FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING
DEGREE PROGRAMME IN WIRELESS COMMUNICATIONS ENGINEERING

MASTER'S THESIS

META-LEARNING APPLICATIONS FOR MACHINE-TYPE WIRELESS COMMUNICATIONS

Author	Hebatalla Issa
Supervisor	Mohammed Shehab
Second Examiner	Hirley Alves

June 2023

Issa H. (2023) Meta-Learning Applications for Machine-Type Wireless Communications. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Wireless Communications Engineering. Master's Thesis, 54 p.

ABSTRACT

Machine Type Communication (MTC) emerged as a key enabling technology for 5G wireless networks and beyond towards the 6G networks. MTC provides two service modes. Massive MTC (mMTC) provides connectivity to a huge number of users. Ultra-Reliable Low Latency Communication (URLLC) achieves stringent reliability and latency requirements to enable industrial and interactive applications. Recently, data-driven learning-based approaches have been proposed to optimize the operation of various MTC applications and allow for obtaining the desired strict performance metrics. In our work, we propose implementing meta-learning alongside other deep-learning models in MTC applications. First, we analyze the model-agnostic meta-learning algorithm (MAML) and its convergence for regression and reinforcement learning (RL) problems. Then, we discuss uncrewed aerial vehicles (UAVs) trajectory planning as a case study in mMTC and RL, illustrating the system model and the main challenges. Hence, we propose the MAML-RL formulation to solve the UAV path learning problem. Moreover, we address the MAML-based few-pilot demodulation problem in massive IoT deployments. Finally, we extend the problem to include the interference cancellation with Non-Orthogonal Multiple Access (NOMA) as a paradigm shift towards non-orthogonal communication thanks to its potential to scale well in massive deployments. We propose a novel, data-driven, meta-learning-aided NOMA uplink model that minimizes the channel estimation overhead and does not require perfect channel knowledge. Unlike conventional deep learning successive interference cancellation (SICNet), Meta-Learning aided SIC (meta-SICNet) can share experiences across different devices, facilitating learning for new incoming devices while reducing training over- head. Our results show the superiority of MAML performance in addressing many problems compared to other deep learning schemes. The simulations also prove that MAML can successfully solve the few-pilot demodulation problem and achieve better performance in terms of symbol error rates (SERs) and convergence latency. Moreover, the analysis confirms that the proposed meta-SICNet outperforms classical SIC and conventional SICNet as it can achieve a lower SER with fewer pilots.

Keywords: machine type communications, meta-learning, MAML, NOMA, SIC, modulation, pilot allocation, UAVs trajectory.

TABLE OF CONTENTS

ABSTRACT
TABLE OF CONTENTS
FOREWORD
LIST OF ABBREVIATIONS AND SYMBOLS
1 INTRODUCTION 7
1.1 Literature Review 7
1.2 Contributions 10
1.3 Outlines 11
2 MODEL AGNOSTIC META-LEARNING AND REINFORCEMENT LEARNING 12
2.1 MAML Algorithm 13
2.1.1 MAML for Regression and Classification 14
2.1.2 MAML for RL 15
2.2 RL Algorithms 16
2.2.1 DQN Approach 16
2.2.2 VPG Approach 17
2.2.3 TRPO Approach 17
2.3 Case Study: UAV Trajectory Planning 18
2.4 Numerical Results 20
2.4.1 MAML Algorithm 20
2.4.2 UAV Path Planning Using RL 21
3 META-LEARNING DEMODULATION FROM FEW PILOTS 26
3.1 System Model 27
3.2 Meta-Learning Algorithm 28
3.3 Simulation Results 29
3.3.1 Symmetric Channel 29
3.3.2 Rayleigh-Fading Channel 31
4 META-LEARNING BASED FEW PILOTS DEMODULATION AND INTERFERENCE CANCELLATION FOR NOMA UPLINK 36
4.1 System Layout and Problem Formulation 36
4.1.1 System Layout 36
4.1.2 Problem Formulation 37
4.2 Deep Learning-Based SIC 38
4.2.1 SICNET 38
4.2.2 Proposed Meta-Learning Approach 39
4.3 Numerical Analysis 40
5 CONCLUSION 46
6 REFERENCES 48

FOREWORD

This thesis presents meta-learning applications for machine-type communication devices. It was done at the Center of Wireless Communications (CWC) at the University of Oulu, Finland. I would like to thank Dr. Hirley Alves for giving me a chance to join the MTC research group and for his guidance during the work. And special thanks to my supervisor and instructor Dr.Mohammad Shehab for his support and guidance all the time.

Oulu, 26th June, 2023

Hebatalla Issa

LIST OF ABBREVIATIONS AND SYMBOLS

5G	Fifth Generation of Cellular Communication
AMR	Automatic modulation recognition
AoI	Age of information
BS	Base station
CG	Conjugate gradient
CNN	Convolutional Neural Networks
CSI	Channel state information
DDPG	Deep deterministic policy gradient
DNN	Deep Neural Networks
DQN	Deep Q-Network
DRL	Deep reinforcement learning
FL	Federated Learning
HTC	Human-Type Communication
IoT	Internet of things
IoV	Internet of vehicles
KPIs	Key performance metrics
LR	Learning rate
LSTM	Long short term memory
MAML	Model-agnostic meta-learning
MARL	Multi-agent reinforcement learning
MDP	Markov decision process
MIMO	multiple-input and multiple-output
MMSE	Minimum-mean square error
MTC	Machine-Type Communication
NOMA	Non-orthogonal multiple access
NR	5G New Radio
OFDM	Orthogonal frequency division multiplexing.
OTFS	Orthogonal Time Frequency Space modulation.
PAM	Pulse Amplitude Modulation
PG	Policy gradient
QAM	Quadrature Amplitude Modulation.
QoS	Quality of service
ReLU	Rectified linear unit activation function
RL	reinforcement learning
RW	Random walk
SIC	Successive interference cancellation
SISO	Single-input single-output
TRPO	Trust region policy gradient
UAV	Unmanned aerial vehicles
VPG	Vanilla policy gradient
WGN	White Gaussian noise
WSN	Wireless sensor network
SNR	Signal-to-noise ratio
$\mathcal{S}$	a set of finite environment states s.

$A(s)$	a set of possible actions.
$R(s)$	a real-valued reward function
$P(s', s\|a)$	a transition probability model
$Q(s, a)$	action-value function
$\mathcal{T}$	a set of tasks
$p(\mathcal{T})$	a distribution over tasks $\mathcal{T}$
$\mathcal{L}$	a set of loss functions
$q(s_1)$	a distribution over initial observations (states)
$q(s_{t+1}\|s_t, a_t)$	a state transitional distribution
H	the task horizon
f_θ	a function parameterized by θ
θ	general learning parameter
θ'	task adaptive parameter
α	outer learning rate (step size)
β	inner learning rate
η	test adaptation learning rate
γ	discount factor
ϵ	exploration rate
$J(\theta)$	reward (cost) function
π_θ	Policy function parameterized by θ
A^{π_θ}	the advantage function for the current policy
$V^\pi(s)$	the policy value function
$Q^\pi(s\|a)$	the policy action-value function
h_k	Complex channel gain between device k and the BS.
n_k	complex additive white Gaussian noise.
$\mathcal{CN}$	Complex normal distribution.
ζ	Noise variance.
s_k	The complex symbol transmitted by the device to the BS.
$\mathcal{D}$	Meta-training dataset.
$\mathcal{D}_T$	Meta-testing dataset.
$\mathcal{D}_k^s$	Meta-training support set for device k.
$\mathcal{D}_k^Q$	Meta-training query set for device k.
N_{tr}^Q	size of the training query set.
N_{tr}^S	size of the training support set.
$p_\theta(s^{(n)}\|y^{(n)})$	Probabilistic demodulator.
N_{te}	Number of pilots transmitted by target/test device.
$x_l^(k)$	The l^th symbol transmitted from the k^th device
L	The number of IoT devices per each meta-group.
P_l	the transmitted power for each device.
$\mathbf{x_k}$	The superimposed channel input signal
ψ_i	the i^{th} constellation symbol in the constellation space.
$\mathbf{p}_l$	the conditional distribution of the corresponding symbol l
M	the modulation order.
$\sigma(.)$	Activation function.

1 INTRODUCTION

The primary goal of the fifth generation, introduced as 5G new radio (NR), is to serve Macine-Type Communication (MTC) besides Human-Type Communication (HTC) [1]. MTC allows the interconnection between devices and sensors without the need for human intervention, therefore allowing for a wide variety of applications related to the Internet of Things (IoT) [2]. MTC use cases can be classified into two main categories: ultra-reliable low latency communication (URLLC), also known as critical machine-type communication (cMTC), and massive machine type communication (mMTC) service classes in 5G NR [3]. However, the need for more stringent and robust performance requirements has led to the design and development of the 6G wireless network. The methodology of an MTC-optimized 6G network exploits the advanced machine learning (ML) tools combined with enhanced 5G technologies such as non-orthogonal multiple access techniques (NOMA).

Machine learning (ML) and artificial intelligence (AI) introduce a wide range of tools such as deep learning, deep reinforcement learning, and distributed learning. that can be utilized to effectively achieve the enhanced MTC applications' key performance indicators (KPIs). Additionally, meta-learning combined with other ML tools allows significant enhancement in the performance of MTC applications improving the latency and the learning time. In this context, implementing meta-learning in applications like UAV path learning, demodulation, and NOMA techniques becomes a key enabler in MTC applications.

This section will cover a detailed literature review of the machine learning-based approaches used in 5G and beyond wireless communication systems which enable the deployment of MTC applications. Afterward, our novel contributions are presented briefly. Finally, The outline of the whole thesis is presented.

1.1 Literature Review

Recently, deep learning has become pervasive in communication to enhance the performance of communication systems including channel estimation [4–6], channel equalization [7], modulation detection [8, 9]. In [9], the authors presented various deep-learning models for automatic modulation recognition (AMR) for multiple-input multiple-output (MIMO) communication systems which achieve better performance than the conventional recognition methods. They show that deep neural network (DNN) and convolutional neural networks (CNN) models can be effectively optimized by transfer learning, few-shot learning, cooperative learning, and data enhancement. However, applying deep learning in such systems introduces many challenges including the model complexity, the training latency, and the models' adaptability to varying conditions like channel models. Moreover, DRL, which combines deep learning based on DNNs and reinforcement learning, has obtained huge potential in many wireless applications thanks to its ability to effectively find the optimal or near-optimal policies in applications with high-dimensional state and action spaces and limited computational resources [10]. DRL has the ability to speed up the learning process and reduce the storage required for the large action space, hence papers [11–13] discuss the applications of DRL in IoT, cloud computing, smart vehicular communications, etc. Therefore, several IoT wireless systems were studied for the

possibility of applying RL and DRL such as Internet of vehicles (IoV) [14, 15], cognitive radio [16], autonomous IoT [12], wireless sensor networks (WSN) [17], and more.

Furthermore, distributed learning has emerged as an effective solution for the large training datasets that are considered a bottleneck in classic centralized machine learning techniques. The use of distributed learning, and specifically federated learning (FL), reduces the need for high computational resources by enabling edge devices to mutually train their models without transporting data across the network, thus enhancing the traffic overhead, latency, and data privacy [18]. The distributed learning underlying frameworks can be exploited in many wireless applications, especially for IoT cases where devices are connected to perform a similar coordinated task (e.g., swarms of drones or vehicles). Multiple approaches have been introduced to enable distributed learning for wireless applications, for example, the multi-agent reinforcement learning (MARL) framework [19, 20] has been widely used to solve different distributed optimization problems. In this case, deep neural networks are combined with reinforcement learning to enable the learning agents to cooperatively perform a specific task without sharing the whole collected data over the network. The use cases include distributed channel access [21], cooperative edge caching [22], and resource allocation for UAV network [23]. In [23], the authors formulate a QoS constraint energy efficiency optimization problem for multi-UAV networks where a MARL-based resource allocation algorithm is utilized for UAV agents. In [24], the UAVs deployed as relay nodes between IoT sensors and the base station are jointly trained to learn the best trajectory that minimizes the age of information (AoI) to maintain the freshness of data transmitted by the sensors to the base station. Moreover, FL was proposed to allow cooperative learning between a cluster of agents by exchanging their local model parameters (weights) instead of raw data [25]. However, FL faces many challenges including complex computational algorithms and maintaining high accuracy without sharing raw data between learning agents. To overcome these challenges, novel FL techniques like federated multi-task learning and model agnostic meta-learning based FL [18] are introduced to preserve privacy while achieving high accuracy and low computational overhead. MAML-based FL tends to utilize the meta-learning approach to find the best-suited ML model for all edge/IoT devices after a few training iterations. Therefore, each device can use its own data to update this general model within a few steps of gradient descent.

The concept of meta-learning can be consolidated with different deep-learning approaches to find a common model which can quickly adapt to the data variations that happen due to the change in environment, variations of channel conditions due to mobility for instance. For example, meta-reinforcement learning aims to generalize the learning experience to new tasks and new environments that have never been encountered during training time [26]. In [27], a proposed MAML-based RL algorithm is used for edge caching in vehicular networks to rapidly adapt to the change in content's popularity due to the mobility of vehicles and the alteration in users' preferences. Furthermore, meta-reinforcement learning is proposed as a potential candidate approach to effectively solve the drones' optimal trajectory planning for energy-constrained drones operating in dynamic networks [28]. The simulation in [28] proves that meta-learning enhances the RL solution and provides faster convergence to unknown environments with low computational complexity.

Despite the recent advances, there are many open challenges concerning NOMA in massive IoT deployments such as latency, complexity of receivers, perfect channel knowl-

edge, and effective utilization of radio resources. These challenges prompted researchers to look into data-driven approaches to handle multi-user connectivity [29–32]. We should note that data-driven learning-based communication systems aim to coexist or replace conventional model-based approaches thanks to their low complexity and ability to adapt well to varying channels. Moreover, the learning-based models can be easily customized to any environment, however, at the cost of new training instances. For instance, [31] argues that although linear processing has been effective in NOMA systems, non-linear processing is sometimes necessary to maintain good performance. The authors propose a neural network architecture that leverages the benefits of both linear and non-linear processing leading to efficient real-time detection performance. The authors in [30] introduce a deep-learning-based user detection solution (DeepMuD) for the uplink in massive MTC NOMA. The proposed DeepMuD employs an offline-trained long short-term memory (LSTM)-based network for multi-user detection without the need for perfect channel state information (CSI). Interestingly, authors report that DeepMuD improves error performance compared to conventional detectors and becomes even better as the number of devices increases.

Authors in [32] design an efficient and high-performance sparse Bayesian learning-based multi-user detection grant-free NOMA for mMTC able to handle scenarios with relatively high or quasi-sparse user activity. Their findings demonstrate that the proposed method outperforms traditional solutions, particularly when the number of active users is unknown and relatively high. NOMA with SIC has been studied using many methods [33]. For instance, the authors of [34] studied the performance of NOMA for short packet transmission. Authors in [29] introduce a deep learning approach for estimating the symbols named SICNet in the downlink NOMA. Contrary to conventional SIC, SICNet replaces the interference cancellation blocks with DNNs to infer the soft information representing the interfering symbols in a data-driven fashion, yielding robustness against changes in the number of users and power allocation.

More recently, ML-based transceivers have gained lots of interest. For instance, the authors of [35] proposed a meta-SICNet detection scheme for massive MIMO. The results showed that their meta-SICNet outperforms the conventional MMSE detector. Furthermore, the work in [36] investigates learning-based transceivers using joint learning and meta-learning techniques. An important limitation of the autoencoder-based approach to end-to-end training is that training should be generally carried out from scratch for each new channel. Despite the enhanced performance, such models can be trained on an end-to-end basis irrespective of the underlying modulation or multiplexing scheme. However, when relying on conventional training schemes, these models are required to be trained from scratch for each separate transmission condition, which is cumbersome in MTC networks with heterogeneous traffic and radio resources. Therefore, such networks would benefit from a learning scheme that accumulates experience to facilitate learning in new conditions.

In this context, meta-learning was introduced [37–39], also known as learning to learn. Often communications networks are expected to operate under a variety of system configurations. Using conventional learning requires the training of a separate model for each system configuration, leading to data and training time and processing costs. Owing to efficiency, we want to train a single model that would perform across all configurations. However, for such joint training, there may not exist a model that is able to perform well. On the other hand, meta-learning uses data from multiple configurations to infer a

model class and learning procedure, enabling learning on configurations of interest. For example, few-shot classification aims to infer a learning procedure that trains a classifier under limited training data from each class. To do so, the learning procedure is inferred from meta-training data that quickly train a classifier on meta-training tasks, rather than training a single model to classify across all tasks [37, 38]. More specifically, in a wireless communication setting, meta-learning outperforms (with respect to symbol error probability) conventional and joint training in IoT scenarios where devices transmit few pilots while adapting to non-linearities and fading. Learning to communicate on noisy or fading channels requires training to be carried out from scratch for every channel, for example, using pilot symbols. In this case, joint training is equivalent to non-coherent transmission.

In [40], authors address the sensing and fusion problem in massive MTC, which aims to collect and process a large amount of information and extract key features representing the observed process. However, sensing and fusion impose communication overheads and data redundancy to perform a given accuracy. The authors then proposed a meta-learning adaptive sensing and reconstruction framework that leverages prediction error and sensing decisions so to reduce the amount of communication overhead while guaranteeing robustness. Besides, they show that meta-learning-based approaches outperform conventional machine-learning algorithms in terms of convergence rate.

1.2 Contributions

The main contribution of the thesis is implementing the model agnostic meta-learning method to enhance the performance and learning experience for MTC applications. We developed an end-to-end, low-latency meta-learning scheme for few-pilot modulation detection to support the deployment of a massive number of IoT devices transmitting their data over fading channels. Additionally, we build upon [29, 30, 41] and propose a novel few pilot-aided detection mechanism, named meta-SICNet, for NOMA uplink in massive MTC. Herein, as in [29], we infer the interfering symbols via deep neural networks. However, we can reduce training over the network by the use of meta-learning, and in addition, as in [41], we reduce the number of pilots required for detection. Therefore, the three main pillars of our contributions are

- We propose a data-driven approach that applies meta-learning for NOMA uplink transceiver design.
- The network model consists of multiple MTC devices transmitting their superimposed signal to the BS. Then, the meta-SICNet framework in the BS learns to recover the transmitted symbols using the meta-training and adaptation approach.
- The meta-SICNet is able to decode symbols with fewer pilots, in the presence of interference, compared to basic SIC and conventional SICNet methods.
- Extensive simulation results elucidate that the proposed meta-SICNet scheme significantly outperforms both classical SIC and conventional SICNet in terms of outage probability and requires a lower number of pilots (down to 2 pilots).

1.3 Outlines

The rest of the thesis is organized as follows: Chapter 2 illustrates the meta-learning techniques focusing on the model agnostic meta-learning with a simulation example, the chapter also investigates the application of reinforcement learning for UAV path planning and discusses the possibility of optimizing the solution using meta-RL. Chapter 3 introduces the meta-learning demodulation techniques from few-pilot data. Chapter 4 discusses the interference cancellation and few-pilot demodulation for NOMA using meta-learning. Finally, the summary and the thesis conclusions are presented in Chapter 5.

2 MODEL AGNOSTIC META-LEARNING AND REINFORCEMENT LEARNING

Machine learning is a massive field that aims to replicate human intelligence into machines or software programs. To achieve that, the learning process should be constructive and generalized, thus shared between the learning agents. Therefore, meta-learning is proposed to be combined with other machine-learning algorithms in order to allow a near-human learning experience.

The key idea of meta-learning, or learning to learn, is to generalize the learning experience to new tasks and new environments that have never been encountered during training time. This holds when the learning involves tasks from the same class but yet different tasks. For example, the well-known image classification task can be solved using few-shot meta-learning where the model trained on specific image classes such as cats learns to classify new non-cats image classes [42, 43]. In [42], the MAML algorithm is used to generate synthetic image data with artificial labels to allow unsupervised fast learning using the meta-learning concept. The model outperforms all the Omniglot and Mini-Imagenet few-shot learning benchmarks. Another example is the maze navigation problem which was well-studied as a reinforcement learning problem, the meta-learning approach can be applied to allow the RL agent to navigate new mazes and reliably determine the exit after training on a few maze examples [44].

Furthermore, reinforcement learning is a machine learning field that allows an agent to interact with the environment and learns from trial and error by observing the effect of its own actions and experiences, the same way a human learns something new. The goal of RL is to find a suitable action model that would maximize the total cumulative reward of the agent. The basic elements of the RL problem are described using the Markov decision process (MDP) framework [45]. An MDP consists of

- a set of finite environment states $\mathcal{S}$,
- a set of possible actions $A(s)$ in each state,
- a real-valued reward function $R(s)$, and
- a transition model $P(s^{'}, s|a)$.

Therefore, RL algorithms utilized the mathematical formulation of the MDP to accomplish the learning process. RL algorithms are divided into two classes: model-based and model-free RL. Model-based learning attempts to model the environment and then choose the optimal policy based on its learned model [46], model-based RL algorithms leverage the learned model to simulate and plan future trajectories, enabling more efficient and sample-efficient learning without relying on try-and-error. However, in model-free learning the agent doesn't follow a particular model, instead, it depends on trial-and-error for finding the optimal policy [47]. Since real-world environments don't often have any prior knowledge of the environment dynamics and conditions, RL problems are mostly based on model-free approaches which are categorized into policy optimization methods and value-based (Q-learning) methods.

In policy optimization methods, the optimal policy is computed by handling the policy directly using policy gradient (PG) algorithms [48], while the value-based approach learns the action-value function $Q(s, a)$ and hence discovers the optimal policy by realizing

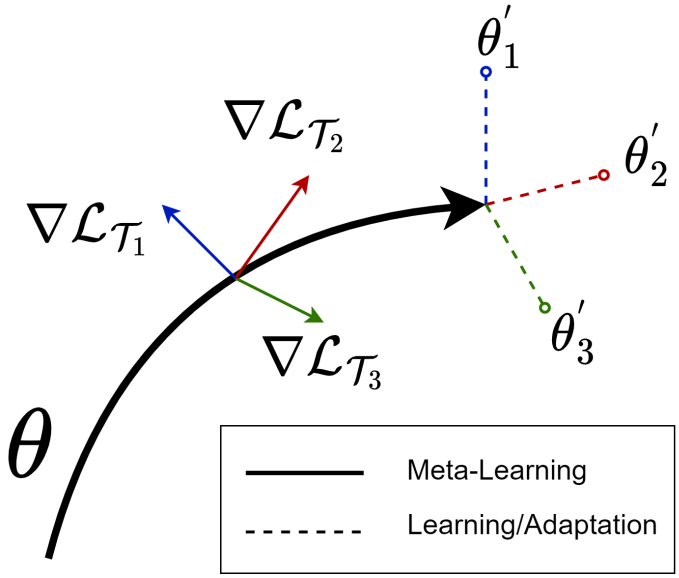


Figure 1: Diagram of MAML Algorithm.

the optimal value function. Policy-based RL is more efficient when dealing with high dimensional or continuous action spaces, it's also beneficial in learning stochastic policies. On the other hand, the value-based RL is most effective in sample efficiency and stability [49]. Examples of policy gradient algorithms are vanilla policy gradient (VPG) [50] and trust-region policy gradient (TRPO) [51].

In Section 2.1, we describe a popular meta-learning approach known as MAML introducing the algorithm and its mathematical formulation. Then, we discuss the MAML for reinforcement learning algorithm and the experimental evaluation of the MAML algorithm for regression and RL problems. Section 2.2 introduces some RL approaches and their algorithmic formulation. Section 2.3 discusses the case study of UAV path learning for minimizing the age of information. Afterward, the simulation results are presented and discussed in Section 2.4.

2.1 MAML Algorithm

The aim of meta-learning is to provide learning from experience and quick adaptation to new tasks from only a few examples. To achieve this goal, the Model agnostic meta-learning is proposed as a general framework that can be directly used for any learning problem or combined with a learning model that is trained with gradient descent procedures such as classification, regression, and policy gradient reinforcement learning. Therefore, the MAML algorithm leads to fast learning on a new task by performing a few gradient updates.

To formulate a general problem setting for the MAML algorithm, a model f that maps observations s to outputs a is trained to adapt to a large number of tasks. Each task $\mathcal{T} = \{\mathcal{L}(\mathrm{s}_1, \mathrm{a}_1, ..., \mathrm{s}_\mathrm{H}, \mathrm{a}_\mathrm{H}), q(\mathrm{s}_1), q(\mathrm{s}_{\mathrm{t}+1}|\mathrm{s}_\mathrm{t}, \mathrm{a}_\mathrm{t}), H\}$ is a set of a loss function $\mathcal{L}$, a distribution over initial observations $q(\mathrm{s}_1)$, a transitional distribution $q(\mathrm{s}_{\mathrm{t}+1}|\mathrm{s}_\mathrm{t}, \mathrm{a}_\mathrm{t})$, and a task horizon H. The model parameters are summarized in table 1. The meta-learning model is

Table 1: Parameters of The MAML Algorithm

Parameter	Definition
$\mathcal{T}$	a set of tasks
$p(\mathcal{T})$	a distribution over tasks $\mathcal{T}$
$q(s_1)$	a distribution over initial observations (states)
$q(s_{t+1}\|s_t, a_t)$	a state transitional distribution
H	the task horizon
$\mathcal{L}$	a set of loss functions
f_θ	a function parameterized by θ
θ	general learning parameter
θ'	task adaptive parameter
α	outer learning rate (step size)
β	inner learning rate

presented by a parameterized function f_θ with vector parameters θ, and the model is trained over a distribution of tasks $p(\mathcal{T})$. When the model adapts to a new task $\mathcal{T}_i$, the model's parameters θ turns to be $\theta_i^{'}$ which is computed using one or more gradient update steps on the task $\mathcal{T}_i$. When applying only one gradient step, the adaptive parameter $\theta_i^{'}$ is computed as

$$\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta) \tag{1}$$

The meta training aims at optimizing the loss function of $f_{\theta_i^{'}}$ with respect to θ over the sampled tasks from $p(\mathcal{T})$ as follows,

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i^{'}}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\Big(f_\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)\Big) \tag{2}$$

The final step is to update the meta parameters across tasks using stochastic gradient descent (SGD) [52] as follows,

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i^{'}}) \tag{3}$$

Figure 1 represents the optimization of the meta-parameter θ to enable quick adaptation to new tasks [37]. The algorithm is summarized in **algorithm 1**.

2.1.1 MAML for Regression and Classification

As established in our discussion for MAML algorithm, MAML can be used for supervised tasks to learn the function representation of a few input/output pairs of specific tasks using the experience acquired from training on similar tasks. For instance, in few-shot regression, the learning objective is to anticipate the outcomes of a continuous function utilizing a few datapoints sampled from that function, after the meta-training on other functions that have similar statistical properties. To formulate the supervised classification/regression problem, we adjust the horizon $H = 1$ and drop the time subscript from

Algorithm 1: MAML Algorithm

Input: distribution over tasks $p(\mathcal{T})$; step size hyperparameters α and β.
Output: Learning parameter vector θ.

Randomly initialize the parameter vector θ.
START META-TRAINING:
while *not done* **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$.
  for *each* $\mathcal{T}_i$ **do**
    Calculate the gradient $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples.
    Compute adapted parameters θ' using (1) as

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

  end
  update the meta-parameter θ:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_\theta')$$

end
END META-TRAINING.

the observation s. The loss function can be defined as a mean-squared error (MSE) loss or cross-entropy loss, the MSE loss for regression problems can be defined as

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{s^{(j)}, y^{(j)} \sim \mathcal{T}_i} ||f_\phi(s^{(j)}), y^{(j)}||_2^2 \tag{4}$$

where $s^{(j)}), y^{(j)}$ represents the input-output pair sampled from the task $\mathcal{T}_i$, the cross-entropy loss function for classification is expressed as

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{s^{(j)}, y^{(j)} \sim \mathcal{T}_i} y^{(j)} \log f_\phi(s^{(j)}) + (1 - y^{(j)})(1 - \log f_\phi(s^{(j)})). \tag{5}$$

2.1.2 MAML for RL

To apply MAML for RL problems, a few-shot meta-learning approach is adopted to enable the agent to quickly learn a policy for a new task by applying only a small amount of experience. A new task might include a new learning goal, for example, an agent might learn how to walk in a certain direction and then quickly learn to walk in a different direction. A new task might also involve accomplishing the same goal in a new unknown environment, for example, an agent might learn how to navigate a certain maze so that when faced with a new maze, it can determine its exit with only a few samples. To formulate the meta-RL problem, the RL task $\mathcal{T}_i$ is defined with its initial state distribution $q_i(x_1)$, a state transition distribution $q_i(s_{t+1}|s_t, a_t)$, and the loss

function $\mathcal{L}_{\mathcal{T}_i}$ corresponding to the (negative) reward function R. The whole learning task is modeled as an MDP that has a horizon of H. In MDP, the learning agent is allowed to access limited sample trajectories for few-shot learning. Thus, the learning model defined as f_θ is a policy that maps the observations or states s_t to an action distribution a_t at each timestep $t \in \{1, ..., H\}$. The loss for task $\mathcal{T}_i$ and model f_ϕ can be defined as

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = -\mathbb{E}_{s_t, a_t \sim f_\phi, q_{\mathcal{T}_i}} \Big[\sum_{t=1}^{H} R_i(s_t, a_t) \Big] \tag{6}$$

The MAML algorithm requires RL models parameterized by differentiable parameters in order to estimate the gradient updates for the meta-optimization. Therefore, the policy gradient methods are employed for that purpose.

2.2 RL Algorithms

In this section, we briefly discuss some popular RL algorithms which have been used effectively in research to solve a variety of RL problems. These algorithms include DQN, VPG, and TRPO approaches.

2.2.1 DQN Approach

The state-action value function, the Q-function $Q_\pi(s, a)$ is used to compute the value of applying an action a after observing a state s. It can be updated each time instant as

$$\begin{aligned} Q\left(s\left(t\right), a\left(t\right)\right) = \; & Q\left(s\left(t\right), a\left(t\right)\right) + \\ \alpha \; & \Big(r\left(t\right) + \gamma \; \max_a Q\left(s\left(t+1\right), a\right) - Q\left(s\left(t\right), a\left(t\right)\right) \Big), \end{aligned} \tag{7}$$

where α is the learning rate, $r(t)$ is the reward, $\gamma\, Q\left(s\left(t+1\right), a\left(t+1\right)\right)$ is the discounted state-action value at time instant $t+1$, γ is the discount factor, and ϵ is the exploration rate. For each episode, the agent randomly explores a new action with a probability of ϵ or selects a greedy action that maximizes the state-action value with probability $1-\epsilon$. In order to achieve the best policy after training for a relatively long period, the value of ϵ decays as the learning progresses. Therefore, random actions are more likely to be chosen at the beginning of training to explore the state space, whereas the model chooses the best actions after training for a while to obtain the best policy [53].

The DQN model consists of two neural networks, the estimator network which estimates the actual Q-function, and the target network, which estimates the target Q-function. Two major strategies are implemented to improve the learning rate; the fixed Q-targets and experience replay buffer. For every fixed number of steps, the Q-function estimator network is updated via the DQN. However, the experience tuples$(s\left(t+1\right), a\left(t+1\right), r\left(t+1\right), s\left(t+1\right))$ are stored in a buffer memory to be sampled and used for training the neural network. Besides utilizing past experience, the experience replay detects and breaks highly correlated samples. The deep Q-learning algorithm is summarized in **algorithm 2** [54].

Algorithm 2: Deep Q-Learning Algorithm

Initialize the replay memory to capacity N.
Initialize action-value function $\mathcal{Q}$ with random weights.
for *each episode* **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for *each time step* $t = 1, \ldots, T$ **do**
    Select a random action a_t with probability ϵ.
    otherwise, select $a_t = max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t and observe the reward r_t and image x_{t+1}.
    Set $s_{t+1} = s_t, a_t, x_{t+1}$, and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transitions $(\phi_t, a_t, r_t, \phi_{t+1})$ in the replay memory.
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from the memory.
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$
  end
end

2.2.2 VPG Approach

RL aims at obtaining the optimal behavior strategy for the agent to achieve higher rewards. Therefore, the policy gradient algorithms emerge as the recommended approach in RL since they operate directly on policy optimization. The VPG algorithm is one of the common PG approaches that has been used often in solving RL problems, It's not only suitable when implementing the MAML algorithm, but also it outperforms the Q-learning approach in many RL tasks. To formulate the VPG problem, the policy is modeled as a parameterized function $\pi_\theta(a|s)$ with respect to the parameter θ. The PG approaches aim at optimizing the parameter θ to achieve the best reward (average return). Thus, the gradient of the reward function $J(\theta)$ for the VPG algorithm is defined as

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \in \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \, log \, \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t) \right] \tag{8}$$

where τ is the trajectory of the states and actions taken by the agent during learning and A^{π_θ} is the advantage function for the current policy defined as

$$A^{\pi_\theta} = Q^\pi(s, a) - V^\pi(s) \tag{9}$$

where $Q^\pi(s, a)$ is the policy action-value function and $V^\pi(s)$ is the policy value function.

2.2.3 TRPO Approach

TRPO is a policy gradient method based on estimating the gradient of the expected return (total rewards) collected from sampled trajectories [55]. TRPO has rapidly gained popularity thanks to its stability compared to VPG and other natural policy gradient

methods [56] that include second-order optimization. As suggested by its name, the algorithm defines a trust region in which the maximum step size for policy exploration is determined to locate the optimal point in this region. Therefore, there are three main pillars of TRPO algorithms to achieve this goal; first, the conjugate gradient method which approximates the updated value without the need to compute the inverse of the fisher information matrix defined as

$$F(\theta) = E_\theta \left[\nabla_\theta \log \pi_\theta(x) \nabla_\theta \log \pi_\theta(x)^T \right] \tag{10}$$

where $\pi_\theta(x)$ is the optimized policy. The second pillar is the line search that shrinks the trust region until the update is satisfactory, and the third is the improvement check which is used to verify whether the update actually improves the policy trajectory. Consequently, TRPO outperforms the former natural gradient methods on many RL tasks. For example, the work introduced in [57] establishes a detailed comparison between TRPO, VPG, deep deterministic policy gradient (DDPG), and DQN with normalized advantage function (DQN-NAF). The authors performed comparative experiments on robotic arm control tasks such as reaching a target at a random position and picking or placing an object. The reported results elucidate the robustness and superiority of TRPO compared to the other algorithms.

2.3 Case Study: UAV Trajectory Planning

Recently, UAVs have been extensively proposed in research for several MTC applications thanks to their eminent performance guarantees. Consequently, UAVs are deployed in large IoT setups as mobile relay units between the IoT devices and the BS to minimize the age of information and maintain the data freshness within limited energy consumption. Hence, several learning schemes such as deep reinforcement learning are proposed to tackle the problem of minimizing the AoI and the energy consumption for IoT sensors [58]. In this section, we investigate two deep RL schemes for optimizing age and compare their behaviors.

The system model is defined as a set of devices randomly distributed over a 2D area, the position of each device is given by $L_d = (x_d, y_d)$. A UAV is deployed to collect update packets from these devices by flying over the service area as shown in figure 2. The main target of the UAVs is to collect information from the devices by following a strategy that reduces the total sum of the age of information (AoI). The AoI metric is used to evaluate the freshness of information, it is defined as the time elapsed since the last update packet received at a UAV was generated. Specifically, we define AoI $A_d(t+1)$ for the device d at time slot $t+1$ as

$$A_d(t+1) = \begin{cases} 1, & \text{if } w(t) = d, \\ \min\{A_{d,max}(t), A_d(t)+1\}, & \text{otherwise}, \end{cases} \tag{11}$$

where $A_{d,max}$ denotes the maximum allowed AoI. The definition guarantees that the AoI of the device d is equal to one when it generates an update packet at time slot t, otherwise, the AoI is increased by one. To formulate the MDP for the RL solution, we define the states, actions, and rewards as follows.

1. State space: The state space of the system at time slot t is expressed as $s(t) = (\boldsymbol{l}(t), \boldsymbol{A}(t))$ where:

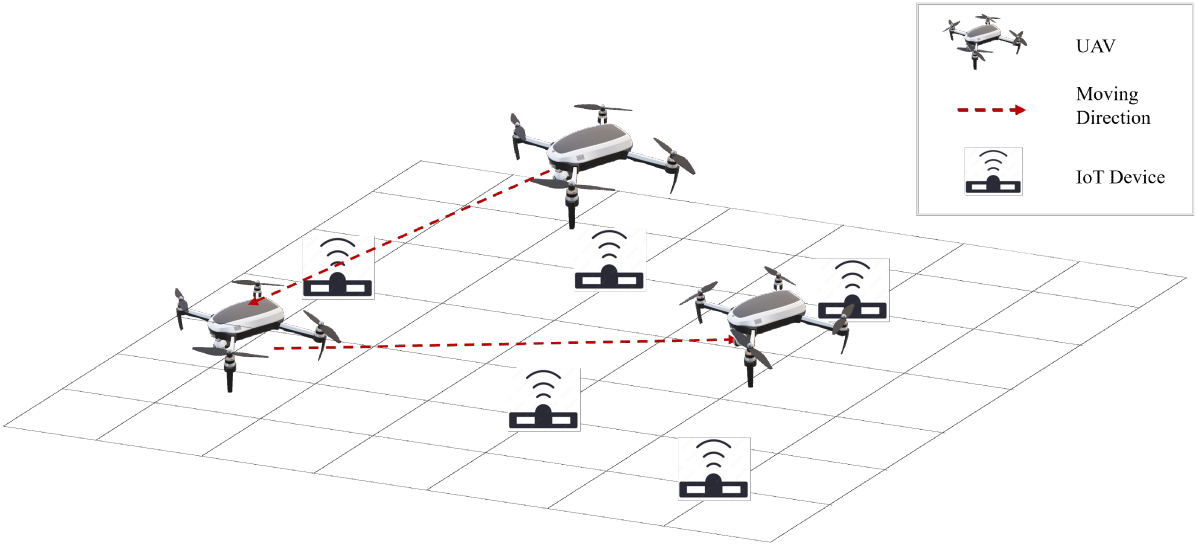


Figure 2: System Model for UAV path learning showing the trajectory of 1 UAV flying over 5 devices.

- $\boldsymbol{l}(t)$ is a location vector that comprises the position of each UAV, u, $l_u(t) \in \mathcal{P}$ at time t.
- $\boldsymbol{A}(t) = (A_1(t), A_2(t), ..., A_N(t))$ $(N < D)$ includes the AoI of all devices served by the UAVs, where $A_N(t) \in \mathcal{A} = [1, 2, ..., A_{N,max}]$.

Hence, the state space of the system is given by $\mathcal{S} = \mathcal{P}^U \times \mathcal{A}^N$, where $U = \{1, \ldots, u, \ldots\}$ is the set of UAVs.

2. Action space: The action space at time t is given by the movement of the UAVs $v_u(t)$ and the scheduling policy $w(t)$ that determines the next device to be served by the UAV. Hence, that action at time t is $a(t) = (v_u(t), w(t))$. Here, $v_u(t) \in \mathcal{V}$ has 5 directional actions (North, South, East, West, and Hovering) as shown in (12). Then, the action space is given by $\mathcal{B} = \mathcal{V}^U \times \mathcal{W}^U$.

$$l_u(t+1) = \begin{cases} l_u(t) + (0, r), & v_u(t) = \text{North}, \\ l_u(t) - (0, r), & v_u(t) = \text{South}, \\ l_u(t) + (r, 0), & v_u(t) = \text{East}, \\ l_u(t) - (r, 0), & v_u(t) = \text{West}, \\ l_u(t), & \text{Hovering}. \end{cases} \tag{12}$$

3. Reward: The reward determines the weighted sum of the age of information for all IoT devices covered by the UAVs. We define the reward r for the UAV at time instant t as

$$r(t) = -\sum_{d=1}^{D} A_d(t) \tag{13}$$

As illustrated in the system model, the UAV experiences a large dimensional state space. Thus, we adopt the deep Q-network (DQN) approach to overcome the dimensionality problem thanks to its ability to approximate the action-value function (Q-function).

We also investigate the performance of the VPG algorithm and compare its behavior with DQN, the simulation analysis is presented in the following section.

2.4 Numerical Results

2.4.1 MAML Algorithm

In this section, we evaluate the performance of the MAML algorithm for both regression and RL problems as in [37].

Regression

The MAML algorithm is tested for a simple regression problem, the learning tasks are defined as a mapping from the input to the output of a sine wave, where the amplitude and phase of the sinusoidal function vary for each task. Therefore, the distribution over tasks $p(\mathcal{T})$ is continuous. The amplitude is chosen within the range $[0.1, 5.0]$, and the phase is defined within $[0, \pi]$, both input and output have a dimensionality of 1. During training and testing, data points x are sampled uniformly from $[-5.0, 5.0]$. The loss is defined as the mean-squared error between the prediction $f(x)$ and the true function value. When training with MAML, we use one gradient update with $K = 10$ examples with a fixed step size $\alpha = 0.01$ and use Adam [59] as the meta-optimizer.

The simulation results are presented in figure 3, figure 3a shows the adaptation process of the meta-learner over gradient update steps. After one update step, the model approaches the true function and after 10 steps the model highly approximates the true function. The figure shows that the model was able to adapt quickly with few data points. Moreover, Figure 3b represents the adaptation for the neural network model with respect to the gradient steps, the plot clearly shows that the neural net poorly approximates the true function values compared to the MAML adaptation, even after 10 update steps. Figure 4 shows the mean-squared error loss of the MAML algorithm and a neural network model without meta-training. The model learned with MAML performs much better and converges faster than the neural network model, the model manages to improve with extra gradient update steps outperforming the conventional neural network setup without MAML.

Reinforcement Learning

To visualize the performance of MAML for RL problems, we train a deep neural network model that has two hidden layers of 100 neurons each, and apply the ReLU activation function [60]. The gradient updates are computed using VPG, while the TRPO is used as the meta-learner.

The tasks are defined based on a simulated planar cheetah robot that runs in a particular direction or at a particular velocity. The learning objective is to achieve a particular velocity when running in specific directions. To achieve the target velocity, the reward is selected as the negative absolute value between the agent's current velocity and the target velocity, which is chosen randomly from a uniform distribution between 0.0 and 2.0. To achieve the target direction, the reward is the magnitude of the velocity when running in the forward or the backward direction, which is selected randomly for each

task in $p(\mathcal{T})$. The task horizon or the number of episodes for each task is $H = 200$, with 20 iterations for each gradient step. The results for the half-cheetah direction task are shown in Figure 5 where the average return is plotted vs. the number of meta tasks or batches. The average return for the 1-gradient step is growing with the number of batches compared to the training with the RL algorithm only which keeps the same value for different batches. The MAML-RL approach accelerates the learning and adaptation process for the RL agents which often consume a lot of training and exploration time to start learning and achieving a higher return.

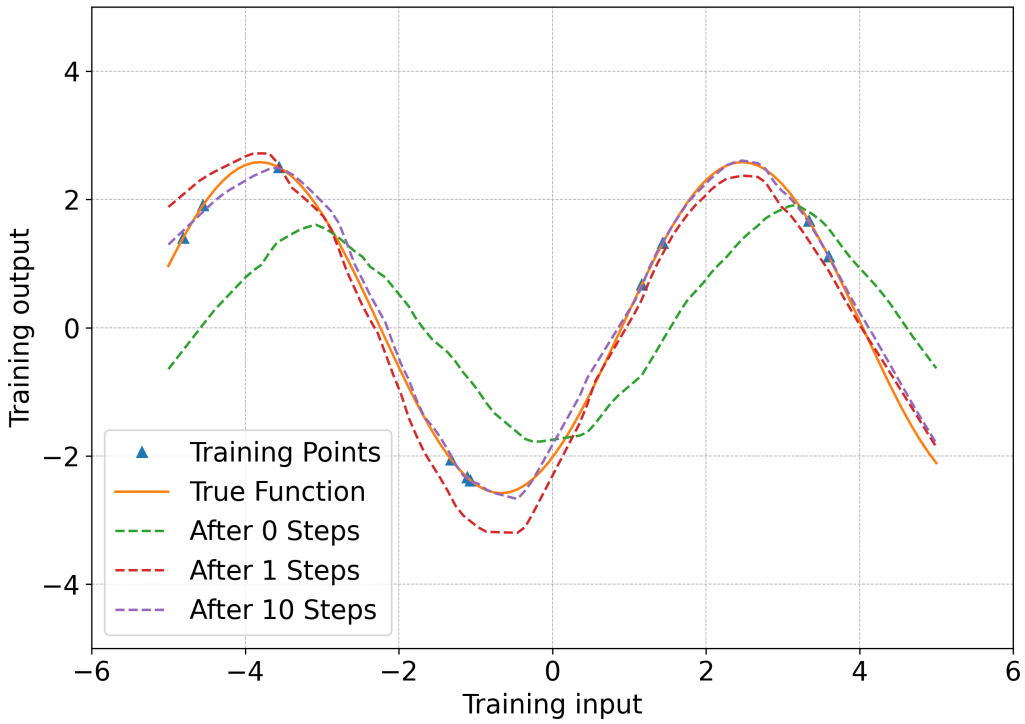
2.4.2 UAV Path Planning Using RL

In this section, we discuss the results obtained from DQN and VPG learning approaches. The achieved AoI is compared to the random walk (RW) baseline. We assume the IoT devices are configured on a space of 1100 m $\times$ 1100 m, which is divided into 11 $\times$ 11 grids. For training the DQN approach, we use a neural network with two hidden layers with 64 neurons for each layer. The learning rate is $\alpha = 10^{-4}$, we adopt the Adam optimizer, ReLU activation function, and replay buffer of size 100000. The discount factor $\gamma = 0.99$, and we train the model with 10,000 trained episodes using Pytorch framework [61].

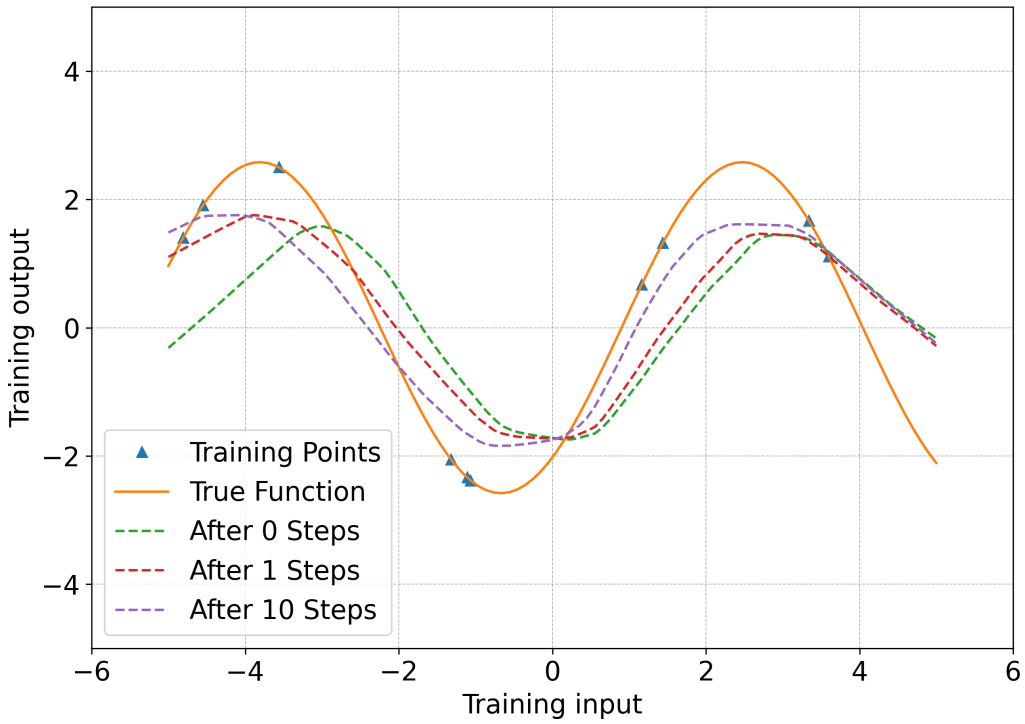
For training the VPG agent, we use two hidden layers with 20 neurons each, the corresponding learning rate $\alpha = 10^{-5}$, the model adapts Adam optimizer, Softmax activation function, $\gamma = 0.99$, and 10,000 trained episodes using Pytorch framework.

Figures 6a and 6b show the average score per 10,000 episodes for DQN and VPG, respectively. The scores represent the total reward or the average return after performing a chosen action on the environment. Thus, they reflect the learning progress and convergence to the optimal policy. The plot shows that the scores tend to improve and converge with the number of training episodes. Figure 7 compares the average age (AoI) for DQN, VPG, and RW agents, the results show that DQN and VPG algorithms improve the average age for all devices compared to the conventional RW approach. Furthermore, the VPG agent achieves lower age than DQN in some setups of IoT devices, which proves that the PG method outperforms the value-based DQN method when tackling the path learning problem.

In this chapter, we discussed the MAML algorithm and its applications in various machine-learning fields. The simulation proved that MAML approach boosts the performance of the ML tasks including the reinforcement learning problems. We also presented the UAV trajectory planning problem in massive IoT setups as a case study in mMTC applications. Moreover, we discussed different RL approaches to solve this problem by finding the optimal UAV trajectory that achieves lower AoI and maintains the information freshness between the BS and the IoT sensors. The next step is to use the MAML-RL method to enhance the performance of the UAV trajectory problem. The meta tasks can be defined based on different setups of IoT devices, hence, the UAVs can quickly adapt to new environments where the distribution of the sensors alters, and be able to determine the best path after a few update steps.



(a) Few-shot adaptation for MAML regression task.



(b) Few-shot adaptation for neural net regression task.

Figure 3: Results for few-shot adaptation using MAML Algorithm.

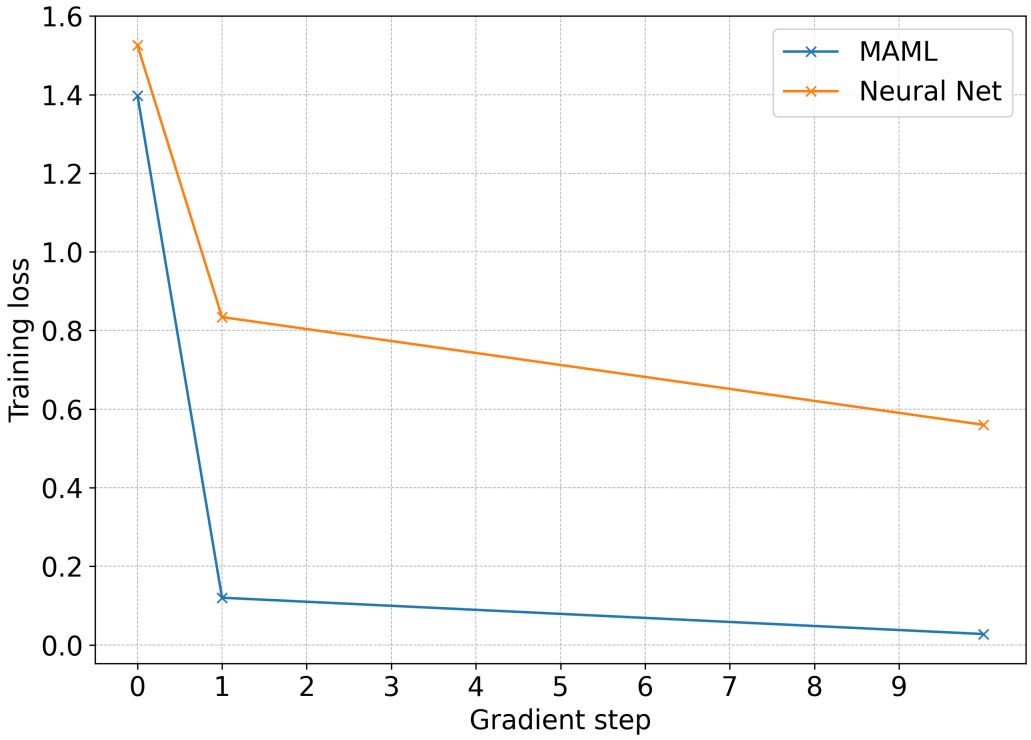


Figure 4: The loss function for MAML and a neural network for 10 gradient steps

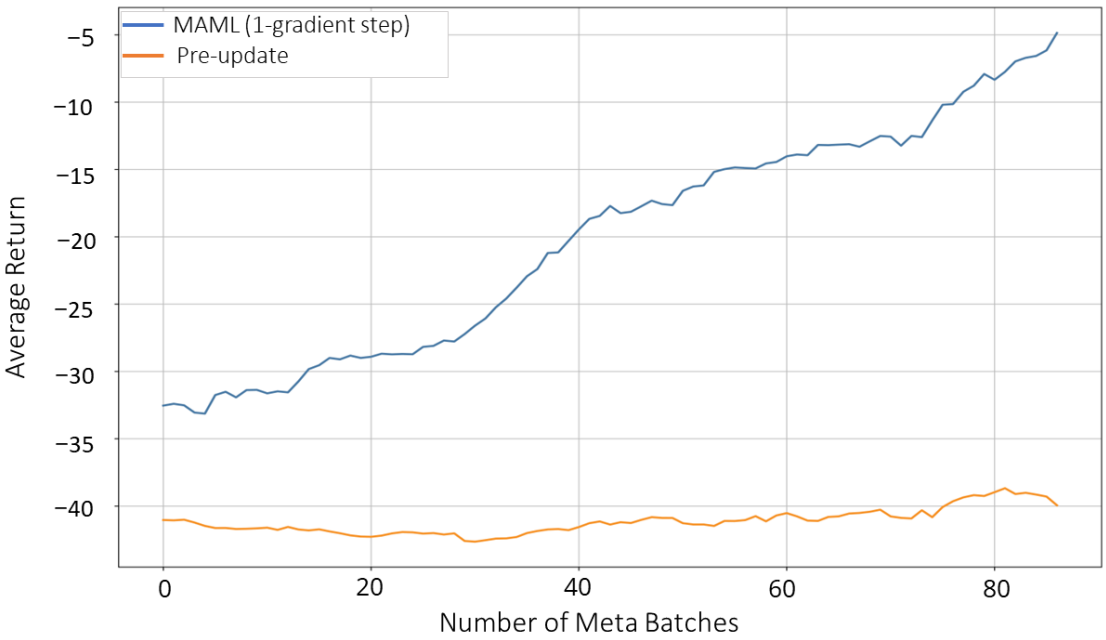
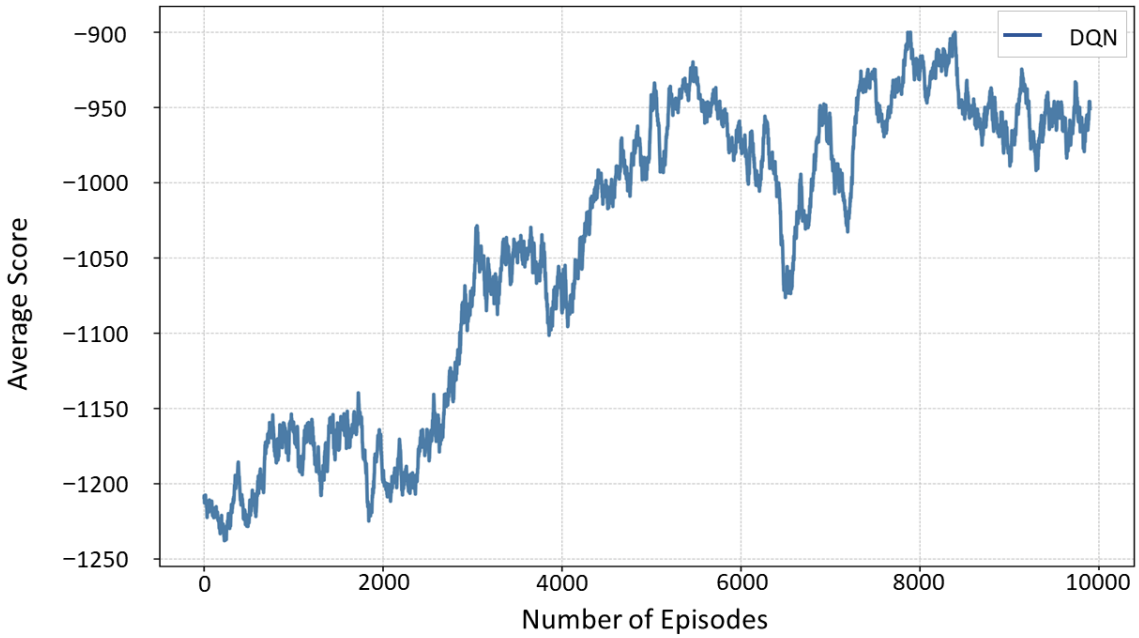
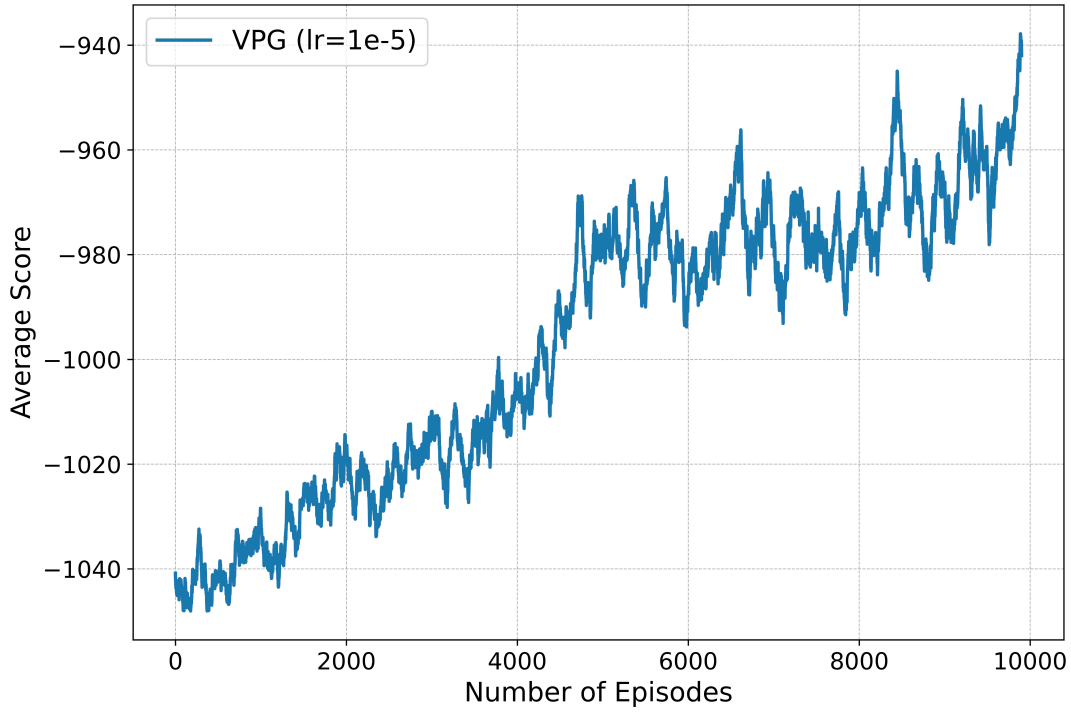


Figure 5: Reinforcement learning results for the half-cheetah direction task.



(a) Average score for DQN learning agent.



(b) Average score for VPG learning agent.

Figure 6: Average reward for DQN and VPG agents.

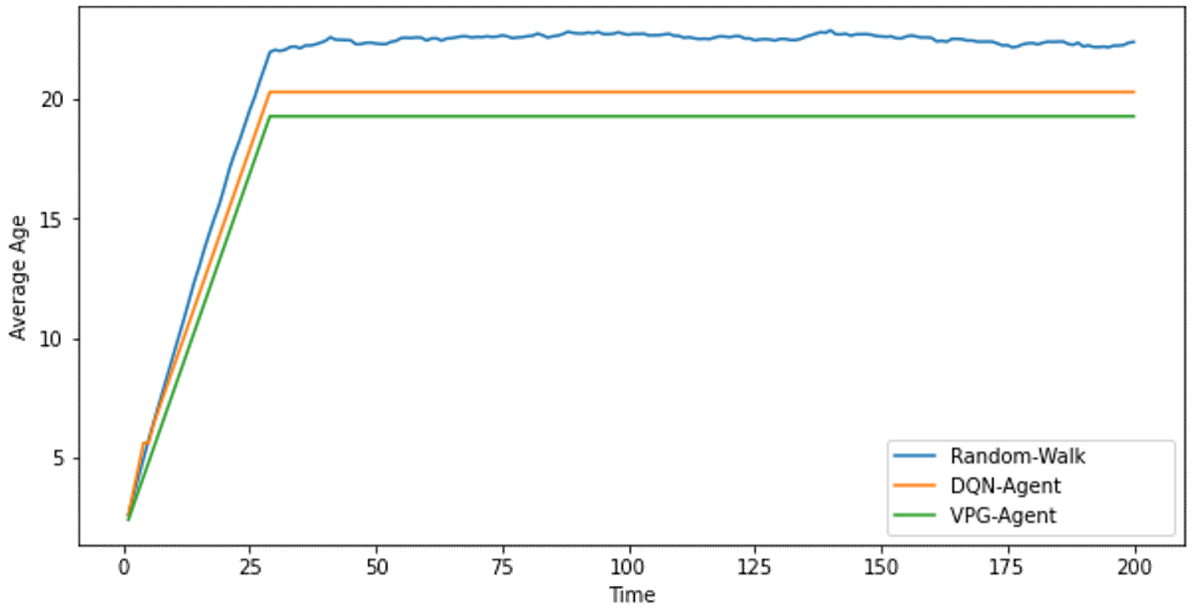


Figure 7: Average AoI for Random Walk vs. DQN-Agent and VPG agent for 1 UAV and 5 Devices.

3 META-LEARNING DEMODULATION FROM FEW PILOTS

For IoT scenarios where a large number of devices transmit a few pilots using short packets over a fading channel, the accurate estimation of the end-to-end channel becomes unfeasible. Therefore, the demodulation problem can be solved using machine learning approaches. Demodulation and decoding can be managed as a classification problem where the input is the received baseband signals and the output is the actually transmitted symbols [41, 62–68]. For this matter, the pilots are used as the training data for the neural network model while the performance is measured in the test time with different channel conditions. However, the training on neural networks requires a large amount of training data, i.e. transmission of a long pilot sequence. To handle the problem of short packet transmission with few pilot symbols, a meta-learning approach for few-shot learning is proposed in [41]. The learning model is based on the MAML algorithm proposed in [37]. The pilots from the previous transmission of other IoT devices are used as meta-training data for the demodulator to be able to adapt quickly to the change in the channel conditions between the test devices and the base station. In [63], the authors extend their meta-learning solution by adopting other offline versions of the MAML approach which include first-order MAML (FOMAML) [37], REPTILE [69], and CAVIA [70]. The simulation results presented in the paper show that the meta-learning-based approaches outperform the conventional learning-based techniques, moreover, the CAVIA algorithm seems to achieve higher performance with fewer pilots than both MAML and REPTILE. The paper also proposed an online meta-learning scheme where the meta-learner is concurrently updated while the pilots are received, their online adaptation model is based on the work presented in [71, 72]. The results also show good performance in online settings compared to conventional learning, however, the online model is slower and achieves a relatively higher symbol error rate than the offline model. In order to optimize the online meta-learning for few-pilot demodulation, the authors in [64] introduce a Bayesian meta-learning approach to obtain a calibrated few-pilot demodulators by optimizing the prior data distribution used in Bayesian inference. The proposed Bayesian model is motivated by the VAMPIRE scheme in [73] which relies on Gaussian variational posteriors [74]. The numerical results discussed in the paper show the model's ability to achieve lower symbol error rates and better calibration performance in terms of expected calibration error (ECE) [75]. Additionally, the Bayesian meta-learning is employed in [65] for demodulation and channel equalization problems. The authors combined Bayesian learning with active meta-learning [76, 77] to effectively reduce the number of tasks required by the meta-learner to collect the training data and adapt to new tasks.

In this Chapter, we reconstruct the system model and the simulation results presented in [41]. We focus on the few-shot MAML approach for few-pilot demodulation and online adaptation to the variations in CSI. The simulations include two channel condition scenarios; a symmetric channel (± 1), and a Rayleigh fading channel. We also adopt two modulation schemes pulse amplitude modulation with modulation order 4 (4-PAM) and quadrature amplitude modulation with modulation order 16 (16-QAM). The analysis proves the superiority of MAML to solve the short packet transmission problem encountered in massive MTC applications and massive IoT scenarios.

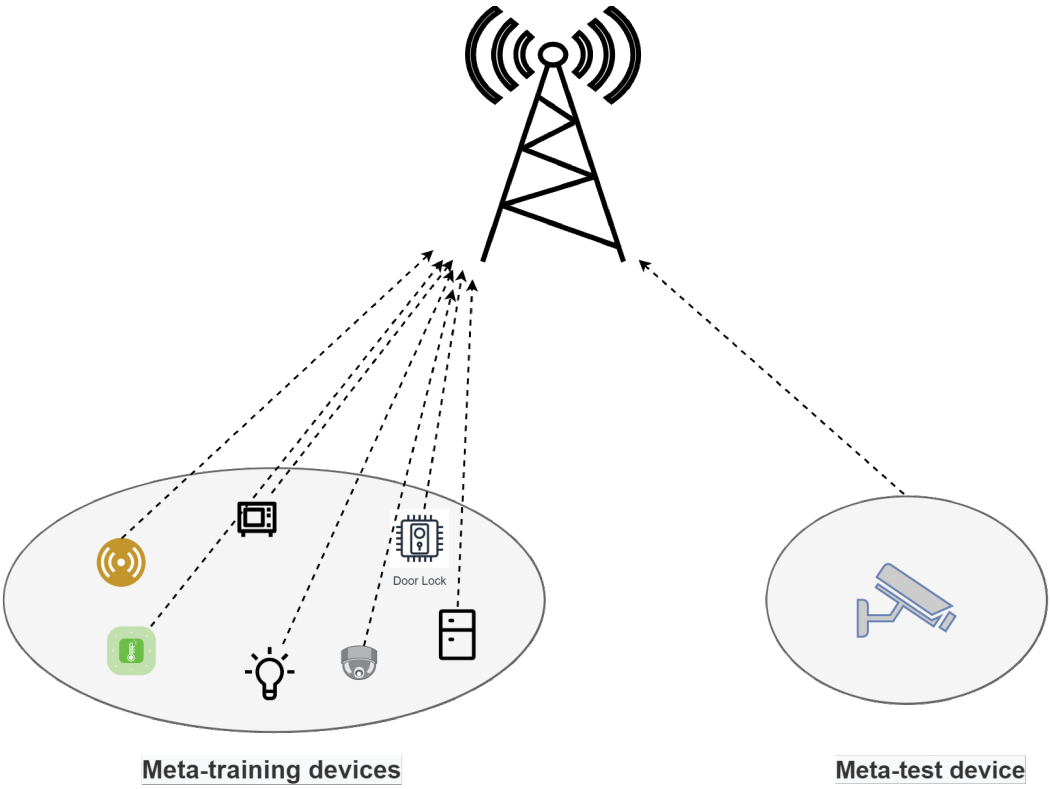


Figure 8: System model for few-pilot demodulation using meta-learning.

3.1 System Model

We consider the IoT system and the meta-learning solution proposed in [41]. The system model illustrated in figure 8 consists of a base station (BS), K meta-training devices, and one meta-test device. For each device, k, s_k denotes the complex symbol transmitted by the device. y_k is the corresponding received signal at the BS such that

$$y_k = h_k s_k + n_k \tag{14}$$

where h_k is the complex channel gain from the device k to the BS, and $n_k \sim \mathcal{CN}(0, \zeta)$ is the complex additive white Gaussian noise (AWGN) and ζ is the noise variance. Based on the transmission of few-pilot symbols, a probabilistic demodulator $p_\theta(s|y)$, parameterized by the learning parameter θ, learns to demodulate the received signal y to recover the transmitted symbols s with a low probability of symbol error. To inspire the model to learn from few-pilot data, the signals received at the BS from previous transmissions by the meta-training devices are gathered and treated as meta-training data. Particularly, the BS contains N pairs of symbols s_k and received signal y_k for each meta-training device $k = 1, \ldots, K$. The meta-training dataset is expressed as $\mathcal{D} = \{D\}_{k=1,\ldots,K}$, where $D_k = \{(s_k^{(n)}, y_k^{(n)}) : n = 1, \ldots, N\}$, and $(s_k^{(n)}, y_k^{(n)})$ are the n^th pilot-received signal pairs for the k^{th} meta-training device.

For the meta-test device, the BS receives N_{te} pilot symbols. The N_{te} pilots received from the test device are gathered in the set $\mathcal{D}_T = \{(s^{(n)}, y^{(n)}) : n = 1, \ldots, N_{te}\}$. Therefore, the demodulator model is trained using the meta-training data $\mathcal{D}$ and then adapted to the pilot data $\mathcal{D}_T$ received from the meta-test device.

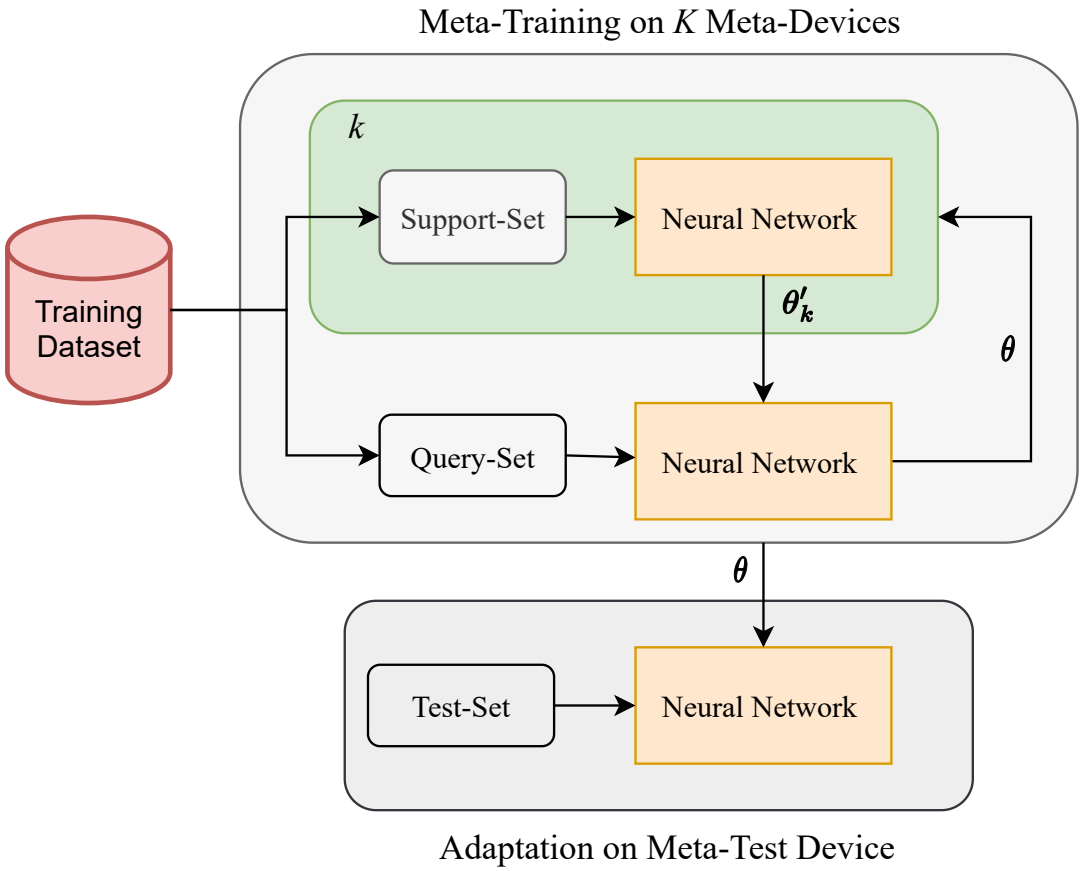


Figure 9: Illustration of the Meta-Learning approach.

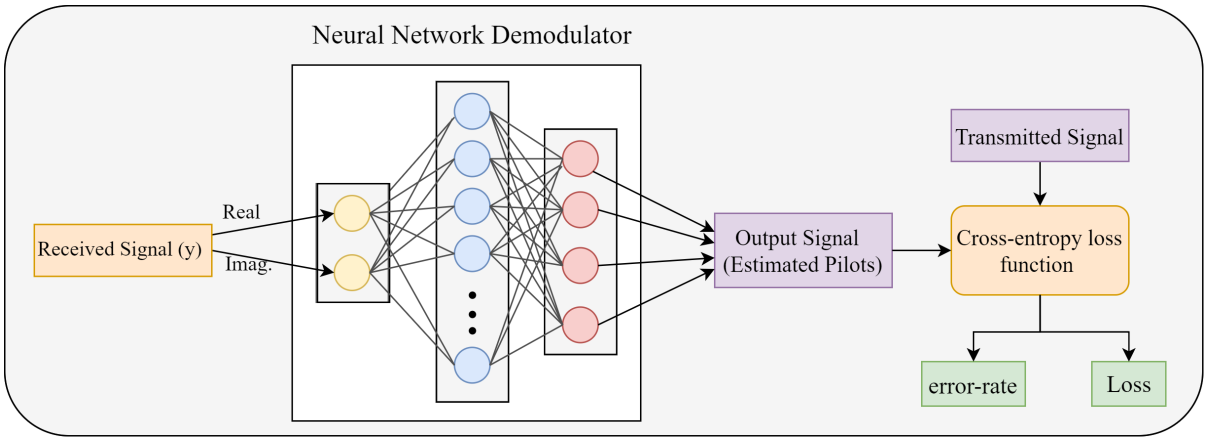


Figure 10: Neural Network Demodulator Diagram

3.2 Meta-Learning Algorithm

In this section, we describe the meta-learning algorithm for the demodulator [41]. The learning is performed in two phases; the meta-learning phase, and the test-adaptation phase. In the meta-learning phase, we use the meta-training data $\mathcal{D}$ and iterate over the number of K meta-tasks (meta-devices) to learn a general parameter vector θ using the inner task parameter θ'_k. This can be done offline at the BS while collecting data from

training devices. For the meta-test adaptation phase, the learned parameter θ is used to enable fast adaptation based on the few pilots N_{te} transmitted by the target (meta-test) devices. Consequently, the test dataset $\mathcal{D}_T$ is used to train a demodulator $p_\theta(s|y)$ to minimize the cross-entropy loss

$$L_{\mathcal{D}_T}(\theta) = -\sum_{(s^{(n)}|y^{(n)})\in\mathcal{D}_T} \log p_\theta(s^{(n)}|y^{(n)}) \tag{15}$$

Then, the stochastic gradient descent algorithm is used to update the parameter θ iteratively as

$$\theta \leftarrow \theta - \eta\nabla_\theta \log(s^{(n)}|y^{(n)}), \tag{16}$$

where the pair $(s^{(n)}|y^{(n)}) \in \mathcal{D}_T$, and η is the step size. As discussed in [37], the purpose of the MAML algorithm is to find the initial parameter θ such that, for any device, the loss after one iteration of 16 applied to the received pilots is minimized. The algorithm is illustrated in figure 9 where the training dataset is split into support and query sets, the support set is used for the meta-training devices to train the neural network parameters θ'_k. The query set is fed into the neural network (NN) to update the meta parameter θ. After the meta-training is performed, the general parameter θ is used to initialize the neural network that adapts to the test dataset. The algorithm is detailed in **algorithm 3** [41].

3.3 Simulation Results

In this Section, we illustrate the simulation results for both symmetric and Rayleigh fading channels using 4-PAM and 16-QAM modulation schemes.

3.3.1 Symmetric Channel

In this experiment, we adopt a simple scenario where we assume the fading is binary, i.e., the channel h_k is symmetric and with values ± 1. Here we assume a pulse-amplitude modulation with four amplitude levels (4-PAM) where the pilot symbols are $\mathcal{S} = \{-3, -1, 1, 3\}$. Pilot symbols in the meta-training dataset $\mathcal{D}$ and meta-test dataset$\mathcal{D}_T$ follow a fixed periodic sequence $\{-3, -1, 1, 3, -3, -1, \ldots, \}$ while transmitted symbols in the test set for the meta-test device are randomly selected from the set $\mathcal{S}$. The number of meta-training devices is $K = 20$, and the number of pilot symbols per device is $N = 8$ for 4-PAM and $N = 32$ for 16-QAM. The devices are divided equally into two sets, support set with channel value $+1$ and the number of pilots $N_{tr}^Q = 4, 16$ and query set with channel value -1 and $N_{tr}^S = 4, 16$ for 4-PAM and 16-QAM, respectively. For the adaptation on the meta-test device, the channel value is chosen randomly between $+1$ and -1.

The demodulator, shown in figure 10, is a neural network with an input layer of 2 neurons, one hidden layer with 30 neurons, and a softmax output layer with 4 neurons in case of 4-PAM and 16 neurons in case of 16-QAM, the activation function $\sigma(.) = \tanh(.)$ for 4-PAM, and $\sigma(.) = \text{ReLU}(.)$ for 16-QAM. For meta-learning with MAML, we use a mini-batch of size 4 with fixed learning rates $\alpha = 0.1$ and $\beta = 0.001$. The weights and biases are all initialized randomly. For the adaptation in the meta-test device, we adopt

Algorithm 3: Meta-Learning Demodulation

Input: Meta-training data $\mathcal{D} = \{\mathcal{D}_{k=1,\ldots,K}\}$, and meta-testing pilot data $\mathcal{D}_T$; N_{tr}^S and N_{tr}^Q; step size α and β.

Output: Learning parameter vector θ; SER.

Randomly initialize the parameter vector θ.
START META-TRAINING:
while *not done* **do**
  for *each meta-training device* k **do**
    Randomly split $\mathcal{D}_k$ into two sets; support-set $\mathcal{D}_k^S$ of size N_{tr}^S, and query-set $\mathcal{D}_k^Q$ of size N_{tr}^Q.
    Calculate the gradient $\nabla_\theta L_{\mathcal{D}_T}(\theta)$ from (20) with $\mathcal{D}_T = \mathcal{D}_k^Q$, and $\nabla_\theta^2 L_{\mathcal{D}_T}(\theta)$.
    Compute adapted parameters θ' using

$$\theta_k' = \theta - \beta \nabla_\theta L_{\mathcal{D}_k^Q}(\theta)$$

  end
  update the meta-parameter θ:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{k=1}^{K} L_{\mathcal{D}_k^S}(\theta_k')$$

end
START META-TESTING ADAPTATION:
for *testing epochs* **do**
  Load the learned parameter vector θ.
  Sample data from $\mathcal{D}_T$.
  Update θ in the direction of the gradient with step size η by

$$\theta \leftarrow \theta - \eta \nabla_\theta L_{\mathcal{D}_T}(\theta)$$

end

a mini-batch of size 1 and learning rate $\alpha = 0.01$. The signal-to-noise ratio (SNR) is 15dB. The simulation parameters are summarized in table 2.

We compare the performance of the meta-learning algorithm with a simple neural network demodulator in which we only train the devices without the experience of meta-learning. The comparison is illustrated in figure 11 in which we plot the symbol error rate with respect to the number of pilots N_{te} for the meta-test device. The figure shows that the MAML approach outperforms the neural network model and quickly adapt to the few pilot transmission of the test device. The MAML algorithm's training convergence is analyzed for error rate and training loss given in 15. The results are illustrated in figure 12a and 12b which show that the MAML update step greatly accelerates the convergence achieving much lower loss and error rate for the same number of the training epochs.

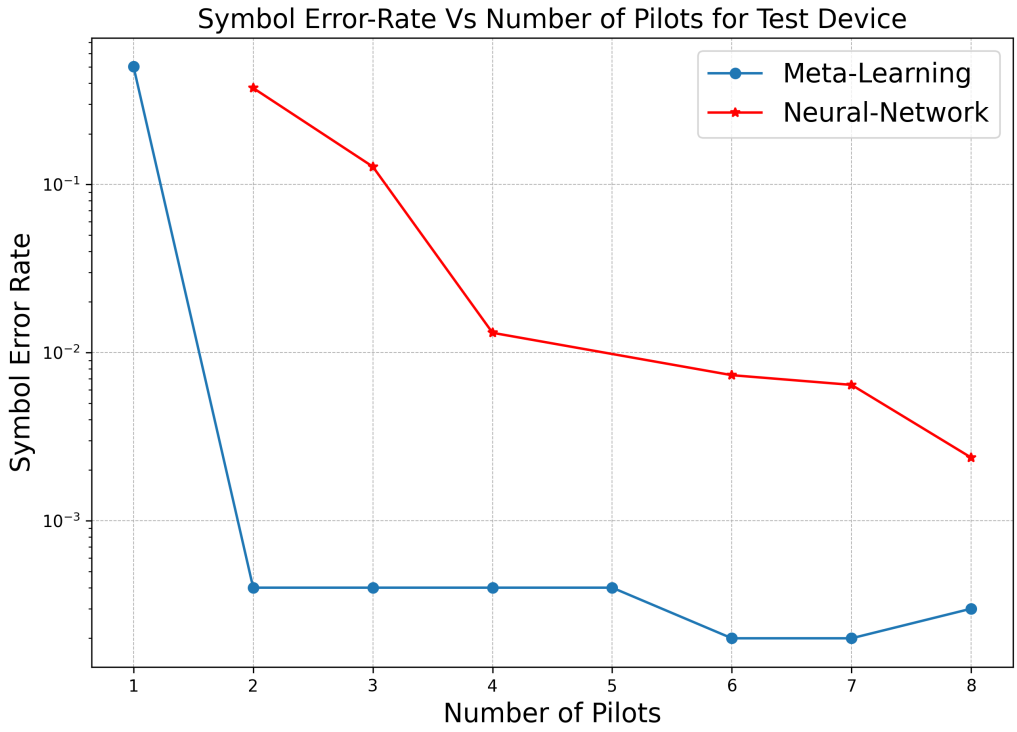
Table 2: Meta-demodulator Model Parameters

Simulation Parameters	**4-PAM**	**16-QAM**
Activation function for hidden layers	tanh	ReLU
Modulation Order M	4	16
Number of pilots symbols per device	4	16
Common Parameters:		
Number of devices K	20	
Optimizer	Adam	
Outer learning rate α	0.1	
Inner learning rate β	0.001	
Learning rate for adaptation η	0.01	
Number of training epochs	10,000	
Size of testing data for target devices	10^6	
Training SNR	15 dB	
Number of adaptation epochs	1000	
Number of DNN layers	3	
Neurons for hidden layer	30	
Activation function (output layers)	Softmax	

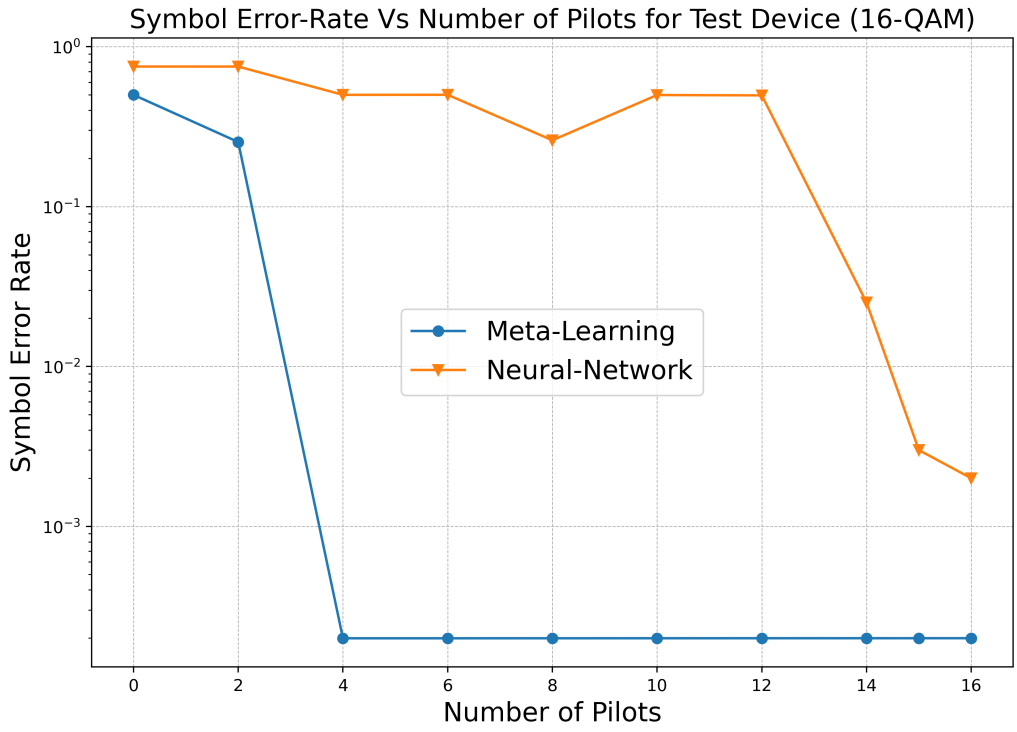
3.3.2 Rayleigh-Fading Channel

We consider a more complicated scenario where we adopt a Rayleigh fading channel where $h_k \sim \mathcal{CN}(0, 1)$. In figure 13, we plot the average symbol error rate with respect to the number of pilots N_{te} transmitted by the meta-test device. The performance is compared to a basic neural network-based model. The simulation results show that the MAML approach adapts more quickly than the NN baseline scheme when encountered with the channel of the target device. However, in the Rayleigh fading scenario, more pilots are needed to achieve the same SER compared to the symmetric channel scenario presented in Figure 11.

The training convergence is also investigated for the Rayleigh fading channel configuration. Figure 14a the error rate convergence vs. training epochs for MAML before and after updating the meta-parameter. The error quickly converges for the meta-training approach. The loss value is also plotted against the training epochs in figure 14b, The MAML loss quickly converges and achieves the minimum value of -1.



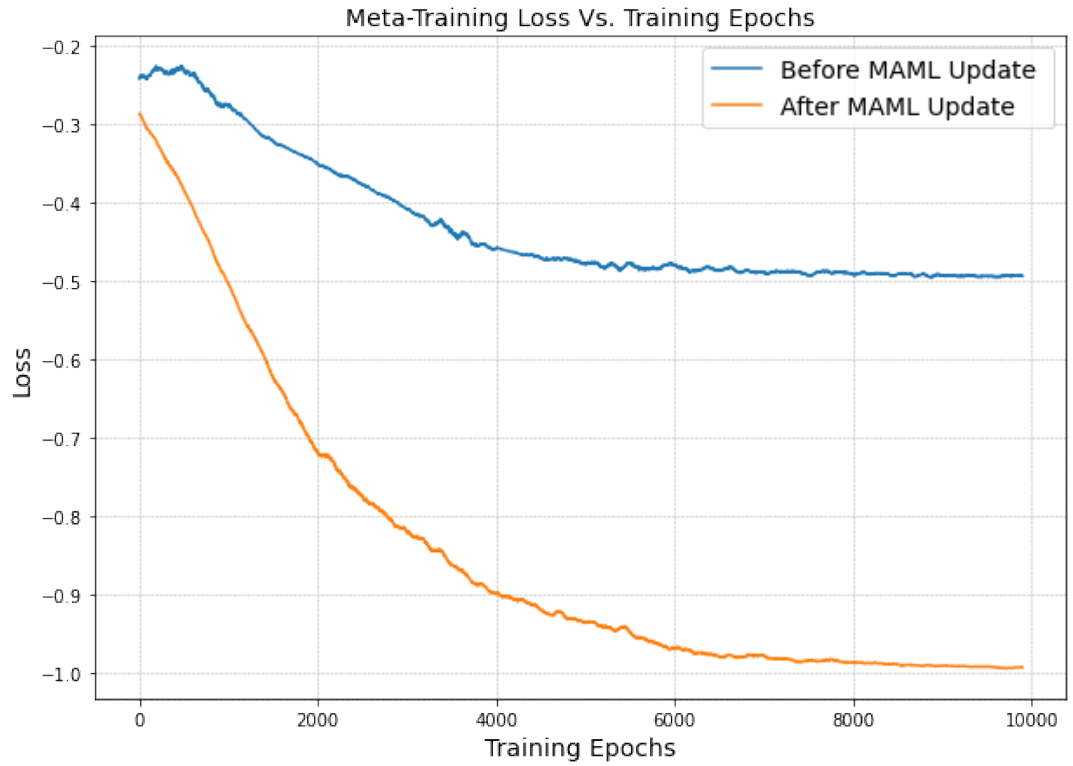
(a) 4-PAM modulation.



(b) 16-QAM modulation.

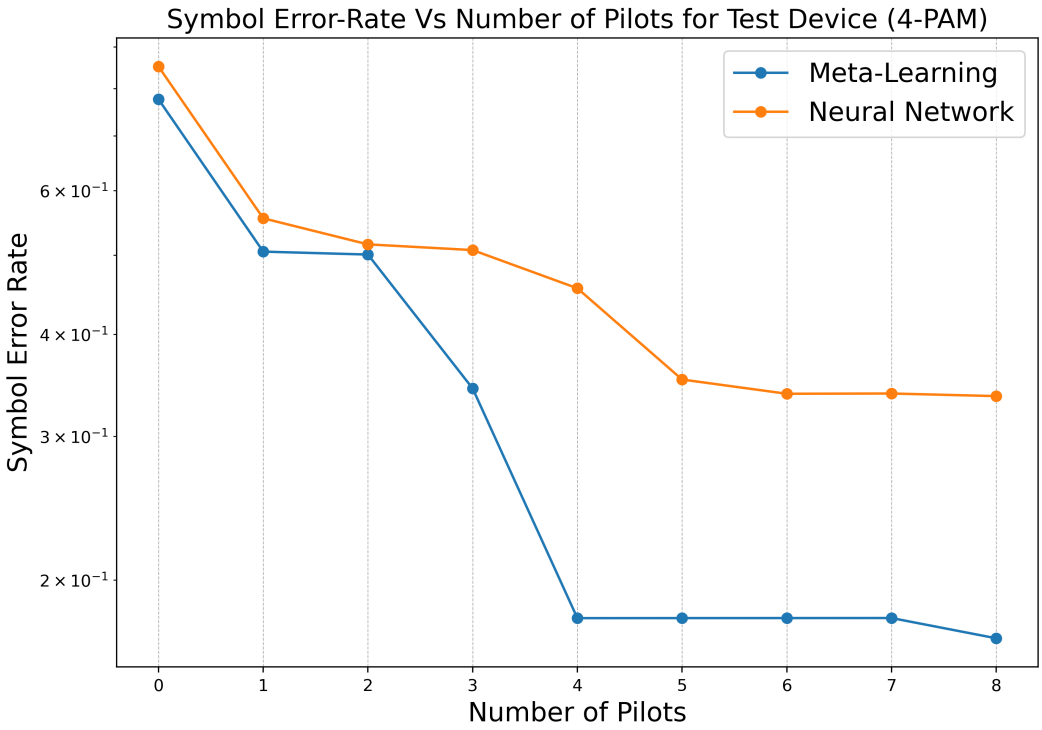
Figure 11: Symbol error rate for a different number of transmitted pilots. The figure compares the error rate for a simple neural network model and the meta-learning approach for the binary-fading (symmetric) channel.

(a) Average error rate for meta-training before and after MAML update step vs. training epochs.

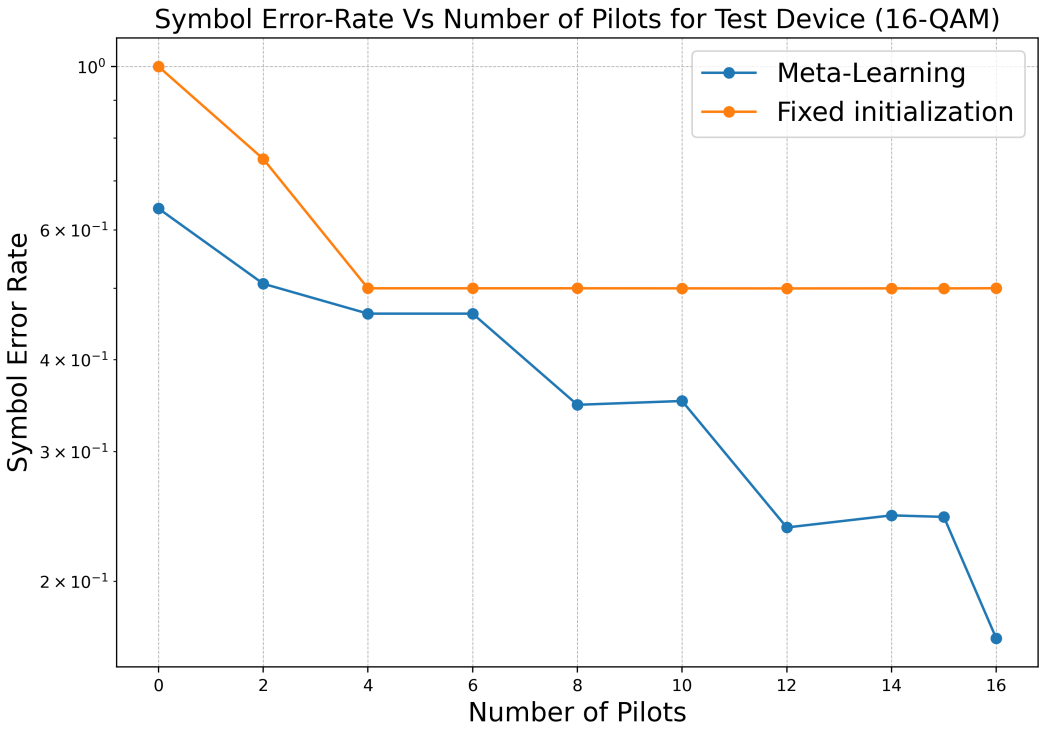


(b) Average training loss for meta-training before and after MAML update step vs. training epochs.

Figure 12: Convergence analysis for MAML-algorithm training phase before and after MAML update step. The simulation is done for 4-PAM modulation symmetric channel.

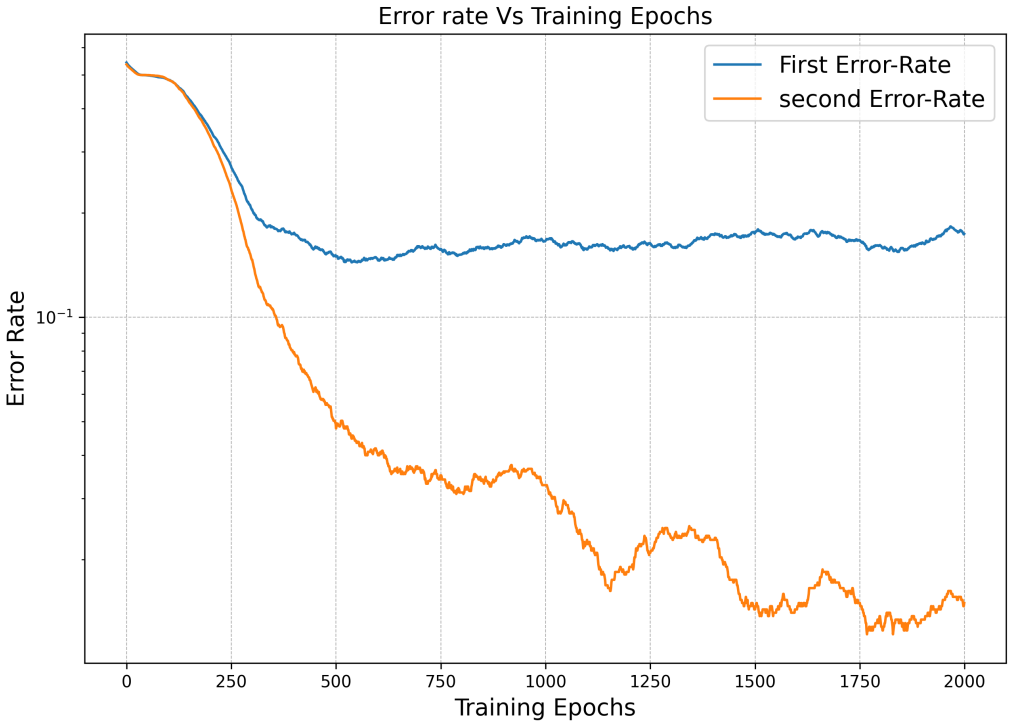


(a) 4-PAM modulation.

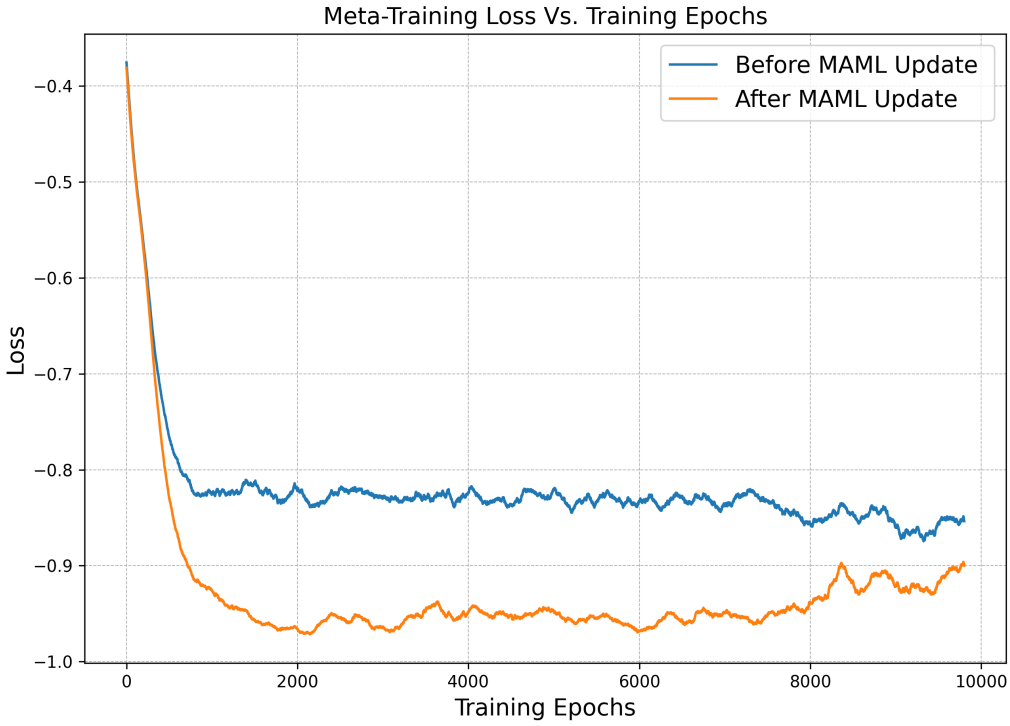


(b) 16-QAM modulation.

Figure 13: Symbol error rate for a different number of transmitted pilots. The figure compares the error rate for a simple neural network model and the meta-learning approach for the Rayleigh-fading channel.



(a) Average error rate for meta-training before and after MAML update step vs. training epochs.



(b) Average training loss for meta-training before and after MAML update step vs. training epochs.

Figure 14: Convergence analysis for MAML-algorithm training phase before and after MAML update step. The simulation is done for 4-PAM modulation Rayleigh fading channel.

4 META-LEARNING BASED FEW PILOTS DEMODULATION AND INTERFERENCE CANCELLATION FOR NOMA UPLINK

The path to future wireless networks is encompassed by a massive deployment of devices, enabling smart cities, autonomous vehicles, and many unforeseen scenarios. Besides a large number of devices, these mMTC devices are often battery constraint, with limited computational capability, and have heterogeneous and sporadic traffic patterns. Combined, these characteristics impose many challenges for the design of efficient random-access procedures and radio resource management. Therefore, researchers are increasingly studying multiple access techniques that are able to scale well to this massiveness while coping with the existence of interference and scarce spectrum. In this context, NOMA with SIC has shown potential and has been the focus of academy and industry in recent years [29,33,34,78–81]. This is because NOMA schemes allow for multiple-user transmission with superior performance compared to conventional orthogonal schemes. These recent surveys evince the popularity and potential of NOMA and overview key characteristics, techniques, and applications [33,80,81].

In this Chapter, we extend the meta-learning for the few-pilot demodulation problem to include interference from other IoT devices. Furthermore, we utilize the non-orthogonal multiple access technique and propose a meta-learning based successive interference cancellation approach to cancel the interference from other IoT devices. The numerical results demonstrate the robust performance of the proposed meta-SICNet model on the demodulation and interference cancellation task using only few pilots and a few meta tasks. **The work in this Chapter is published in [82]**.

The rest of the chapter is organized as follows: Section 4.1 depicts the system model and the problem formulation. Conventional SICNet and the proposed meta-learning-based SICNET solutions are presented in Section 4.2, and Section 4.3 elucidates the results.

4.1 System Layout and Problem Formulation

4.1.1 System Layout

We consider a non-orthogonal uplink channel where we have K meta-training device groups and one meta-testing group. For each group k, L IoT devices transmit their data to the BS within the same time and frequency resources as illustrated in Figure 15. The devices transmit $\{x_{l=1,\dots,L}^{(k)=1,\dots,K}\}$ symbols to the BS using superposition coding, where the symbol $x_l^{(k)}$ is the l^{th} symbol transmitted from the k^{th} device. Specifically, the symbol $x_l^{(k)}$ is amplified with the transmitted power P_l for $l = 1, \dots, L$. The channel input is the superimposed signal $\mathbf{x_k}$ given by

$$\mathbf{x_k} = \sum_{l=1}^{L} \sqrt{P_l} x_l^{(k)} \tag{17}$$

The symbols are sampled from an M-point constellation $\mathcal{S}$, and assumed to be mutually independent with unit mean power, i.e., $\mathbb{E}\left[\left|x_l^{(k)}\right|\right] = 1$. For simplicity, we assume all

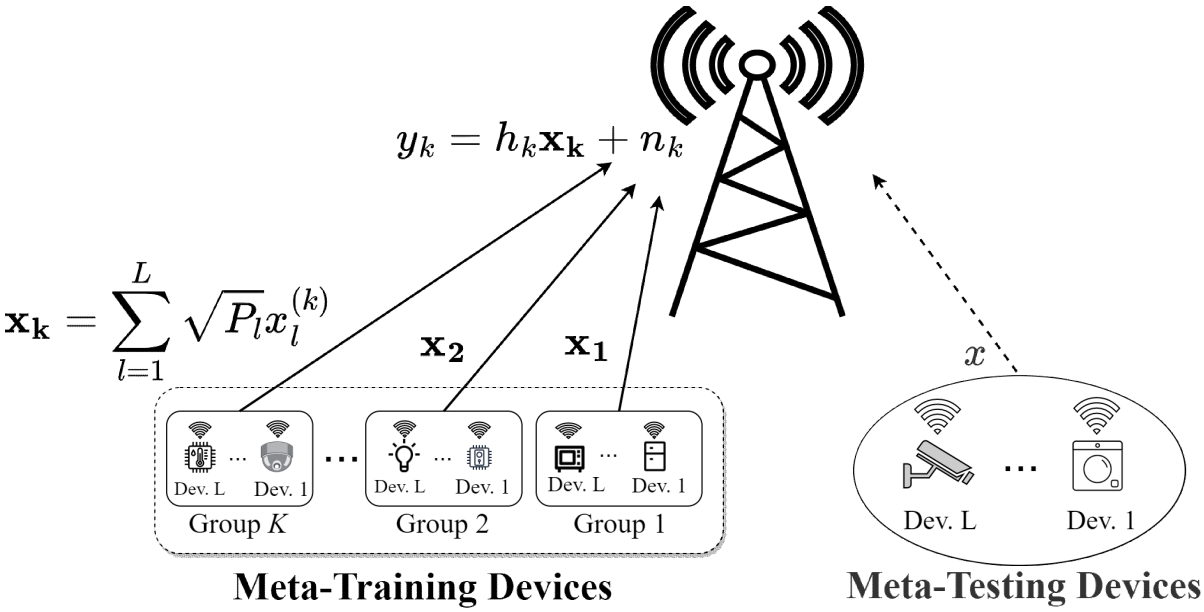


Figure 15: The system model comprises k sets of meta-training devices and one Meta-testing device. Each meta-training set, k, transmits a superimposed signal, $\mathbf{x_k}$ of L-devices to the BS.

devices have the same modulation order M. The channel output at the BS for each k group denoted as y_k for $k = 1, \ldots, K$ is given by

$$y_k = h_k \mathbf{x_k} + n_k, \tag{18}$$

where $h_k \in \mathbb{C}$ is the channel coefficient between the BS and the device-group k, and $n_k \sim \mathcal{CN}(0, \zeta)$ is the complex AWGN.

4.1.2 Problem Formulation

We aim to construct a symbol detection demodulator based on a short packet transmission of a few pilot symbols. For each device l in the device group k, the detected symbol, i.e., $\hat{x}_l^{(k)}$, is estimated from the channel output y_k. To enable symbol recovery using few-pilot learning, we use a data-driven approach based on meta-learning and DNNs. Assuming no prior information about the channel model at the receiver, the BS can use the signals received from the previous pilot transmissions of K other IoT device groups, which are referred to as meta-training devices and their data as meta-training data. In particular, the BS has available N pairs of pilots $\mathbf{x_k}$ and received signal y_k for each meta-training group $k = 1, \ldots, K$. The meta-training dataset is denoted as $\mathcal{D} = \{D\}_{k=1,\ldots,K}$, where $D_k = \{(\mathbf{x_k}^{(n)}, y_k^{(n)}) : n = 1, \ldots, N\}$, and $(\mathbf{x_k}^{(n)}, y_k^{(n)})$ are the pilot-received signal pairs for the kth meta-training group of devices.

For the meta-test devices, the BS receives N_{te} pilot symbols. It collects the N_{te} pilots received from the target device in set $\mathcal{D}_T = \{(x^{(n)}, y^{(n)}) : n = 1, \ldots, N_{te}\}$. The demodulator can be trained using meta-training data $\mathcal{D}$ and the pilot symbols $\mathcal{D}_T$ from the meta-test devices. To recover the symbols, the successive interference cancellation algorithm is utilized, such that the power allocations for the superimposed symbols satisfy $P_1 >$

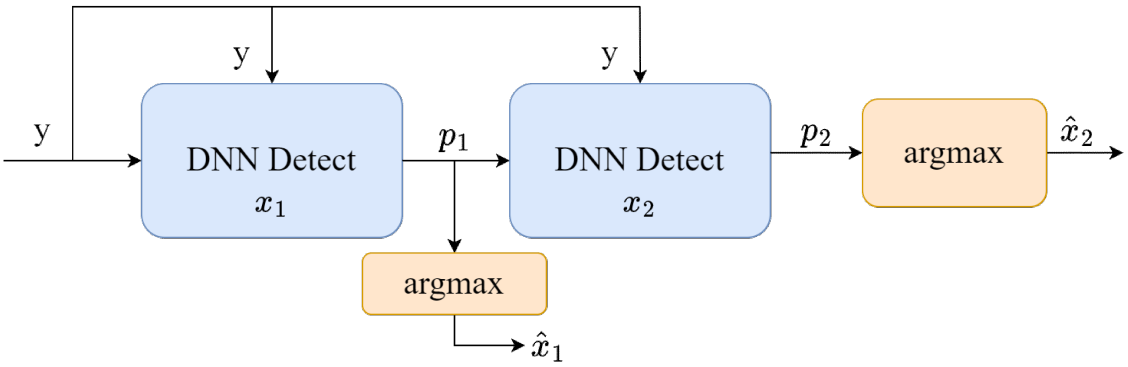


Figure 16: Architecture of SICNet for DNN-based SIC for $L = 2$ devices.

$P_2 \ldots > P_L$. Hence, the meta-learning-based DNN successively detects each symbol from the channel output signal y_k.

4.2 Deep Learning-Based SIC

In this section, we describe the data-driven solutions based on DNNs to perform SIC and symbol detection. First, we introduce the conventional DNN-based SICNet; then we illustrate the proposed meta-SICNet approach.

4.2.1 SICNET

The DNN-based SIC, called SICNet, was introduced in [29] to estimate the transmitted symbols in the downlink NOMA users scheme. For the sake of illustration, we consider an uplink scheme with $L = 2$ devices in each device group. As depicted in Figure 16, the architecture of SICNet is implemented using sequential DNN blocks. Each block performs symbol recovery, which is considered a classification problem. Therefore, the structure of SICNet consists of L DNN stages, where each stage estimates the transmitted symbol of each device. Each stage uses the received signal y_k and the output vector $\mathbf{p}_l$ for $l = 1, \ldots, L$ that represents the conditional distribution of the corresponding symbol, expressed as

$$\mathbf{p}_l = \begin{bmatrix} \hat{p}(x_l = \psi_1 | y_k, \mathbf{p}_1, \ldots, \mathbf{p}_{l-1}) \\ \vdots \\ \hat{p}(x_l = \psi_M | y_k, \mathbf{p}_1, \ldots, \mathbf{p}_{l-1}) \end{bmatrix}, \tag{19}$$

where ψ_i is the i^{th} constellation symbol in the constellation space $\mathcal{S}$ for $i = 1, \ldots, M$, and $\hat{p}(x_l = \psi_i | y_k, \mathbf{p}_1, \ldots, \mathbf{p}_{l-1})$ is a parametric estimation of the probability of x_l given y_k and the previous estimates $\mathbf{p}_1, \ldots, \mathbf{p}_{l-1}$.

The data-driven feature of the SICNet increases its capability to detect symbols reliably without requiring full knowledge of the channel model. Concretely, it works in a model-agnostic manner contrary to the conventional SIC algorithm [29], which requires

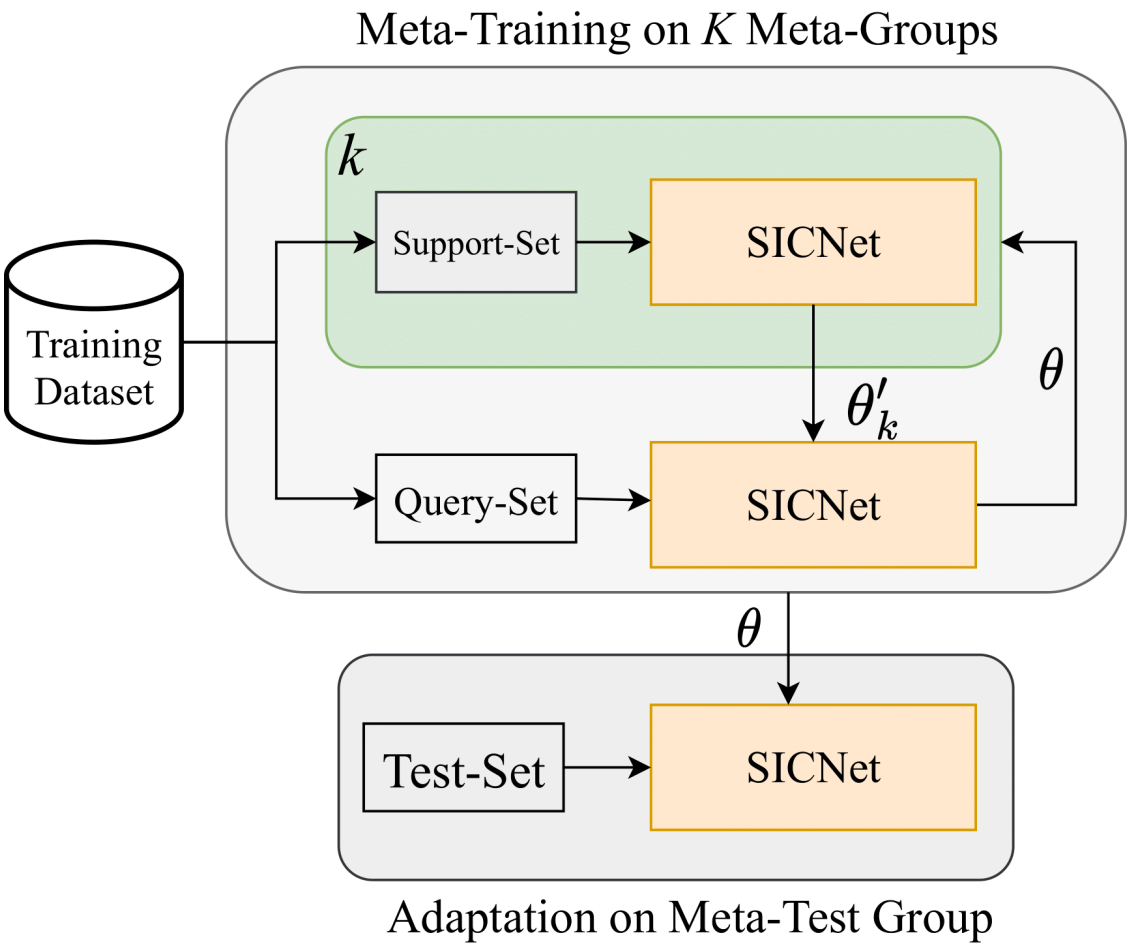


Figure 17: Architecture of meta-SICNet

a restricted channel model [83]. Thus, SICNet operates with arbitrary channel models depending on the classification process of the data-driven DNN. Furthermore, the SICNet architecture only requires knowledge of the modulation order M and the power coefficients of the transmitted signals.

4.2.2 Proposed Meta-Learning Approach

In this section, we describe the meta-learning approach for demodulation based on [41]. In our proposed method, the meta-learning-based SICNet allows learning the interference cancellation and signals demodulation using a few-pilot transmission. Moreover, the model can quickly adapt to the change in the channel conditions, in contrast to the classical SICNet.

In our model, the learning is performed in two phases; the meta-learning phase and the test-adaptation phase, as shown in Figure 17. In the meta-learning phase, we use the meta-training data $\mathcal{D}$ and iterate over the number of K meta-tasks (meta-devices) to learn a general parameter vector θ using the inner task parameter θ'_k. This can be done offline at the BS while collecting data from training devices. For the meta-test

adaptation phase, the learned parameter θ is adapted to enable fast adaptation based on the few pilots P transmitted by the target (meta-test) devices. Consequently, the test data-set $\mathcal{D}_T$ is used to train a demodulator $p_\theta(x|y)$ to minimize the cross-entropy loss given by

$$L_{\mathcal{D}_T}(\theta) = -\sum_{(x^{(n)}|y^{(n)})\in\mathcal{D}_T} \log p_\theta(x^{(n)}|y^{(n)}). \tag{20}$$

Then, the stochastic gradient descent is used to update the parameter θ iteratively as

$$\theta \leftarrow \theta - \eta\nabla_\theta \log(x^{(n)}|y^{(n)}), \tag{21}$$

where the pair $(x^{(n)}|y^{(n)}) \in \mathcal{D}_T$, and η is the step size. As discussed in [37], the purpose of the MAML algorithm is to find the initial parameter θ such that, for any device, the loss after one iteration of (21) applied to the received pilots is minimized. The training and testing (adaptation) steps are illustrated in **Algorithm 4**.

4.3 Numerical Analysis

In this section, we evaluate the performance of the proposed meta-SICNet in terms of symbol error rate (SER) and compare it to the performance of classical SIC and conventional SICNet. The simulation parameters for meta-SICNeT are presented in Table 3. The model considers $K = 20$ meta-device groups for training, and each group includes $L = 2$ non-orthogonal interference devices for the UL system. The transmitted signal to the BS is modulated using the BPSK modulation scheme of modulation order 2, where the signal is superimposed by the power coefficients $P_1 = 4$ and $P_2 = 1$. The SICNet architecture comprises 2 DNN blocks, and each DNN consists of 4 layers; the input and output layers and two hidden layers. The number of neurons in each hidden layer is shown in Table 3. The training loss criteria is based on the *combined loss* from both devices. For the end-to-end training, we used the Adam optimizer with the meta-learning rate $\alpha = 0.1$ and inner learning rate $\beta = 0.001$. For simplicity, we consider a symmetric channel model of ± 1, where half of the training groups have a channel of $+1$ and the other half -1. For adaption on target devices, the channel is chosen randomly as $+1$ or -1. The training is performed on an 11-th Gen Intel Core i5 2.40GHz CPU and 16-GB RAM.

Figure 18 depicts the performance of SER for both meta-SICNet and SICNet for a different number of pilots. The simulation was performed for $N_{te} = 1, 2, \ldots, 8$ with an SNR value of 15 dB. The results show that the meta-learning approach outperforms the SICNet in terms of SER. Moreover, it is clear that increasing the number of pilots improves learning performance. However, for meta-SICNet, the SER relatively saturates faster at 2 or 3 pilots. Note that device 2 always performs better since interference is already removed after decoding the signal from device 1, which captures the SIC effect.

In Figure 19, we plot the symbol outage probability as a function of the SNR for SNR values $\{0, 2, \ldots, 18\}$ using 4 pilot symbols. It appears that the meta-SICNet can capture the effect of SNR as the symbol error rate significantly improves when increasing the SNR. Again, it is obvious that the meta-SICNet outperforms conventional SICNet and the classical SIC for all SNR values, which proves the superiority of the proposed meta-

Algorithm 4: Meta-Learning based SIC

Input: Meta-training data $\mathcal{D} = \{\mathcal{D}_{k=1,\ldots,K}\}$, and meta-testing pilot data $\mathcal{D}_T$; N_{tr}^S and N_{tr}^Q; step size α and β.

Output: Learning parameter vector θ; SER.

Randomly initialize the parameter vector θ.
START META-TRAINING:
while *not done* **do**
  for *each meta-training group* k **do**
    Randomly split $\mathcal{D}_k$ into two sets; support-set $\mathcal{D}_k^S$ of size N_{tr}^S, and query-set $\mathcal{D}_k^Q$ of size N_{tr}^Q.
    Calculate the gradient $\nabla_\theta L_{\mathcal{D}_T}(\theta)$ from (20) with $\mathcal{D}_T = \mathcal{D}_k^Q$, and $\nabla_\theta^2 L_{\mathcal{D}_T}(\theta)$.
    Compute adapted parameters θ' using (21) as

$$\theta_k' = \theta - \beta \nabla_\theta L_{\mathcal{D}_k^Q}(\theta)$$

  end
  update the meta-parameter θ:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{k=1}^{K} L_{\mathcal{D}_k^S}(\theta_k')$$

end
START META-TESTING ADAPTATION:
for *testing epochs* **do**
  Load the learned parameter vector θ.
  Sample data from $\mathcal{D}_T$.
  Update θ in the direction of the gradient with step size η by

$$\theta \leftarrow \theta - \eta \nabla_\theta L_{\mathcal{D}_T}(\theta)$$

end

learning solution for in NOMA uplink. Note that the classical SIC is the worst performer here, so we just added one curve for illustration.

In Figure 20, we address the performance of meta-SICNet for a different number of meta-training tasks (device groups) with 15dB SNR and $N_{te} = \{4, 8\}$ pilots. The simulation shows that the capacity of learning for the meta-SICNet model improves when increasing the meta-training device groups. Therefore, the model can achieve lower SER if the training experience is shared between more device groups.

Table 4 illustrates the training complexity for both meta-SICNet and SICNet in terms of the number of parameters, training time per epoch, and adaptation (testing) time. Although the meta-SICNet approach achieves higher computational complexity, the training can be done offline among the meta-training device groups. Then, the online adaptation process is performed to continuously update the model parameters without having to train the model again. On the contrary, the SICNet model requires completing the

Table 3: meta-SICNet Model Parameters

Parameter	**Value**
Number of device groups for meta-SICNet K	20
Number of devices in each group L	2
Modulation Number M	2
Power factors of the devices P_1, P_2	$4, 1$
Number of DNN layers for SICNet	4
Neurons for DNN block 1	24-12
Neurons for DNN block 2	32-16
Activation function for hidden layers	ReLU
Activation function for output layers	Softmax
Optimizer	Adam
Outer learning rate α	0.1
Inner learning rate β	0.001
Number of training epochs	300
Number of pilots for training data N_{tr}^S, N_{tr}^Q	4, 4
Size of testing data for target devices	10^6
Training SNR	6 dB
Learning-rate for adaptation on test devices η	0.001
Number of adaptation epochs	1000

Table 4: Complexity analysis of meta-SICNet vs. SICNet, 8 pilots

Model	**# Parameters**	**Training time**	**Test-time**
Meta-SICNet	2240	50.3 ms	4.01 ms
SICNet	1120	10.9 ms	41.01 ms

whole training process online whenever the channel condition alters. Therefore, the testing (adaptation) time for the meta-based approach is much less than SICNet for the same number of pilots. Hence, meta-SICNet can achieve lower SER with relatively low online complexity (i.e, only 4.01 ms of adaptation time).

We studied the convergence of our model in Figures 21 and 22. The convergence is measured for two interference devices before and after the MAML update through the meta-training. Both the training loss and error rate converge at 300 training epochs, moreover, the MAML update drives the model to converge faster achieving a lower error rate and loss value. As illustrated in the figures, the performance of the first device is lower than the second since the SICNet cancels the interference and recovers the symbols in order. Thus, the symbols which are recovered first have a higher error rate and loss value than the symbols recovered last.

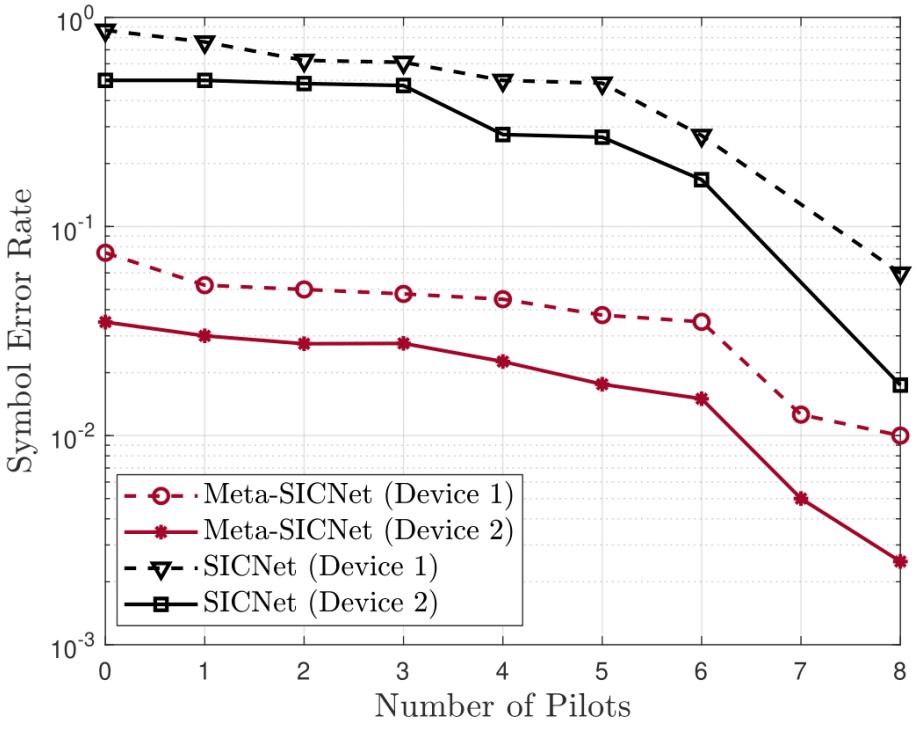


Figure 18: Symbol error rate vs. number of pilots for Meta-testing devices and SICNet (SNR = 15 dB).

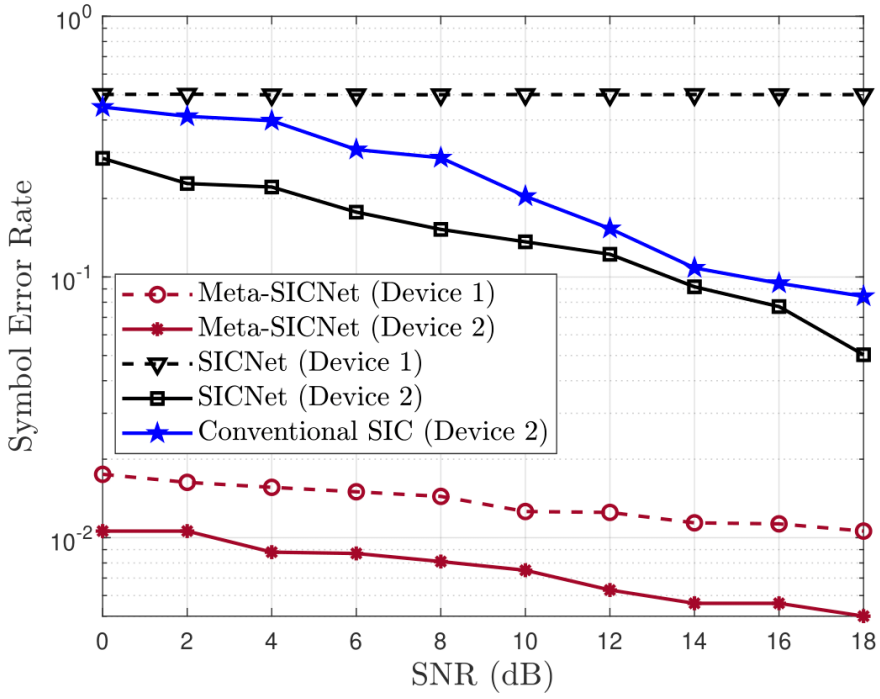


Figure 19: Symbol error rate vs. SNR for Meta-testing devices, SICNet, and conventional SIC (number of pilots = 4).

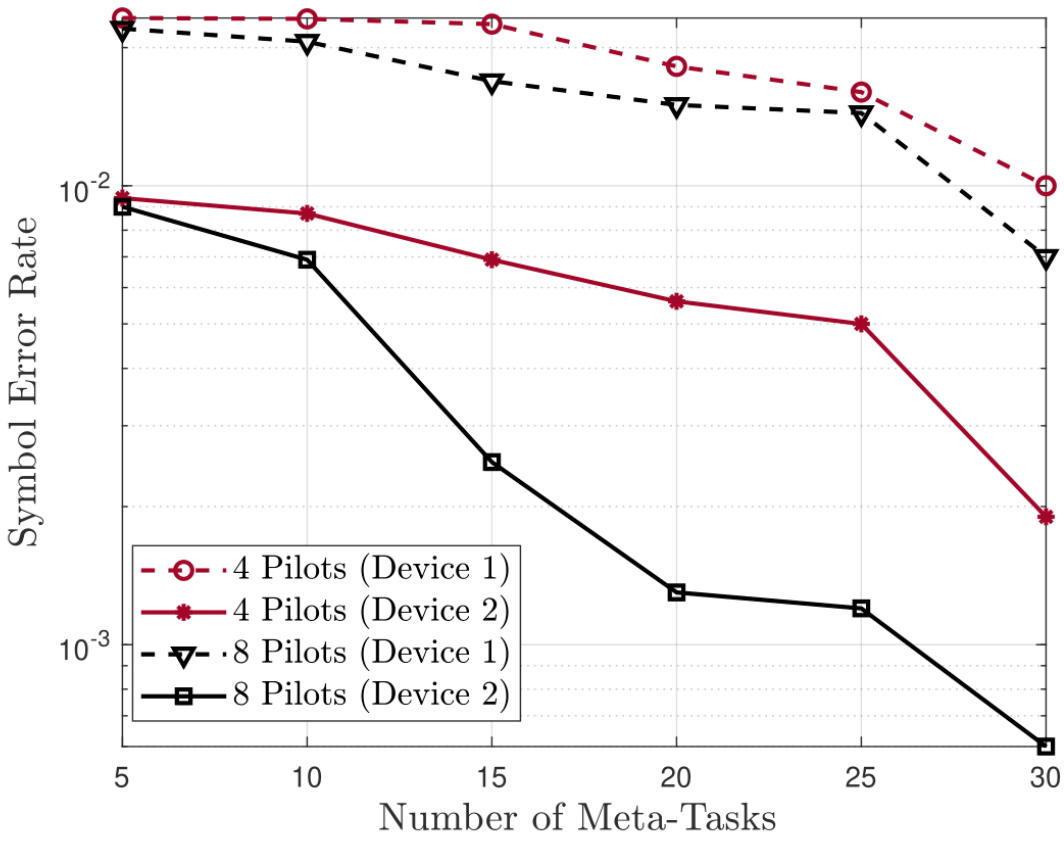


Figure 20: Symbol error rate vs. number of meta-training tasks (number of device groups) for SNR = 15 dB, and number of pilots 4, 8.

Figure 21: Convergence of error rate vs. the training epochs.

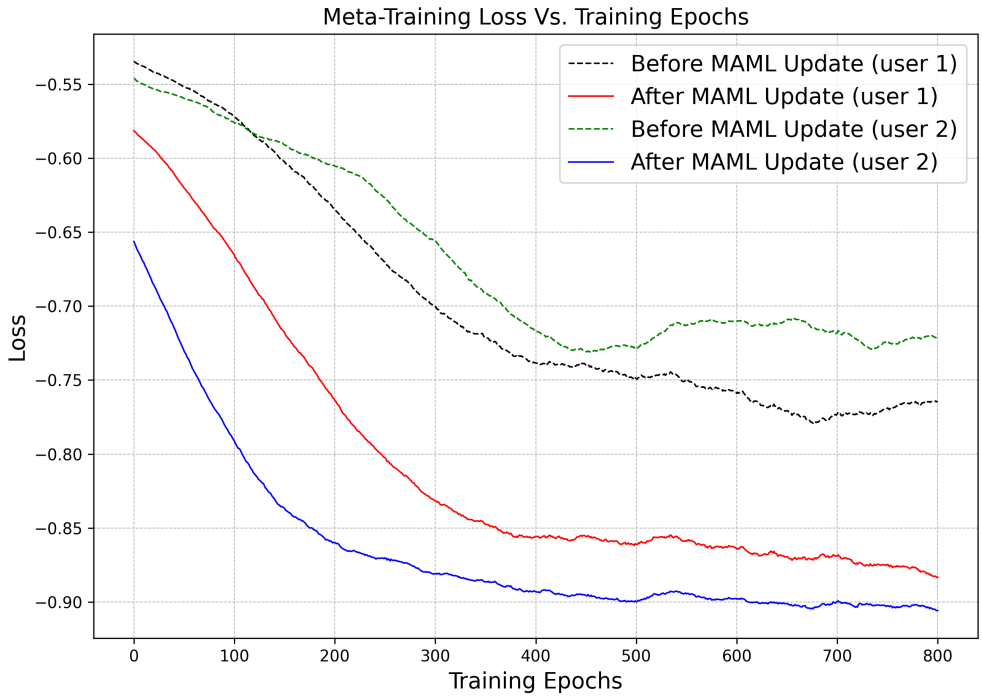


Figure 22: Convergence of loss function vs. training epochs.

5 CONCLUSION

In this thesis, we present the meta-learning approach to optimize the performance of various massive machine-type communication applications where a large number of IoT devices are deployed, especially in URLLC applications where stringent latency and reliability requirements are required. We analyzed the model-agnostic meta-learning algorithm and its applications in different machine-learning fields such as regression, classification, and reinforcement learning. The models are built and implemented in Python using the Pytorch framework for deep learning development. Moreover, we propose implementing MAML in three mMTC applications; *i)* UAV path learning and trajectory planning using reinforcement learning, *ii)* few-pilot demodulation for massive IoT configurations, and *iii)* interference cancellation and demodulation for NOMA schemes. First, we studied the MAML algorithm and analyzed its implementation in regression and RL problems. The simulation results demonstrated the advantages of MAML and meta-learning in optimizing the learning process, allowing the DL model to collect the learning experience and grow monotonically when encountered with unknown situations that require accomplishing new tasks. Moreover, we introduced the UAV path learning problem in terms of AoI and data freshness indicator. We illustrate the detailed system model and the mathematical formulation of the problem. Furthermore, we investigated the different RL solutions and algorithms, i.e., DQN and VPG. The results show that the VPG algorithm outperforms the DQN approach and achieves lower AoI maintaining more freshness of information. Additionally, we discussed the potential of applying the MAML-RL scheme for UAV trajectory planning in cases of varying configurations of IoT sensors. The application of MAML is expected to optimize the AoI and provide the possibility to adapt to the dynamics of UAVs and the change in the relative positions between the sensors and the UAV relays.

Following that, we presented the meta-learning based few-pilot demodulation as a promising solution for short packet transmission in massive IoT scenarios. Consequently, we proposed a MAML-based approach that achieves lower symbol error rates and quick adaptation to the varying channel condition compared to a neural network-based approach. The simulation results manifest the model convergence and its superior performance in cases of a few number of pilots and low SNR value.

Furthermore, in [82], we introduced a data-driven meta-learning-aided NOMA uplink model. Unlike classical SIC and conventional deep learning SIC, the proposed meta-SICNet can accumulate experience across different devices, facilitating the learning process for newly introduced devices and reducing the training overhead. Our results confirm that meta-SICNet outperforms conventional SICNet as it can achieve a lower symbol outage probability. Moreover, meta-SICNet can converge faster and renders significantly good performance for a few pilots (only 2 or 3 pilots). Although meta-SICNet consumes a longer time for training, the training phase is performed offline, and the online adaptation phase consumes a very short time ($\approx$ 4 ms) compared to conventional SIC, which reduces the online complexity.

There are plenty of possible extensions to the meta-SICNet approach proposed here, among which full-duplex NOMA [84] with self-interference cancellation, adaptation to a higher number of devices, and meta-learning aided massive MIMO [85]. Moreover, further future extensions to the work presented in this thesis can be attained, which leads to the realization of the 6G network. A possible extension that can be addressed

by meta-learning is the novel two-dimensional modulation technique, the orthogonal time frequency space modulation (OTFS) [86]. OTFS is designed to effectively perform modulations in time-varying channels by transforming the multipath signal into a two-dimensional delay-Doppler domain. OTFS provides full diversity and supports the large antenna structures deployed in massive MIMO systems. Therefore, combining the power of OTFS with the potential of meta-learning and the MAML algorithm will enhance the performance and the latency of the OTFS scheme.

While data-driven and machine-learning approaches have emerged in various wireless communication fields, our research shows that meta-learning has a huge potential in mMTC and URLLC applications. The reason for that is the ability of meta-learning to satisfy the key performance metrics required by many applications, e.g., energy consumption, reliability, latency, connectivity, and network capacity. However, there are many challenges that need to be addressed in order to achieve the full potential of learning from experience. One of the challenges is obtaining ample knowledge about the problem and the environment. The choice of tasks can be misleading and repetitive and does not fully represent the aspects of the learning problem or the altering conditions of the environment. Hence, it is crucial to construct a distribution of tasks that allow the model to generalize its learning process and gain sufficient experience. Another shortcoming that concerns the implementation of MAML is the assumption that the MAML training encounters new data samples at each iteration (task) which guarantees the model convergence to a generalized learning stage. However, the training often passes through the same data points multiple times, since the training acquires a large distribution of realized datapoints and randomly samples the meta-data for each training task. Therefore, the MAML training requires more refinement to be ble to distinguish between the training error and the so-called generalization error that might compromise the whole meta-learning process and the training convergence.

6 REFERENCES

[1] Kodheli O., Guidotti A. & Vanelli-Coralli A. (2017) Integration of Satellites in 5G through LEO Constellations. In: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, pp. 1–6.

[2] Popovski P., Nielsen J.J., Stefanovic C., De Carvalho E., Strom E., Trillingsgaard K.F., Bana A.S., Kim D.M., Kotaba R., Park J. et al. (2018) Wireless access for ultra-reliable low-latency communication: Principles and building blocks. Ieee Network 32, pp. 16–23.

[3] Mahmood N.H., Böcker S., Moerman I., López O.A., Munari A., Mikhaylov K., Clazzer F., Bartz H., Park O.S., Mercier E. et al. (2021) Machine type communications: key drivers and enablers towards the 6G era. EURASIP Journal on Wireless Communications and Networking 2021, p. 134.

[4] Soltani M., Pourahmadi V., Mirzaei A. & Sheikhzadeh H. (2019) Deep learning-based channel estimation. IEEE Communications Letters 23, pp. 652–655.

[5] Ye H., Li G.Y. & Juang B.H. (2017) Power of deep learning for channel estimation and signal detection in OFDM systems. IEEE Wireless Communications Letters 7, pp. 114–117.

[6] He H., Wen C.K., Jin S. & Li G.Y. (2018) Deep learning-based channel estimation for beamspace mmWave massive MIMO systems. IEEE Wireless Communications Letters 7, pp. 852–855.

[7] Erdogmus D., Rende D., Principe J.C. & Wong T.F. (2001) Nonlinear channel equalization using multilayer perceptrons with information-theoretic criterion. In: Neural Networks for Signal Processing XI: Proceedings of the 2001 IEEE Signal Processing Society Workshop (IEEE Cat. No. 01TH8584), IEEE, pp. 443–451.

[8] O'shea T. & Hoydis J. (2017) An introduction to deep learning for the physical layer. IEEE Transactions on Cognitive Communications and Networking 3, pp. 563–575.

[9] Zhang F., Luo C., Xu J., Luo Y. & Zheng F. (2022) Deep learning based automatic modulation recognition: Models, datasets, and challenges. Digital Signal Processing , p. 103650.

[10] Frikha M.S., Gammar S.M., Lahmadi A. & Andrey L. (2021) Reinforcement and deep reinforcement learning for wireless Internet of Things: A survey. Computer Communications 178, pp. 98–113.

[11] Luong N.C., Hoang D.T., Gong S., Niyato D., Wang P., Liang Y.C. & Kim D.I. (2019) Applications of deep reinforcement learning in communications and networking: A survey. IEEE Communications Surveys & Tutorials 21, pp. 3133–3174.

[12] Lei L., Tan Y., Zheng K., Liu S., Zhang K. & Shen X. (2020) Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. IEEE Communications Surveys & Tutorials 22, pp. 1722–1760.

[13] Nguyen T.T., Nguyen N.D. & Nahavandi S. (2020) Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE transactions on cybernetics 50, pp. 3826–3839.

[14] Yang F., Wang S., Li J., Liu Z. & Sun Q. (2014) An overview of internet of vehicles. China communications 11, pp. 1–15.

[15] Althamary I., Huang C.W. & Lin P. (2019) A survey on multi-agent reinforcement learning methods for vehicular networks. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), IEEE, pp. 1154–1159.

[16] Wang Y., Ye Z., Wan P. & Zhao J. (2019) A survey of dynamic spectrum allocation based on reinforcement learning algorithms in cognitive radio networks. Artificial intelligence review 51, pp. 493–506.

[17] Kumar D.P., Amgoth T. & Annavarapu C.S.R. (2019) Machine learning algorithms for wireless sensor networks: A survey. Information Fusion 49, pp. 1–25.

[18] Chen M., Gündüz D., Huang K., Saad W., Bennis M., Feljan A.V. & Poor H.V. (2021) Distributed learning in wireless networks: Recent progress and future challenges. IEEE Journal on Selected Areas in Communications 39, pp. 3579–3605.

[19] Busoniu L., Babuska R. & De Schutter B. (2008) A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38, pp. 156–172.

[20] Feriani A. & Hossain E. (2021) Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial. IEEE Communications Surveys & Tutorials 23, pp. 1226–1252.

[21] Guo Z., Chen Z., Liu P., Luo J., Yang X. & Sun X. (2022) Multi-agent reinforcement learning-based distributed channel access for next generation wireless networks. IEEE Journal on Selected Areas in Communications 40, pp. 1587–1599.

[22] Zhong C., Gursoy M.C. & Velipasalar S. (2019) Deep multi-agent reinforcement learning based cooperative edge caching in wireless networks. In: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, pp. 1–6.

[23] Cui J., Liu Y. & Nallanathan A. (2019) Multi-agent reinforcement learning-based resource allocation for uav networks. IEEE Transactions on Wireless Communications 19, pp. 729–743.

[24] Eldeeb E., de Souza Sant'Ana J.M., Pérez D.E., Shehab M., Mahmood N.H. & Alves H. (2022) Multi-uav path learning for age and power optimization in iot with uav battery recharge. IEEE Transactions on Vehicular Technology .

[25] Bonawitz K., Eichner H., Grieskamp W., Huba D., Ingerman A., Ivanov V., Kiddon C., Konečnỳ J., Mazzocchi S., McMahan B. et al. (2019) Towards federated learning at scale: System design. Proceedings of machine learning and systems 1, pp. 374–388.

[26] Nagabandi A., Clavera I., Liu S., Fearing R.S., Abbeel P., Levine S. & Finn C. (2018) Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. arXiv preprint arXiv:1803.11347 .

[27] Sakr H. & Elsabrouty M. (2023) Meta-reinforcement learning for edge caching in vehicular networks. Journal of Ambient Intelligence and Humanized Computing 14, pp. 4607–4619.

[28] Hu Y., Chen M., Saad W., Poor H.V. & Cui S. (2020) Meta-reinforcement learning for trajectory design in wireless uav networks. In: GLOBECOM 2020-2020 IEEE Global Communications Conference, IEEE, pp. 1–6.

[29] Van Luong T., Shlezinger N., Xu C., Hoang T.M., Eldar Y.C. & Hanzo L. (2022) Deep learning based successive interference cancellation for the non-orthogonal downlink. IEEE transactions on vehicular technology 71, pp. 11876–11888.

[30] Emir A., Kara F., Kaya H. & Yanikomeroglu H. (2021) DeepMuD: Multi-User Detection for Uplink Grant-Free NOMA IoT Networks via Deep Learning. IEEE Wireless Communications Letters 10, pp. 1133–1137.

[31] Schaufele D., Marcus G., Binder N., Mehlhose M., Keller A. & Stańczak S. (2022) GPU-Accelerated Machine Learning in Non-Orthogonal Multiple Access. In: 2022 30th European Signal Processing Conference (EUSIPCO), pp. 667–671.

[32] Zhang X., Fan P., Liu J. & Hao L. (2022) Bayesian Learning-Based Multiuser Detection for Grant-Free NOMA Systems. IEEE Transactions on Wireless Communications 21, pp. 6317–6328.

[33] Ding Z., Lei X., Karagiannidis G.K., Schober R., Yuan J. & Bhargava V.K. (2017) A Survey on Non-Orthogonal Multiple Access for 5G Networks: Research Challenges and Future Trends. IEEE Journal on Selected Areas in Communications 35, pp. 2181–2195.

[34] Dosti E., Shehab M., Alves H. & Latva-aho M. (2019) On the performance of non-orthogonal multiple access in the finite blocklength regime. Ad Hoc Networks 84, pp. 148–157. URL: https://www.sciencedirect.com/science/article/pii/S157087051830708X.

[35] Khani M., Alizadeh M., Hoydis J. & Fleming P. (2020) Adaptive neural signal detection for massive mimo. IEEE Transactions on Wireless Communications 19, pp. 5635–5648.

[36] Park S., Simeone O. & Kang J. (2020) Meta-learning to communicate: Fast end-to-end training for fading channels. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5075–5079.

[37] Finn C., Abbeel P. & Levine S. (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning, PMLR, pp. 1126–1135.

[38] Chen L., Jose S.T., Nikoloska I., Park S., Chen T. & Simeone O. (2023) Learning with limited samples: Meta-learning and applications to communication systems. Foundations and Trends in Signal Processing 17, pp. 79–208. URL: http://dx.doi.org/10.1561/2000000115.

[39] Beck J., Vuorio R., Liu E.Z., Xiong Z., Zintgraf L., Finn C. & Whiteson S. (2023), A Survey of Meta-Reinforcement Learning. URL: http://arxiv.org/abs/2301.08028.

[40] Wu H., Zhang Z., Jiao C., Li C. & Quek T.Q.S. (2019) Learn to Sense: A Meta-Learning-Based Sensing and Fusion Framework for Wireless Sensor Networks. IEEE Internet of Things Journal 6, pp. 8215–8227.

[41] Park S., Jang H., Simeone O. & Kang J. (2019) Learning how to demodulate from few pilots via meta-learning. In: 2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), IEEE, pp. 1–5.

[42] Khodadadeh S., Boloni L. & Shah M. (2019) Unsupervised meta-learning for few-shot image classification. Advances in neural information processing systems 32.

[43] Ren M., Triantafillou E., Ravi S., Snell J., Swersky K., Tenenbaum J.B., Larochelle H. & Zemel R.S. (2018) Meta-learning for semi-supervised few-shot classification. arXiv preprint arXiv:1803.00676 .

[44] Li K., Gupta A., Reddy A., Pong V.H., Zhou A., Yu J. & Levine S. (2021) Mural: Meta-learning uncertainty-aware rewards for outcome-driven reinforcement learning. In: International conference on machine learning, PMLR, pp. 6346–6356.

[45] Li Y. (2017) Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274 .

[46] Moerland T.M., Broekens J., Plaat A., Jonker C.M. et al. (2023) Model-based reinforcement learning: A survey. Foundations and Trends® in Machine Learning 16, pp. 1–118.

[47] Çalışır S. & Pehlivanoğlu M.K. (2019) Model-free reinforcement learning algorithms: A survey. In: 2019 27th Signal Processing and Communications Applications Conference (SIU), IEEE, pp. 1–4.

[48] Riedmiller M., Peters J. & Schaal S. (2007) Evaluation of policy gradient methods and variants on the cart-pole benchmark. In: 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, IEEE, pp. 254–261.

[49] Arulkumaran K., Deisenroth M.P., Brundage M. & Bharath A.A. (2017) Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine 34, pp. 26–38.

[50] Williams R.J. (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning 8, pp. 229–256.

[51] Schulman J., Levine S., Abbeel P., Jordan M. & Moritz P. (2015) Trust region policy optimization. In: International conference on machine learning, PMLR, pp. 1889–1897.

[52] Bottou L. (2012) Stochastic gradient descent tricks. Neural Networks: Tricks of the Trade: Second Edition , pp. 421–436.

[53] Kiran B.R., Sobh I., Talpaert V., Mannion P., Al Sallab A.A., Yogamani S. & Pérez P. (2021) Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems 23, pp. 4909–4926.

[54] Mnih V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D. & Riedmiller M. (2013) Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 .

[55] Peters J. & Schaal S. (2008) Reinforcement learning of motor skills with policy gradients. Neural networks 21, pp. 682–697.

[56] Kakade S.M. (2001) A natural policy gradient. Advances in neural information processing systems 14.

[57] Franceschetti A., Tosello E., Castaman N. & Ghidoni S. (2021) Robotic arm control and task training through deep reinforcement learning. In: International Conference on Intelligent Autonomous Systems, Springer, pp. 532–550.

[58] Eldeeb E., Pérez D.E., de Souza Sant'Ana J.M., Shehab M., Mahmood N.H., Alves H. & Latva-Aho M. (2022) A learning-based trajectory planning of multiple uavs for aoi minimization in iot networks. In: 2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), IEEE, pp. 172–177.

[59] Kingma D.P. & Ba J. (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

[60] Agarap A.F. (2018) Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 .

[61] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L. et al. (2019) Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32.

[62] Jiang Y., Kim H., Asnani H. & Kannan S. (2019) Mind: Model independent neural decoder. In: 2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), IEEE, pp. 1–5.

[63] Park S., Jang H., Simeone O. & Kang J. (2020) Learning to demodulate from few pilots via offline and online meta-learning. IEEE Transactions on Signal Processing 69, pp. 226–239.

[64] Cohen K.M., Park S., Simeone O. & Shamai S. (2021) Learning to learn to demodulate with uncertainty quantification via Bayesian meta-learning. In: WSA 2021; 25th International ITG Workshop on Smart Antennas, VDE, pp. 1–6.

[65] Cohen K.M., Park S., Simeone O. & Shamai S. (2022) Bayesian Active Meta-Learning for Reliable and Efficient AI-Based Demodulation. IEEE Transactions on Signal Processing 70, pp. 5366–5380.

[66] Park S., Simeone O. & Kang J. (2020) End-to-end fast training of communication links without a channel model via online meta-learning. In: 2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), IEEE, pp. 1–5.

[67] Raviv T., Park S., Simeone O., Eldar Y.C. & Shlezinger N. (2023) Online meta-learning for hybrid model-based deep receivers. IEEE Transactions on Wireless Communications .

[68] Simeone O., Park S. & Kang J. (2020) From learning to meta-learning: Reduced training overhead and complexity for communication systems. In: 2020 2nd 6G Wireless Summit (6G SUMMIT), IEEE, pp. 1–5.

[69] Nichol A., Achiam J. & Schulman J. (2018) On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 .

[70] Zintgraf L., Shiarli K., Kurin V., Hofmann K. & Whiteson S. (2019) Fast context adaptation via meta-learning. In: International Conference on Machine Learning, PMLR, pp. 7693–7702.

[71] Finn C., Rajeswaran A., Kakade S. & Levine S. (2019) Online meta-learning. In: International Conference on Machine Learning, PMLR, pp. 1920–1930.

[72] Simeone O. & Spagnolini U. (2004) Adaptive pilot pattern for OFDM systems. In: 2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577), vol. 2, IEEE, vol. 2, pp. 978–982.

[73] Nguyen C., Do T.T. & Carneiro G. (2020) Uncertainty in model-agnostic meta-learning using variational inference. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 3090–3100.

[74] Louizos C. & Welling M. (2016) Structured and efficient variational deep learning with matrix Gaussian posteriors. In: International conference on machine learning, PMLR, pp. 1708–1716.

[75] Guo C., Pleiss G., Sun Y. & Weinberger K.Q. (2017) On calibration of modern neural networks. In: International conference on machine learning, PMLR, pp. 1321–1330.

[76] Kaddour J., Sæmundsson S. et al. (2020) Probabilistic active meta-learning. Advances in Neural Information Processing Systems 33, pp. 20813–20822.

[77] Kirsch A., Van Amersfoort J. & Gal Y. (2019) Batchbald: Efficient and diverse batch acquisition for deep Bayesian active learning. Advances in neural information processing systems 32.

[78] Liu Y., Qin Z., Elkashlan M., Ding Z., Nallanathan A. & Hanzo L. (2017) Nonorthogonal multiple access for 5g and beyond. Proceedings of the IEEE 105, pp. 2347–2381.

[79] Yuan Y., Yuan Z. & Tian L. (2020) 5G Non-Orthogonal Multiple Access Study in 3GPP. IEEE Communications Magazine 58, pp. 90–96.

[80] Shahab M.B., Abbas R., Shirvanimoghaddam M. & Johnson S.J. (2020) Grant-free non-orthogonal multiple access for iot: A survey. IEEE Communications Surveys & Tutorials 22, pp. 1805–1838.

[81] Elbayoumi M., Kamel M., Hamouda W. & Youssef A. (2020) NOMA-Assisted Machine-Type Communications in UDN: State-of-the-Art and Challenges. IEEE Communications Surveys & Tutorials 22, pp. 1276–1304.

[82] Issa H., Shehab M. & Alves H. (2023) Meta-Learning Based Few Pilots Demodulation and Interference Cancellation For NOMA Uplink. arXiv preprint arXiv:2306.05848 .

[83] Andrews J.G. (2005) Interference cancellation for cellular systems: a contemporary overview. IEEE Wireless Communications 12, pp. 19–29.

[84] Ding Z., Fan P. & Poor H.V. (2018) On the coexistence between full-duplex and NOMA. IEEE Wireless Communications Letters 7, pp. 692–695.

[85] Lu L., Li G.Y., Swindlehurst A.L., Ashikhmin A. & Zhang R. (2014) An overview of massive MIMO: Benefits and challenges. IEEE journal of selected topics in signal processing 8, pp. 742–758.

[86] Hadani R., Rakib S., Tsatsanis M., Monk A., Goldsmith A.J., Molisch A.F. & Calderbank R. (2017) Orthogonal time frequency space modulation. In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, pp. 1–6.