K_1 & \cdots & K_1 \\
\vdots & \ddots & \vdots \\
K_{N_{co}} & \cdots & K_{N_{co}}
\end{pmatrix} \ast \begin{pmatrix}
X_1 \\
\vdots \\
X_{N_{co}}
\end{pmatrix} = (W \otimes I_{N_{ho} \times N_{ho}}) \begin{pmatrix}
K_1 & \cdots & K_1 \\
\vdots & \ddots & \vdots \\
K_{N_{co}} & \cdots & K_{N_{co}}
\end{pmatrix} \ast \begin{pmatrix}
X_1 \\
\vdots \\
X_{N_{co}}
\end{pmatrix}.
\]

(26)

Equation (26) shows that multiplying a matrix \( (W \otimes I_{N_k \times N_k}) \) after convolution is equivalent to multiplying the matrix on the kernels or applying another convolutional layer with \( 1 \times 1 \) kernels, i.e.,

\[
\begin{pmatrix}
K_1 & \cdots & K_1 \\
\vdots & \ddots & \vdots \\
K_{N_{co}} & \cdots & K_{N_{co}}
\end{pmatrix} = (W \otimes I_{N_k \times N_k}) \begin{pmatrix}
K_1 & \cdots & K_1 \\
\vdots & \ddots & \vdots \\
K_{N_{co}} & \cdots & K_{N_{co}}
\end{pmatrix}.
\]

(27)

Hence, for convolutional layers, we only need to replace the rearranging method in Fig. 3. The difference between the rearranging methods of full-connected and convolutional layers is shown in Fig. 4. We note that, in Fig. 4, only the intermediate output corresponding to a piece of input data is shown, whereas a batch of outputs is transmitted in each iteration in practice.

B. Comparisons Among the Implementations

In this section, we compare all the proposed implementations in terms of the amounts of parameters, calculation, and communication, which are described by the number of complex parameters, the number of complex multiply-accumulate operations (MACs), and the number of forward transmissions per batch, respectively. The MAC operation is defined as calculating \( a \leftarrow a + b \times c \) by a multiplier accumulator unit, which is widely adopted in the hardware field. We show the results of the comparison in Table II.

We note that \( r \) is always no more than \( \min\{N_r, N_i\} \), even probably being far smaller than \( \min\{N_r, N_i\} \) under sparse channels. From this fact and Table II, we know that for most cases, although the separated design requires one more time of matrix multiplication, it usually uses fewer parameters and costs less in computation because of smaller matrices. Moreover, whether the transmitter parameterized designs or the receiver parameterized designs are better depends on the relationship between the input and output sizes. If the input size is larger than the output size, the receiver parameterized designs are better, and vice versa.

V. NUMERICAL RESULTS

In this section, we evaluate the performance of the proposed system and provide numerical results based on the CIFAR-10 image classification dataset. We use the full CIFAR 10 dataset with 50000 training images and 10000 testing images in 10 categories. All the results are reported in terms of testing accuracy.

A. Experimental Settings

1) Communication System Settings: We adopt common MIMO communication settings, where the channel is composed of \( N_p \) different paths, as for the gain, transmission direction angle, and arrival direction angle of each path are independently and randomly generated by uniform distribution, i.e.,

\[
H = \sum_{n=1}^{N_p} a_n (1, \cdots, e^{j(N_r-1)\theta_n})^H (1, \cdots, e^{j(N_i-1)\phi_n}),
\]

(28)
where $\theta_n \sim U(-\pi, \pi)$, $\phi_n \sim U(-\pi, \pi)$, $|a_n| \sim U(0.5, 1.5)$, and $\text{angle}(a_n) \sim U(-\pi, \pi)$ for each $n$.

The total transmitting power of each transmission is normalized to 1, the noise power of each receiving antenna is the same, and the SNR is defined by

$$\text{SNR} = \|H\|_2^2 N_r / P_n,$$  \tag{29}$$

where $P_n$ is the total noise power.

We consider the following settings in this section. Firstly, we consider a 2-node complex channel setting, where $N_t = N_r = 16$ and $N_p = 20$, such that the channel is with full rank. We also consider a 3-node sparse channel setting, where $N_t = N_r = 16$ and $N_p = 4$ for both channels. Considering the massive MIMO settings, in reality, we consider a massive MIMO setting where there are 3 nodes, and for both channels, $N_t = N_r = 64$ and $N_p = 8$. Finally, we consider a 3-node moving setting where $N_t = N_r = 16$ and $N_p = 4$ for both channels. To ensure fairness, the channels of all settings are pre-generated and remain the same in all experiments.

2) NN Settings: We use the 16-layer complex ResNet to evaluate the performance of our proposed system and algorithms in this section. The NN structure follows the one in [23] and is shown in Fig. 5. “Conv3 * 3 * N_c, stride = 2” refers to a convolutional layer with $N_c$ kernels with size $3 * 3$ and stride 2, and stride is 1 if not written. “Downsampling” refers to cast downsampling such that the outputs of the shortcut connection and the convolutional layers have the same size, which is realized by the convolutional layer with $1 * 1$ kernels.

“Average pooling” and “Linear” stand for the corresponding layer with the following parameters. Each layer, except the pooling layer, is followed by a BN layer. For 2-node settings, the NN is split at the 9th layer, and for 3-node settings, the NN is split at the 5th and 13th layers. We also consider a VGGNet structure as also shown in Fig. 5. It is another famous convolutional NN structure that is wider but shallower than the ResNet. For 2-node settings, the VGGNet is split at the 5th layer, and for 3-node settings, the VGGNet is split at the 3rd and 8th layers. We use batch size 64 and Adam optimizer [29] with the learning rate of 0.005 to train the NN, and set $\alpha = 0.99$. Since the output sizes of the convolutional layers are no more than the corresponding input sizes, we use the receiver parameterized designs. According to Table II, we choose the separated design for $r < 16$ and the combined design for $r = 16$ for the minimum computation and storage cost.

B. Comparison Algorithms

We verify the proposed system and algorithm by comparing them with the straightforward algorithms below. The first is the centralized algorithm, where the NN is trained in a centralized manner. We also use traditional split ML and MIMO-based split ML, where the communication of split ML is conducted by traditional MIMO. We note that both of them are identical to the centralized algorithm in performance since, in such systems, communication is just a tube. The last comparison algorithm is called the ideal case, where the channel is...
assumed to be known exactly. Hence, from (22)-(24), we can use the following precoding and combining matrix:

$$
P^H = V_{:,N_r-r},$$

(30)

$$
C^H = U_{:,N_r-r},$$

(31)

where the matrices are calculated by (22). When training NNs, the matrices $P$ and $C$ is determined by the channel and is not trainable in the whole progress.

The costs of all considered algorithms are compared in Table III, where the smallest value in each column is marked in bold font. We assume that 16 QAM modulation is used in digital systems. Hence a 64-bit complex float number is represented by 16 digital symbols or an analog symbol. Moreover, digital communication systems require channel coding and retransmissions according to channel quality. Thus, analog communication requires no more than 1/16 times of transmission compared with digital communication. In Table III, $C_{NN}$, $C_{Trans}$, and $C_{CE}$ are defined as the total cost of calculating the NN (in terms of MACs), transmission (in terms of communication rounds), and channel estimation (in terms of times) in traditional split ML, respectively. $C_{MIMO}$ represents the total native computation cost by the MIMO system (precoding and combining, in terms of MACs) in traditional ML. According to the calculation in Table II, we always have $C_{MIMO} \ll C_{NN}$. We emphasize that our MIMO-based OAC design greatly reduces the transmission cost compared to traditional split ML without OAC. It also saves the system costs and/or improves its overall efficiency by eliminating explicit channel estimation when compared with the ideal case. According to Table III, the proposed algorithm yields the lowest cost, which reflects the high communication and transmission efficiency of the considered MIMO-based OAC structure.

| Cost Comparisons Among Different Algorithms | Computation | Transmission | Channel Estimation |
|---------------------------------------------|-------------|--------------|-------------------|
| Traditional Split ML                        | $C_{NN}$    | $C_{Trans}$  | $C_{CE}$          |
| MIMO-Based Split ML                         | $C_{NN} + C_{MIMO}$ | $C_{Trans}/r$ | $C_{CE}/r$ |
| Ideal Case                                  | $C_{NN} + C_{MIMO}$ | at most $C_{Trans}/(16r)$ | at most $C_{CE}/(16r)$ |
| Proposed                                    | $C_{NN} + C_{MIMO}$ | at most $C_{Trans}/(16r)$ | 0 |

C. Results Under Stable Complex Channel

In this section, we consider the 2-node complex channel setting. In the setting, the channel is almost of full rank since there are 20 independent randomly generated paths. We suppose the pre-estimation of the rank $r$ to be 4, 8, 12, or 16 to verify the performance of the algorithms.

Fig. 6 shows the convergence trend of various algorithms of the ResNet. In this figure, we consider a high-quality channel, where SNR = 35 dB and $r = 16$. In Fig. 6, both the ideal case and the proposed algorithm do not show a significant difference in terms of convergence rate, but with a slight performance loss, which verifies our analysis that the proposed system implementation is equivalent to the centralized ML system if there is almost no noise. From the centralized case, we can also find both structures have similar learning abilities.

In Figs. 7 to 10, we present the effect of SNR with different pre-estimation $r$s. Although the pre-estimation $r = 16$ is the most accurate, smaller $r$ lead to better performance. This is because when using smaller $r$,
smaller data size is transmitted per transmission, and fewer subchannels are used, both equivalent to higher SNR.

We note that when $r = 16$, all possible subchannels are used, and hence the optimization for communication does not valid. For other values of $r$, we can find that the proposed algorithm performs better with low SNR, whereas the ideal case performs better with high SNR. This is mainly because when the channel is well, the best strategy is to simply transmit all the outputs to the receiver. Hence the ideal case performs better. However, when the channel becomes poor, the best strategy becomes to stress transmitting only a part of the outputs to obtain higher quality. This strategy can make sense mainly because the output of NNs is usually sparse [30], and the performance loss caused by the reduction of width is smaller than that by the increase of the noise on the intermediate output.

In Fig. 8, we show the results of the proposed algorithm for both networks with $r = 16$. The results show that the performance of VGGNet is better than that of ResNet, especially that its performance drops later as the channel becomes worse. That is probably because the VGGNet is wider, requiring more communication load in each transmission, which is usually detrimental for split ML. However, in the proposed scheme, wider NNs can retain more information in the intermediate results, which is somehow similar to retransmission. From these results, the NN structure suitable for the proposed split ML system needs further discussion.

D. Results Under Stable Sparse Channels

In this section, we consider the 3-node sparse channel setting with ResNet. In the setting, the channel is of rank 4. We suppose the pre-estimation of the rank $r$ to be 4 or 8 to verify the performance of the algorithms. In Fig. 11, we show the performance of different algorithms and different pre-estimations of $r$. The results and conclusions are similar to those of complex channels. However, it is counterintuitive that the performance when $r = 8$ does not decline sharply, although it does not satisfy the conditions of Theorem 1. This phenomenon is mainly caused by the sparsity of NNs [30]. All the preconditions are derived by the suppose that the parameters $W$ can take arbitrary values in $\mathbb{C}^{N_i \times N_o}$, whereas the optimal $W$ is usually sparse in reality, which may explain the small performance loss.

E. Results Under Stable Massive MIMO Channels

We show the results under stable massive MIMO channels with $r = 8$ and ResNet in Fig. 12, which does not show apparent discrimination compared to Fig. 11, showing
the effectiveness of our proposed system in massive MIMO scenarios. However, in Fig. 12, the proposed algorithm cannot outperform the ideal case, which is mainly caused by that the proposed algorithm brings much error when the dimension becomes larger.

**F. Results Under Time-Varying Channels**

We show the results under dynamic channel settings here. In this setting, both the channel matrices vary according to the following memory model:

\[
\begin{align*}
a_{n,t+1} &= (1 - \rho)a_{n,t} + \rho \hat{a}_n, \\
\theta_{n,t+1} &= (1 - \rho)\theta_{n,t} + \rho \hat{\theta}_n, \\
\phi_{n,t+1} &= (1 - \rho)\phi_{n,t} + \rho \hat{\phi}_n,
\end{align*}
\]

where \(a_n\), \(\theta_n\), and \(\phi_n\) are the value of the corresponding values in (28) at \(t\)-th time slot, respectively. \(\hat{a}_n\), \(\hat{\theta}_n\), and \(\hat{\phi}_n\) are generated randomly following the way in (28), and \(\rho\) indicates the changing rate. The channel \(H_t\) at each time slot is also generated by (28). In Fig. 13, we show the average values and standard deviations of the results of 5 independent runs. The standard deviations are shown by the shadows. We can find that both the ideal case and the proposed algorithm perform well even under moving channels. In the ideal case, both the precoding and the combining matrices suddenly change when the channel changes, while the proposed algorithm can learn to adapt to the change of channel. Hence, the proposed algorithm outperforms the ideal case with smooth channel change, while performing poorly when the channel changes fiercely. It is mainly because the channel is considered implicit parameters in the NN. The training method is an iterative gradient algorithm, which cannot adapt the implicit parameters’ change very well. Moreover, the performance of VGGNet is still better since it is wider as we have explained above.

**VI. CONCLUSIONS AND FUTURE DIRECTIONS**

In this paper, we proposed a MIMO-based OAC system for implementing split ML over wireless networks, which significantly improves communication efficiency. In the proposed system, the precoder and combiner design, together with the implicit MIMO channel matrix, contribute to a trainable layer, which can be natively integrated into a NN. We theoretically found the inherent equivalent procedure between the reciprocity of MIMO channels and the training process of NNs. Based on this finding, the explicit channel estimation
process is eliminated in our system, which can improve the system’s efficiency. Besides, we provided basic principles for implementing the proposed system such that the proposed structure can be mathematically equivalent to any fully connected or convolutional layer in NNs. Numerical results show that the proposed system has a similar converge rate with slight performance degradation under high-quality channels compared to the centralized algorithm. We also observed that our algorithm could work well under various channels and is robust to slowly varying channels. However, since iterative algorithms cannot adapt to fast-changing implicit parameters, the proposed system can only be applied to systems with quasi-stable channels.

The numerical results show that the NN structure may affect the performance of the proposed system, requiring a specific design of the NN structure such as that in [31]. In this paper, we considered channel reciprocity and the extension to natural channels without strict reciprocity remains to be a future direction. The proposed MIMO OAC-based split ML structure can play an essential role in native distributed learning problems for wireless communications, such as distributed sensing, channel estimation, and prediction, location estimation and prediction, distributed network optimization, interference coordination, etc.

**Appendix**

**Proof of Theorem 2**

Proof: To prove Theorem 2, we introduce the regret to analyze the performance of the learning algorithm [32]. In particular, the regret is defined as

\[ R(T) = \sum_{t=1}^{T} f_t(\theta(t)) - \sum_{t=1}^{T} f_t(\theta^*) \]

where \( \theta^* \) is the optimal parameter set. Since function \( f_t(\theta) \) is convex, we have

\[ f_t(\theta(t)) - f_t(\theta^*) \leq \langle g_t, \theta(t) - \theta^* \rangle, \]

which shows that

\[ R(T) \leq \sum_{t=1}^{T} \langle g_t, \theta(t) - \theta^* \rangle. \]

According to the update procedure of SGD with noise, we have

\[ \theta(t+1) = \theta(t) - \eta_t (g_t + \Lambda_t n_t), \]

where \( \Lambda_t \) is a non-negative diagonal matrix, each element of which is no more than \( \xi \), and \( n_t \sim N(0^{N^2}, \sigma I^{N^2}) \) is the noise. Hence, we have

\[
\begin{align*}
\|\theta(t+1) - \theta^*\|^2_2 &= \|\theta(t) - \theta^* - \eta_t (g_t + \Lambda_t n_t)\|^2_2 \\
&= \|\theta(t) - \theta^*\|^2_2 - 2\eta_t \langle g_t, \theta(t) - \theta^* \rangle + \eta_t^2 \|g_t\|^2_2 \\
&= \|\theta(t) - \theta^*\|^2_2 + \eta_t^2 \|\Lambda_t\|^2_2 \|n_t\|^2_2 - 2\eta_t \langle \Lambda_t n_t, \theta(t) - \theta^* \rangle + 2\eta_t \langle \Lambda_t n_t, g_t \rangle.
\end{align*}
\]

According to (39), we can obtain

\[
\begin{align*}
\langle g_t, \theta(t) - \theta^* \rangle &= \frac{1}{2\eta_t} \left( \|\theta(t) - \theta^*\|^2_2 - \|\theta(t+1) - \theta^*\|^2_2 \right) \\
&\quad + \frac{\eta_t}{2} \|\Lambda_t\|^2_2 \|n_t\|^2_2 + \eta_t \|g_t\|^2_2 \\
&\quad - \langle \Lambda_t n_t, \theta(t) - \theta^* \rangle + \eta_t \langle \Lambda_t n_t, g_t \rangle \\
&\leq \frac{1}{2\eta_t} \left( \|\theta(t) - \theta^*\|^2_2 - \|\theta(t+1) - \theta^*\|^2_2 \right) \\
&\quad + \frac{\eta_t}{2} \|\Lambda_t\|^2_2 \|n_t\|^2_2 + \frac{\eta_t G^2}{2} \\
&\quad - \langle \Lambda_t n_t, \theta(t+1) - \theta^* \rangle,
\end{align*}
\]

where the last inequality comes from the fact that \( \|g_t\|^2_2 \leq G . \) Since there are always many parameters in a NN, i.e., \( N \) is very large, we can approximately get \( \|n_t\|^2_2 \approx N \sigma^2 \) based on the law of large numbers. Substituting (40) to (37), we have

\[
\begin{align*}
R(T) &\leq \sum_{t=1}^{T} \frac{1}{2\eta_t} \left( \|\theta(t) - \theta^*\|^2_2 - \|\theta(t+1) - \theta^*\|^2_2 \right) \\
&\quad + \frac{\eta_t}{2} \|\Lambda_t\|^2_2 \|n_t\|^2_2 + \frac{\eta_t G^2}{2} \\
&\quad - \langle \Lambda_t n_t, \theta(t+1) - \theta^* \rangle.
\end{align*}
\]

Due to the fact that the parameters are bounded and learning rate is decreasing over time, we have

\[
\begin{align*}
&\sum_{t=1}^{T} \frac{1}{2\eta_t} \left( \|\theta(t) - \theta^*\|^2_2 - \|\theta(t+1) - \theta^*\|^2_2 \right) \\
&= \frac{1}{2\eta_1} \|\theta(1) - \theta^*\|^2_2 - \frac{1}{2\eta_T} \|\theta(T+1) - \theta^*\|^2_2 \\
&\quad + \sum_{t=2}^{T} \left( \frac{1}{2\eta_t} - \frac{1}{2\eta_{t-1}} \right) \|\theta(t) - \theta^*\|^2_2 \\
&\leq \frac{1}{2\eta_1} D^2 + \sum_{t=2}^{T} \left( \frac{1}{2\eta_t} - \frac{1}{2\eta_{t-1}} \right) D^2 \\
&= \frac{D^2}{2\eta_1}.
\end{align*}
\]

Since \( n_t \) follows a standard Gaussian distribution, we can obtain expectation \( \mathbb{E}\{1/T \sum_{t=1}^{T} \langle \Lambda_t n_t, \theta(t) - \theta^* \rangle \} = 0 \) and standard deviation \( \sigma \{1/T \sum_{t=1}^{T} \langle \Lambda_t n_t, \theta(t) - \theta^* \rangle \} = \mathcal{O}(\sigma T^{-1/2}) \). When we choose \( \eta_t = C(t)^{1/2} \), we have

\[
R(T)/T \leq \frac{D^2}{2CT^{1/2}} + \frac{C(\xi^2 + G^2)}{2T} \sum_{t=1}^{T} \frac{1}{T} - \frac{1}{T} \sum_{t=1}^{T} \langle \Lambda_t n_t, \theta(t) - \theta^* \rangle.
\]

In the right hand of (43), the first item is of \( \mathcal{O}(T^{-1/2}) \), the second item is approximately by \( \mathcal{O}(\sigma^2 T^{-1/2}) \), and the third item is a random item with zero expectation and standard deviation of \( \mathcal{O}(\sigma T^{-1/2}) \). Hence, the convergence rate of the algorithm is \( \mathcal{O}(\sigma^2 T^{-1/2}) \). Comparing (43) with the regret of SGD provided in [32] that

\[
R(T)/T \leq \frac{D^2}{2CT^{1/2}} + \frac{CG^2}{T} \sum_{t=1}^{T} \frac{1}{T},
\]

we have
we can easily observe that noise power has a linear effect on the convergence rate of SGD with noise. Theorem 2 is proved.

REFERENCES

[1] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, "Toward self-learning edge intelligence in 6G," IEEE Commun. Mag., vol. 58, no. 12, pp. 34–40, Dec. 2020.

[2] M. Jankowski, D. Gunduz, and K. Mikolajczyk, "Wireless image retrieval at the edge," IEEE J. Sel. Areas Commun., vol. 39, no. 1, pp. 89–100, Jan. 2021.

[3] S. Dörner, S. Cammerer, J. Hoydis, and S. T. Brink, "Deep learning based communication over the air," IEEE J. Sel. Topics Signal Process., vol. 14, no. 2, pp. 330–339, Apr. 2020.

[4] M. Goldenbaum and S. Stanczak, "Robust analog function computation via wireless multiple-access channels," IEEE Trans. Commun., vol. 61, no. 9, pp. 3863–3877, Sep. 2013.

[5] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," IEEE Trans. Wireless Commun., vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[6] M. Gastpar, "Uncoded transmission is exactly optimal for a simple Gaussian 'sensor' network," IEEE Trans. Inf. Theory, vol. 54, no. 7, pp. 5247–5251, Nov. 2008.

[7] T. S. Nazaré, G. B. P. da Costa, W. A. Contato, and M. Ponti, "Deep convolutional neural networks and noisy images," in Proc. Pattern Recognit., Image Anal., Comput. Vis., Appl. (CIARP), Madrid, Spain, Nov. 2018, pp. 846–854.

[8] S. Wager, S. Wang, and P. Liang, "Dropout training as adaptive regularization," in Proc. 26th Int. Conf. Neural Information Process. Syst., Red Hook, NY, USA, vol. 1, 2013, pp. 351–359.

[9] Y. Kang et al., "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," ACM SIGARCH Comput. Archit. News, Xi’an, China, Apr. 2017, vol. 45, no. 1, pp. 615–629.

[10] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive DNN surgery for inference acceleration on the edge," in Proc. IEEE Conf. Comput. Commun. (INFOCOM), Paris, France, Apr. 2019, pp. 1423–1431.

[11] S. Wang, X. Zhang, H. Uchiyama, and H. Matsuda, "HiveMind: Towards cellular native machine learning model splitting," IEEE J. Sel. Areas Commun., vol. 40, no. 2, pp. 626–640, Feb. 2022.

[12] A. E. Esrratifar, A. Esmailli, and M. Pedram, "Bottleneck: A deep learning architecture for intelligent mobile cloud computing services," in Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED), Lausanne, Switzerland, Jul. 2019, pp. 1–6.

[13] J. Shao and J. Zhang, "Bottlenets+: An end-to-end approach for feature compression in device-edge co-inference systems," in Proc. IEEE Int. Conf. Comput. Commun. Workshops (ICC Workshops), Dublin, Ireland, Jun. 2020, pp. 1–6.

[14] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "JALAD: Joint accuracy and latency aware deep structure decoupling for edge-cloud execution," in Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS), Singapore, Dec. 2018, pp. 671–678.

[15] Y. Tian, Z. Zhang, Z. Yang, and Q. Yang, "JMSNAS: Joint model split and neural architecture search for learning over mobile edge networks," in Proc. IEEE Int. Conf. Comput. Commun. Workshops (ICC Workshops), May 2022, pp. 103–108.

[16] G. Zhu, Y. Wang, and D. Huband, "Broccoli: Analog aggregation for low-latency federated edge learning," IEEE Trans. Wireless Commun., vol. 19, no. 1, pp. 491–506, Jan. 2019.

[17] Y. Shao, D. Gunduz, and Z. Chang Liew, "Federated edge learning with misaligned over-the-air computation," 2021, arXiv:2102.13604.

[18] G. Zhu and K. Huang, "MIMO over-the-air computation for high-mobility multimodal sensing," IEEE Internet Things J., vol. 6, no. 4, pp. 6089–6103, Aug. 2019.

[19] T. W. Hughes, I. A. D. Williamson, M. Minkov, and S. Fan, "Wave physics as an analog recurrent neural network," Sci. Adv., vol. 5, no. 12, Dec. 2019, Art. no. eaay9646.

[20] X. Sui, Q. Wu, J. Liu, Q. Chen, and G. Gu, "A review of optical neural networks," IEEE Access, vol. 8, pp. 70773–70783, 2020.

[21] A. M. Elbirk and K. Vijay Mishra, "A survey of deep learning architectures for intelligent reflecting surfaces," 2020, arXiv:2009.02540.

[22] S. Garcia and J. H. Seo et al., "AirNet: Neural networks with over-the-air convolution via reconfigurable intelligent surfaces," 2022, arXiv:2202.03399.
Yuqing Tian (Student Member, IEEE) received the B.S.Eng. degree in information engineering from Zhejiang University, Hangzhou, China, in 2020. She is currently pursuing the Ph.D. degree in information and communication engineering with Zhejiang University, under the supervision of Prof. Zhaoyang Zhang. Her current research interests include machine learning for wireless networks, distributed algorithms, and neural architecture search.

Zhaohui Yang (Member, IEEE) received the Ph.D. degree from Southeast University, Nanjing, China, in 2018. From 2018 to 2020, he was a Post-Doctoral Research Associate at the Center for Telecommunications Research, Department of Informatics, King’s College London, U.K. From 2020 to 2022, he was a Research Fellow at the Department of Electronic and Electrical Engineering, University College London, U.K. He is currently a ZJU Young Professor with the Zhejiang Key Laboratory of Information Processing Communication and Networking, College of Information Science and Electronic Engineering, Zhejiang University, and also a Research Scientist with Zhejiang Laboratory. His research interests include joint communication, sensing, and computation, federated learning, and semantic communication. He was the Co-Chair for international workshops with more than ten times including IEEE ICC, IEEE GLOBECOM, IEEE WCNC, IEEE PIMRC, and IEEE INFOCOM. He is an Associate Editor for the IEEE COMMUNICATIONS LETTERS, IET Communications, and EURASIP Journal on Wireless Communications and Networking. He has served as a Guest Editor for several journals including IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.

Chongwen Huang (Member, IEEE) received the B.Sc. degree from Nankai University in 2010, the M.Sc. degree from the University of Electronic Science and Technology of China in 2013, and the Ph.D. degree from the Singapore University of Technology and Design (SUTD) in 2019. From October 2019 to September 2020, he was a Post-Doctoral Researcher at SUTD. Since September 2020, he joined at Zhejiang University, as a tenure-track young Professor. His main research interests are focused on holographic MIMO surface/reconfigurable intelligent surface, B5G/6G wireless communication, mmWave/THz communications, and deep learning technologies for wireless communications. He was a recipient of IEEE Marconi Prize Paper Award in wireless communications and IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award in 2021. He has served as an Editor of IEEE COMMUNICATIONS LETTERS, Signal Processing (Elsevier), EURASIP Journal on Wireless Communications and Networking, and Physical Communication, since 2021.

Caijun Zhong (Senior Member, IEEE) received the B.S. degree in information engineering from Xi’an Jiaotong University, Xi’an, China, in 2004, and the M.S. degree in information security and the Ph.D. degree in telecommunications from University College London, London, U.K., in 2006 and 2010, respectively. From September 2009 to September 2011, he was a Research Fellow at the Institute for Electronics, Communications and Information Technologies (ECIT), Queen’s University Belfast, Belfast, U.K. Since September 2011, he has been with Zhejiang University, Hangzhou, China, where he is currently a Professor. His current research interests include reconfigurable intelligent surfaces assisted communications and artificial intelligence based wireless communications. He was a recipient of the 2013 IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award. He and his coauthors was awarded the Best Paper Award at the IEEE ICC 2019 and IEEE GLOBECOM 2020. He is an Editor of Science China Information Science and China Communications. He was an Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE COMMUNICATIONS LETTERS.

Kai-Kit Wong (Fellow, IEEE) received the B.Eng., M.Phil., and Ph.D. degrees in electrical and electronic engineering from the Hong Kong University of Science and Technology, Hong Kong, in 1996, 1998, and 2001, respectively. After graduation, he took up academic and research positions at the University of Hong Kong, Lucent Technologies, Bell-Laboratories, Holmdel, the Smart Antennas Research Group of Stanford University, and the University of Hull, U.K. He is the Chair in wireless communications with the Department of Electronic and Electrical Engineering, University College London, U.K. His current research centers around 5G and beyond mobile communications. He is fellow of IET and also the editorial board of several international journals. He was a co-recipient of the 2013 IEEE SIGNAL PROCESSING LETTERS Best Paper Award and the 2000 IEEE VTS Japan Chapter Award at the IEEE Vehicular Technology Conference, Japan, in 2000, and a few other international best paper awards. He is the Editor-in-Chief for IEEE WIRELESS COMMUNICATIONS LETTERS since 2020.