

# Learning to Demodulate from Few Pilots via Offline and Online Meta-Learning

Sangwoo Park, *Student Member, IEEE*, Hyeryung Jang, *Member, IEEE*,  
Osvaldo Simeone, *Fellow, IEEE*, and Joonhyuk Kang, *Member, IEEE*

**Abstract**—This paper considers an Internet-of-Things (IoT) scenario in which devices sporadically transmit short packets with few pilot symbols over a fading channel. Devices are characterized by unique transmission non-idealities, such as I/Q imbalance. The number of pilots is generally insufficient to obtain an accurate estimate of the end-to-end channel, which includes the effects of fading and of the transmission-side distortion. This paper proposes to tackle this problem by using meta-learning. Accordingly, pilots from previous IoT transmissions are used as meta-training data in order to train a demodulator that is able to quickly adapt to new end-to-end channel conditions from few pilots. Various state-of-the-art meta-learning schemes are adapted to the problem at hand and evaluated, including Model-Agnostic Meta-Learning (MAML), First-Order MAML (FOMAML), REPTILE, and fast Context Adaptation VIA meta-learning (CAVIA). Both offline and online solutions are developed. In the latter case, an integrated online meta-learning and adaptive pilot number selection scheme is proposed. Numerical results validate the advantages of meta-learning as compared to training schemes that either do not leverage prior transmissions or apply a standard joint learning algorithms on previously received data.

**Index Terms**—Machine learning, meta-learning, online meta-learning, Model-Agnostic Meta-Learning (MAML), First-Order MAML (FOMAML), REPTILE, fast Context Adaptation VIA meta-learning (CAVIA), IoT, demodulation.

## I. INTRODUCTION

### A. Motivation

For many standard channel models, such as additive Gaussian noise and fading channels with receive Channel State Information (CSI), the design of optimal demodulators and decoders is well understood. Most communication links hence use pilot sequences to estimate CSI, which is then plugged into

This work was presented in part at IEEE SPAWC 2019 [1].

S. Park and J. Kang are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mails: {sangwoop, jkang}@kaist.ac.kr).

H. Jang and O. Simeone are with the Department of Informatics, King’s College London, London WC2R 2LS, U.K. (e-mails: {hyeryung.jang, osvaldo.simeone}@kcl.ac.uk).

The work of S. Park was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (No.2018-0-00170, Virtual Presence in Moving Objects through 5G).

The work of H. Jang and O. Simeone was supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 725731).

The work of J. Kang was supported in by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute of Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2020-0-01787.

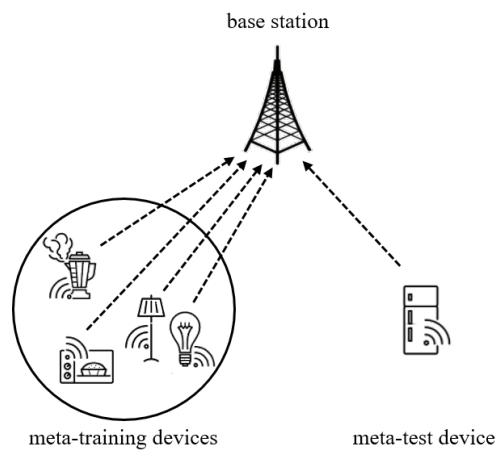


Fig. 1: Illustration of few-pilot training for an IoT system via meta-learning.

the optimal receiver with ideal receive CSI (see, e.g., [2]). This standard model-based approach is inapplicable if: (i) an accurate channel model is unavailable; and/or (ii) the optimal receiver for the given transmission scheme and channel is of prohibitive complexity or unknown. Examples of both scenarios are reviewed in [3], [4], and include new communication set-ups, such as molecular channels, which lack well-established models; and links with strong non-linearities, such as satellite channels with non-linear transceivers, whose optimal demodulators can be highly complex [3], [5]. This observation has motivated a long line of work on the application of machine learning methods to the design of demodulators or decoders, from the 90s [3] to many recent contributions, including [6], [7], [8] and references therein.

Demodulation and decoding can be interpreted as classification tasks, where the input is given by the received baseband signals and the output consists of the transmitted symbols for demodulation, and of the transmitted binary messages for decoding. Pilot symbols can hence be used as training data to carry out the supervised learning of a parametric model for the demodulator or decoder, such as Support Vector Machines (SVMs) or neural networks. The performance of the trained “machine” as a demodulator or a decoder generally depends on how representative the training data are for the channel conditions encountered during test time and on the suitability of the parametric model in terms of trade-off between bias and variance [9].

To the best of our knowledge, all the prior works reviewed

above assume that training is carried out using pilot signals from the same transmitter whose data is to be demodulated or decoded. This generally requires the transmission of long pilot sequences for training. In this paper, we consider an Internet-of-Things (IoT)-like scenario, illustrated in Fig. 1, in which devices sporadically transmit short packets with few pilot symbols. The number of pilots is generally insufficient to obtain an accurate estimate of the end-to-end channel, which generally includes the effects of fading and of the transmitter’s non-linearities [10]. We propose to tackle this problem by using *meta-learning* [11].

### B. Meta-Learning

Meta-learning, also sometimes referred to as “learning to learn”, aims at leveraging training and test data from different, but related, tasks for the purpose of acquiring an inductive bias that is suitable for the entire class of tasks of interest [11]. The inductive bias can be optimized by selecting either a model class, e.g., through a feature extractor, or a training algorithm, e.g., through an initialization of model parameters or the learning rate [12], [13]. An important application of meta-learning is the acquisition of a training procedure that allows a quick adaptation to a new, but related, task using few training examples, also known as *few-shot learning* [14]. For instance, one may have training and test labelled images for binary classifiers of different types of objects, such as cats vs dogs or birds vs bikes. These can be used as meta-training data to quickly learn a new binary classifier, say for handwritten binary digits, from few training examples.

Meta-learning has recently received renewed attention, particularly thanks to advances in the development of methods based on Stochastic Gradient Descent (SGD), including Model-Agnostic Meta-Learning (MAML) [15], REPTILE [16], and fast Context Adaptation VIA meta-learning (CAVIA) [17]. Such techniques can be generally classified as either *offline*, in which case the meta-training data is fixed and given [15], [16], [17]; or *online*, in which case all prior data from related tasks is treated as meta-training data in a streaming fashion [18].

### C. Main Contributions

As illustrated in Figs. 2 and 3, the key idea of this paper is to use pilots from previous transmissions of other IoT devices as meta-training data in order to train a procedure that is able to quickly adapt a demodulator to new end-to-end channel conditions from few pilots. We consider both an offline formulation, in which the set of previous transmissions is fixed, and an online set-up, in which the meta-training set is updated as transmitted pilots are received. The main contributions are as follows:

- We adapt to the problem at hand a number of state-of-the-art offline meta-learning solutions, namely MAML [15], First-Order MAML (FOMAML) [15], REPTILE [16], and CAVIA [17]. Their relative merits and a unified interpretation in terms of the Expectation-Maximization (EM) algorithm are discussed;

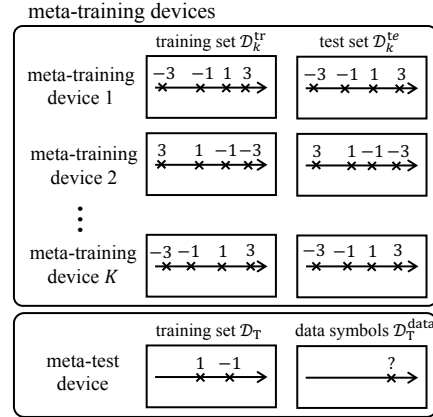


Fig. 2: Offline meta-learning: Meta-training and meta-test data for 4-PAM transmission from set  $\mathcal{S} = \{-3, -1, 1, 3\}$ . The figure assumes  $N = 8$  pilot symbols divided into  $N^{\text{tr}} = 4$  for meta-training and  $N^{\text{te}} = 4$  for meta-testing, and  $P = 2$  pilots for the meta-test device. Crosses represent received signals  $y_k^{(n)}$ , and the number above each cross represents the corresponding label, i.e., the pilot symbol  $s_k^{(n)}$ .

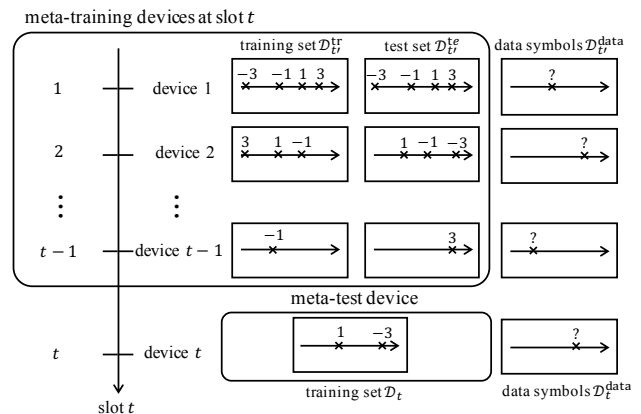


Fig. 3: Online meta-learning: Meta-training and meta-test data for 4-PAM transmission from set  $\mathcal{S} = \{-3, -1, 1, 3\}$ . Meta-training data are accumulated as the BS observes subsequent slots  $t = 1, 2, \dots$ , with one device transmitting pilots and data symbols in each slot.

- We validate the advantage of meta-learning with extensive numerical results that provide comparisons with conventional model-based and learning-based communication schemes. A comparative study of the performance of various meta-learning solutions is also presented;
- We propose a novel online solution that integrates meta-learning with an adaptive selection of the number of pilots and compare the proposed solution with conventional non-adaptive solutions in terms of receiver’s performance and number of pilots.

The results in this paper have been partially presented in [1]. In particular, reference [1] derives an offline MAML-based algorithm, and offers some preliminary numerical results. As compared to the preliminary conference version [1], this paper presents additional analysis, including a general framework

for meta-learning based on EM; novel algorithms, introducing a comprehensive evaluation of a larger number of meta-learning offline schemes and the study of online meta-learning for demodulation, along with a new adaptive pilot number selection algorithm; and more extensive discussions in terms of both algorithm definition and extra experiments.

#### D. Related Works

In [19], which is concurrent to [1], the authors train a neural network-based decoder that can adapt to the new channel condition with a minimal number of pilot symbols using meta-learning via FOMAML. In [20], the authors train a neural network-based channel estimator in OFDM system with meta-learning via FOMAML in order to obtain an effective channel estimation given a small number of pilots. After the first submission of this paper, several additional papers have considered meta-learning for communication. Reference [13] provides a review of meta-learning with applications to communication systems. In [21], meta-learning is used for downlink/uplink channel conversion in Frequency-Division Duplex massive MIMO channels. Papers [22] and [23] consider meta-learning for end-to-end training of physical layer with and without a channel model, respectively. Finally, reference [24] considers a related approach based on hypernetworks to aid neural network-based MIMO detection.

*Paper organization:* The rest of the paper is organized as follows. In Sec. II, we detail system model and offline meta-learning problem. In Sec. III, various offline meta-learning solutions are covered within a unified interpretation. In Sec. IV, we redefine system model for an online setting and propose a novel online solution, including adaptive pilot number selection. Numerical results are presented in Sec. V and conclusions are presented in Sec. VI.

## II. MODEL AND PROBLEM

### A. System Model

In this paper, we consider the IoT system illustrated in Fig. 1, which consists of a number of devices and a base station (BS). For each device  $k$ , the complex symbol transmitted by the device and the corresponding received signal at the BS are denoted as  $s_k \in \mathcal{S}$  and  $y_k$ , respectively. We also denote as  $\mathcal{S}$  the set of all constellation symbols as determined by the modulation scheme. The end-to-end channel for a device  $k$  is defined as

$$y_k = h_k x_k + z_k, \quad (1)$$

where  $h_k$  is the complex channel gain from device  $k$  to the BS, which is constant over a transmission block according to the standard quasi-static fading model typically assumed for short-packet transmissions;  $z_k \sim \mathcal{CN}(0, N_0)$  is additive white complex Gaussian noise; and

$$x_k \sim p_k(\cdot | s_k) \quad (2)$$

is the output of a generally random transformation defined by the conditional distribution  $p_k(\cdot | s_k)$ . This conditional distribution accounts for transmitter's non-idealities such as phase

noise [25], I/Q imbalance [26], and amplifier's characteristics [10] of the IoT device. Throughout the paper for each device  $k$ , we assume pilots and data symbols to follow the same constellation  $\mathcal{S}$  and to be subject to the transmitter's non-idealities defined by  $p_k(\cdot | s_k)$ . The average transmitted energy per symbol is constrained as  $\mathbb{E}[|x_k|^2] \leq E_x$  for some positive value  $E_x$  for both pilot and data symbols. As an example for the transmitter's non-idealities (2), a common model for the I/Q imbalance assumes the following transformation [27]

$$x_k = (1 + \epsilon_k) \cos \delta_k \text{Re}\{s_k\} - (1 - \epsilon_k) \sin \delta_k \text{Im}\{s_k\} \\ + j((1 - \epsilon_k) \cos \delta_k \text{Im}\{s_k\} - (1 + \epsilon_k) \sin \delta_k \text{Re}\{s_k\}), \quad (3)$$

where  $\epsilon_k$  and  $\delta_k$  represent the amplitude imbalance factor and phase imbalance factor, which are real constants or random variables. Note that we only explicitly model imperfections at the transmitter side. This is because, in practice, non-idealities on the receiver processing chain at the BS are expected to be much less significant than the mentioned non-idealities for IoT devices. Furthermore, receiver-side non-linearities at the BS can be also mitigated through offline designs prior to deployment.

Based on the reception of few pilots from a target device, we aim at determining a demodulator that correctly recovers the transmitted symbol  $s$  from the received signal  $y$  with high probability. The demodulator is defined by a conditional probability distribution  $p(s|y, \varphi)$ , which depends on a trainable parameter vector  $\varphi$ .

### B. Offline Meta-Learning Problem

Following the nomenclature of meta-learning [15], the target device is referred to as the *meta-test device*. To enable few-pilot learning, we assume here that the BS can use the signals received from the previous pilot transmissions of  $K$  other IoT devices, which are referred to as *meta-training devices* and their data as *meta-training data*. Specifically, as illustrated in Fig. 2, the BS has  $N$  pairs of pilot  $s_k$  and received signal  $y_k$  for each meta-training device  $k = 1, \dots, K$ . The meta-training dataset is denoted as  $\mathcal{D} = \{\mathcal{D}_k\}_{k=1, \dots, K}$ , where  $\mathcal{D}_k = \{(s_k^{(n)}, y_k^{(n)}) : n = 1, \dots, N\}$ , and  $(s_k^{(n)}, y_k^{(n)})$  are the  $n$ -th pilot-received signal pairs for the  $k$ th meta-training device. This scenario is referred to as offline meta-learning since the meta-training dataset  $\mathcal{D}$  is fixed and given. Online meta-training will be discussed in Sec. IV.

For the target, or the meta-test, device, the BS receives  $P$  pilot symbols. We collect the  $P$  pilots received from the target device in set  $\mathcal{D}_T = \{(s^{(n)}, y^{(n)}) : n = 1, \dots, P\}$ . The demodulator can be trained using meta-training data  $\mathcal{D}$  and the pilot symbols  $\mathcal{D}_T$  from the meta-test device.

Training requires the selection of a parametric model  $p(s|y, \varphi)$  for the demodulator. The choice of the parametric model  $p(s|y, \varphi)$  should account for the standard trade-off between capacity of the model and overfitting [28], [29]. To fix the ideas, we will assume that the demodulator  $p(s|y, \varphi)$  is given by a multi-layer neural network with  $L$  layers, with

a softmax non-linearity in the final,  $L$ th, layer. This can be written as

$$p(s|y, \varphi) = \frac{\exp\left([f_{\varphi^{(L-1)}}(f_{\varphi^{(L-2)}}(\cdots f_{\varphi^{(1)}}(y)))\right]_s}{\sum_{s' \in \mathcal{S}} \exp\left([f_{\varphi^{(L-1)}}(f_{\varphi^{(L-2)}}(\cdots f_{\varphi^{(1)}}(y)))\right]_{s'}}, \quad (4)$$

where  $f_{\varphi^{(l)}}(x) = \sigma(W^{(l)}x + b^{(l)})$  represents the non-linear activation function of the  $l$ th layer with parameter  $\varphi^{(l)} = \{W^{(l)}, b^{(l)}\}$ , with  $W^{(l)}$  and  $b^{(l)}$  being the weight matrix and bias vector of appropriate size, respectively;  $[\cdot]_s$  stands for the element regarding  $s$ ; and  $\varphi = \{\varphi^{(l)}\}_{l=1, \dots, L-1}$  is the vector of parameters. The non-linear function  $\sigma(\cdot)$  can be, e.g., a Rectified Linear Unit (ReLU) or a hyperbolic tangent function. The input  $y$  in (4) can be represented as a two-dimensional vector comprising real and imaginary parts of the received signal.

### III. OFFLINE META-LEARNING ALGORITHMS

In this section, we adapt state-of-the-art offline meta-learning algorithms to design the demodulator in (4) given meta-training and meta-test data. As discussed in Sec. I, we view demodulation as a classification task. To set the notation, for any set  $\mathcal{D}_0$  of pairs  $(s, y)$  of transmitted symbol  $s$  and received signal  $y$ , the standard cross-entropy loss function is defined as a function of the demodulator parameter vector  $\varphi$  as

$$L_{\mathcal{D}_0}(\varphi) = - \sum_{(s, y) \in \mathcal{D}_0} \log p(s|y, \varphi). \quad (5)$$

#### A. Joint Training

As a benchmark, we start by considering a conventional approach that uses the meta-training data  $\mathcal{D}$  and the training data  $\mathcal{D}_T$  for the *joint training* of the model  $p(s|y, \varphi)$ . Joint training pools together all the pilots received from the meta-training devices and the meta-test device, and carries out the optimization of the cumulative loss  $L_{\mathcal{D} \cup \mathcal{D}_T}(\varphi)$  in (5) using SGD. Accordingly, the parameter vector  $\varphi$  is updated iteratively based on the rule

$$\varphi \leftarrow \varphi + \eta \nabla_{\varphi} \log p(s^{(n)}|y^{(n)}, \varphi), \quad (6)$$

by drawing one pair  $(s^{(n)}, y^{(n)})$  at random from the set  $\mathcal{D} \cup \mathcal{D}_T$ . In (6), the step size  $\eta$  is assumed to be fixed for simplicity of notation but it can in practice be adapted across the updates (see, e.g., [30]). Furthermore, this rule can be generalized by summing the gradient in (6) over a mini-batch of pairs from the dataset  $\mathcal{D} \cup \mathcal{D}_T$  at each iteration [30].

#### B. A Unified View of Meta-Learning

A useful way to introduce meta-learning in terms of the graphical model is illustrated in Fig. 4. Accordingly, meta-learning assumes a demodulator  $p(s|y, \phi, \theta)$  that depends on a shared parameter  $\theta$  common to all tasks, or users, and on a latent context variable  $\phi$ , which is specific to each user. The specific parameterization  $p(s|y, \phi, \theta)$  and its relationship

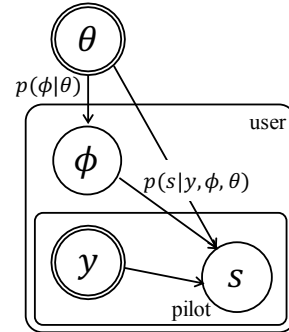


Fig. 4: Graphical model assumed by meta-learning: The demodulator  $p(s|y, \phi, \theta)$  depends on a user-specific, or context, random variable  $\phi$ , as well as on a shared parameter  $\theta$ , which may also affect the prior distribution of the context variable  $\phi$ . Double circles denote parameters, and the tile notation (see, e.g., [31]) defines multiple users and pilots per user.

with (4) depend on the meta-learning scheme, and they will be discussed below. Note that, as illustrated in Fig. 4, the context vector  $\phi$  is assumed to be random, while  $\theta$  is a shared (deterministic) parameter. Furthermore, from Fig. 4, the shared variable  $\theta$  can also affect the prior distribution of the context variable  $\phi$ . In this framework, the key idea is that meta-training data  $\mathcal{D}$  are used to estimate the shared parameters  $\theta$  via the process of meta-learning, while the context variable  $\phi$  is inferred from the meta-test data  $\mathcal{D}_T$ .

To elaborate, a principled way to train the model in Fig. 4 would be to estimate parameter  $\theta$  using the Expectation-Maximization (EM) algorithm based on the meta-training data  $\mathcal{D}$ . The EM algorithm is in fact the standard tool to tackle the problem of maximum likelihood estimation in the presence of latent variables, here the context variable  $\phi$  (see, e.g., [28], [29], [31]). EM maximizes the sum of marginal likelihoods

$$p(s|y, \theta) = \mathbb{E}_{\phi \sim p(\phi|\theta, \mathcal{D}_k)} [p(s|y, \phi, \theta)] \quad (7)$$

over the data pairs  $(s, y)$  from all data sets  $\mathcal{D}_k$  in the meta-training data set  $\mathcal{D}$ . In (7), the average is taken with respect to the posterior distribution  $p(\phi|\theta, \mathcal{D}_k)$  of the context variable given the training data  $\mathcal{D}_k$  of the  $k$ th meta-training device. After EM training, one can consider the obtained parameter  $\theta$  as fixed when inferring a data symbol  $s$  given a new observed signal  $y$  and the pilots  $\mathcal{D}_T$  for the meta-test device. This last step would ideally yield the demodulator

$$p(s|y, \theta) = \mathbb{E}_{\phi \sim p(\phi|\theta, \mathcal{D}_T)} [p(s|y, \phi, \theta)], \quad (8)$$

where the average is taken over the posterior distribution  $p(\phi|\theta, \mathcal{D}_T)$  of the context variable given the training data of the meta-test device.

The computation of the posteriors  $p(\phi|\theta, \mathcal{D}_k)$  in (7) and  $p(\phi|\theta, \mathcal{D}_T)$  in (8) is generally of infeasible complexity. Therefore, state-of-the-art meta-learning techniques approximate this principled solution by either employing point estimate of latent context variable  $\phi$  [15], [16], [17] or by a direct approximation of its posterior distribution [32], [33], [34]. In this paper, we focus on the more common point estimate based meta-learning techniques, which are reviewed next.



### C. MAML

For any meta-training device  $k$ , MAML [15] assumes a demodulator  $p(s|y, \phi_k)$  given by (4) with model weights  $\varphi$  equal to the context variable  $\phi_k$ . The user-specific variable  $\phi_k$ , rather than being obtained from the ideal posterior  $p(\phi_k|\theta, \mathcal{D}_k)$  as in (7), is computed via SGD-based training from the data  $\mathcal{D}_k$ . Specifically, the key idea in MAML is to identify an initial shared parameter  $\theta$  during meta-training such that starting from it, the SGD updates (6) for any meta-training device  $k$  (i.e., for  $\mathcal{D}_0 = \mathcal{D}_k$ ) produce a parameter vector  $\phi_k$  that yields a low value of the loss function (5). As we will detail, it is possible to include one or multiple SGD updating steps in (6) [35]. After meta-training, the initial parameter  $\theta$  is used for the SGD updates of the target device based on the pilots in set  $\mathcal{D}_T$ .

To elaborate, assume first that the exact average loss  $L_k(\phi_k) = \mathbb{E}[-\log p(s_k|y_k, \phi_k)]$  for all meta-training devices  $k = 1, \dots, K$  is given. The average in  $L_k(\phi_k)$  is taken over the distribution  $p(s_k, y_k) = p(s_k)p(y_k|s_k)$ , where  $p(s_k)$  is the prior distribution of the transmitted symbol  $s_k$  and  $p(y_k|s_k)$  is defined by (1) and (2). Note that in practice, this distribution is not known since the channel and the transmitters' model are not known a priori. During meta-training, MAML seeks an initial value  $\theta$  such that, for every device  $k$ , the losses  $L_k(\phi_k)$  obtained after one or more SGD updates starting from  $\theta$  are collectively minimized. As discussed, the SGD updates can be interpreted as producing a point estimate of the context variables  $\phi_k$  in the model in Fig. 4 [12]. Mathematically, with a single SGD iteration, the estimate is obtained as

$$\phi_k = \theta - \eta \nabla_{\theta} L_k(\theta). \quad (9)$$

More generally, with  $m \geq 1$  local SGD updates we obtain  $\phi_k = \phi_k^m$ , where

$$\phi_k^i = \phi_k^{i-1} - \eta \nabla_{\phi_k^{i-1}} L_k(\phi_k^{i-1}), \quad (10)$$

for  $i = 1, \dots, m$ , with  $\phi_k^0 = \theta$ . The identification of a shared parameter  $\theta$  is done by minimizing the sum  $\sum_{k=1}^K L_k(\phi_k)$  over  $\theta$ .

The losses  $L_k(\phi_k)$  for all meta-training devices are not known and need to be estimated from the available data. To this end, in the meta-training phase, each set  $\mathcal{D}_k$  of  $N$  pairs of pilots and received signals for meta-training device  $k$  is randomly divided into a training set  $\mathcal{D}_k^{\text{tr}}$  of  $N^{\text{tr}}$  pairs and a test set  $\mathcal{D}_k^{\text{te}}$  of  $N^{\text{te}}$  pairs, as shown in Fig. 2. The updated context variable  $\phi_k$  is computed by applying the SGD-based rule in (6) by using the training subset  $\mathcal{D}_k^{\text{tr}}$ , as in (10), e.g.,  $\phi_k = \theta - \eta \nabla_{\theta} L_{\mathcal{D}_k^{\text{tr}}}(\theta)$  for a single update. In practice,  $m$  SGD updates can be carried out using mini-batches of training samples at each iteration. The loss  $L_k(\phi_k)$  is then estimated by using the test subset  $\mathcal{D}_k^{\text{te}}$  as  $L_{\mathcal{D}_k^{\text{te}}}(\phi_k)$ . Finally, MAML minimizes the overall estimated loss  $\sum_{k=1}^K L_{\mathcal{D}_k^{\text{te}}}(\phi_k)$  by performing an SGD-based update in the opposite direction of the gradient  $\nabla_{\theta} \sum_{k=1}^K L_{\mathcal{D}_k^{\text{te}}}(\phi_k)$  with step size  $\kappa$ .

Considering first a single local SGD update (9) for the context variables, the meta-training update is finally given as

$$\begin{aligned} \theta &\leftarrow \theta - \kappa \nabla_{\theta} \sum_{k=1}^K L_{\mathcal{D}_k^{\text{te}}}(\phi_k) = \theta - \kappa \sum_{k=1}^K (\mathbf{J}_{\theta} \phi_k) \nabla_{\phi_k} L_{\mathcal{D}_k^{\text{te}}}(\phi_k) \\ &= \theta - \kappa \sum_{k=1}^K (I - \eta \nabla_{\theta}^2 L_{\mathcal{D}_k^{\text{tr}}}(\theta)) \nabla_{\phi_k} L_{\mathcal{D}_k^{\text{te}}}(\phi_k), \end{aligned} \quad (11)$$

where  $\mathbf{J}_{\theta}$  represents the Jacobian operation, and  $\kappa > 0$  is the step size. With multiple local SGD updating steps in (10), the meta-training update can be similarly written as

$$\begin{aligned} \theta &\leftarrow \theta - \kappa \sum_{k=1}^K \left[ (I - \eta \nabla_{\theta}^2 L_{\mathcal{D}_k^{\text{tr}}}(\theta)) \cdots \right. \\ &\quad \left. (I - \eta \nabla_{\phi_k^{m-1}}^2 L_{\mathcal{D}_k^{\text{tr}}}(\phi_k^{m-1})) \nabla_{\phi_k^m} L_{\mathcal{D}_k^{\text{te}}}(\phi_k^m) \right]. \end{aligned} \quad (12)$$

In practice, the meta-update in (11) and (12) can be carried out over a subset of meta-training devices at each iteration. Computation of the Hessian matrices needed in (11) and (12) can be significantly accelerated using a finite difference approximation for Hessian-vector product calculation [36], [37], which is reviewed in Appendix A. The MAML algorithm is summarized in Algorithm 1.

In Algorithm 1, meta-training is carried out for a given fixed number of iterations. Among these iterations, we choose the iterate that has the smallest meta-training loss  $\sum_{k \in K'} L_{\mathcal{D}_k^{\text{te}}}(\phi_k)$ , evaluated on the subset  $K'$  of meta-training devices sampled for the corresponding meta-training update. Similarly, adaptation on the meta-test device chooses the iterate that has the lowest loss  $L_{\mathcal{D}_T^{\text{tr}}}(\phi_T)$ , where  $\mathcal{D}_T^{\text{tr}}$  is the mini-batch of training pairs  $(s, y)$  drawn from  $\mathcal{D}_T$  for the corresponding update.

### D. FOMAML

First-order MAML (FOMAML) [15] is an approximation of MAML that ignores the second-derivative terms in the meta-training updates (11)–(12). Accordingly, the meta-training update is given as

$$\theta \leftarrow \theta - \kappa \nabla_{\theta} \sum_{k=1}^K L_{\mathcal{D}_k^{\text{te}}}(\phi_k). \quad (13)$$

As a result, FOMAML updates parameter  $\theta$  in the opposite direction of the gradient  $\nabla_{\theta} \sum_{k=1}^K L_{\mathcal{D}_k^{\text{te}}}(\phi_k)$  instead of  $\nabla_{\theta} \sum_{k=1}^K L_{\mathcal{D}_k^{\text{te}}}(\phi_k)$ . For some neural network architectures and loss functions, e.g., networks with ReLU activation functions [38], FOMAML has been reported to perform almost as well as MAML [15]. Algorithm 1 provides a summary.

### E. REPTILE

REPTILE [16] is a first-order gradient-based meta-learning algorithm as FOMAML. It uses the same local update (9)–(10) for the context variables  $\phi_k$ , but the meta-training update is given as

$$\theta \leftarrow (1 - \kappa)\theta + \kappa \sum_{k=1}^K \phi_k. \quad (14)$$

---

**Algorithm 1:** Few-Pilot Demodulator Meta-Learning via MAML, FOMAML, REPTILE
 

---

**Input:** Meta-training set  $\mathcal{D} = \{\mathcal{D}_k\}_{k=1,\dots,K}$  and pilots  $\mathcal{D}_T$  from the target device;  $N^{\text{tr}}$  and  $N^{\text{te}}$ ; step size hyperparameters  $\eta$  and  $\kappa$ ; number of meta-training iterations  $I$ ; number of local updates during meta-training  $m$  and meta-testing  $I_T$

**Output:** Learned shared initial parameter vector  $\theta$  and target-device specific parameter vector  $\phi_T$

---

```

initialize parameter vector  $\theta$ 
meta-learning phase
for  $I$  meta-training iterations do
  randomly select a subset  $K'$  of meta-training devices
  for each selected meta-training device  $k$  do
    randomly divide  $\mathcal{D}_k$  into two sets  $\mathcal{D}_k^{\text{tr}}$  of size  $N^{\text{tr}}$  and  $\mathcal{D}_k^{\text{te}}$  of size  $N^{\text{te}}$ 
    do  $m$  local updates for the context variable  $\phi_k$  using (9)–(10)
  end
  update shared parameter  $\theta$  via (11)–(12) for MAML, via (13) for FOMAML, via (14) for REPTILE
end
return iterate  $\theta$  with minimum meta-training loss  $\sum_{k \in K'} L_{\mathcal{D}_k^{\text{te}}}(\theta)$ 

adaptation on the meta-test device
initialize context parameter vector  $\phi_T \leftarrow \theta$ 
for  $I_T$  meta-testing iterations do
  draw a mini-batch  $\mathcal{D}'_T$  of pairs  $(s^{(n)}, y^{(n)})$  from  $\mathcal{D}_T$ 
  update context variable  $\phi_T$  in the direction of the gradient  $\sum_{(s,y) \in \mathcal{D}'_T} \nabla_{\phi_T} \log p(s|y, \phi_T)$  with step size  $\eta$ 
end
return iterate  $\phi_T$  with minimum training loss  $L_{\mathcal{D}'_T}(\phi_T)$ 

```

---

The REPTILE update rule (14) is essentially equivalent to that used for federated learning [39]. We refer to [16] for a justification of the method.

#### F. CAVIA

Unlike the meta-learning techniques discussed so far, CAVIA [17] interprets context variable  $\phi$  as an additional input to the demodulator, so that the demodulator  $p(s|y, \phi, \theta)$  can be written as in (4) with input given by the concatenation  $\tilde{y} = [y, \phi]$  and model weights  $\varphi$  equal to the shared parameter vector  $\theta$ . Using (4), the demodulator is hence in the form  $p(s|\tilde{y}, \theta)$ , where the shared parameter  $\theta$  defines the weights of the demodulator model. After meta-training, the shared parameter  $\theta$  is fixed, and the pilots in set  $\mathcal{D}_T$  of the meta-test device are used to optimize the additional input vector  $\phi$ .

During meta-training, given the current value of the shared parameter  $\theta$ , the context variable  $\phi_k$  is optimized by one or

---

**Algorithm 2:** Few-Pilot Demodulator Meta-Learning via CAVIA
 

---

**Input:** Meta-training set  $\mathcal{D} = \{\mathcal{D}_k\}_{k=1,\dots,K}$  and pilots  $\mathcal{D}_T$  from the target device;  $N^{\text{tr}}$  and  $N^{\text{te}}$ ; step size hyperparameters  $\eta$  and  $\kappa$ ; number of meta-training iterations  $I$ ; number of local updates during meta-training  $m$  and meta-testing  $I_T$

**Output:** Learned parameter vector  $\theta$  and target-device context parameter vector  $\phi_T$

---

```

initialize parameter vector  $\theta$ 
meta-learning phase
for  $I$  meta-training iterations do
  randomly select a subset  $K'$  of meta-training devices
  for each selected meta-training device  $k$  do
    initialize context parameter vector  $\phi_k$ 
    randomly divide  $\mathcal{D}_k$  into two sets  $\mathcal{D}_k^{\text{tr}}$  of size  $N^{\text{tr}}$  and  $\mathcal{D}_k^{\text{te}}$  of size  $N^{\text{te}}$ 
    do  $m$  local updates for the context variable  $\phi_k$  using (15)
  end
  update shared parameter  $\theta$  via (16)
end
return iterate  $\theta$  with minimum meta-training loss  $\sum_{k \in K'} L_{\mathcal{D}_k^{\text{te}}}(\theta)$ 

adaptation on the meta-test device
initialize context parameter vector  $\phi_T$ 
for  $I_T$  meta-testing iterations do
  draw a mini-batch  $\mathcal{D}'_T$  of pairs  $(s^{(n)}, y^{(n)})$  from  $\mathcal{D}_T$ 
  update context parameter vector  $\phi_T$  in the direction of the gradient  $\sum_{(s,y) \in \mathcal{D}'_T} \nabla_{\phi_T} \log p(s|\tilde{y}, \theta)$  with step size  $\eta$  where  $\tilde{y} = [y, \phi_T]$ 
end
return iterate  $\phi_T$  with minimum training loss  $L_{\mathcal{D}'_T}(\theta)$ 

```

---

more SGD-based updates to minimize the loss  $L_{\mathcal{D}_k^{\text{te}}}(\theta)$  as

$$\phi_k \leftarrow \phi_k - \eta \nabla_{\phi_k} L_{\mathcal{D}_k^{\text{te}}}(\theta). \quad (15)$$

Note that the loss  $L_{\mathcal{D}_k^{\text{te}}}(\theta)$  is a function of  $\phi_k$  through the additional input  $\phi_k$ . With the obtained additional input  $\phi_k$ , the meta-training update is given as

$$\theta \leftarrow \theta - \kappa \nabla_{\theta} \sum_{k=1}^K L_{\mathcal{D}_k^{\text{te}}}(\theta). \quad (16)$$

After meta-training, as mentioned, parameter  $\theta$  is fixed, and the context vector  $\phi_T$  is obtained by using SGD updates as

$$\phi_T \leftarrow \phi_T - \eta \nabla_{\phi_T} L_{\mathcal{D}_T}(\theta). \quad (17)$$

CAVIA is summarized in Algorithm 2.

#### IV. ONLINE META-LEARNING ALGORITHM

In this section, we consider an online formulation in which packets from devices, containing both pilots and a data

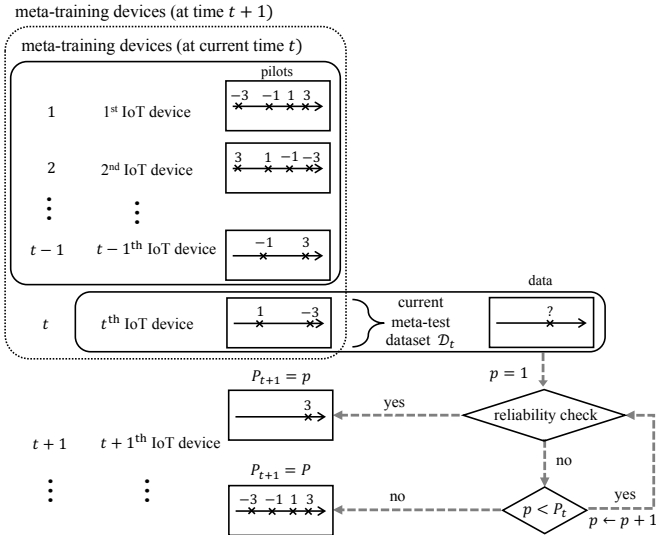


Fig. 5: Illustration of adaptive pilot number selection based on online meta-learning for 4-PAM transmission. The number of transmitted pilots  $P_{t+1}$  for the IoT device active in the next time slot is determined by the BS from the performance of the meta-learned demodulator in the current slot  $t$ .

payload, are sequentially received at the BS. Therefore, as illustrated in Fig. 3, meta-training data are accumulated at the BS over time. The formulation follows the basic framework of online meta-learning introduced in [18], which proposes an online version of MAML. Here, we adapt the online meta-learning framework to the demodulation problem at hand, and extend it to integrate all the meta-training solutions discussed in the previous section, namely MAML, FOMAML, REPTILE, and CAVIA. Moreover, a novel adaptive pilot number selection scheme is proposed that leverages the fast adaptation property of meta-learning to reduce the pilot overhead.

### A. System Model

As illustrated in Fig. 3, in each slot  $t = 1, 2, \dots$ , the BS receives a packet from a new device, from which the BS obtains the set  $\mathcal{D}_t = \{(s_t^{(n)}, y_t^{(n)}) : n = 1, \dots, N_t\}$  of  $N_t$  pilots  $\{s_t^{(n)}\}$  and corresponding received signals  $\{y_t^{(n)}\}$ . Each received packet also contains a payload of data  $\mathcal{D}_t^{\text{data}} = \{(y_t^{(n)}) : n = 1, 2, \dots\}$ . Therefore, at each slot  $t$ , the BS has meta-training data-carrying symbols  $\mathcal{D}^{t-1} = \{\mathcal{D}_{t'}\}_{t'=1}^{t-1}$  from previously active devices, as well as meta-test data  $\mathcal{D}_t$  from the currently active device. The goal is training a demodulator  $p(s|y, \varphi_t)$  that performs well on the payload data  $\mathcal{D}_t^{\text{data}}$  after adaptation on the received pilots in set  $\mathcal{D}_t$  by using the meta-training data  $\mathcal{D}^{t-1}$  as well.

### B. Online Learning (Joint Training)

Before discussing online meta-learning, here we briefly summarize the standard online learning set-up as applied to the problem introduced above. This can be considered as the counterpart of joint training for the offline problem studied in Sec. III-A. In online learning, the goal of the online learner

is to determine a model parameter vector  $\varphi_t$ , sequentially at each slot  $t$ , that performs well on the loss sequence  $L_{\mathcal{D}_t}(\varphi_t)$  for  $t = 1, 2, \dots$  (recall (5)). As a benchmark, typical online learning formulations use the best single model  $\varphi$  that can be obtained using knowledge of the losses  $L_{\mathcal{D}_t}(\cdot)$  in hindsight for all relevant values of  $t$ , i.e.,  $\varphi \in \arg \min_{\varphi} \sum_t L_{\mathcal{D}_t}(\varphi)$ , where the sum is over the time horizon of interest [40].

A standard online learning algorithm is Follow The Leader (FTL) [41], which determines the parameter  $\varphi_t$  that performs best on the previous data  $\mathcal{D}^{t-1}$ . For the problem at hand, FTL determines the parameter  $\varphi_t$  at slot  $t$  by tackling the problem

$$\varphi_t = \arg \min_{\varphi} \sum_{k=1}^t L_{\mathcal{D}_k}(\varphi). \quad (18)$$

Note that in standard online learning formulations the sum in (18) would be performed up to time slot  $t-1$  due to the typical assumption that no data is a priori known at time  $t$  about loss  $L_{\mathcal{D}_t}(\cdot)$  [40]. From (18), FTL can be interpreted as a form of joint training carried out in an online manner. From a theoretical standpoint, FTL can be shown to obtain a sub-linearly growing regret with respect to slot  $t$  as compared to the discussed benchmark learner with hindsight information (see [40] for precise statements).

### C. Online Meta-Learning

With meta-learning, as discussed in Sec. III-B (see Fig. 4), the demodulator  $p(s|y, \phi, \theta)$  is defined by a shared parameter  $\theta$  and by a context, device-dependent, variables  $\phi$ . In the online setting, in each slot  $t$ , we propose to estimate the shared parameter  $\theta_t$  from the meta-training data  $\mathcal{D}^{t-1}$ , while the context variable  $\phi_t$  for the currently active device is estimated from  $\mathcal{D}_t$ . These steps can be carried out for different meta-learning strategies as described in Sec. III, with set  $\mathcal{D}^{t-1}$  in lieu of the meta-training set  $\mathcal{D}$  and set  $\mathcal{D}_t$  for the meta-test set  $\mathcal{D}_T$ . As a special case, if MAML is used, this recovers the Follow The Meta Leader (FTML) algorithm [18], which determines the shared parameter  $\theta_t$  by solving the problem

$$\theta_t = \arg \min_{\theta} \sum_{k=1}^{t-1} L_{\mathcal{D}_k^e}(\phi_t), \quad (19)$$

where the context variable  $\phi_t$  is computed from the local updates (9)–(10) starting from the initial value  $\theta$ . The general algorithm for online meta-learning is summarized in Algorithm 3.

### D. Integrated Online Meta-Learning and Pilot Number Selection

In order to further reduce the pilot overhead, we now consider the possibility to adapt the number of transmitted pilot symbols in each slot  $t$  based on the performance of the demodulator meta-learned in the previous slots. We note that in [42] the idea of adapting the number of pilots was proposed for a single device by leveraging the temporal correlation of the channels for an individual device. In contrast, the method proposed here works by using information from different

---

**Algorithm 3:** Few-Pilot Demodulator Online Meta-Learning

---

**Input:** Data sets  $\{\mathcal{D}_t, \mathcal{D}_t^{\text{data}}\}$  for  $t = 1, 2, \dots$ ; step size hyperparameters  $\eta$  and  $\kappa$ ; number of transmitted pilots  $p$

**Output:** Learned parameter vector  $\theta_t$  and context vector  $\phi_t$ , for  $t = 1, 2, \dots$

---

```
initialize parameter vector  $\theta_1$ 
initialize the meta-training dataset  $\mathcal{D}$  as empty, i.e.,  $\mathcal{D} \leftarrow [ ]$ 
for  $t = 1, \dots$  do
    receive  $\mathcal{D}_t = \{(s_t^{(n)}, y_t^{(n)}) : n = 1, \dots, p\}$ 
    adaptation on the current device
    setting  $\mathcal{D}_T \leftarrow \mathcal{D}_t$ 
    follow adaptation on the meta-test device in
        Algorithm 1 or Algorithm 2 to obtain context
        vector  $\phi_T \rightarrow \phi_t$ 
    use  $\phi_t, \theta_t$  to demodulate  $\mathcal{D}_t^{\text{data}}$ 
    meta-learning phase
    add current dataset  $\mathcal{D}_t$  to meta-training dataset  $\mathcal{D}$ 
    as  $\mathcal{D} \leftarrow \bigcup_{k=1}^t \mathcal{D}_k$ 
    follow meta-learning phase in Algorithm 1 or
        Algorithm 2 to obtain shared parameter  $\theta_{t+1}$ 
end
```

---

devices without making any assumption about temporal correlations. In practice, the adaptive assignment of the number of pilots requires a low-rate control downlink channel to be available for transmission prior to uplink transmission. The proposed method is hence not applicable to IoT systems whose communication protocols do not allow for the downlink control channels. It is noted that IoT systems such as NB-IoT do include downlink control channels [43].

In the proposed scheme, at each slot  $t$ , a device transmits  $P_t$  pilots. The BS carries out demodulation of the data payload by using the demodulator  $p(s|y, \phi_t^{(p)}, \theta_t)$ , where the shared parameter  $\theta_t$  is obtained as discussed in Sec. IV-C and the context variable  $\phi_t^{(p)}$  is obtained by using  $p \leq P_t$  pilots via Algorithm 3. By trying different values of  $p = 1, \dots, P_t$ , the BS determines the minimum value of  $p \leq P_t$  such that demodulation of the data in set  $\mathcal{D}_t^{\text{data}}$  meets some reliability requirement. If such a value of  $p$  is found, then the BS assigns the number of pilots for the next slot as  $P_{t+1} = p$ . Otherwise,  $P_{t+1}$  is set to the maximum value  $P$ . The overall online meta-learning procedure with this pilot number selection scheme is summarized in Algorithm 4, and an illustration of the proposed adaptive pilot number selection strategy can be found in Fig. 5.

In practice, the reliability level can be estimated in different ways. For example, it can be obtained by evaluating the output of a cyclic redundancy check (CRC) field at the output of a decoder operating on the demodulated symbols from the payload  $\mathcal{D}_t^{\text{data}}$ . Here, a simpler approach is considered that uses directly the output of the demodulator  $p(s|y, \phi_t, \theta_t)$  without having to run a decoder. This is done by comparing the cross-

---

**Algorithm 4:** Few-Pilot Demodulator Learning via Online Meta-Learning with Adaptive Pilot Number Selection

---

**Input:** Data sets  $\{\mathcal{D}_t, \mathcal{D}_t^{\text{data}}\}$  for  $t = 1, 2, \dots$ ; step size hyperparameters  $\eta$  and  $\kappa$

**Output:** Learned parameter vector  $\theta_t$  and context vector  $\phi_t$ , for  $t = 1, 2, \dots$

---

```
initialize parameter vector  $\theta_1$ 
initialize the meta-training dataset  $\mathcal{D}$  as empty, i.e.,  $\mathcal{D} \leftarrow [ ]$ 
initialize number of transmitted pilots  $P_1 \leftarrow P$ 
for  $t = 1, \dots$  do
    receive  $\mathcal{D}_t = \{(s_t^{(n)}, y_t^{(n)}) : n = 1, \dots, P_t\}$ 
    adaptation on the current device
    for  $p = 1, \dots, P_t$  do
        setting  $\mathcal{D}_T \leftarrow \{(s_t^{(n)}, y_t^{(n)}) : n = 1, \dots, p\}$ 
        follow adaptation on the meta-test device in
            Algorithm 1 or Algorithm 2 to obtain context
            vector  $\phi_T \rightarrow \phi_t$ 
        if (reliability check passed) then
            | set  $P_{t+1} = p$  and exit
        else if (reliability check not passed) and
            ( $p = P_t$ ) then
            | set  $P_{t+1} = P$ 
    end
    use  $\phi_t, \theta_t$  to demodulate  $\mathcal{D}_t^{\text{data}}$ 
    meta-learning phase
    add current dataset  $\mathcal{D}_t$  to meta-training dataset  $\mathcal{D}$ 
    as  $\mathcal{D} \leftarrow \bigcup_{k=1}^t \mathcal{D}_k$ 
    follow meta-learning phase in Algorithm 1 or
        Algorithm 2 to obtain shared parameter  $\theta_{t+1}$ 
end
```

---

entropy loss (5) on the demodulated data

$$- \sum_{y \in \mathcal{D}_t^{\text{data}}} \max_s [\log p(s|y, \phi_t^{(p)}, \theta_t)] \quad (20)$$

to some prescribed threshold: if (20) is below a threshold, then the reliability check is considered successful.

## V. EXPERIMENTS

In this section, we provide numerical results in order to bring insights into the advantages of meta-learning<sup>1</sup>.

### A. Offline Meta-Learning: Binary Fading

We begin by considering the offline set-up and focusing on a simple example, in which the transmitter is ideal, i.e.,  $x_k = s_k$  and fading is binary, i.e., the channel  $h_k$  in (1) can take values  $\pm 1$ . We emphasize that this set-up is intended to yield useful intuitions on the operation of meta-learning in the simplest possible setting. A more realistic scenario is studied in the next subsection. In this set-up, pulse-amplitude modulation with four amplitude levels (4-PAM), i.e.,  $\mathcal{S} =$

<sup>1</sup>Code is available at <https://github.com/kclip/meta-demodulator>.



$\{-3, -1, 1, 3\}$ , is adopted. Pilot symbols in the meta-training dataset  $\mathcal{D}$  and meta-test dataset  $\mathcal{D}_T$  follow a fixed periodic sequence  $-3, -1, 1, 3, -3, -1, \dots$ , while transmitted symbols in the test set for the meta-test device are randomly selected from the set  $\mathcal{S}$ . The channel of the meta-test device is selected randomly between  $+1$  and  $-1$  with equal probability, while the channels for half of the meta-training devices are set as  $+1$  and for the remaining half as  $-1$ .

Other numerical details are as follows. The number of meta-training devices is  $K = 20$ ; the number of pilot symbols per meta-training device is  $N = 1000$ , which are divided into  $N^{\text{tr}} = 1$  training symbols and  $N^{\text{te}} = 999$  testing symbols. Only one pilot symbol is transmitted by the meta-test device, i.e.  $P = 1$ . The demodulator (4) is a neural network with  $L = 3$  layers, i.e., an input layer with 2 neurons, one hidden layer with 30 neurons, and a softmax output layer with 4 neurons. The network adopts a hyperbolic tangent function  $\sigma(\cdot) = \tanh(\cdot)$  as the activation function. For meta-learning, fixed learning rates  $\eta = 0.1$  and  $\kappa = 0.001$  are used with a single local update, i.e.,  $m = 1$ , and the ADAM optimizer [44] is used to update the shared parameter  $\theta$ . To train for the meta-test device, one adaptation step is adopted, i.e.,  $I_T = 1$ , with learning rate  $\eta = 0.1$ . The average signal-to-noise ratio (SNR) per real symbol is given as  $2E_x/N_0 = 18$  dB.

We compare the performance of the proposed meta-learning approach via MAML, FOMAML, REPTILE, and CAVIA with: (i) conventional learning where data from the meta-training devices are not used and the weights of the demodulator are randomly initialized; (ii) joint training with the meta-training dataset  $\mathcal{D}$  as explained in Sec. III-A; (iii) optimal ideal demodulator that assumes perfect CSI. In order to improve the performance of (i), instead of single adaptation step for the meta-test device, we allow for multiple SGD updates with learning rate 0.001; while, for (ii), we set the learning rate to 0.001 and the mini-batch size to 4. The probability of symbol error of the optimal demodulator (iii) can be computed as  $P_e = 3/2Q(\sqrt{\text{SNR}/5})$  using standard arguments.

In Fig. 6, we plot the symbol error rate with respect to number  $I$  of iterations during meta-training. Conventional learning and joint training curves are not shown since they both failed to reach symbol error rates smaller than 0.25. FOMAML and REPTILE curves are shown up to 400 and 1400 meta-training iterations, respectively, since further iterations for both schemes show unstable behavior [35], [45]. Despite having only one pilot for adaptation  $P = 1$ , both MAML and CAVIA can approach the performance of the optimal demodulator, with MAML outperforming CAVIA after sufficiently many meta-training iterations. This advantage of MAML stems from its adaptation of a larger number of parameters  $\phi_T$ , namely the demodulator weight vector, with respect to CAVIA, which only updates an auxiliary input vector. Overall, these results confirm the claim that, unlike conventional solutions, meta-training can effectively transfer information from meta-training devices to a new target device to achieve near-optimal demodulator with few pilots, here only one.

In order to gain intuition on how meta-learning learns from the meta-training devices, in Fig. 7, we plot the probabilities defined by the demodulator (4) for the four symbols in the

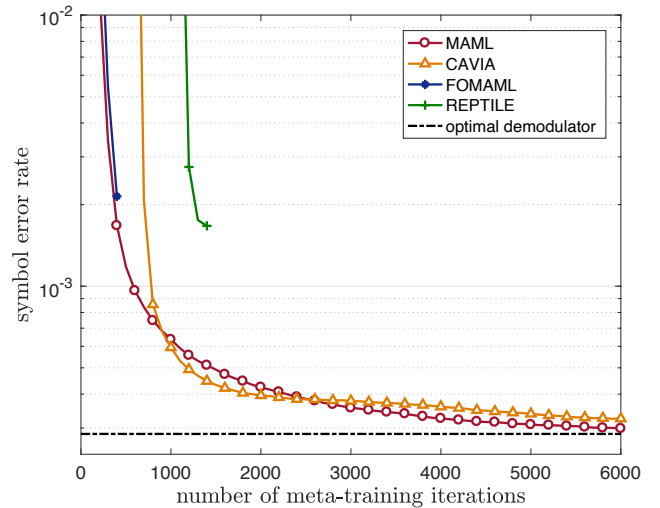


Fig. 6: Symbol error rate with respect to number of iterations during meta-training for an offline meta-learning example with binary fading.  $P = 1$  pilot is used for meta-test device. The symbol error rate is averaged over  $10^6$  data symbols and 100 meta-test devices. The unstable behavior of FOMAML and REPTILE after 400 and 1400 iterations, respectively, are not shown. The symbol error rate for conventional training and joint training are not shown as their symbol error rate is above 0.25.

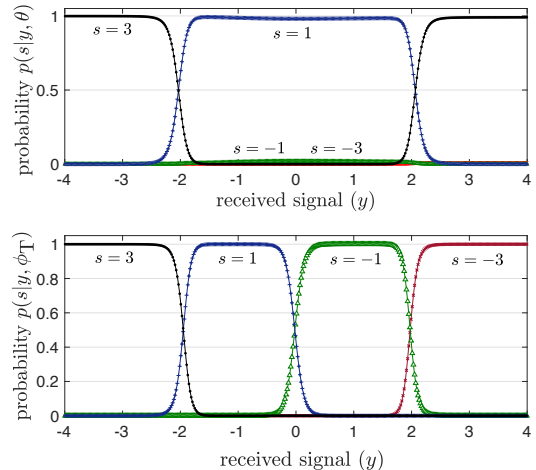


Fig. 7: (Top) Demodulator (4) for the shared parameter vector  $\theta$  obtained via offline meta-learning phase in Algorithm 1 using MAML; (Bottom) Updated demodulator (4) with target-device specific parameter vector  $\phi_T$  using  $P = 1$  pilot from the meta-test device.

constellation  $\mathcal{S}$  with the shared parameter vector  $\theta$  obtained from the meta-learning phase in Algorithm 1 (top) and with target-device specific parameter vector  $\phi_T$  after adaptation using the pilots of the target meta-test device (bottom). Here, MAML is adopted as the meta-learning algorithm. The class probabilities identified by meta-learning in the top figure have the interesting property of being approximately symmetric with respect to the origin. From this symmetric initialization, the resulting decision region can be adapted to the channel of

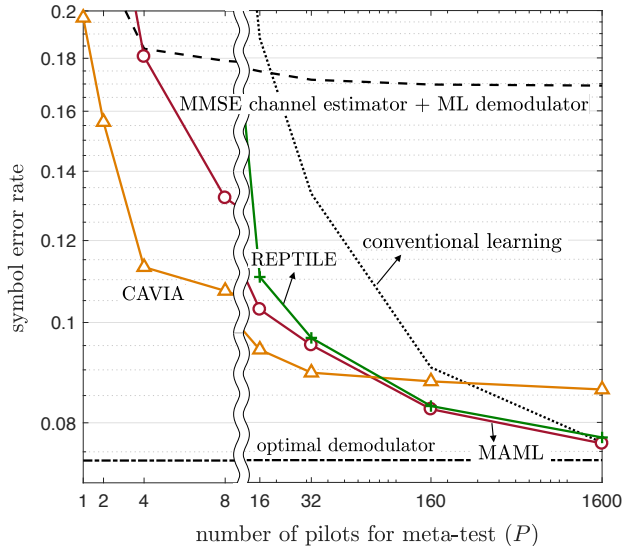


Fig. 8: Symbol error rate with respect to the number  $N^{\text{tr}} = P$  of training pilots used for both meta-training and meta-testing for offline meta-learning with 16-QAM, Rayleigh fading, and I/Q imbalance with  $K = 1000$  meta-training devices,  $N^{\text{tr}} + N^{\text{te}} = 3200$  pilots for meta-training devices. In order to plot average extra symbol error probability on the meta-test device, the symbol error rate is evaluated by averaging over 100 different meta-test devices (i.e., channel realizations), each with 10000 data symbols.

the target device, which may take values  $\pm 1$  in this example. The adapted probabilities in the bottom figure illustrate how the initial demodulator obtained via MAML is specialized to the channel of the target device.

### B. Offline Meta-Learning: Rayleigh Fading with I/Q Imbalance

We now consider a more realistic scenario including Rayleigh fading channels  $h_k \sim \mathcal{CN}(0, 1)$  and model (3) to account for I/Q imbalance at the transmitters. We set  $\epsilon_k = 0.15\epsilon'_k$  and  $\delta_k = 15^\circ\delta'_k$ , where  $\epsilon'_k$  and  $\delta'_k$  are independent random variables with beta distribution  $\text{Beta}(5, 2)$ . Note that this implies that  $\epsilon_k$  and  $\delta_k$  are limited in the intervals  $[0, 0.15]$  and  $[0, 15^\circ]$ , respectively. We assume 16-ary quadrature amplitude modulation (16-QAM) for constellation  $\mathcal{S}$ , and the sequence of pilot symbols in the meta-training dataset  $\mathcal{D}$  and meta-test dataset  $\mathcal{D}_{\text{T}}$  is fixed by cycling through the symbols in  $\mathcal{S}$ , while the transmitted symbols in the test set for the meta-test device are randomly selected from  $\mathcal{S}$ . The number of meta-training devices is set as  $K = 1000$ ; the number of pilot symbols per device is  $N = 3200$ , which are divided into  $N^{\text{tr}}$  training samples and  $N^{\text{te}} = N - N^{\text{tr}}$  testing samples. The average SNR per complex symbol is given as  $E_x/N_0 = 20$  dB. Further details on the numerical set-up can be found in Appendix B.

In Fig. 8, the symbol error rate with respect to the number  $P$  of pilots for the meta-test device is illustrated when using an equal number of pilots for meta-training, i.e.,  $N^{\text{tr}} = P$ . As in Fig. 6, we compare the performance of meta-training methods with conventional learning and joint training strategies, along

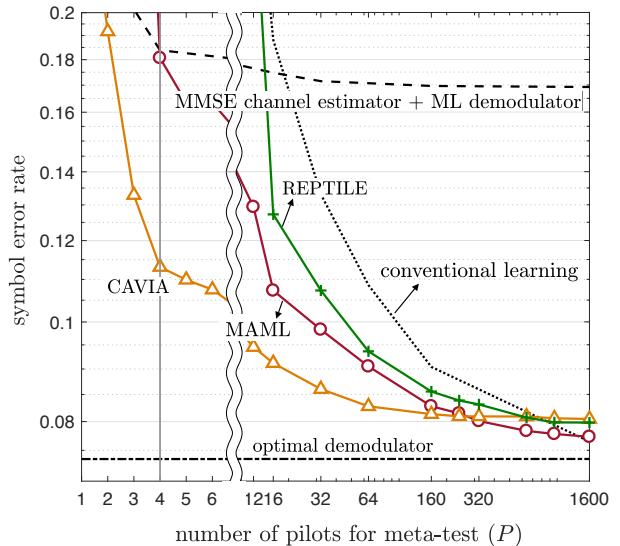


Fig. 9: Symbol error rate with respect to the number  $P$  of pilots (used during meta-testing) for offline meta-learning with 16-QAM, Rayleigh fading, and I/Q imbalance for  $K = 1000$  meta-training devices,  $N^{\text{tr}} = 4$  (vertical line),  $N^{\text{te}} = 3196$ . The symbol error is averaged over 10000 data symbols and 100 meta-test devices.

with a conventional communication scheme based on Minimum Mean Square Error (MMSE) channel estimation with  $P$  pilots followed by a Maximum Likelihood (ML) demodulator. All the schemes with worse performance as compared to this conventional communication scheme are not shown. MAML, REPTILE, and CAVIA are seen to adapt quickly to the channel and I/Q imbalance of the target device, outperforming the conventional scheme based on MMSE channel estimation, which is agnostic to the I/Q imbalance. MAML shows the best performance for sufficiently large  $P$ , while CAVIA is seen to outperform MAML when fewer pilots are available. It is worth noting that, when there is a sufficient number  $P$  of pilots for the meta-test device, conventional learning can outperform meta-learning schemes. In this case, the inductive bias inferred by meta-training can hence cause a performance degradation [22].

In Fig. 9, we consider the case where the number  $P$  of pilots for the meta-test device is different from the number  $N^{\text{tr}}$  used for meta-training, here set to  $N^{\text{tr}} = 4$ . Despite this mismatch between meta-training and meta-testing condition, MAML, CAVIA, and REPTILE are seen to outperform conventional communication when there is a sufficient number  $P$  of pilots for meta-test. In a manner similar to results in Fig. 8, CAVIA shows the best performance with extremely few pilots, e.g., 4 pilots, while MAML is preferable for larger values of  $P$ . In fact, comparing Fig. 8 and Fig. 9 reveals that having  $P > N^{\text{tr}}$  can even be advantageous for some meta-training schemes, such as CAVIA. This may be interpreted in terms of meta-overfitting, which refers to a degradation in meta-testing performance due to an excessive dependence of the meta-trained shared parameters on the meta-training data [46]. Using fewer pilots during meta-training can potentially reduce meta-

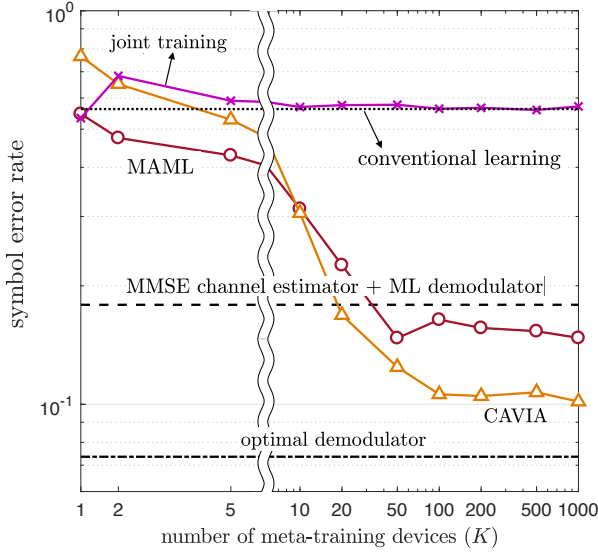


Fig. 10: Symbol error rate with respect to number  $K$  of meta-training devices for offline meta-learning with 16-QAM, Rayleigh fading, and I/Q imbalance with  $N^{\text{tr}} = 4$  and  $N^{\text{te}} = 3196$  for meta-training devices,  $P = 8$  pilots for meta-test devices. The symbol error is averaged over by 10000 data symbols and 100 meta-test devices.

overfitting, by making the shared parameters less dependent on meta-training data, and improve meta-testing performance.

Finally, in Fig. 10, the symbol error rate with respect to the number  $K$  of the meta-training devices is demonstrated. Joint training has a performance similar to conventional learning, hence being unable to transfer useful information from the  $K$  meta-training devices. In contrast, MAML and CAVIA show better performance when given data from more meta-training devices, up to a point where the gain saturates. This matches well with the intuition that there is only a limited amount of common information among different users that can be captured by meta-learning. Confirming the results in Fig. 8 and Fig. 9, MAML and CAVIA are seen to offer better performance than the conventional communication scheme with a sufficient number  $K$  of meta-training devices. Furthermore, CAVIA needs a larger value of  $K$  than MAML. This accounts again for CAVIA's architectural difference as compared to MAML: CAVIA needs to find a shared parameter vector  $\theta$  for the demodulator  $p(s|y, \phi_{\text{T}}, \theta)$  that is not adapted to the training symbols of the current device.

### C. Online Meta-Learning: Rayleigh Fading with I/Q Imbalance

We now move on to consider the online scenario under same assumptions on Rayleigh fading, transmitters' I/Q imbalance, modulation scheme, and SNR as in the offline set-up presented in Sec. V-B.

The maximum number of pilots is set as  $P = 32$ , and the number of pilots  $P_t$  in any slot  $t$  is determined by using adaptive pilot number selection scheme in Algorithm 4. In a manner similar to the offline set-up, we compare the performance with: (i) a conventional learning scheme that only adapts to current

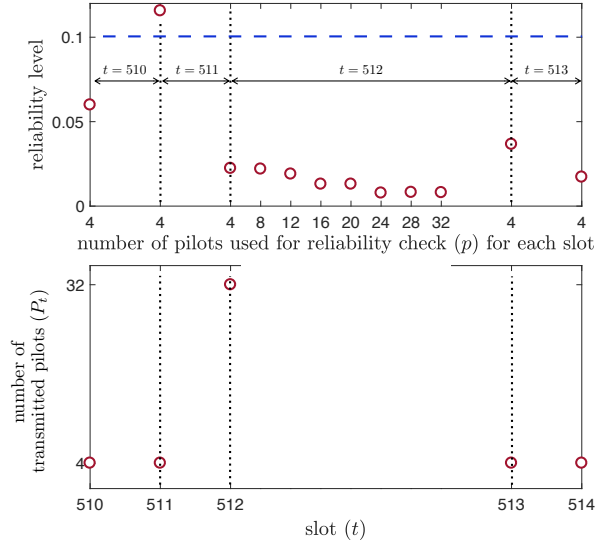


Fig. 11: Illustration of the procedure of adaptive pilot number selection scheme: (top) reliability level in (20) versus the number  $p$  of pilots during slots  $t = 510, \dots, 513$  (the prescribed reliability threshold value, 0.1, is dashed); (bottom) number of transmitted pilots  $P_t$  for each slot  $t = 510, \dots, 514$ .

device based on current pilot data  $\mathcal{D}_t$  with number of pilots  $P_t$  fixed as constant to a prescribed value; (ii) joint training as described in Sec. IV-B; (iii) conventional communication scheme with MMSE channel estimation and ML demodulator; and (iv) optimal ideal demodulator as described in the previous section. Other details on the numerical set-up can be found in Appendix B.

In Fig. 11, we first describe the procedure used by the proposed adaptive pilot number selection scheme. As discussed in Sec. IV-D, reliability levels for different values of the number  $p$  of pilots are evaluated by using (20) as shown in Fig. 11 (top) and the number of transmitted pilots  $P_{t+1}$  in the next slot is selected accordingly (bottom). The adaptive pilot number selection scheme is performed here with CAVIA, while the prescribed threshold value is set as 0.1. For instance, for slot 513, the number of transmitted pilots is chosen as  $P_{513} = 4$  based on the result from previous slot 512 that passed reliability check at  $p = 4$ . In contrast, for slot 512, the number of transmitted pilots  $P_{512} = P$  has been chosen as maximum value  $p = 32$  due to the failure of reliability check pass at slot 511. In the following, we assess whether the adaptive pilot number selection scheme can maintain reasonable performance in terms of probability of symbol error in the payload data  $\mathcal{D}_t^{\text{data}}$ , despite the illustrated reduction in the pilot overhead.

To this end, in Fig. 12, we plot the average symbol error rate for payload data  $\mathcal{D}_t^{\text{data}}$  versus the average number of transmitted pilots as evaluated during slots  $t = 500, \dots, 519$ . For MAML and CAVIA, the corresponding curve is obtained by selecting the threshold values (0, 0.01, 0.05, 0.1) and (0, 0.001, 0.02, 0.05, 0.1, 0.15), respectively, for the reliability level. For conventional learning, each point on the curve corresponds to the given fixed number of pilots defined by the

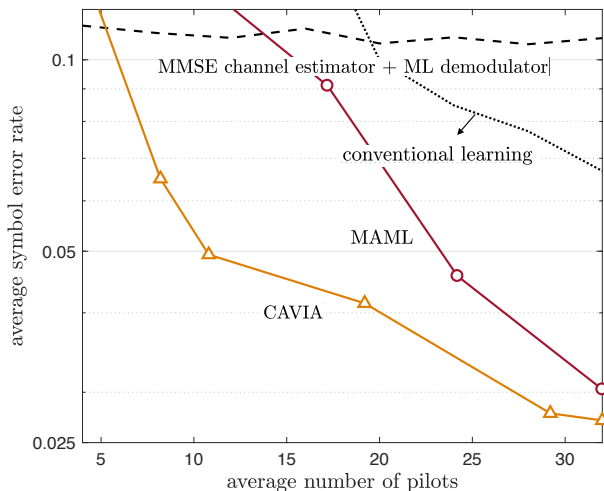


Fig. 12: Average symbol error rate with respect to average number of pilots over slots  $t = 500, \dots, 520$  for online meta-learning.

horizontal axis. The proposed adaptive pilot number selection scheme is seen to improve both over conventional learning and a conventional model-based demodulator. In particular, in a manner consistent with the discussion so far, MAML and CAVIA can operate with fewer pilots than conventional schemes, with CAVIA being generally more efficient in reducing the number of pilots due to its adaptation of a smaller number of parameters.

## VI. CONCLUSIONS AND EXTENSIONS

In communication systems with short packets, such as IoT, meta-learning techniques can adapt quickly based on few training examples by transferring knowledge from previously observed pilot information from other devices. In this paper, we have proposed the use of offline and online meta-learning for IoT scenarios by adapting state-of-the-art meta-learning schemes, namely MAML, FOMAML, REPTILE, and CAVIA, in a unified framework. For the online setting, meta-learning has been further integrated with an adaptive pilot number

selection scheme to reduce the pilot overhead. Extensive numerical results have validated the advantage of meta-learning in both offline and online cases as compared to conventional machine learning schemes. Moreover, comparisons among the mentioned meta-learning schemes reveal that MAML and CAVIA are preferable, with each scheme outperforming the other in different regimes in terms of amount of available meta-training data. For online meta-learning, the proposed pilot selection scheme was demonstrated to have a decreased pilot overhead with negligible performance degradation of the demodulator.

Meta-learning, first introduced in the conference version [1] of this work and in [19] for use in communications systems, may be useful in a number of other network functionalities characterized by reduced overhead and correlation across successive tasks. Examples include prediction of traffic from sets of IoT devices, e.g., in grant-free access [47], [48]; channel estimation [20]; and precoding in multi-antenna systems.

A complementary approach for reducing the pilot overhead involves model-based compressive sensing (CS) methods that leverage sparsity of the channel in some domain, typically space or multi-path dimensions [49], [50]. An interesting direction for future work would be to combine meta-learning with CS in order to further reduce the number of pilots for frequency-selective or multi-antenna channels. Furthermore, more advanced meta-training solutions can also be considered that are based on a probabilistic estimate of the context variables [34]. Finally, this work may motivate the development of novel meta-training techniques that reap the complementary benefits of CAVIA and MAML.

## APPENDIX A

### HESSIAN-VECTOR PRODUCT CALCULATION

In order to compute the updates in (11) and (12), we adopt a finite difference method for Hessian-vector product calculation [36]. This allows us to avoid computing Hessian matrix and obtain an approximate value of the product of the Hessian matrix and a vector. Given a loss function  $L(\theta)$  defined and doubly continuously differentiable over a local neighborhood of the value  $\theta$  of interest, the finite difference

TABLE I: Numerical Set-Up

	Figs. 6, 7	Figs. 8, 9, 10	Figs. 11, 12
number of meta-training iterations ( $I$ )	-	50000	500 (at each time slot)
number of local updates during meta-training ( $m$ )	1	1	1
number of local updates during meta-testing ( $I_T$ )	1	1000	1000
mini-batch size during meta-training	$N^{\text{tr}} = 1$	$N^{\text{tr}}$	$N^{\text{tr}} = 4$
mini-batch size during meta-testing for $i_T$ th update, for $i_T = 1, \dots, m$	1	$\min(P, N^{\text{tr}})$	$\min(P, N^{\text{tr}})$
mini-batch size during meta-testing for $i_T$ th update, for $i_T = m + 1, \dots, I_T$	-	$\min(P, 16)$	$\min(P, 16)$
learning rate $\eta$ during meta-training	0.1	0.1	0.1
learning rate $\kappa$ during meta-training	0.001	0.001	0.001
learning rate $\eta$ during meta-testing for $i_T$ th update, for $i_T = 1, \dots, m$	0.1	0.1	0.1
learning rate $\eta$ during meta-testing for $i_T$ th update, for $i_T = m + 1, \dots, I_T$	-	0.005	0.005



method approximates the Hessian-vector product  $Hg$ , where  $H = \nabla_{\theta}^2 L(\theta)$  is the Hessian matrix and  $g$  is any vector. The Hessian-vector product  $Hg$  can be approximately computed as [36]

$$Hg \approx \frac{1}{\alpha} (\nabla_{\theta} L(\theta + \alpha g) - \nabla_{\theta} L(\theta)), \quad (21)$$

where  $\alpha$  is a sufficiently small constant value. In (21), we follow [37] to choose  $\alpha$  as

$$\alpha = \frac{2\sqrt{\epsilon}(1 + \|\theta\|)}{\|g\|}, \quad (22)$$

where  $\|\cdot\|$  indicates Euclidean norm and  $\epsilon = 1.1920929e-7$ , which is an upper bound on the relative error due to rounding in single precision floating-point arithmetic [51].

## APPENDIX B DETAILS ON NUMERICAL SET-UP

### A. Offline Meta-Learning

The following is the fixed sequence that is used for pilot symbols in the meta-training dataset  $\mathcal{D}$  and meta-test dataset  $\mathcal{D}_T$  in the offline scenario for 16-QAM:  $-3-3j, -3+1j, 1+1j, 1-3j, -3+3j, 3+1j, 1-1j, -1-3j, 3+3j, 3-1j, -1-1j, -1+3j, 3-3j, -3-1j, -1+1j, 1+3j, -3-3j, -3+1j, \dots$ . Pilots in meta-test dataset  $\mathcal{D}_T$  and training set  $\mathcal{D}_k^{\text{tr}}$  follow this same fixed sequence. For the experiments in Sec. V-B, every demodulator (4) except for CAVIA is a neural network with  $L = 5$  layers, i.e., an input layer with 2 neurons, three hidden layers with 10, 30, 30 neurons, and a softmax output layer with 16 neurons. For CAVIA, we use a neural network with an input layer of 12 neurons, three hidden layers with 10, 30, 30 neurons, and a softmax output layer with 16 neurons, so that the dimension of the context parameter  $\phi$  is 10. For the activation function, we adopt a ReLU function as  $\sigma(\cdot) = \text{ReLU}(\cdot)$ . Detailed settings are described in Table I.

### B. Online Meta-Learning

We trained the demodulator (4) with the same architecture with same mini-batch sizes and learning rates described above for offline meta-learning. For this experiment, we set  $\mathcal{D}_k^{\text{e}} = \mathcal{D}_k$ . Other details are described in Table I.

## REFERENCES

- [1] S. Park, H. Jang, O. Simeone, and J. Kang, "Learning how to demodulate from few pilots via meta-learning," in *Proc. IEEE Signal Processing Advances in Wireless Commun. (SPAWC)*, Cannes, France, July 2019.
- [2] J. Östman, G. Durisi, E. G. Ström, M. C. Coşkun, and G. Liva, "Short packets over block-memoryless fading channels: Pilot-assisted or noncoherent transmission?" *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1521–1536, Feb. 2019.
- [3] M. Ibnkahla, "Applications of neural networks to digital communications—a survey," *Signal Processing*, vol. 80, no. 7, pp. 1185–1215, July 2000.
- [4] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Trans. Cognitive Commun. and Netw.*, vol. 4, no. 4, pp. 648–664, Nov. 2018.
- [5] S. Bouchired, D. Roviras, and F. Castanié, "Equalisation of satellite mobile channels with neural network techniques," *Space Communications*, vol. 15, no. 4, pp. 209–220, 1998/1999.
- [6] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognitive Commun. and Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.
- [7] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2017.
- [8] M. Khani, M. Alizadeh, J. Hoydis, and P. Fleming, "Adaptive neural signal detection for massive MIMO," *arXiv preprint arXiv:1906.04610*, 2019.
- [9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements Of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer, 2009.
- [10] A. G. Helmy, M. Di Renzo, and N. Al-Dhahir, "On the robustness of spatial modulation to I/Q imbalance," *IEEE Commun. Letters*, vol. 21, no. 7, pp. 1485–1488, July 2017.
- [11] S. Thrun, "Lifelong learning algorithms," in *Learning to Learn*. Springer, 1998, pp. 181–209.
- [12] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting gradient-based meta-learning as hierarchical bayes," *arXiv preprint arXiv:1801.08930*, 2018.
- [13] O. Simeone, S. Park, and J. Kang, "From Learning to Meta-Learning: Reduced Training Overhead and Complexity for Communication Systems," in *Proc. 6G Wireless Summit*, Lapland, Finland, Mar. 2020.
- [14] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 3630–3638, Barcelona, Spain, Dec. 2016.
- [15] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.
- [16] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.
- [17] L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson, "Fast context adaptation via meta-learning," in *Proc. International Conference on Machine Learning (ICML)*, pp. 7693–7702, Long Beach, California, June 2019.
- [18] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," *arXiv preprint arXiv:1902.08438*, 2019.
- [19] Y. Jiang, H. Kim, H. Asnani, and S. Kannan, "Mind: Model independent neural decoder," in *Proc. IEEE Signal Processing Advances in Wireless Commun. (SPAWC)*, Cannes, France, July 2019.
- [20] H. Mao, H. Lu, Y. Lu, and D. Zhu, "RoemNet: Robust meta learning based channel estimation in OFDM systems," in *Proc IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019.
- [21] Y. Yang, F. Gao, Z. Zhong, B. Ai, and A. Alkhateeb, "Deep Transfer Learning Based Downlink Channel Prediction for FDD Massive MIMO Systems," *arXiv preprint arXiv:1912.12265*, 2019.
- [22] S. Park, O. Simeone, and J. Kang, "Meta-learning to communicate: Fast end-to-end training for fading channels," in *Proc. IEEE 45th Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020.
- [23] S. Park, O. Simeone, and J. Kang, "End-to-End Fast Training of Communication Links Without a Channel Model via Online Meta-Learning," *arXiv preprint arXiv:2003.01479*, 2020.
- [24] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep HyperNetwork-Based MIMO Detection," *arXiv preprint arXiv:2002.02750*, 2020.
- [25] E. Costa and S. Pupolin, "M-QAM-OFDM system performance in the presence of a nonlinear amplifier and phase noise," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 462–472, Mar. 2002.

- [26] M. Windisch and G. Fettweis, "On the performance of standard-independent I/Q imbalance compensation in OFDM direct-conversion receivers," in *Proc. IEEE European Signal Processing Conference*, pp. 1–5, Antalya, Turkey, Sep. 2005.
- [27] D. Tandur and M. Moonen, "Joint adaptive compensation of transmitter and receiver IQ imbalance under carrier frequency offset in OFDM-based systems," *IEEE Trans. Signal Processing*, vol. 55, no. 11, pp. 5246–5252, Nov. 2007.
- [28] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
- [29] O. Simeone, "A brief introduction to machine learning for engineers," *Foundations and Trends in Signal Processing*, vol. 12, no. 3-4, pp. 200–431, 2018.
- [30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [31] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
- [32] S. Ravi and A. Beatson, "Amortized bayesian meta-learning," in *Proc. International Conference on Learning Representations (ICLR)*, New Orleans, United States, May 2019.
- [33] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. Turner, "Meta-learning probabilistic inference for prediction," in *Proc. International Conference on Learning Representations (ICLR)*, New Orleans, United States, May 2019.
- [34] C. Nguyen, T.-T. Do, and G. Carneiro, "Uncertainty in model-agnostic meta-learning using variational inference," *arXiv preprint arXiv:1907.11864*, 2019.
- [35] A. Antoniou, H. Edwards, and A. Storkey, "How to train your MAML," *arXiv preprint arXiv:1810.09502*, 2018.
- [36] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. pp. 9–48, Springer, 2012.
- [37] N. Andrei, "Accelerated conjugate gradient algorithm with finite difference Hessian/vector product approximation for unconstrained optimization," *Journal of Computational and Applied Mathematics*, vol. 230, no. 2, pp. 570–582, Aug. 2009.
- [38] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
- [39] P. Kairouz *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.
- [40] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.
- [41] J. Hannan, "Approximation to bayes risk in repeated play," *Contributions to the Theory of Games*, vol. 3, pp. 97–139, 1957.
- [42] O. Simeone and U. Spagnolini, "Adaptive pilot pattern for OFDM systems," in *Proc IEEE Int. Conf. Commun. (ICC)*, pp. 978–982, Paris, France, June 2004.
- [43] Y.-P. E. Wang, *et al.*, "A primer on 3GPP narrowband Internet of Things," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 117–123, Mar. 2017.
- [44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [45] N. S. Keskar and R. Socher, "Improving generalization performance by switching from ADAM to SGD," *arXiv preprint arXiv:1712.07628*, 2017.
- [46] R. Amit and R. Meir, "Meta-learning by adjusting priors based on extended PAC-Bayes theory," *arXiv preprint arXiv:1711.01244*, 2017.
- [47] R. Kassab, O. Simeone, and P. Popovski, "Information-centric grant-free access for IoT fog networks: Edge vs cloud detection and learning," *arXiv preprint arXiv:1907.05182*, 2019.
- [48] N. Jiang, Y. Deng, O. Simeone, and A. Nallanathan, "Online supervised learning for traffic load prediction in framed-ALOHA networks," *IEEE Commun. Letters*, vol. 23, no. 10, pp. 1778–1782, Oct. 2019.
- [49] X. Rao and V. K. N. Lau, "Compressive sensing with prior support quality information and application to massive MIMO channel estimation with temporal correlation," *IEEE Trans. Signal Processing*, vol. 63, no. 18, pp. 4914–4924, Sept. 15, 2015.
- [50] Z. Gao, L. Dai, Z. Wang and S. Chen, "Spatially common sparsity based adaptive channel estimation and feedback for FDD massive MIMO," *IEEE Trans. Signal Processing*, vol. 63, no. 23, pp. 6169–6183, Dec. 1, 2015.
- [51] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys (CSUR)*, vol. 23, no. 1, pp. 5–48, 1991.



**Sangwoo Park** (Student Member, IEEE) received his B.S. degree in physics in 2014 and the M.S.E degree in electrical engineering in 2016, all from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. He is currently working toward the Ph.D. degree at KAIST. His research interests lie in the field of communication theory, machine learning, and statistical signal processing, with emphasis on meta-learning usage for communication systems. From April to December 2019, he was a visiting researcher at King's Communications, Learning and Information Processing lab, King's College London, United Kingdom. He has served as a Reviewer for many journals, including the IEEE Journal on Selected Areas in Communications and the IEEE Transactions on Cognitive Communications and Networking.



**Hyeryung Jang** (Member, IEEE) received her B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, in 2010, 2012, and 2017, respectively. She is currently a research associate in the Department of Informatics, King's College London, United Kingdom. Her recent research interests lie in the mathematical modeling, learning, and inference of probabilistic graphical models, with a specific focus on spiking neural networks and communication systems. Her past research works also include network economics, game theory, and distributed algorithms in communication networks.



**Osvaldo Simeone** (Fellow, IEEE) is a Professor of Information Engineering with the Centre for Telecommunications Research at the Department of Engineering of King's College London, where he directs the King's Communications, Learning and Information Processing lab. He received an M.Sc. degree (with honors) and a Ph.D. degree in information engineering from Politecnico di Milano, Milan, Italy, in 2001 and 2005, respectively. From 2006 to 2017, he was a faculty member of the Electrical and Computer Engineering (ECE) Department at New Jersey Institute of Technology (NJIT), where he was affiliated with the Center for Wireless Information Processing (CWIP). His research interests include information theory, machine learning, wireless communications, and neuromorphic computing. Dr Simeone is a co-recipient of the 2019 IEEE Communication Society Best Tutorial Paper Award, the 2018 IEEE Signal Processing Best Paper Award, the 2017 JCN Best Paper Award, the 2015 IEEE Communication Society Best Tutorial Paper Award and of the Best Paper Awards of IEEE SPAWC 2007 and IEEE WRECOM 2007. He was awarded a Consolidator grant by the European Research Council (ERC) in 2016. His research has been supported by the U.S. NSF, the ERC, the Vienna Science and Technology Fund, as well as by a number of industrial collaborations. He currently serves in the editorial board of the IEEE Signal Processing Magazine and is the vice-chair of the Signal Processing for Communications and Networking Technical Committee of the IEEE Signal Processing Society. He was a Distinguished Lecturer of the IEEE Information Theory Society in 2017 and 2018. Dr Simeone is a co-author of two monographs, two edited books published by Cambridge University Press, and more than one hundred research journal papers. He is a Fellow of the IET and of the IEEE.



**Joonhyuk Kang** (Member, IEEE) received the B.S.E. and M.S.E. degrees from Seoul National University, Seoul, South Korea, in 1991 and 1993, respectively, and the Ph.D. degree in electrical and computer engineering from The University of Texas at Austin, Austin, in 2002. From 1993 to 1998, he was a Research Staff Member at Samsung Electronics, Suwon, South Korea, where he was involved in the development of DSP-based real-time control systems. In 2000, he was with Cwill Telecommunications, Austin, TX, USA, where he

participated in the project for multicarrier CDMA systems with antenna array. He was a Visiting Scholar with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA, from 2008 to 2009. He is currently serving as Head of the School of Electrical Engineering (EE), KAIST, Daejeon, South Korea. His research interests include signal processing for cognitive radio, cooperative communication, physical-layer security, and wireless localization. He is a member of the Korea Information and Communications Society and the Tau Beta Pi (the Engineering Honor Society). He was a recipient of the Texas Telecommunication Consortium Graduate Fellowship, from 2000 to 2002.