Federated Edge Learning with Misaligned Over-The-Air Computation

Yulin Shao, *Member, IEEE*, Deniz Gündüz, *Senior Member, IEEE*, Soung Chang Liew, *Fellow, IEEE*

***Abstract*—Over-the-air computation (OAC) is a promising technique to realize fast model aggregation in the uplink of federated edge learning (FEEL). OAC, however, hinges on accurate channel-gain precoding and strict synchronization among edge devices, which are challenging in practice. As such, how to design the maximum likelihood (ML) estimator in the presence of residual channel-gain mismatch and asynchronies is an open problem. To fill this gap, this paper formulates the problem of misaligned OAC for FEEL and puts forth a whitened matched filtering and sampling scheme to obtain oversampled but independent, samples from the misaligned and overlapped signals. Given the whitened samples, a sum-product ML (SP-ML) estimator and an aligned-sample estimator are devised to estimate the arithmetic sum of the transmitted symbols. In particular, the computational complexity of our SP-ML estimator is linear in the packet length, and hence is significantly lower than the conventional ML estimator. Extensive simulations on the test accuracy versus the average received energy per symbol to noise power spectral density ratio (EsN0) yield two main results: 1) In the low EsN0 regime, the aligned-sample estimator can achieve superior test accuracy provided that the phase misalignment is non-severe. In contrast, the ML estimator does not work well due to the error propagation and noise enhancement in the estimation process. 2) In the high EsN0 regime, the ML estimator attains the optimal learning performance regardless of the severity of phase misalignment. On the other hand, the aligned-sample estimator suffers from a test-accuracy loss caused by phase misalignment.**

***Index Terms*—Federated edge learning, over-the-air computations, asynchronous, maximum likelihood estimation, sum product algorithm.**

I. Introduction

With the increasing adoption of Internet of Things (IoT) devices and services, exponentially growing amount of data is collected at the wireless network edge. Increasingly complex machine learning models are trained and deployed to gather intelligence from the data collected by edge devices [1], [2]. While this is conventionally done at a cloud server [3], offloading huge amounts of edge data to centralized cloud servers is not sustainable, and will potentially cause significant network congestion [4]. Moreover, data from edge devices contain user-specific features, and centralized processing also causes privacy concerns. Federated learning (FL) has been proposed as an alternative distributed solution to enable collaborative on-device learning without sharing private training data [5]–[7].

FL is an iterative distributed learning algorithm. In its basic implementation, orchestrated by a parameter server (PS), each iteration of FL consists of four main steps [6]: 1) Downlink (DL) broadcast – a PS maintains a global model and periodically broadcasts the latest global model to the edge devices; 2) Local training – upon receiving the latest global model, each edge device trains the model locally using its local data set; 3) Uplink (UL) model aggregation – after training, all, or a subset, of devices transmit their model updates back to the PS; 4) Global model update – the PS updates the global model using the model updates collected from the edge devices, typically by taking their average.

In the case of edge devices, often the devices that collaborate to learn a common model are within physical proximity of each other, and are coordinated by a nearby access point, e.g., a base station acting as the PS. In this, so-called federated edge learning (FEEL) scenario [4], the UL model aggregation step is particularly challenging as the wireless medium is shared among all the participating devices. Traditional radio access network (RAN) technologies distribute channel resources among the devices by means of orthogonal multiple-access technologies [8] (e.g., TDMA, CDMA, OFDMA). However, such orthogonal resource allocation techniques significantly limit the quality of model updates that can be sent from individual devices due to the limited channel resource that can be allocated to each device. We note that the number of real values to be transmitted by each device scales according to the neural network size. For today's neural networks, this number can easily run into hundreds of millions or more [9], and hence, is a heavy burden for the RANs.

Analog over-the-air computation (OAC) is a promising technique to realize uplink model aggregation in an efficient manner [10]–[18]. The basic idea of OAC is to create and leverage inter-user interferences over the multiple-access channel (MAC) rather than trying to avoid it. When operated with OAC, devices send their model updates in an uncoded fashion by directly mapping each model parameter to a channel symbol: each device first precodes the transmitted symbols by the inverse of the UL channel gain (assumed to be known to the transmitter in advance) and then transmits the precoded symbols to the receiver in an analog fashion. All the participating devices transmit simultaneously in the same communication link such that their signals overlap at the PS. Provided that the channel-gain precoding and transmission timing are accurate, the fading MAC reduces to a Gaussian MAC and the signal overlapped from the devices to the PS

Y. Shao and D. Gündüz are with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: y.shao@imperial.ac.uk, d.gunduz@imperial.ac.uk). S. C. Liew is with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong (e-mail: soung@ie.cuhk.edu.hk).

This work was supported by the European Research Council project BEACON under grant number 677854, and by CHIST-ERA grant CHIST-ERA-18-SDCDN-001 (funded by EPSRC-EP/T023600/1).

over-the-air naturally produces the arithmetic sum of the local model-updates [17].

Compared with the traditional digital multiple-access schemes, wherein the communication and computation constitute separate processes, OAC is a joint computation-and-communication scheme exploiting the fact that the MAC inherently yields an additive superposed signal.

The successful operation of OAC hinges on accurate channel-gain precoding and strict synchronization among the participating devices [11], [12]. In practice, however, both requirements may not be perfectly fulfilled. On the one hand, the channel-gain precoding at the edge devices can be imperfect due to the inaccurate channel estimation and non-ideal hardware. The consequence is that there can be residual channel-gain mismatch in the overlapped signals. On the other hand, to meet the synchronization requirement, each device has to carefully calibrate the transmission timing – based on its distance from the PS and its moving speed – so that their signals overlap exactly with each other at the PS. This strict synchronization across different devices is very expensive to realize in practice, and there can be residual asynchronies among the signals from different devices.

With the residual channel gains and residual asynchronies in the system, which we refer to as the *misaligned OAC*, an open problem is how to estimate the arithmetic sum of the transmitted symbols from different devices. This paper fills this gap and addresses the key problem in the misaligned OAC on how to devise the maximum likelihood (ML) estimator in the face of the channel-gain and time misalignments among signals.

Our main contributions are as follows:

1) We formulate the problem of misaligned OAC for FEEL considering a time-domain realization of OAC. We put forth a whitened matched filtering and sampling (WMFS) scheme that yields oversampled, but independent, samples from the overlapped signals. An ML estimator for the arithmetic sum based on the whitened samples is devised.
2) To tackle the inter-symbol and inter-device interferences in the misaligned OAC, ML estimation requires the inversion of a large coefficient matrix, and hence, is computationally intensive. In view of this, we dissect the inner structure of the whitened samples and put forth a factor-graph based ML estimator exploiting the sparsity of the coefficient matrix. This factor-graph estimator, dubbed sum-product ML (SP-ML) estimator, interprets the compositions of samples by a factor graph and computes the likelihood functions via an analog message passing process on the graph. With the SP-ML estimator, the computational complexity of ML estimation is significantly reduced from $\Omega(L^2 \log L)$ to $\Omega(L)$ for a packet of length L.
3) We identify two main problems of ML estimation in the misaligned OAC: error propagation and noise enhancement. As a result, ML estimation does not work well in the low average received energy per symbol to noise power spectral density ratio (EsN0) regime. To tackle this problem, we further put forth an aligned-sample estimator for the misaligned OAC leveraging a subsequence of whitened samples, wherein the symbols from different devices are "aligned", i.e., the indexes of symbols from different devices are consistent in these samples. This estimator is shown to be a good alternative to the ML estimator in the low-EsN0 regime. The complexity of the aligned-sample estimator is also linear in the packet length.
4) With the ML and aligned-sample estimators for the misaligned OAC, we perform extensive simulations on the CIFAR dataset varying the degrees of time misalignment, phase misalignment, and EsN0. The learning performance is measured by means of test accuracy, i.e., the achieved accuracy of the learned neural network on the test data set. We find that i) When there is no phase misalignment, the test accuracies of the ML estimator and the aligned-sample estimator are on the same footing for various degrees of time misalignment. ii) When there is phase misalignment, the ML estimator works only in the high-EsN0 regime whereas the aligned-sample estimator works in both the low and high EsN0 regimes. Nevertheless, the aligned-sample estimator suffers from a loss in the test accuracy due to phase misalignment. In particular, the larger the phase misalignment, the greater the test-accuracy loss. In the case of severe phase misalignment, the aligned-sample estimator leads to learning divergence even in the noiseless case. The ML estimator, on the other hand, does not incur such test-accuracy loss even with severe phase misalignment. iii) When there is phase misalignment, the ML estimator benefits from time asynchronicity while the aligned-sample estimator suffers from time asynchronicity. iv) Overall, the aligned-sample estimator is preferred in the low-EsN0 regime and the ML estimator is preferred in the high-EsN0 regime.

Notations – We use boldface lowercase letters to denote column vectors (e.g., $\boldsymbol{\theta}$, $\boldsymbol{s}$) and boldface uppercase letters to denote matrices (e.g., $\boldsymbol{V}$, $\boldsymbol{D}$). For a vector or matrix, $(\cdot)^\top$ denotes the transpose, $(\cdot)^*$ denotes the complex conjugate, $(\cdot)^H$ denotes the conjugate transpose, and $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse. $\mathbb{R}$ and $\mathbb{C}$ stand for the sets of real and complex numbers, respectively. $(\cdot)^{\mathfrak{r}}$ and $(\cdot)^{\mathfrak{i}}$ stand for the real and imaginary components of a complex symbol or vector, respectively. The imaginary unit is represented by j. $\mathcal{N}$ and $\mathcal{CN}$ stand for the real and complex Gaussian distributions, respectively. The cardinality of a set $\mathcal{V}$ is denoted by $|\mathcal{V}|$. The sign function is denoted by $\mathrm{sgn}(\cdot)$.

II. System Model

We consider FEEL with the help of a wireless PS where nearby edge devices with distinct local datasets collaborate over the shared wireless medium to train a common model, as shown in Fig. 1. The learning process goes through many iterations. Without loss of generality, let us focus on one of the iterations, wherein M active devices participate in the training. The iteration proceeds as follows [6]:

1) DL broadcast: at the beginning of the iteration, the PS broadcasts the global model $\boldsymbol{\theta} \in \mathbb{R}^d$ to the M edge devices;

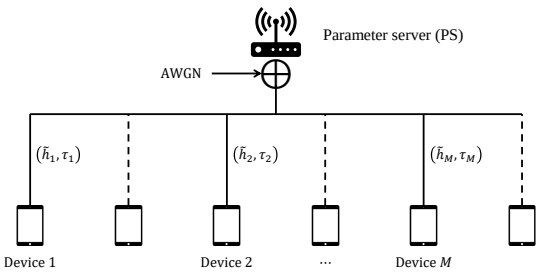

Figure 1. In FEEL, edge devices collaboratively train a shared model with the help of a wireless access point acting as a PS.

2) Local training: each of the M devices trains the global model $\boldsymbol{\theta}$ on its local dataset $\mathcal{B}_m$ of size B_m and obtains a new model $\tilde{\boldsymbol{\theta}}_m \in \mathbb{R}^d$;
3) UL aggregation: each device scales the local model update $\tilde{\boldsymbol{\theta}}_m - \boldsymbol{\theta}$ by B_m and transmits the scaled model update $\boldsymbol{\theta'_m} = B_m(\tilde{\boldsymbol{\theta}}_m - \boldsymbol{\theta}) \in \mathbb{R}^d$ back to the PS;
4) Arithmetic-sum estimation: the PS estimates the arithmetic sum of the transmitted model-updates $\boldsymbol{\theta'_m}$ from the edge devices:

$$\boldsymbol{\theta}_+ = \sum_{m=1}^{M} \boldsymbol{\theta'_m}; \tag{1}$$

5) Model update: the PS updates the global model by

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta} + \frac{1}{\sum_m B_m}\boldsymbol{\theta}_+. \tag{2}$$

The updated global model $\boldsymbol{\theta}_{\text{new}}$ is then broadcasted in the next iteration and the cycle continues.

Remark. *To compute* (2)*, the sum of the dataset sizes $\sum_m B_m$ has to be known to the PS a priori. Thus, we let each device transmit the local dataset size B_m to the PS reliably in advance of data transmission (in a digital manner over a control channel, with channel coding and automatic repeat request, for example).*

Among the above five steps, the uplink model aggregation poses the greatest challenge to the RAN. In this step, each device has to transmit d real numbers to the PS, where d can run into hundreds of millions or more. We consider analog OAC to realize the uplink model aggregation in this paper.

When operated with OAC, edge devices transmit their raw model updates $\boldsymbol{\theta'_m}$ simultaneously to the PS in an analog manner (without digital modulation and channel coding). The PS, on the other hand, estimates the sum of the model updates $\boldsymbol{\theta}_+$ directly from the received overlapped signal.

The signal flow is detailed as follows. Each device partitions its sequence of scaled model update $\boldsymbol{\theta'_m} \in \mathbb{R}^d$ to two subsequences $\boldsymbol{\theta'_m} = [(\boldsymbol{s}^{\mathfrak{r}}_{\boldsymbol{m}})^\top, (\boldsymbol{s}^{\mathfrak{i}}_{\boldsymbol{m}})^\top]^\top$, where $\boldsymbol{s}^{\mathfrak{r}}_{\boldsymbol{m}}$, $\boldsymbol{s}^{\mathfrak{i}}_{\boldsymbol{m}} \in \mathbb{R}^{d/2}$, and constructs a complex sequence $\boldsymbol{s_m} \in \mathbb{C}^{d/2}$: $\boldsymbol{s_m} = \boldsymbol{s}^{\mathfrak{r}}_{\boldsymbol{m}} + j\boldsymbol{s}^{\mathfrak{i}}_{\boldsymbol{m}}$, that is, the raw model update information $\boldsymbol{\theta'_m}$ is carried on both the real and imaginary parts of $\boldsymbol{s_m}$.

Time is divided into slots, and each device transmits a packet of L symbols in each slot where $\boldsymbol{s_m} = [s_m[1], s_m[2], ..., s_m[L]]^\top$. Therefore, to transmit the total number of $d/2$ complex symbols, $\lceil d/2L \rceil$ slots are needed. Without loss of generality, we focus on the signal processing in one slot.

The time-domain signal transmitted by the m-th device in one slot is given by

$$x_m(t) = \alpha_m \sum_{\ell=1}^{L} s_m[\ell]p(t - \ell T), \tag{3}$$

where 1) $p(t) = 1/2\,[\text{sgn}(t+T) - \text{sgn}(t)]$ is a rectangular pulse of duration T; 2) α_m is the channel precoding factor. Given an estimated channel coefficient $\bar{h}_m$ at the m-th device, α_m is designed to be $\alpha_m = 1/\bar{h}_m$.[1] Each of the M edge devices then calibrates the transmission timing, based on its distance from the PS and its moving speed, so that the signals from different devices arrive at the PS simultaneously.

In practice, however, both the channel-gain precoding and transmission-timing calibration can be imperfect due to the non-ideal hardware and inaccurate estimation of the channel gains and transmission delays. The received signal $r(t)$ at the PS can be written as

$$r(t) = \sum_{m=1}^{M} \tilde{h}_m x_m(t - \tau_m) + z(t), \tag{4}$$

where

1) $\tilde{h}_m$ is the time-domain complex channel response. We consider flat fading (frequency nonselective) and slow fading (time nonselective) channels [19]. That is, for the channel between each device and the PS, the maximum delay spread is less than the symbol period T so that there is only one resolvable path with the channel response $\tilde{h}_m$ at the receiver, and $\tilde{h}_m$ remains constant over one packet.[2]
2) Without loss of generality, we sort the M devices so that the symbols from the devices with smaller indexes arrive at the receiver earlier. The delay of the first device is set to $\tau_1 = 0$, and the relative delay of the m-th device with respect to the first device is denoted by τ_m. We assume the time offsets τ_m, $\forall\, m$, are less than the symbol duration T, as shown in Fig. 2. In the ideal case where the timing calibrations are perfect, the relative delays among packets are $\tau_m = 0$, $\forall m$.
3) $z(t)$ is the zero-mean baseband complex additive white Gaussian noise (AWGN), the double-sided power spectral densities of each of its real and imaginary parts is $N_0/2$ for an aggregate of N_0.

Substituting (3) into (4) gives us

$$\begin{aligned} r(t) &= \sum_{m=1}^{M} \tilde{h}_m \alpha_m \sum_{\ell=1}^{L} s_m[\ell]p(t - \tau_m - \ell T) + z(t) \\ &= \sum_{\ell=1}^{L}\sum_{m=1}^{M} h'_m s_m[\ell]p(t - \tau_m - \ell T) + z(t), \end{aligned} \tag{5}$$

[1]In practice, the channel-gain precoding is limited by the maximum transmission power of the edge devices. In the case of deep fading α_m would be very large, and we would have to clip α_m to satisfy the peak or average transmission-power constraint. In our formulation, this may be one cause of the residual channel gains at the receiver.

[2]To ease exposition, the main body of this paper considers only the channel-gain misalignment, time misalignment, and slow fading channel. However, the system model can be easily generalized to OAC with residual carrier frequency offset (CFO) and fast fading channel, as detailed in Appendix A.

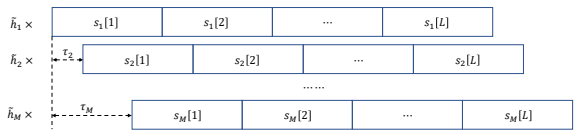

Figure 2. In each slot, the transmitted packets from different devices overlap at the PS with channel misalignments and relative time offsets.

where $h'_m = \tilde{h}_m/\bar{h}_m$ is the residual channel-fading coefficient between the m-th device and the PS.

Succinctly speaking, there can be two kinds of misalignments among the signals transmitted from different devices: 1) channel-gain misalignment h'_m caused by inaccurate channel-gain precoding; and 2) time misalignment τ_m caused by inaccurate calibration of the transmission timing.

The objective of the PS is to estimate $\boldsymbol{\theta}_+ = \sum_{m=1}^M \boldsymbol{\theta'_m}$, i.e., the arithmetic sum of the local model updates. This is equivalent to estimating the arithmetic sum of the transmitted complex symbols $\boldsymbol{s}_+$, where

$$s_+[i] \triangleq \sum_{m=1}^M s_m[i],$$

because $\boldsymbol{\theta'_m}$ are carried on the real and imaginary parts of the $\boldsymbol{s_m}$ sequence. In other words, given the estimated sequence $\hat{\boldsymbol{s}}_+$, the estimated arithmetic sum of the local model updates is $\hat{\boldsymbol{\theta}}_+ = \left[(\hat{\boldsymbol{s}}^{\mathsf{r}}_+)^\top, (\hat{\boldsymbol{s}}^{\mathsf{i}}_+)^\top\right]^\top$. Therefore, we shall focus on the estimation of the complex symbols $\boldsymbol{s}_+$ in the following sections.

Remark. *This paper formulates misaligned OAC for FEEL considering a time-domain realization. OAC can also be realized in the frequency domain via OFDM. The connections and differences between the two realizations are discussed in the Conclusion section.*

III. Aligned and Misaligned OAC

A. Aligned OAC

Prior works on OAC, with the exception of [17], considered only the perfectly aligned case [10], [11], [13], [14], [20], where there is neither channel-gain misalignment nor time misalignment, which we refer to as the *aligned OAC*. In this case, we have $\alpha_m = 1/\tilde{h}_m$ and $\tau_m = 0$, $\forall m$, and the received signal is given by

$$r(t) = \sum_{\ell=1}^L \sum_{m=1}^M s_m[\ell] p(t-\ell T) + z(t). \tag{6}$$

Matched filtering $r(t)$ by the same rectangular pulse $p(t)$ and sampling at $t = iT$, $i = 1, 2, ..., L$, gives us

$$r[i] = \frac{1}{T}\int_{(i-1)T}^{iT} r(t)dt = \sum_{m=1}^M s_m[i] + z[i] = s_+[i] + z[i], \tag{7}$$

where the noise sequence $z[i]$ in the samples is independent and identically distributed (i.i.d.), $z[i] \sim \mathcal{CN}(0, \frac{N_0}{T})$.

As can be seen, the target signal $s_+[i]$ appears explicitly on the right hand side (RHS) of (7). In this context, the fading MAC degenerates to a Gaussian MAC and the M devices can be abstracted as a single device transmitting the arithmetic sum of the local model updates directly to the PS. In practice, however, the channel-gain precoding and the calibration of transmission timing can be inaccurate. With either channel-gain or time misalignment, clean samples as in (7) with $\boldsymbol{s}_+$ explicitly present are no longer available.

B. Misaligned OAC

With channel-gain and time misalignments, the received signal $r(t)$ is given in (5) and illustrated in Fig. 2. Let us first follow the standard signal processing flow in digital communications to process the received signal. Specifically, we first matched filter $r(t)$ by the rectangular pulse $p(t)$ and then oversample the matched filtered signal at $\{iT + \tau_k : i = 1, 2, ..., L; k = 1, 2, ..., M\}$ to collect sufficient statistics [21]. In so doing, the samples we get, denoted by $\{r_k[i] : k = 1, 2, ..., M; i = 1, 2, ..., L\}$, can be written as

$$\begin{aligned} r_k[i] &= \frac{1}{T}\int_{(i-1)T+\tau_k}^{iT+\tau_k} r(t) * p(t)\, dt \qquad (8)\\ &= \frac{1}{T}\sum_{m=1}^M \int_{(i-1)T+\tau_k}^{(i-\mathbb{1}_{m>k})T+\tau_m} h'_m s_m[i-\mathbb{1}_{m>k}]\, d\zeta + \frac{1}{T}\sum_{m=1}^M \\ &\quad \int_{(i-\mathbb{1}_{m>k})T+\tau_m}^{iT+\tau_k} h'_m s_m[i+\mathbb{1}_{m<k}]\, d\zeta + \frac{1}{T}\int_{(i-1)T+\tau_k}^{iT+\tau_k} z(\zeta)\, d\zeta \\ &\triangleq \sum_{m=1}^M c_{m,k}[i] s_m[i-\mathbb{1}_{m>k}] + \sum_{m=1}^M c'_{m,k}[i] s_m[i+\mathbb{1}_{m<k}] + z_k[i], \end{aligned}$$

where $\mathbb{1}$ is the indicator function and

$$\begin{aligned} c_{m,k}[i] &= \frac{h'_m}{T}[(1-\mathbb{1}_{m>k})T + \tau_m - \tau_k], \\ c'_{m,k}[i] &= \frac{h'_m}{T}[\mathbb{1}_{m>k}T + \tau_k - \tau_m]. \end{aligned}$$

As shown in Appendix A, when there is residual CFO and the channel is fast fading, the discrete samples can be written in the same form as (8) with the coefficients $c_{m,k}[i]$ and $c'_{m,k}[i]$ given in (30) and (31).

The noise sequence $\{z_k[i]\}$ in (8) is colored since

$$\begin{aligned} \mathbb{E}[z_k[i] z_{k'}[i']] &= \frac{1}{T^2}\int_{(i-1)T+\tau_k}^{iT+\tau_k} \int_{(i'-1)T+\tau_{k'}}^{i'T+\tau_{k'}} z(\zeta) z(\zeta')\, d\zeta\, d\zeta' \\ &= \begin{cases} \frac{N_0}{T^2}\lambda(i,i',k,k') & \text{if } \lambda(i,i',k,k') \in [0,T), \\ \frac{N_0}{T^2}[2T - \lambda(i,i',k,k')] & \text{if } \lambda(i,i',k,k') \in [T,2T), \\ 0 & \text{otherwise}, \end{cases} \qquad (9) \end{aligned}$$

where $\lambda(i,i',k,k') = (i'-i-1)T + \tau_{k'} - \tau_k$.

Given the samples $\{r_k[i]\}$ in (8), we now set out to estimate the desired arithmetic sum $\boldsymbol{s}_+$. First, the sequence of samples $y_k[i]$ can be written in a more compact form as

$$\boldsymbol{r} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{z}, \tag{10}$$

where the dimensionalities of $\boldsymbol{r}$, $\boldsymbol{s}$, and $\boldsymbol{z}$ are $ML \times 1$, and the dimensionality of matrix $\boldsymbol{A}$ is $ML \times ML$ (see the detailed form of (10) at the top of the next page). Denoting by Σ_z the covariance matrix of the noise sequence $\boldsymbol{z}$, each element of Σ_z can then be computed from (9).

$$\begin{bmatrix} r_1[1] \\ \cdots \\ r_M[1] \\ r_1[2] \\ \cdots \\ r_M[2] \\ \cdots \\ r_1[L] \\ \cdots \\ r_M[L] \end{bmatrix} = \begin{bmatrix} c_{1,1}[1] & c'_{2,1}[1] & c'_{3,1}[1] & \cdots & c'_{M,1}[1] & & & & & & & & & & \\ c_{1,2}[1] & c_{2,2}[1] & c'_{3,2}[1] & \cdots & c'_{M,2}[1] & c'_{1,2}[2] & & & & & & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & & & & & & & & \\ c_{1,M}[1] & c_{2,M}[1] & c_{3,M}[1] & \cdots & c_{M,M}[1] & c'_{1,M}[2] & c'_{2,M}[2] & \cdots & c'_{M-1,M}[2] & & & & & & \\ & c_{2,1}[1] & c_{3,1}[1] & \cdots & c_{M,1}[1] & c_{1,1}[2] & c'_{2,1}[2] & c'_{3,1}[2] & \cdots & c'_{M,1}[2] & & & & & \\ & & c_{3,2}[1] & \cdots & c_{M,2}[1] & c_{1,2}[2] & c_{2,2}[2] & c'_{3,2}[2] & \cdots & c'_{M,2}[2] & c'_{1,2}[3] & & & & \\ & & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & & & & \\ & & & & c_{1,M}[2] & c_{2,M}[2] & c_{3,M}[2] & \cdots & c_{M,M}[2] & c'_{1,M}[3] & c'_{2,M}[3] & \cdots & c'_{M-1,M}[3] & & \\ & & & & & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} s_1[1] \\ \cdots \\ s_M[1] \\ s_1[2] \\ \cdots \\ s_M[2] \\ \cdots \\ s_1[L] \\ \cdots \\ s_M[L] \end{bmatrix} + \begin{bmatrix} z_1[1] \\ \cdots \\ z_M[1] \\ z_1[2] \\ \cdots \\ z_M[2] \\ \cdots \\ z_1[L] \\ \cdots \\ z_M[L] \end{bmatrix}$$

The desired sequence $\boldsymbol{s}_+$, on the other hand, can be written as a linear transformation of the complex vector $\boldsymbol{s}$:

$$\boldsymbol{s}_+ = \boldsymbol{V}\boldsymbol{s}, \tag{11}$$

where the $L \times ML$ matrix $\boldsymbol{V}$ is given by

$$\boldsymbol{V} = \begin{bmatrix} \mathbf{1}_{1\times M} & & & \\ & \mathbf{1}_{1\times M} & & \\ & & \cdots & \\ & & & \mathbf{1}_{1\times M} \end{bmatrix},$$

in which $\mathbf{1}_{1\times M}$ represents a $1 \times M$ all-ones matrix.

Multiplying both sides of (10) by $\boldsymbol{V}\boldsymbol{A}^{-1}$ gives us

$$\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{r} = \boldsymbol{s}_+ + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{z}, \tag{12}$$

based on which an ML estimator can be devised, as in Definition 1.

Definition 1 (ML estimation for misaligned OAC)**.** *Given a sequence of samples $\boldsymbol{r} \in \mathcal{C}^{ML}$ in* (10)*, the ML estimate of sequence $\boldsymbol{s}_+ \in \mathcal{C}^L$ is*

$$\hat{\boldsymbol{s}}_+^{ml} = \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{r}. \tag{13}$$

Eq. (13) follows directly from (12) since the likelihood function of $\boldsymbol{s}_+$ is an L-dimensional Gaussian distribution. Specifically, given $\boldsymbol{r}$, the likelihood function of $\boldsymbol{s}_+$ is

$$f(\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{r}|\boldsymbol{s}_+) \propto \mathcal{CN}(\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{r}, \boldsymbol{V}\boldsymbol{A}^{-1}\Sigma_z\boldsymbol{A}^{-\boldsymbol{H}}\boldsymbol{V}^{\boldsymbol{H}}).$$

Differentiating $f(\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{r}|\boldsymbol{s}_+)$ with respect to $\boldsymbol{s}_+$ gives us the ML estimate $\hat{\boldsymbol{s}}_+^{\text{ml}}$ in (13).

An important implication of (13) is that, the maximum-likelihood $\boldsymbol{s}_+$ can be obtained by first finding the maximum-likelihood transmitted vector $\hat{\boldsymbol{s}}^{\text{ml}} = \boldsymbol{A}^{-1}\boldsymbol{r}$, and then performing the arithmetic sum $\hat{\boldsymbol{s}}_+^{\text{ml}} = \boldsymbol{V}\hat{\boldsymbol{s}}^{\text{ml}}$. Said in another way, the ML estimation of $\boldsymbol{s}_+$ boils down to multi-user estimation/detection (MUE) when we have misaligned channel gains at the receiver.

Remark. *The transmitted vector $\boldsymbol{s}$ carries the local weight-updates of a neural network. Therefore, $\boldsymbol{s}$ is by no means i.i.d. considering the strong correlations among the weights of the neural network. This also suggests that the prior information of $\boldsymbol{s}$ is hard to obtain and is unlikely to be known to the PS in advance. As a result, ML estimation is the only choice at the receiver.*

Remark. *The result that the ML estimation of $\boldsymbol{s}_+$ boils down to MUE is exclusive to the misaligned OAC system, wherein the transmitted symbols are continuous valued and the channel gains are misaligned. In digital communications, we do not have such result for ML estimation. The reason for this divergence is as follows. OAC is an analog communication system wherein the transmitted symbols $\boldsymbol{s}$ are continuous complex values. In contrast, the transmitted symbols in digital communications are discrete constellations. Whether the prior probability distribution of the transmitted symbols is available to the receiver or not, the constellation itself serves as a kind of prior information as the detection space is naturally narrowed down to the possible constellation points. As a result, when we perform ML estimation in digital communications, we inherently assume all the constellations are equiprobable. In that case, the likelihood function $f(\boldsymbol{r}|\boldsymbol{s})$ is a Gaussian mixture instead of Gaussian and the MUE-and-sum estimation is no longer optimal if we were to estimate the arithmetic sum $\boldsymbol{s}_+$.*

On the other hand, when we perform ML estimation in OAC, all the complex space is assumed to be equiprobable. The only information we have is the noise-contaminated sample and the likelihood function is a Gaussian centered around the noisy sample. Two implications about the ML estimation in OAC are thus 1) it faces an infinitely large estimation space; 2) it can be very susceptible to noise.

The ML estimator in (13) is not a practical estimator due to the prohibitive computational complexity of matrix inversion. To invert an n by n matrix, the best proven lower bound of the computational complexity is $\Omega(n^2 \log n)$ [22]. Notice that the dimensionality of $\boldsymbol{A}$ is ML by ML. Thus, the computational complexity of (13) is $\Omega(L^2M^2\log(LM))$. In practical OAC systems, M cannot be too large due to the saturation of the receiver (that is, the received signal power can exceed the dynamic range of the receiver if M is too large), but the packet length L can be extremely large. Let us fix M as a constant, the computational complexity of (13) is then $\Omega(L^2 \log L)$.

To address this problem and devise an ML estimator with acceptable computational complexity, we put forth in Section IV a factor-graph based ML estimator by exploiting the sparsity of the coefficient matrix. Compared with the ML estimator in (13), the computational complexity of the factor-graph based ML estimator is only $\Omega(L)$.

IV. A Sum-Product ML Estimator and The Aligned-sample Estimator

Before we dive deeper to dissect the inner structure of the coefficient matrix, let us first introduce a new matched filtering and sampling scheme that gives us oversampled but independent samples, which we refer to as the *whitened matched filtering and sampling* (WMFS). Two benefits of the WMFS scheme are 1) the independent samples obtained from the scheme allows us to construct a factor graph with a simple structure, based on which a low-complexity SP-ML estimator

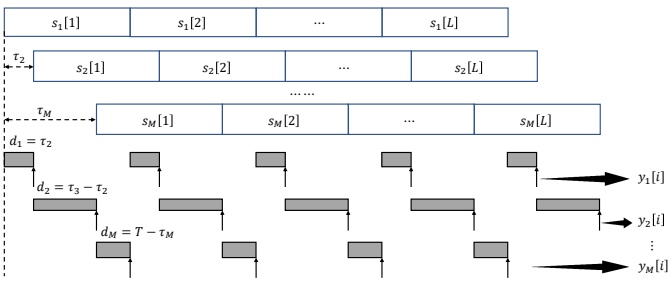

Figure 3. Matched filtering the received signal by a bank of M filters of lengths $d_k = \tau_{k+1} - \tau_k$.

can be devised; 2) the whitened scheme yields a subsequence of samples in which the indexes of symbols from different devices are consistent – in these samples, the symbols from different devices are aligned in time, as shown in (25). This admits an aligned-sample estimator for the misaligned OAC.

A. WMFS

The key idea of the WMFS scheme is to use a bank of matched filters of different lengths to collect power judiciously from $r(t)$. Specifically, instead of using the rectangular pulse $p(t)$ as the matched filter as in (8), we define M matched filters $\{p'_k(t) : k = 1, 2, ..., M\}$ as follows:

$$p'_k(t) = \frac{1}{2}\big[\text{sgn}(t+T) - \text{sgn}(t+T-d_k)\big], \tag{14}$$

where the length of the k-th matched filter is $d_k = \tau_{k+1} - \tau_k$, $k = 1, 2, ..., M$. For completeness, we define $\tau_{M+1} = T$.

The matched filtering and sampling processes are illustrated in Fig. 3. As shown, the signal filtered by the k-th matched filter is given by

$$y_k(t) = \frac{1}{d_k}\int_{-\infty}^{\infty} r(\zeta) p'_k(t-\zeta)\, d\zeta,$$

and we sample $y_k(t)$ at $(i-1)T + \tau_{k+1} : i = 1, 2, ..., L+1$, giving

$$\begin{aligned} y_k[i] &= y_k(t = (i-1)T + \tau_{k+1}) = \frac{1}{d_k}\int_{(i-1)T+\tau_k}^{(i-1)T+\tau_{k+1}} \\ &\sum_{m=1}^{M} h'_m s_m[i - \mathbb{1}_{m>k}]\, d\zeta + \frac{1}{d_k}\int_{(i-1)T+\tau_k}^{(i-1)T+\tau_{k+1}} z(\zeta)\, d\zeta \\ &\triangleq \sum_{m=1}^{M} h'_m s_m[i - \mathbb{1}_{m>k}] + \tilde{z}_k[i], \end{aligned} \tag{15}$$

where we have defined $s_m[0] = s_m[L+1] = 0$, $\forall m$, for completeness.

An important observation from (15) is that the noise term $\tilde{z}_k[i] \sim \mathcal{CN}(0, N_0/d_k)$ is independent for different k and i. This is because

$$\begin{aligned} &\mathbb{E}\left[\tilde{z}_k[i]\tilde{z}_{k'}[i']\right] \\ &= \mathbb{E}\left[\frac{1}{d_k}\int_{(i-1)T+\tau_k}^{(i-1)T+\tau_{k+1}} \tilde{z}(\zeta)\, d\zeta \frac{1}{d_{k'}}\int_{(i'-1)T+\tau_{k'}}^{(i'-1)T+\tau_{k'+1}} \tilde{z}(\zeta')\, d\zeta'\right] \\ &= \frac{1}{d_k d_{k'}}\int_{(i-1)T+\tau_k}^{(i-1)T+\tau_{k+1}}\int_{(i'-1)T+\tau_{k'}}^{(i'-1)T+\tau_{k'+1}} \mathbb{E}\left[\tilde{z}(\zeta)\tilde{z}(\zeta')\right]\, d\zeta d\zeta' \\ &= \frac{1}{d_k d_{k'}}\int_{(i-1)T+\tau_m}^{(i-1)T+\tau_{m+1}} N_0\delta\left((i-i')(k-k')\right)\, d\zeta \\ &= \frac{N_0}{d_k}\delta\left((i-i')(k-k')\right). \end{aligned} \tag{16}$$

Similarly to (10), we can rewrite (15) in a matrix form as

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{s} + \tilde{\boldsymbol{z}}, \tag{17}$$

where the sequence of transmitted symbols $\boldsymbol{s}$ is the same as that in (10). Unlike (10), the vectors $\boldsymbol{y}$ and $\tilde{\boldsymbol{z}}$ in (17) are $M(L+1)-1$ by 1 dimensional, giving

$$\begin{aligned} \boldsymbol{y} = \Big[&y_1[1], y_2[1], ..., y_M[1], y_1[2], y_2[2], ..., y_M[2], ..., \\ &y_1[L], y_2[L], ..., y_M[L], y_1[L+1], y_2[L+1], ..., y_{M-1}[L+1]\Big]^\top, \\ \tilde{\boldsymbol{z}} = \Big[&\tilde{z}_1[1], \tilde{z}_2[1], ..., \tilde{z}_M[1], \tilde{z}_1[2], \tilde{z}_2[2], ..., \tilde{z}_M[2], ..., \\ &\tilde{z}_1[L], \tilde{z}_2[L], ..., \tilde{z}_M[L], \tilde{z}_1[L+1], \tilde{z}_2[L+1], ..., \tilde{z}_{M-1}[L+1]\Big]^\top, \end{aligned}$$

and the coefficient matrix $\boldsymbol{D}$ is $M(L+1)-1$ by ML, giving

$$\boldsymbol{D} = \begin{bmatrix} h'_1 & & & & & & & \\ h'_1 & h'_2 & & & & & & \\ \dots & h'_2 & \dots & & & & & \\ h'_1 & \dots & \dots & h'_M & & & & \\ & h'_2 & \dots & h'_M & h'_1 & & & \\ & & \dots & \dots & h'_1 & h'_2 & & \\ & & & h'_M & \dots & h'_2 & \dots & \\ & & & & h'_1 & \dots & \dots & h'_M \\ & & & & & h'_2 & \dots & h'_M \\ & & & & & & \dots & \dots \\ & & & & & & & h'_M \end{bmatrix}. \tag{18}$$

When there is residual CFO and the channel is fast fading, matrix $\boldsymbol{D}$ can be written in the same form, as given in (33).

The desired sequence at the PS is $\boldsymbol{s}_+ = \boldsymbol{V}\boldsymbol{s}$, as in (11).

Eq. (16) validates that the noise sequence $\tilde{\boldsymbol{z}}$ is white, that is, (17) can be viewed as a whitened model of (10), and hence the name WMFS.

We emphasize that the signal model in (17) is equivalent to that in (10) since they are sampled from the same received signal $r(t)$ and no information is lost. More specifically, (17) can be transformed back to (10) after some elementary row operations and row deletions. Nevertheless, the model in (17) is more favorable than (10) thanks to the following factors:

1) The whitened noise. As will be shown later, white noise admits a sample-by-sample factorization of the likelihood function and a much simpler structure of the factor graph.
2) The alleviated inter-symbol and inter-user interferences. A sample $y_k[i]$ is related to only M complex symbols, each of which comes from a different device. In contrast, a sample $r_k[i]$ in (10) is related to $2M-1$ symbols.

Remark. *Since* (10) *and* (17) *are equivalent, we can also design the ML estimator from* (17)*. Denoting the covariance matrix of $\tilde{\boldsymbol{z}}$ by $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}$ (it is a diagonal matrix since $\tilde{z}$ is white), the ML estimator is given by*

$$\hat{\boldsymbol{s}}_+^{ml} = \boldsymbol{V}(\boldsymbol{D}^{\boldsymbol{H}}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^{-\mathbf{1}}\boldsymbol{D})^{-1}\boldsymbol{D}^{\boldsymbol{H}}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^{-\mathbf{1}}\boldsymbol{y}. \tag{19}$$

Again, the inversion of $\boldsymbol{D}^{\boldsymbol{H}}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^{-\mathbf{1}}\boldsymbol{D}$ is computationally demanding, as in (13)*.*

B. A Factor Graph Approach

A possible way to reduce the complexity of the ML estimation is to exploit the sparsity of the coefficient matrix $\boldsymbol{D}$. To this end, let us focus on the ML estimate of a single entry in the desired sequence $\boldsymbol{s}_+$, i.e., $\hat{s}_+^{\rm ml}[i]$.

Let $\boldsymbol{s[i]} = \big[s_1[i], s_2[i], ..., s_M[i]\big]^\top$. Given an observed sample sequence $\boldsymbol{y}$, we have

$$\begin{aligned}\hat{s}_+^{\rm ml}[i] &= \arg\max_{s_+[i]} f(\boldsymbol{y}|s_+[i]) \\ &= \arg\max_{s_+[i]} \int_{\boldsymbol{1}^\top \boldsymbol{s[i]}=s_+[i]} f(\boldsymbol{y}|\boldsymbol{s[i]})\, d\boldsymbol{s[i]}.\end{aligned}$$

In particular, $f(\boldsymbol{y}|\boldsymbol{s[i]})$ is a marginal function of $f(\boldsymbol{y}|\boldsymbol{s})$. Thus, to find the ML estimate $\hat{s}_+^{\rm ml}[i]$, a first step is to analyze $f(\boldsymbol{y}|\boldsymbol{s})$. In the following analysis, we will call $f(\boldsymbol{y}|\boldsymbol{s})$ the global likelihood function and $f(\boldsymbol{y}|\boldsymbol{s[i]})$ the marginal likelihood function.

In ML estimation, the transmitted symbols $\boldsymbol{s}$ are treated as constants. Randomness is only introduced by the noise sequence $\tilde{\boldsymbol{z}}$. Therefore, in the whitened model (17), the elements of $\boldsymbol{y}$ are independent of each other. We can then factorize the likelihood function $f(\boldsymbol{y}|\boldsymbol{s})$ as

$$f(\boldsymbol{y}|\boldsymbol{s}) \propto \prod_{k=1}^{M}\prod_{i=1}^{L+1} f(y_k[i]|\boldsymbol{s}) \overset{(a)}{=} \prod_{k=1}^{M}\prod_{i=1}^{L+1} f(y_k[i]|\mathcal{V}(y_k[i])), \tag{20}$$

where (a) follows because a sample $y_k[i]$ is related to only M complex symbols in $\boldsymbol{s}$. As per (15), we denote these symbols by $\mathcal{V}(y_k[i]) = \{s_1[i], s_2[i], ..., s_k[i], s_{k+1}[i-1], s_{k+2}[i-1], ..., s_M[i-1]\}$ and call them the *neighbor symbols* of $y_k[i]$. A sample $y_k[i]$ is then fully determined by the values of its neighbor symbols. Note that the number of non-zero symbols in $\mathcal{V}(y_k[i])$ is the number of non-zero elements in the corresponding row of $\boldsymbol{D}$, giving

$$|\mathcal{V}(y_k[i])| = \begin{cases} k, & \text{when } i = 1; \\ M, & \text{when } 1 \le i \le L; \\ M-k, & \text{when } i = L+1. \end{cases} \tag{21}$$

Based on the factorization in (20), $f(\boldsymbol{y}|\boldsymbol{s})$ can be depicted by a graphical model [23]–[25]. As shown in Fig. 4, we use a Forney-style factor graph [24] to represent the factorization. In particular, each edge in the graph corresponds to a variable in (20), i.e., a complex symbol $s_m[\ell]$, an observation $y_k[i]$, or a noise term $\tilde{z}_k[i]$. To simplify notations, we denote them by $s_{m,i}$, $y_{k,i}$, and $\tilde{z}_{k,i}$ in Fig. 4, respectively.

As can be seen, each sample $y_k[i]$ is related to a set of symbols $\mathcal{V}(y_k[i])$; each complex symbol $s_k[i]$, on the other hand, is related to M samples. Thus, we duplicate each symbol $s_{m,i}$ for M times and connect them to M consecutive samples – the equality constraint function "=" means that the values of the variables connecting to this function must be equal. The output degree of each equality constraint function is M, so is the input degree of each plus function "+" (except for the $M-1$ samples at both ends of the packet). The target symbols $\boldsymbol{s}_+$ are shown at the top of Fig. 4.

The marginal likelihood function $f(\boldsymbol{y}|\boldsymbol{s[i]})$ can be obtained from the global likelihood function $f(\boldsymbol{y}|\boldsymbol{s})$ by a marginalization process operated on the factor graph, which can be implemented efficiently via the sum-product (SP) algorithm. However, note in Fig. 4 that we have a loopy graph. Moreover, the girth of the graph (the length of the shortest loop) is 4. Such short loops prevent the sum-product algorithm from converging [26]. Even if they converge, the performance of the sum-product algorithm often degrades greatly, and the equilibrium posterior distribution is only an approximation of the true posterior distribution of the variables. Said in another way, the independence assumption of the extrinsic information passed along the edges no longer holds because the messages can circulate indefinitely around the loops.

To circumvent this problem, below we transform the loopy graph in Fig. 4 to a loop-free graph, at the expense of increasing the dimension of the variables [23]. For each sample $y_k[i]$, we cluster all its neighbor symbols (i.e., the edges/variables that connect to $y_k[i]$ in Fig. 4) and construct a new higher-dimensional variable. Let us denote the new high-dimensional variable by $\boldsymbol{W}_{k,i} = \mathcal{V}(y_k[i])$. As illustrated in Fig. 5, each sample $y_k[i]$ is now connected to a single high-dimensional variable $\boldsymbol{W}_{k,i}$ after clustering and the loops are removed.

After clustering, the new high-dimensional variables $\boldsymbol{W}_{k,i}$ are correlated with each other because they contain common symbols. For example, $\boldsymbol{W}_{1,1} = \{s_1[1]\}$, $\boldsymbol{W}_{2,1} = \{s_1[1], s_2[1]\}$, and $\boldsymbol{W}_{3,1} = \{s_1[1], s_2[1], s_3[1]\}$, the common symbol between $\boldsymbol{W}_{1,1}$ and $\boldsymbol{W}_{2,1}$ is $s_1[1]$, that between $\boldsymbol{W}_{1,1}$ and $\boldsymbol{W}_{3,1}$ is $s_1[1]$, and that between $\boldsymbol{W}_{2,1}$ and $\boldsymbol{W}_{3,1}$ are $s_1[1]$, $s_2[1]$. Therefore, we have to add constraints among the new high-dimensional variables to ensure that the values of the common symbols are consistent across different variables.

In Fig. 5, the compatibility function δ is added on the adjacent variables to represent the above constraints. Specifically, for the two adjacent variables $\boldsymbol{W}$ and $\boldsymbol{W}'$ connected to the same delta function, the compatibility function $\delta(\boldsymbol{W}, \boldsymbol{W}')$ is defined as

$$\delta(\boldsymbol{W}, \boldsymbol{W}') = \begin{cases} 1, \text{ if the values of all common symbols} \\ \quad \text{between } \boldsymbol{W} \text{ and } \boldsymbol{W}' \text{ are equal;} \\ 0, \text{ otherwise.} \end{cases}$$

That is, function δ is an on-off function that ensures that the messages passed from $\boldsymbol{W}$ to $\boldsymbol{W}'$ and from $\boldsymbol{W}'$ to $\boldsymbol{W}$ satisfy the constraint that the values of the common symbols between $\boldsymbol{W}$ and $\boldsymbol{W}'$ are equal.

Remark. *It is worth noting that adding compatibility functions between two adjacent variables is enough to depict all the constraints. For example, let us consider $\boldsymbol{W}_{1,1}$, $\boldsymbol{W}_{2,1}$ and $\boldsymbol{W}_{3,1}$. In Fig. 5, we only add compatibility functions $\delta(\boldsymbol{W}_{1,1}, \boldsymbol{W}_{2,1})$ and $\delta(\boldsymbol{W}_{2,1}, \boldsymbol{W}_{3,1})$. There is no need to add an extra compatibility function $\delta(\boldsymbol{W}_{1,1}, \boldsymbol{W}_{3,1})$ between $\boldsymbol{W}_{1,1}$, $\boldsymbol{W}_{3,1}$ although they do have a common symbol $s_1[1]$. This is because $\boldsymbol{W}_{1,1}$ and $\boldsymbol{W}_{3,1}$ are independent conditioned on the compatibility functions $\delta(\boldsymbol{W}_{1,1}, \boldsymbol{W}_{2,1})$ and $\delta(\boldsymbol{W}_{2,1}, \boldsymbol{W}_{3,1})$. More specifically, $\delta(\boldsymbol{W}_{1,1}, \boldsymbol{W}_{2,1})$ has ensured that the value of $s_1[1]$ in $\boldsymbol{W}_{1,1}$ equals that in $\boldsymbol{W}_{2,1}$ and $\delta(\boldsymbol{W}_{2,1}, \boldsymbol{W}_{3,1})$ has ensured that the values of $s_1[1]$ in $\boldsymbol{W}_{2,1}$ equals that in $\boldsymbol{W}_{3,1}$. Thus, the values of $s_1[1]$ in $\boldsymbol{W}_{1,1}$ and $\boldsymbol{W}_{3,1}$ must be equal.*

Overall, the factor graph in Fig. 5 presents a tree structure. Compared with Fig. 4, although the dimensions of the

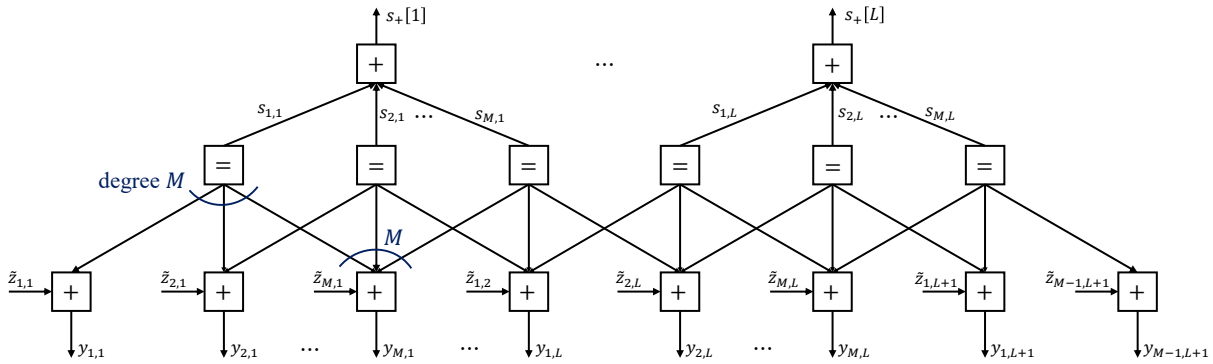

Figure 4. A graphical interpretation of the factorization in (20). To simplify notations, we denote $y_k[i]$, $z_k[i]$ and $s_m[i]$ by $y_{k,i}$, $z_{k,i}$ and $s_{m,i}$ in the figure, respectively.

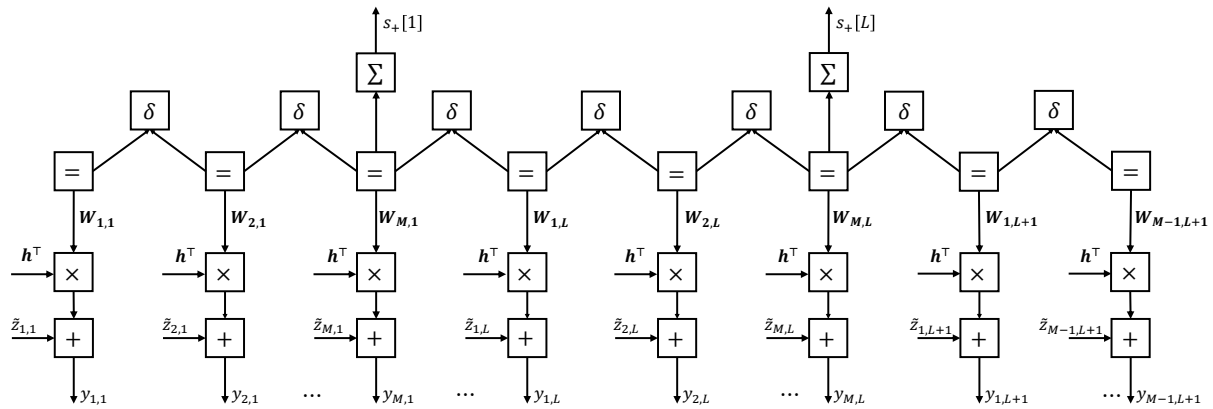

Figure 5. An equivalent tree structure to the loopy graph in Fig. 4. The variables connected to the same sample are clustered as a new high-dimensional variable. Each observation node is connected to a single variable node after clustering.

variables are M times larger, the marginal likelihood function $f(\boldsymbol{y}|\boldsymbol{s}[\boldsymbol{i}])$ can now be computed exactly via the sum-product algorithm thanks to the tree structure.

C. Analog Message Passing and the SP-ML Estimator

In standard sum-product algorithms, the messages passed on the edges are the probability mass function (PMF) of the variables associated with the edges, i.e., a probability vector of finite length [23], [25]. This is because the transmitted symbols in digital communications are chosen from finite-size constellations. However, in the case of OAC, the transmitted symbols $\boldsymbol{s_m}$ are continuous complex numbers. Hence, the messages passed on the edges should be the probability density functions (PDFs) of the associated variables, which is a continuous function rather than a finite-length vector.

To enable message passing, a straightforward idea is to quantize the PDFs so that we can employ the digital message passing. The output of the SP algorithm is then the marginal PDF in a quantized form. However, quantization suffers from the "curse of dimensionality" – when the dimension of the variables increases, the volume of the space increases exponentially fast. In order to get a sound result, a significantly larger number of quantization levels is required in each dimension compared with the low-dimensional case [27]. In our problem, we have deliberately increased the dimensionality of the variables to remove the loops in Fig. 4. Thus, the standard sum-product algorithm with quantization cannot be used considering its prohibitive complexity.

Now that quantization is not an option, the problem is how to pass continuous PDFs along the edges of Fig. 5. A natural idea is then to parameterize the PDFs and pass their parameters [24], [28]. In particular, we point out an important results from our companion paper [29]: *if all inputs to a graph are Gaussian, then all the variables on the graph are multivariate Gaussian random variables.* In Fig. 5, the only input to the factor graph is the likelihood function $f(y_{k,i}|\boldsymbol{W}_{k,i})$. We next show that this message is a multivariate Gaussian distribution with respect to $\boldsymbol{W_{k,i}}$.

Let us consider a complex random variable as a pair of real random variables: the real part and the imaginary part. Then, $\boldsymbol{W_{k,i}}$ can be viewed as a $2M$-dimensional real random variable and denoted by $\boldsymbol{w_{k,i}} = \left(b_1^{\mathrm{r}}, b_2^{\mathrm{r}}, ..., b_M^{\mathrm{r}}, b_1^{\mathrm{i}}, b_2^{\mathrm{i}}, ..., b_M^{\mathrm{i}}\right)$.

As per (15), we have

$$\begin{aligned} y_{k,i} &= \sum_{k=1}^{M} (h_k^{\mathrm{r}} + jh_k^{\mathrm{i}})(b_k^{\mathrm{r}} + jb_k^{\mathrm{i}}) + z_{k,i}^{\mathrm{r}} + jz_{k,i}^{\mathrm{i}} \\ &= \sum_{k=1}^{M} [(h_k^{\mathrm{r}} b_k^{\mathrm{r}} - h_k^{\mathrm{i}} b_k^{\mathrm{i}} + z_{k,i}^{\mathrm{r}}) + j(h_k^{\mathrm{r}} b_k^{\mathrm{i}} + h_k^{\mathrm{i}} b_k^{\mathrm{r}} + z_{k,i}^{\mathrm{i}})], \end{aligned}$$

where $z^{\mathfrak{r}}_{k,i}$, $z^{\mathfrak{i}}_{k,i} \sim \mathcal{N}(0, \frac{N_0}{2d_k})$. Thus, the likelihood function

$$f(y_{k,i}|\boldsymbol{w_{k,i}}) \propto \exp\left\{-\frac{d_k}{N_0}[y^{\mathfrak{r}}_{k,i} - \sum_k (h^{\mathfrak{r}}_k b^{\mathfrak{r}}_k - h^{\mathfrak{i}}_k b^{\mathfrak{i}}_k)]^2\right\} \times \exp\left\{-\frac{d_k}{N_0}[y^{\mathfrak{i}}_{k,i} - \sum_k (h^{\mathfrak{r}}_k b^{\mathfrak{i}}_k + h^{\mathfrak{i}}_k b^{\mathfrak{r}}_k)]^2\right\}$$
$$\propto \mathcal{N}\left(\frac{2d_k}{N_0}\begin{bmatrix}\boldsymbol{\beta_1}\\ \boldsymbol{\beta_2}\end{bmatrix}\begin{bmatrix}y^{\mathfrak{r}}_{k,i}\\ y^{\mathfrak{i}}_{k,i}\end{bmatrix}, \frac{2d_k}{N_0}\begin{bmatrix}\boldsymbol{\beta_1}\boldsymbol{\beta_1}^\top & \boldsymbol{\beta_1}\boldsymbol{\beta_2}^\top\\ \boldsymbol{\beta_2}\boldsymbol{\beta_1}^\top & \boldsymbol{\beta_1}\boldsymbol{\beta_1}^\top\end{bmatrix}\right),$$

where $\boldsymbol{\beta_1}$ and $\boldsymbol{\beta_1}$ are composed of channel coefficients, giving

$$\boldsymbol{\beta_1} = \begin{bmatrix} h^{\mathfrak{r}}_1 & h^{\mathfrak{i}}_1\\ h^{\mathfrak{r}}_2 & h^{\mathfrak{i}}_2\\ \dots & \dots\\ h^{\mathfrak{r}}_M & h^{\mathfrak{i}}_M\end{bmatrix}, \quad \boldsymbol{\beta_2} = \begin{bmatrix} -h^{\mathfrak{i}}_1 & h^{\mathfrak{r}}_1\\ -h^{\mathfrak{i}}_2 & h^{\mathfrak{r}}_2\\ \dots & \dots\\ -h^{\mathfrak{i}}_M & h^{\mathfrak{r}}_M\end{bmatrix}.$$

Since the likelihood functions $f(y_{k,i}|\boldsymbol{W}_{k,i})$, $\forall k, i$ are Gaussian, all the messages passed on Fig. 5 are multivariate Gaussian distributions. The Gaussian PDF can then be parameterized by its mean vector and covariance matrix – passing these parameters is equivalent to passing the continuous PDF on the edges.

To ease reading, we summarize the main results of analog message passing in the following. The detailed proof can be found in Appendix B of our companion paper [29].

1) The marginal likelihood function $f(\boldsymbol{y}|\boldsymbol{s[i]})$ is a multivariate complex Gaussian distribution of dimension M. The likelihood function $f(\boldsymbol{y}|s_+[i])$ is a single-variate complex Gaussian distribution.
2) A complex random variable can be viewed as a pair of real random variables (i.e., the real part and the imaginary part of the complex random variable). Thus, we can denote the M-dimensional complex Gaussian $f(\boldsymbol{y}|\boldsymbol{s[i]})$ by a $2M$-dimensional real Gaussian
$$f(\boldsymbol{y}|\boldsymbol{s[i]}) \sim \mathcal{N}\left(\boldsymbol{s[i]}, \boldsymbol{\mu_{s[i]}} = \begin{bmatrix}\boldsymbol{\mu^{\mathfrak{r}}_{s[i]}}\\ \boldsymbol{\mu^{\mathfrak{i}}_{s[i]}}\end{bmatrix}, \boldsymbol{\Sigma_{s[i]}} = \begin{bmatrix}\boldsymbol{\Sigma^{\mathfrak{rr}}_{s[i]}} & \boldsymbol{\Sigma^{\mathfrak{ri}}_{s[i]}}\\ \boldsymbol{\Sigma^{\mathfrak{ir}}_{s[i]}} & \boldsymbol{\Sigma^{\mathfrak{ii}}_{s[i]}}\end{bmatrix}\right), \tag{22}$$
where $\boldsymbol{\mu_{s[i]}}$ is a $2M$ by 1 real vector consisting of the real and imaginary parts of the mean of $\boldsymbol{s[i]}$. That is, $\boldsymbol{\mu^{\mathfrak{r}}_{s[i]}}$ and $\boldsymbol{\mu^{\mathfrak{i}}_{s[i]}}$ are the real and imaginary parts of sequence $\mathbb{E}[\boldsymbol{s[i]}]^\top$. The covariance matrix $\boldsymbol{\Sigma_{s[i]}}$ is a $2M$ by $2M$ covariance matrix. The moment parameters $(\boldsymbol{\mu_{s[i]}}, \boldsymbol{\Sigma_{s[i]}})$ can be computed by analog message passing.
3) Given the marginal likelihood function $f(\boldsymbol{y}|\boldsymbol{s[i]})$ in (22), $f(\boldsymbol{y}|s_+[i])$ can be constructed by
$$f(\boldsymbol{y}|s_+[i]) \sim \mathcal{N}\left(s_+[i], \boldsymbol{\mu}_{s_+[i]} = \begin{bmatrix}\mu^{\mathfrak{r}}_{s_+[i]}\\ \mu^{\mathfrak{i}}_{s_+[i]}\end{bmatrix}, \boldsymbol{\Sigma}_{s_+[i]} = \begin{bmatrix}\Sigma^{\mathfrak{rr}}_{s_+[i]} & \Sigma^{\mathfrak{ri}}_{s_+[i]}\\ \Sigma^{\mathfrak{ir}}_{s_+[i]} & \Sigma^{\mathfrak{ii}}_{s_+[i]}\end{bmatrix}\right), \tag{23}$$
where
$$\mu^{\mathfrak{r}}_{s_+[i]} = \mathbf{1}^\top \boldsymbol{\mu^{\mathfrak{r}}_{s[i]}}, \quad \mu^{\mathfrak{i}}_{s_+[i]} = \mathbf{1}^\top \boldsymbol{\mu^{\mathfrak{i}}_{s[i]}},$$
$$\Sigma^{\mathfrak{rr}}_{s_+[i]} = \mathbf{1}^\top \boldsymbol{\Sigma^{\mathfrak{rr}}_{s[i]}} \mathbf{1}, \quad \Sigma^{\mathfrak{ri}}_{s_+[i]} = \mathbf{1}^\top \boldsymbol{\Sigma^{\mathfrak{ri}}_{s[i]}} \mathbf{1},$$
$$\Sigma^{\mathfrak{ir}}_{s_+[i]} = \mathbf{1}^\top \boldsymbol{\Sigma^{\mathfrak{ir}}_{s[i]}} \mathbf{1}, \quad \Sigma^{\mathfrak{ii}}_{s_+[i]} = \mathbf{1}^\top \boldsymbol{\Sigma^{\mathfrak{ii}}_{s[i]}} \mathbf{1}.$$
4) Following (23), an SP-ML estimator can be designed, as given in Definition 2.

Definition 2 (SP-ML estimation)**.** *The WMFS scheme gives us the whitened samples $\boldsymbol{y}$ in* (17)*. To estimate the designed sequence $\boldsymbol{s}_+$, an SP-ML estimator first computes the moment parameters of the multivariate Gaussian $f(\boldsymbol{y}|\boldsymbol{s[i]})$, $\forall i$, by an analog sum-product process, and then estimates each element of $\boldsymbol{s}_+$ by*

$$\hat{s}^{ml}_+[i] = \mathbf{1}^\top \boldsymbol{\mu^{\mathfrak{r}}_{s[i]}} + j\mathbf{1}^\top \boldsymbol{\mu^{\mathfrak{i}}_{s[i]}}, \tag{24}$$

where $\boldsymbol{\mu_{s[i]}} = [\boldsymbol{\mu^{\mathfrak{r}}_{s[i]}}, \boldsymbol{\mu^{\mathfrak{i}}_{s[i]}}]^\top$ is the mean of $f(\boldsymbol{y}|\boldsymbol{s[i]})$.

The reason behind (24) is as follows. It has been shown that $f(\boldsymbol{y}|s_+[i])$ is conditionally Gaussian. As per the ML rule, we should choose the mean of $f(\boldsymbol{y}|s_+[i])$ as the estimate of $s_+[i]$ as it maximizes the likelihood function. This gives us (24).

Remark (maximum likelihood sequence estimation (MLSE) versus Bahl-Cocke-Jelinek-Raviv (BCJR))**.** *The ML estimators in* (13) *and* (19) *aim to find the ML sequence $\boldsymbol{s}_+$ in the space $\mathbb{C}^L$. In the language of digital communications, they are ML optimal in the sense of MLSE [30]. On the other hand, the ML estimator in* (24) *aims to maximize the likelihood function of each element of $\boldsymbol{s}_+$. Thus, it is ML optimal in the BCJR [31] sense.*

When we perform ML estimation in digital communications, MLSE-optimal and BCJR-optimal are different criteria because the former minimizes the block error rate (BLER) while the latter minimizes the bit error rate (BER). For the ML estimation in OAC, however, the two criteria are equivalent. The reason for this discrepancy is again that the discrete constellations used in digital communications serve as a kind of prior information to the receiver while an OAC receiver has no prior information at all.

More specifically, let us consider the message passing in Fig. 5. For the ML estimation in OAC, we have shown that all the messages passed on the graph, including $f(\boldsymbol{y}|\boldsymbol{s}_+)$ and $f(\boldsymbol{y}|s_+[i])$, are Gaussian. Thus, the ML sequence $\boldsymbol{s}_+$ also gives us the ML $s_+[i]$, $\forall i$, after marginalization. As a result, MLSE-optimal and BCJR-optimal are equivalent, and the ML estimators in (13)*,* (19)*, and* (24) *are identical. In contrast, when we perform ML estimation in digital communications, the prior information imposes each transmitted symbol to belong to a finite constellation. As a consequence, the messages passed on the graph are Gaussian mixtures, and neither $f(\boldsymbol{y}|\boldsymbol{s}_+)$ nor $f(\boldsymbol{y}|s_+[i])$ is Gaussian distributed. As a result, MLSE-optimal and BCJR-optimal are different criteria.*

Computational complexity – Finally, we evaluate the computational complexity of the SP-ML estimator. With analog message passing, messages passed on the graph are simply the parameters of the Gaussian distributions instead of continuous Gaussian PDFs. Computations involved in analog message passing are simply 1) the sum of 2M-dimensional vectors/matrices, and 2) 2M-dimensional matrix inversion. Therefore, the computational complexity of the SP-ML estimator is $\Omega(LM^2 \log M)$. If we fix M as a constant, the SP-ML estimator significantly reduces the computational complexity of ML estimation from $\Omega(L^2 \log L)$ to $\Omega(L)$.

D. Aligned-Sample Estimator

As stated in the beginning of this section, another benefit of the WMFS scheme is that it yields a sequence of samples wherein the indexes of symbols from different devices are consistent. Specifically, let us consider the outputs of the M-th matched filter.

Let $k = M$ in (15), we have $\mathcal{V}(y_M[i]) = \{s_1[i], s_2[i], ..., s_M[i]\}$, and

$$y_M[i] = \sum_{m=1}^{M} h'_M[i] s_m[i] + z_M[i], \tag{25}$$

where $z_M[i] \sim \mathcal{CN}(0, N_0/d_M)$ and d_M is the duration of the M-th matched filter. As can be seen, unlike the outputs of other matched filters, the neighbor symbols of $y_M[i]$ have the same index (that is, the symbol indexes are aligned within the integral interval of the M-th matched filter). Therefore, we can utilize the outputs of the M-th matched filter to devise an aligned-sample estimator.

Definition 3 (Aligned-sample estimator for misaligned OAC)**.** *Given the outputs of the WMFS $\{y_k[i]\}$, aligned-sample estimator estimates the desired sequence $\boldsymbol{s}_+ \in \mathcal{C}^L$ symbol-by-symbol by*

$$\hat{s}_+[i] = y_M[i]. \tag{26}$$

Eq. (25) is an underdetermined equation since we have one equation for M unknowns, and the estimator in (26) is our best prediction about $\hat{s}_+[i]$. When there is no or mild channel-gain misalignment (i.e., $h'_M \to 1$), the estimator (26) is supposed to perform very well.

V. Simulation Results

This section evaluates the system performance of FEEL with misaligned OAC considering two estimators at the receiver: the ML estimator and the aligned-sample estimator. In particular, for ML estimation, we use our SP-ML estimator since the ML estimators in (13) and (19) are computationally prohibitive.[3] We implement a FEEL system wherein 40 devices collaboratively train a convolution neural network (CNN) to solve the CIFAR-10 classification task [32]. The CIFAR-10 dataset has a training set of $50,000$ examples and a test set of $10,000$ examples in 10 classes. Each example is a 32×32 colour image. The non-i.i.d. training examples are assigned to the 40 devices in the following manner: 1) we first let each device randomly sample $1,000$ samples from the dataset; 2) for the remaining $10,000$ examples, we sort them by their labels and group into 40 shards of size 250 [6]. Each device is then assigned one shard.

The implemented CNN is a ShuffleNet V2 network [33] with $d = 1.26 \times 10^6$ parameters (this corresponds to 6.32×10^5 complex values). In each iteration, we assume $M = 4$ devices are active and participate in the training. Each device will train the global model locally for 5 epochs and then transmit the model-update to the PS in 44 packets (the packet length is $L = 1.44 \times 10^4$) in each iteration. The packets from different transmitters overlap at the PS with time and channel-gain misalignments, and the PS employs the ML and aligned-sample estimators to estimate the arithmetic sum of transmitted symbols, i.e., $\boldsymbol{s}_+$ (and hence, $\boldsymbol{\theta}_+$). The estimated arithmetic sum $\boldsymbol{\theta}_+$ is then used to update the global model, as per (2). All the source codes are available online [34].

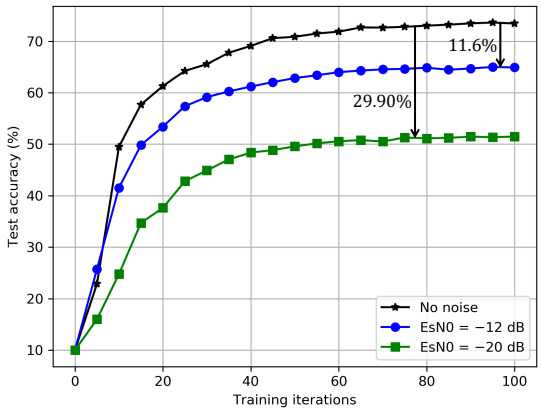

Figure 6. Test accuracy of the learned model over the course of training, with and without noise. There is no time or channel-gain misalignment. We use the ML estimator at the PS.

The metric we use to assess the performance of an estimator is the *test accuracy*. Specifically, when operated with a given estimator, we will train the global model for 100 iterations and take the prediction accuracy of the learned model on the test set as the performance indicator of the estimator. An example is given in Fig. 6.

As shown in Fig. 6, we run the FEEL system for 100 iterations with and without noise, and plot the test accuracy over the course of training. There are no misalignments in this simulation and ML estimator is used at the PS. First, the dark curve corresponds to the noiseless case and the test accuracy after 100 iterations is 73.43%. We point out that this is the global optimal test accuracy since the MAC is ideal, i.e., there is no time misalignment, channel-gain misalignment, or noise in the MAC. The other two curves in Fig. 6 correspond to the learning performance when noise is presented in the received signal. In particular, noise is added according to a given EsN0, i.e., the average received energy per symbol to noise power spectral density ratio, defined as

$$\text{EsN0} = \frac{\mathbb{E}_i \left[\left| \sum_{m=1}^{M} h'_m s_m[i] \right|^2 \right]}{N_0}. \tag{27}$$

As shown, noise is detrimental to the test accuracy after convergence. Compared with the noiseless case, the test accuracy drops by 11.6% with an EsN0 of -12 dB, and by 29.9% with an EsN0 of -20 dB.

In addition to noise, we next introduce time misalignment into the received signal. The received signal is given in (15), where the noise term is $z[i] \sim \mathcal{CN}(0, N_0/d_k)$, and we set the residual channel gain to $h'_m = 1$, $\forall\ m$. Without loss of generality, symbol duration is set to $T = 1$. As can be seen from (15), time offsets $\{\tau_m : m = 1, 2, ..., M\}$ determine the

[3] We have performed additional simulations to validate that the three estimators in (13), (19), and (24) are equivalent (using a much shorter packet length L). The simulation results are omitted here to conserve space.

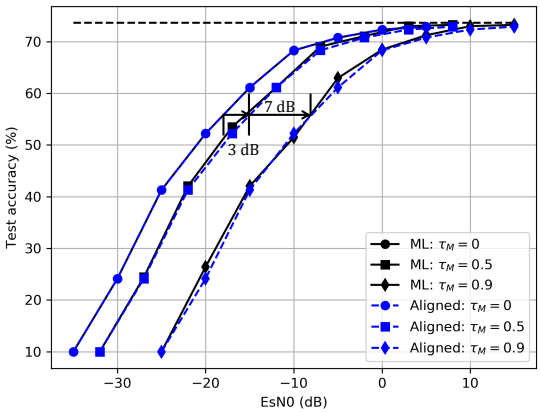

Figure 7. Test accuracy (after 100 iterations) of the asynchronous OAC under various EsN0. There is no channel-gain misalignment in the simulation and we use both the ML estimator and the aligned-sample estimator.

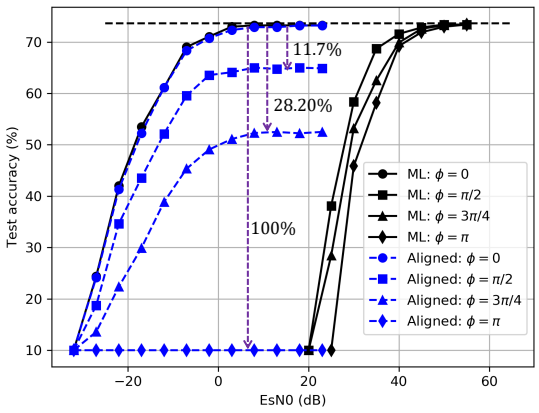

Figure 8. Test accuracies of the ML and aligned-sample estimators under different degrees of phase misalignments. The maximum time offset $\tau_M = 0.5$ and the maximum phase misalignment $\phi = 0$ (no), $\pi/2$ (mild), $3\pi/4$ (moderate), and π (severe), respectively.

noise power of samples from different matched filters (since $d_k = \tau_{k+1} - \tau_k$). In the simulation, the time offsets τ_m, $\forall\ m$, are set in the following manner: first, we fix a maximum time offset τ_M (and hence d_M); then, we generate the other time offsets uniformly in $(0, \tau_M)$.

The simulation results are presented in Fig. 7. When $\tau_M = 0$, there is neither time nor channel-gain misalignment in the received signal. The ML estimator and the aligned-sample estimator are equivalent in this case, and yield the same test accuracy. As we increase τ_M, the performance of both estimators deteriorate.

1) For the aligned-sample estimator, the performance deterioration is easy to understand since the inputs to the estimator are the outputs of the M-th matched filter $y_M[i]$. As a result, the performance of the aligned-sample estimator is governed by the maximum time offset τ_M – the larger the τ_M, the worse the performance. As shown in Fig. 7, the introduction of time misalignment results in an EsN0 penalty for the aligned-sample estimator. When $\tau_M = 0.5$, the EsN0 penalty is 3 dB because d_M is reduced by a factor of 2 (from 1 to 0.5). Likewise, the EsN0 penalty is 10 dB when $\tau_M = 0.9$ since d_M is reduced by 10 times (from 1 to 0.1).
2) For different τ_M, the performance gain of the ML estimator over the aligned-sample estimator is negligible. The aligned-sample estimator utilizes only the outputs of the M-th matched filter. The ML estimator, on the other hand, utilizes the outputs of all matched filters and attempts to estimate the mostly likely $\boldsymbol{s}_+$. It turns out that both estimators yield nearly the same performance when there is only time misalignment and noise.

It should be noted that the above result does not imply that the samples of the matched filters other than the M-th one are useless, because the ML estimator cannot achieve the same performance as the aligned-sample estimator using only the outputs of the M-th matched filter: recall from (26) that the outputs of the M-th matched filter results in a set of underdetermined equations. If we perform ML estimation based on the samples in (26), the estimation error can be arbitrarily large as $s_+[i]$ can be any value.

Remark (Error propagation)**.** *In the misaligned OAC, ML estimation boils down to MUE and faces an infinitely large estimation space. It is then very susceptible to noise and suffers from error propagation. Take the SP-ML estimator for instance. In the forward message passing, the successful estimation of a likelihood function (of a multivariate variable) hinges on the accurate estimations of the likelihood functions on the left. When a sample is contaminated by noise, the mean of the likelihood function deviates from the true value of the noiseless sample. This estimation error will be propagated along the tree all the way to the rightmost leaf, because there are no known messages (i.e., prior information) in between to alleviate/correct the error. This can be one cause of the results in Fig. 7.*

In the third simulation, let us further introduce channel-gain misalignment into the received signal (15). For each device, the residual channel gain is $h_m = |h_m|e^{j\phi_m}$. We set $|h_m| = 1$, $\forall m$, and focus on the impact of the phase offsets ϕ_m only. In particular, we assume $\{\phi_m : m = 1, 2, ..., M\}$ are uniformly distributed in $(0, \phi)$, where ϕ is the maximum phase offset (i.e., $\phi_m \sim U(0, \phi)$). It is worth noting that ϕ_m can be any distribution in general.

Fig. 8 presents the test accuracy of the ML and aligned-sample estimators versus EsN0 (in dB), wherein the maximum time offset τ_M is fixed to 0.5 and the maximum phase offset $\phi = 0$ (no phase misalignment), $\pi/2$ (mild), $3\pi/4$ (moderate), and π (severe), respectively.

Remark. *A caveat here is that ϕ is the maximum phase offset – the phase offsets of all devices are uniformly distributed in $[0, \phi]$. If we look at the phase misalignment between any two devices, however, the average pairwise-phase-misalignment is only $\phi/3$. That is why we classify $\pi/2$ as mild phase misalignment because the average pairwise-phase-misalignment is only $\pi/6$.*

We have the following observations from Fig. 8:

1) When there is no phase misalignment ($\phi = 0$), the two curves coincide just as in Fig. 7.
2) When there is mild phase misalignment ($\phi = \pi/2$), the

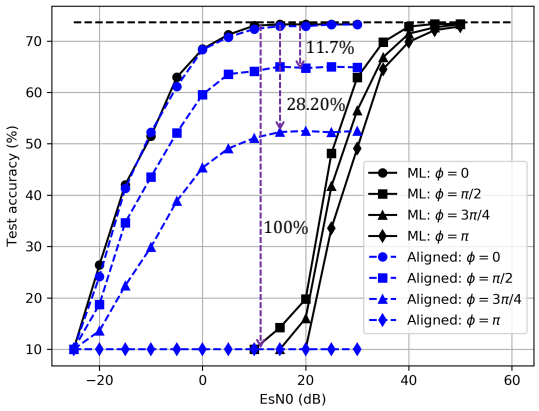

Figure 9. Test accuracies of the ML and aligned-sample estimators under different degrees of phase misalignments. The maximum time offset $\tau = 0.9$ and the maximum phase misalignment $\phi = 0$ (no), $\pi/2$ (mild), $3\pi/4$ (moderate), and π (severe), respectively.

aligned-sample estimator suffers from two penalties: i) a small EsN0 penalty for about 5 dB, i.e., we need a 5 dB higher EsN0 to achieve the same test accuracy; ii) a 11.7% test-accuracy loss, i.e., the test accuracy after convergence is 11.7% less than the phase-aligned case.

The ML estimator, on the other hand, suffers from a large EsN0 penalty. The reason behind is that the phase misalignment enhances the error/noise propagation in ML estimation, which we refer to as *noise enhancement*. As a result, ML estimation does not work in the low-EsN0 regime when there is phase misalignment. On the bright side, ML estimator performs very well in the high-EsN0 regime: it suffers from no test-accuracy loss – the test accuracy after convergence is the same as the phase-aligned case.

3) When we further increase the maximum phase misalignment ϕ, the aligned-sample estimator suffers from larger EsN0 and test-accuracy penalties. In the case of moderate phase misalignment ($\phi = 3\pi/4$), the test-accuracy penalty is up to 28.2%. In the case of severe phase misalignment ($\phi = \pi$), the learning diverges with the aligned-sample estimator.

   In contrast, the ML estimator is more robust to moderate and severe phase misalignments in the high-EsN0 regime – there is no test-accuracy loss and only a small EsN0 penalty.

Fig. 8 studies the impact of phase misalignment under a fixed maximum time offset $\tau_M = 0.5$. Next, we consider a larger time offset $\tau_M = 0.9$ and repeat the simulations in Fig. 8. The simulation results are presented in Fig. 9.

For the aligned-sample estimator, time offset only incurs an EsN0 penalty. Thus, each performance curve of the aligned-sample estimator in Fig. 9 is simply a right-shift of the corresponding curve in Fig. 8 by 7 dB.

On the other hand, we observe that ML estimation benefits from larger time misalignment when there is phase misalignment. For example, with a mild phase misalignment ($\phi = \pi/2$), the EsN0 gain is about 2 dB when the maximum time offset τ_M is increased from 0.5 to 0.9, as shown in Fig. 8 and Fig. 9. In contrast, when there is no phase misalignment, ML estimation suffers from larger time misalignment, as shown in Fig. 7.

To conclude this section, we summarize the main simulation results as follows.

1) When there is no phase misalignment, the ML and aligned-sample estimators are on equal footing as far as the learning performance is concerned.
2) When there is mild or moderate phase misalignment, the aligned-sample estimator outperforms the ML estimator in the low-EsN0 regime, but is worse than the ML estimator in the high-EsN0 regime.
3) When there is severe phase misalignment, the aligned estimator leads to divergence of learning, but ML estimation still works in the high-EsN0 regime.

VI. Time-domain Realization versus Frequency-domain Realization

In this paper, we considered a time-domain realization of OAC. OAC can also be realized in the frequency domain via OFDM. An interesting direction to extend the current work is to compare the two realizations and their abilities to combat misalignments. On the other hand, our study in this paper also sheds light on the frequency-domain realization of OAC. A brief comparison between the two realizations is given below to provide some operational insights.

Time-domain or frequency-domain realization of wireless communication systems has been a long-standing debate. When it concerns misalignments, the time-domain realization is sensitive to time offsets among edge devices, while the frequency-domain realization is sensitive to the carrier frequency offsets (CFOs) among edge devices.

1) Time misalignment. With time-domain realization, time misalignment leads to an EsN0 penalty to the learning performance, as verified in Section V. OFDM, on the other hand, deliberately introduces redundancies known as the cyclic prefix (CP) and transforms the time offset of each device τ_m to the frequency domain as extra phase offsets $e^{j2\pi\ell\tau_m/LT}$, $\ell = 1, 2, ..., L$, on the L subcarriers. If we look at one subcarrier, it is equivalently a synchronous time-domain realization with phase misalignments – the frequency domain samples are the same as (25) by setting $h'_m = e^{j2\pi\ell\tau_m/LT}$ and $d_M = T$. It is then an underdetermined equation and we can use the aligned-sample estimator in Definition 3 to estimate the arithmetic sum. We emphasize that the phase misalignment can be severe for large τ_M (maximum time offset) and ℓ (subcarrier index). For example, when $\tau_M = T$ and $\ell = L$, the phase misalignment is up to 2π.

   When there is no CFO, the performance of the OFDM system for the misaligned OAC can be predicted by the performance of the aligned-sample estimator in Fig. 8 after left-shifting for 3 dB (i.e., let $\tau_M = 0$).

   To summarize, OFDM systems introduce redundancies to transform time misalignment to phase misalignment. It is equivalent to trading off test-accuracy loss for EsN0 loss. It is worth noting that we can also insert some redundancies,

e.g., pilots, in the time-domain realizations to improve the performance of ML estimation

2) CFO. For the time-domain realization considered in this paper, residual CFO in the overlapped signal simply introduces additional channel-gain misalignments among devices. For a frequency-domain realization, however, CFO introduces inter-carrier interference (ICI), a dual problem of the inter-symbol and inter-user interferences in the time-domain realization of OAC. As a result, we have to devise an ML estimator to combat ICI, and ML estimation in OFDM with residual CFO falls into the same scope of ML estimation in time-domain realization with time misalignment. In this context, the analysis in this paper about the properties of ML estimation for the misaligned OAC still holds, and the SP-ML estimator devised in Section IV can also be used in OFDM systems to perform ML estimation.

VII. Conclusion

As a joint computation-and-communication technique, OAC exploits the property of the MAC that its output is the arithmetic sum of the inputs. OAC is an efficient scheme to speed up the uplink aggregation of models from the edge devices in FEEL. This paper filled the research gap of the misaligned OAC by devising two estimators, an SP-ML estimator and an aligned-sample estimator, to estimate the arithmetic sum of the symbols from different devices in the face of channel-gain and time misalignments. The underpinning of the proposed estimators is an oversampled matched filtering and sampling scheme that yields: a) whitened samples with alleviated inter-symbol and inter-user interferences; b) a subsequence of samples wherein the indexes of transmitted symbols from different devices are aligned.

The ML estimator – The whitened samples in a) allows us to construct a factor graph with a simple structure to represent the compositions of the samples, whereby an SP-ML estimator can be devised to compute the ML estimate of the arithmetic sum from an analog message passing process. Compared with conventional ML estimator that is computationally prohibitive, the complexity of the SP-ML estimator grows linearly with the packet length, and hence, is computationally more efficient.

In the OAC system, symbols transmitted from the edge devices are continuous values instead of discrete constellations. Therefore, we have no prior information about the transmitted symbols and ML estimation is the only option. In this context, the arithmetic-sum estimation boils down to MUE and the estimation space is infinitely large. Two problems with the ML estimation are error propagation and noise enhancement. Specifically, the estimation error introduced by noise in a sample can propagate to other samples, causing larger and larger deviations from the true values for all the samples in between. The error propagation is further intensified by phase misalignment since the error/noise can be amplified in the propagation.

As a result, ML estimation does not perform well in the low EsN0 regime, especially when there is phase misalignment. To address this problem, a possible solution is to insert pilots in the transmitted symbols to cut off the error propagation.

The aligned-sample estimator – The subsequence of the "aligned" samples in b), on the other hand, allows an aligned-sample estimator to be used in misaligned OAC. The upside of the aligned-sample estimator is that it does not suffer from error propagation and noise enhancement issues; and hence, it emerges a good alternative to the ML estimator in the low-EsN0 regime. The downsides, however, are that it suffers from both phase misalignment and time misalignment – phase misalignment causes a test-accuracy loss and time misalignment causes an EsN0 penalty to the aligned-sample estimator.

The computational complexities of both ML and aligned-sample estimators grow linearly with the packet length. The aligned-sample estimator is preferred in the low-EsN0 regime and the ML estimator is preferred in the high-EsN0 regime.

Appendix A

In this appendix, we generalize the system model in Section II by considering fast fading channels $\tilde{h}_m(t)$ and residual CFO in the received signal $r(t)$.

At each transmitter, to compensate the fast channel fading and CFO between the transmitter and the receiver, α_m in (3) is set as $\alpha_m(t) = e^{-j\bar{\varepsilon}_m t}/\bar{h}_m(t)$, where $\bar{\varepsilon}_m$ is the estimated CFO at the m-th device.

The received signal $r(t)$ is then given by

$$r(t) = \sum_{m=1}^{M} \tilde{h}_m(t) e^{j\tilde{\varepsilon}_m t} x_m(t-\tau_m) + z(t). \tag{28}$$

Unlike (4), each $\tilde{h}_m(t)$ is now a fast fading channel and $\tilde{\varepsilon}_m$ is the CFO between the m-th device and the PS.

Substituting (3) into (28) gives us

$$\begin{aligned} r(t) &= \sum_{m=1}^{M} \tilde{h}_m(t) e^{j\tilde{\varepsilon}_m t} \alpha_m(t) \sum_{\ell=1}^{L} s_m[\ell] p(t-\tau_m-\ell T) + z(t) \\ &= \sum_{\ell=1}^{L} \sum_{m=1}^{M} h'_m(t) e^{j\varepsilon'_m t} s_m[\ell] p(t-\tau_m-\ell T) + z(t), \end{aligned} \tag{29}$$

where $h'_m(t) = \tilde{h}_m(t)/\bar{h}_m(t)$ is the residual channel-fading coefficient and $\varepsilon'_m = \tilde{\varepsilon}_m - \bar{\varepsilon}_m$ is the residual CFO between the m-th device and the PS. As can be seen, CFO introduces additional channel-gain misalignments among devices.

With the revised signal $r(t)$, the discrete samples can be written in the same form as (8) with different coefficients $c_{m,k}[i]$ and $c'_{m,k}[i]$, giving,

$$\begin{aligned} c_{m,k}[i] &= \frac{1}{T} \int_{(i-1)T+\tau_k}^{(i-\mathbb{1}_{m>k})T+\tau_m} h'_m(t) e^{j\varepsilon'_m \zeta}\, d\zeta = \frac{1}{j\varepsilon'_m T} \times \\ &\left(e^{j\varepsilon'_m[(i-\mathbb{1}_{m>k})T+\tau_m]} - e^{j\varepsilon'_m[(i-1)T+\tau_k]}\right) \int_{(i-1)T+\tau_k}^{(i-\mathbb{1}_{m>k})T+\tau_m} h'_m(t)\, dt, \end{aligned} \tag{30}$$

$$\begin{aligned} c'_{m,k}[i] &= \frac{1}{T} \int_{(i-\mathbb{1}_{m>k})T+\tau_m}^{iT+\tau_k} h'_m(t) e^{j\varepsilon'_m \zeta}\, d\zeta = \frac{1}{j\varepsilon'_m T} \times \\ &\left(e^{j\varepsilon'_m[iT+\tau_k]} - e^{j\varepsilon'_m(i-\mathbb{1}_{m>k})T+\tau_m}\right) \int_{(i-\mathbb{1}_{m>k})T+\tau_m}^{iT+\tau_k} h'_m(t)\, dt. \end{aligned} \tag{31}$$

On the other hand, for our WMFS scheme, the $y_k[i]$ samples in (15) can be written to

$$y_k[i] = \sum_{m=1}^{M} g_{m,k}[i] s_m[i - \mathbb{1}_{m>k}] + \tilde{z}_k[i], \tag{32}$$

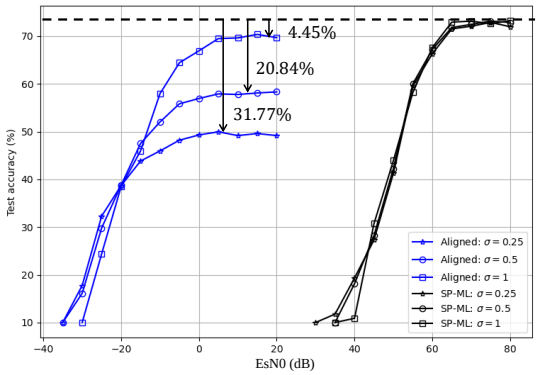

Figure 10. Test accuracies of the aligned-sample and ML estimators under different degrees of channel misalignments. The maximum time offset is $\tau_M = 0.5$, the maximum phase misalignment $\phi = \pi/2$, and the amplitude of the residual channel is $|h_m| \sim \text{Rayleigh}(\sigma)$, where σ is set to 0.25, 0.5, and 1, respectively.

where

$$g_{m,k}[i] = h'_m \frac{1}{d_k} \int_{(i-1)T+\tau_k}^{(i-1)T+\tau_{k+1}} e^{j\varepsilon'_m \zeta} d\zeta = \frac{h'_m}{d_k} \frac{e^{j\varepsilon'_m d_k} - 1}{j\varepsilon'_m} e^{j\varepsilon'_m [(i-1)T+\tau_k]}.$$

As a result, if the channel is fast fading and there is residual CFO in the received signal, we can simply modify the coefficient matrix $\boldsymbol{D}$ as

$$\boldsymbol{D} = \begin{bmatrix} g_{1,1}[1] & & & & & & & & & & \\ g_{1,2}[1] & g_{2,1}[1] & & & & & & & & & \\ \cdots & g_{2,2}[1] & \cdots & & & & & & & & \\ g_{1,M}[1] & \cdots & \cdots & g_{M,1}[1] & & & & & & & \\ & g_{2,M}[1] & \cdots & g_{M,2}[1] & g_{1,1}[2] & & & & & & \\ & & \cdots & \cdots & g_{1,2}[2] & g_{2,1}[2] & & & & & \\ & & & g_{M,M}[1] & \cdots & g_{2,2}[2] & \cdots & & & & \\ & & & & g_{1,M}[2] & \cdots & \cdots & g_{M,1}[2] & & & \\ & & & & & g_{2,M}[2] & \cdots & g_{M,2}[2] & \cdots & & \\ & & & & & & \cdots & \cdots & \cdots & & \\ & & & & & & & g_{M,M}[2] & \cdots & & \\ & & & & & & & & \cdots & & \end{bmatrix}. \quad (33)$$

APPENDIX B
ADDITIONAL SIMULATION RESULTS

A. Channels with mismatched amplitudes

In Section V, we only consider phase-misaligned channels, but it is easy to generalize the simulations to the misaligned-amplitude case. Below, we repeat the simulation in Fig. 8, assuming mismatched amplitudes of the residual channel gains.

As in Fig. 8, we fix the maximum time offset to $\tau_M = 0.5$ in the simulation. The residual channel gain of the m-th device is $h_m = |h_m|e^{j\phi_m}$. In particular, the phase term is uniformly distributed, i.e., $\phi_m \sim U(0, \phi)$, where ϕ is fixed to $\pi/2$. Unlike Fig. 8, where $|h_m| = 1$, we assume $|h_m|$ follows the Rayleigh distribution, i.e., $|h_m| \sim \text{Rayleigh}(\sigma)$, where σ is set to 0.25, 0.5, and 1, respectively.

Fig. 10 presents the test accuracies of the learned model in the presence of mismatched channel amplitudes, where both the aligned-sample and SP-ML estimators are used. In general, the observations from Fig. 10 match those from Fig. 8. Specifically,

1) The aligned-sample estimator works well in the low-EsN0 regime while the ML estimator works well in the high-EsN0 regime.
2) Due to the channel-gain misalignment in both amplitude and phase, the aligned-sample estimator suffers from a test-accuracy loss. On the other hand, the ML estimator does not suffer from test-accuracy loss – the test accuracy after convergence is the same as the aligned case.

B. Effect of the number of active devices

In Section V, we consider a FEEL system with $M = 40$ edge devices. $50,000$ training examples are assigned to the 40 devices in a non-i.i.d. manner. In each FEEL iteration, a random subset of K devices are active and K is set to 4 in the simulations. In the following, we further investigate the impact of K on the learning performance.

Fig. 11 presents the learning performance of FEEL versus the number of active devices considering both aligned and misaligned OAC. Let us first focus on the aligned OAC, in which case only AWGN is introduced in the model aggregation process and there are no misalignments. The test-accuracy performance is shown in Fig. 11(a).

1) A first observation is that the learning performance is irrelevant to the number of active devices in each iteration when the noise power is zero (i.e., the noiseless case, where FEEL reduces to federated learning (FL) with ideal model aggregation). We conjecture that this is due to the non-i.i.d. data across devices. Similar results are observed in the convergence proof of FL [35], where the authors showed that the number of active devices has very limited influence on the convergence of FL when the data are non-i.i.d..
2) On the other hand, when there is noise, the learning performance benefits from more participating devices in each FEEL iteration. In general, more participating devices yield a smaller variance of the aggregated gradients, thereby stabilizing the learning in a noisy environment.

When there are misalignments in the overlapped signals, the same results can be observed, as shown in Fig. 11(b). Specifically, when there are only misalignments but no noise, the learning performance is relatively stable with the increase in the number of active devices. When there are both misalignments and noise, however, more participating devices are beneficial to the learning performance.

We further performed larger-scale simulations with more than 40 devices, as required by the reviewer, and present the results in Fig. 12 below. In these simulations, we assume there are $M = 80$ edge devices in the FEEL system and the number of active devices in each iteration, K, ranges from 4 to 56. As can be seen, the results are in line with those in Fig. 11.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
[2] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2184–2199, 2019.

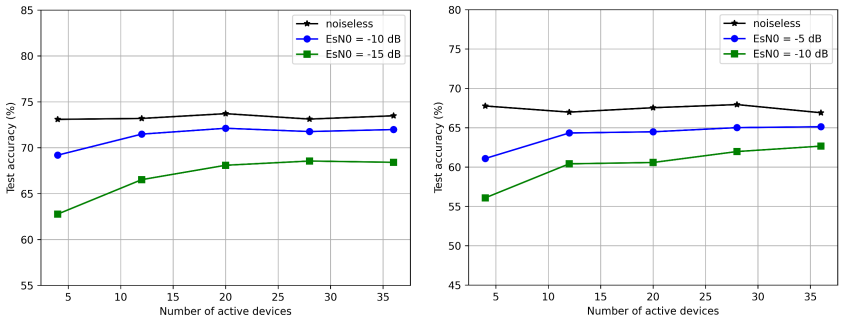

Figure 11. Test accuracy versus the number of active devices in FEEL, where $M = 40$. (a) Aligned OAC, where there is neither time nor channel-gain misalignment, only AWGN is introduced in the federated averaging process. (b) Misaligned OAC, where the maximum time offset $\tau_M = 0.5$ and the maximum phase misalignment $\phi = \pi/2$. We use the aligned-sample estimator at the PS to estimate the aggregated model.

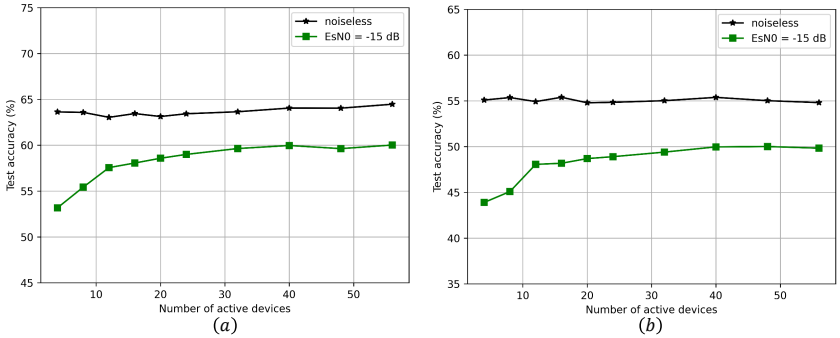

Figure 12. Test accuracy versus the number of active devices in FEEL, where $M = 80$. (a) Aligned OAC, where there is only AWGN but no time or channel-gain misalignment. (b) Misaligned OAC, where the maximum time offset $\tau_M = 0.5$, the maximum phase misalignment $\phi = \pi/2$, and the aligned-sample estimator is used to estimate the aggregated model.

[3] Y. Shao, A. Rezaee, S. C. Liew, and V. Chan, "Significant sampling for shortest path routing: a deep reinforcement learning solution," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2234 – 2248, 2020.

[4] D. Gunduz, D. B. Kurka, M. Jankowski, M. M. Amiri, E. Ozfatura, and S. Sreekumar, "Communicate to learn at the edge," *IEEE Commun. Magazine*, vol. 58, no. 12, pp. 14–19, 2020.

[5] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," *arXiv preprint, arXiv:1610.05492*, 2016.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AI Statistics*. PMLR, 2017, pp. 1273–1282.

[7] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: system design," *arXiv preprint, arXiv:1902.01046*, 2019.

[8] A. Gupta and R. K. Jha, "A survey of 5G network: architecture and emerging technologies," *IEEE access*, vol. 3, pp. 1206–1232, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

[10] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.

[11] ——, "Federated learning over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, 2020.

[12] G. Zhu and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 491–506, Jan. 2020.

[13] T. Sery, N. Shlezinger, K. Cohen, and Y. Eldar, "Over-the-air federated learning from heterogeneous data," *arXiv preprint arXiv:2009.12787*, 2020.

[14] 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, 2020.

[15] G. Zhu, Y. Du, D. Gunduz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis," *IEEE Trans. Wireless Commun.*, 2020.

[16] M. M. Amiri, T. M. Duman, and D. Gunduz, "Collaborative machine learning at the wireless edge with blind transmitters," in *Proc. IEEE Global Conf. on Signal and Info. Proc. (GlobalSIP)*, Dec. 2019.

[17] M. M. Amiri, T. M. Duman, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Blind federated edge learning," *arXiv cs.IT.2010.10030*, 2020.

[18] Y. Sun, S. Zhou, and D. Gündüz, "Energy-aware analog aggregation for federated learning with redundant data," in *IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[19] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[20] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, 2020.

[21] S. Verdu, *Multiuser detection*. Cambridge university press, 1998.

[22] A. Tveit, "On the complexity of matrix inversion," *Mathematical Note*, 2003.

[23] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on inf. theory*, vol. 47, no. 2, pp. 498–519, 2001.

[24] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. IEEE*, vol. 95, no. 6, pp. 1295–1322, 2007.

[25] Y. Shao, S. C. Liew, and L. Lu, "Asynchronous physical-layer network coding: symbol misalignment estimation and its effect on decoding," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6881–6894, 2017.

[26] K. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: an empirical study," *arXiv preprint arXiv:1301.6725*, 2013.

[27] N. Noorshams and M. J. Wainwright, "Belief propagation for continuous state spaces: stochastic message-passing with quantitative guarantees," *J. Mach. Lear. Research*, vol. 14, no. 1, pp. 2799–2835, 2013.

[28] T. Wang, L. Shi, S. Zhang, and H. Wang, "Gaussian mixture message passing for blind known interference cancellation," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4268–4282, 2019.

[29] Y. Shao, D. Gunduz, and S. C. Liew, "Bayesian over-the-air computation," *Technical report, available on ArXiv*, 2021.
[30] G. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of inter-symbol interference," *IEEE Trans. Inf. Theory*, vol. 18, no. 3, pp. 363–378, 1972.
[31] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, 1974.
[32] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Technical report*, 2009.
[33] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: practical guidelines for efficient cnn architecture design," in *ECCV*, 2018, pp. 116–131.
[34] Y. Shao, G. Deniz, and S. C. Liew, "Federated edge learning with misaligned over-the-air computation," *source code, available online at: https://github.com/lynshao/MisAlignedOAC*, 2020.
[35] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-iid data," *arXiv preprint:1907.02189*, 2019.