Synthesis Lectures on
Learning, Networks, and Algorithms

SYNTHESIS
COLLECTION OF TECHNOLOGY

Hang Wang · Sen Lin · Junshan Zhang

# Continual and Reinforcement Learning for Edge AI

## Framework, Foundation, and Algorithm Design

Springer

# Synthesis Lectures on Learning, Networks, and Algorithms

**Series Editor**

Lei Ying, ECE, University of Michigan, Ann Arbor, MI, USA

The series publishes short books on the design, analysis, and management of complex networked systems using tools from control, communications, learning, optimization, and stochastic analysis. Each Lecture is a self-contained presentation of one topic by a leading expert. The topics include learning, networks, and algorithms, and cover a broad spectrum of applications to networked systems including communication networks, data-center networks, social, and transportation networks.

Hang Wang · Sen Lin · Junshan Zhang

# Continual and Reinforcement Learning for Edge AI

## Framework, Foundation, and Algorithm Design

Springer

Hang Wang
University of California
Davis, CA, USA

Sen Lin
University of Houston
Katy, TX, USA

Junshan Zhang
University of California
Davis, CA, USA

ISSN 2690-4306 ISSN 2690-4314 (electronic)
Synthesis Lectures on Learning, Networks, and Algorithms
ISBN 978-3-031-84362-4 ISBN 978-3-031-84363-1 (eBook)
https://doi.org/10.1007/978-3-031-84363-1



This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

With the advances in artificial intelligence (AI) and edge computing, edge-AI has recently emerged as the marriage of these two groundbreaking technologies and attracted significant interest from both industry and academia. Particularly, edge-AI seeks to push the AI frontiers to the resource-limited network edge that is closer to the data-generating sources, e.g., mobile devices and other Internet-of-Things (IoT) devices. This new inter-discipline has great potentials in enabling a wide range of on-device AI applications, spanning from video surveillance to personal assistant to autonomous driving. Nevertheless, local data is usually collected online and the underlying data distributions can continuously shift due to the environment changes, which requires the edge AI models to be adapted accordingly in a lifelong manner and efficiently considering the limited computing resources at the network edge. This online and non-stationary nature calls for a new formulation of edge AI from the perspective of continual learning, however, for which the research is still in an infancy stage and a dedicated venue for exchanging the recent advances is lacking.

To fill this void, we present in this book a survey of recent research progress, with 'bias' towards our own research efforts in edge AI, from supervised learning to reinforcement learning. More specifically, we first introduce the background and the motivation of continual learning in edge AI, followed by the potential frameworks and design considerations therein. Next, we identify and provide a detailed overview of key machine learning technologies to enable continual learning and reinforcement learning for edge AI. To better demonstrate the research directions and problems in edge continual AI, we also showcase four of our own research projects and summarize the recent progress in this field. We discuss the promising applications and future research opportunities at the end. We hope that this book will bring up more attentions, spark fruitful discussions and motivate further research ideas on continual and reinforcement learning for edge AI.

This work is supported in part by NSF Grants CNS-2203239, CNS-2148253, CCSS-2413529 and ARO Grant W911NF-2410046.

Davis, USA — Hang Wang
Katy, USA — Sen Lin
Davis, USA — Junshan Zhang
September 2024

# Contents

# About the Authors

**Hang Wang** is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering at University of California, Davis. Previously, he received the B.E. from University of Science and Technology of China (USTC) in 2018. His research aims to establish a fundamental understanding of reinforcement learning, multi-agent systems, and human-AI interaction, as well as practical applications in the domains like Autonomous Driving and Edge Computing. His contributions have been published in NeurIPS, ICML, AAMAS and his recent work on Warm-start Reinforcement Learning also garnered attention and acclaim via an oral presentation at ICML.

**Sen Lin** is an Assistant Professor in the Department of Computer Science at University of Houston. Previously, he was a Postdoc in the NSF AI-EDGE Institute at The Ohio State University. He received his Ph.D. degree from Arizona State University, M.S. from HKUST and B.E. from Zhejiang University. His research interests broadly fall in the intersection of machine learning and wireless networking. Currently, his research focuses on developing algorithms and theories in continual learning, meta-learning, reinforcement learning, adversarial machine learning and bilevel optimization, with applications in multiple domains, e.g., edge computing, security, network control. His research results have been published in top conferences and journals in machine learning and networking. The recognition his papers have received includes the WiOpt'18 Best Student Paper Award and Spotlight presentations in ICLR and ICML.

**Junshan Zhang** has been a professor in the ECE Department at University of California Davis since 2021. He received his Ph.D. degree from the School of ECE at Purdue University in August 2000, and was on the faculty of the School of ECEE at Arizona State University from 2000 to 2021. His research interests fall in the general field of information networks and data science, including edge AI, reinforcement learning, continual learning, network optimization and control, game theory. He is a Fellow of the IEEE, and a recipient of the ONR Young Investigator Award in 2005 and the NSF CAREER award in 2003.

His papers have won a few awards, including the Best Student paper atWiOPT 2018, the Kenneth C. Sevcik Outstanding Student Paper Award of ACM SIGMETRICS/IFIP Performance 2016, the Best Paper Runner up Award of IEEE INFOCOM 2009 and IEEE INFOCOM 2014, and the Best Paper Award at IEEE ICC 2008 and ICC 2017. He is currently serving as Editor-in-Chief for IEEE/ACM Transactions on Networking. He served as Editor-in-Chief for IEEE Transactions onWireless Communication during 2019–2022. He was TPC co-chair for IEEE INFOCOM 2012 and ACM MOBIHOC 2015.


# Introduction to Continual and Reinforcement Learning for Edge AI

1

## 1.1 Introduction

Since its birth in 1956, artificial intelligence (AI) has gone through multiple summers and winters. Driven by the significant advancements of neural network architectures, computing power and big data, the past decade has witnessed an unparalleled growth in machine learning (ML) and AI, leading to phenomenal breakthroughs in a wide spectrum of applications, e.g., speech recognition [191], image classification [141, 276], object detection [219, 323], etc. In fact, GPU throughput and memory have increased 10× in the last four years. By leveraging the parallelism of the GPU hardware and more training data, the transformer architecture can now train much more expressive models than ever, giving rise to a new era of foundation models. It is widely recognized that these intelligent applications will significantly enrich people's lifestyle and improve human productivity.

In general, the machine learning problems can be mainly divided into three categories:

- *Supervised Learning.* Supervised learning is the process of learning a function that maps an input to an output (label) given a training dataset of inputs and their corresponding labels, such that the labels for unseen inputs can be accurately inferred based on the function. Regression and classification are examples of supervised learning, which have led to unparalleled successes in computer vision.
- *Unsupervised Learning.* In unsupervised learning, there are no labels given for the training data, and the objective is to learn the underlying structure in the data. The most common unsupervised learning tasks are clustering, e.g., finding groups in data, and density estimation, e.g., summarizing the distribution of data.
- *Self-Supervised Learning and Reinforcement Learning.* Self-supervised learning aims to build models that automatically find patterns in data and reveal these patterns explicitly with a representation. In particular, reinforcement learning (RL), a popular approach for



H. Wang et al., *Continual and Reinforcement Learning for Edge AI*,
Synthesis Lectures on Learning, Networks, and Algorithms,
https://doi.org/10.1007/978-3-031-84363-1_1

self-supervised learning, is used to train an agent that can not only perceive and understand their surroundings through vision and other sensing modalities but also can navigate and interact with their physical environments to achieve goals, engage in planning, and perform reasoning. In essence, reinforcement learning is a learning paradigm that learns how to use past data, e.g., offline data or data collected through online interaction, to enhance the future manipulation of a dynamical system, which can be cast as an optimal control problem when the system dynamics are unknown [32]. The fruitful applications, e.g., games, robotics, and substantive interactions with other disciplines, e.g., operation research, control theory and game theory, make reinforcement learning a very popular and critical research direction nowadays.

One of the key driving forces behind AI is the development of deep learning and deep neural networks (DNNs) since 2010s, which have achieved astonishing successes in solving ML problems and demonstrated great superiority over classical ML approaches, e.g., decision tree and Baysian networks. Notably, consisting of a series of layers, artificial neural networks (ANNs) can extract the underlying features from data in a hierarchical manner and provide a universal function approximator for ML problems. Multiplayer Perceptrons (MLPs) are the most basic ANNs with fully connected neurons and non-linear activation functions. To capture the spatial correlation in the input data, especially for images, Convolution Neural Networks (CNNs) [179] replace the basic linear operations in MLPs with convolution operations, making them very popular for computer vision tasks. Recurrent Neural Networks (RNNs) [386] are another type of ANNs which specialize in handling sequential data and hence are widely used for natural language processing (NLP) tasks, e.g., machine translation and question answering. Unlike feedforward neural networks such as MLPs and CNNs which process data in a single pass, RNNs are able to process data across multiple time steps, with an internal memory to remember the knowledge of previous inputs and use that for learning at current time steps. Another special kind of ANN architectures is the generative model, which aims to solve generative tasks, e.g., image generation. Generative adversarial networks (GANs) [117] are in this category, which consist of two separate networks, namely generator network and discriminator network. The generator seeks to generate new data to mimic real data in the training dataset, whereas the discriminator seeks to distinguish the fake data generated by the generator from the real data. Recently, another powerful type of generative models named as diffusion models [69] has attracted much attention, which gradually add random noise to the input data and then learn to reconstruct desired data samples from the noise by reversing the diffusion process. In 2017, a new ANN architecture, namely Transformer [335], was proposed to address the limitation of RNN-based encoder-decoder architecture in solving sequence-to-sequence tasks, by leveraging the attention mechanism to capture the long-range dependencies across data inputs in a highly parallel manner. Due to its superior performance and computational efficiency, the Transformer now becomes the mainstream architecture and is widely used in pretrained foundation models such as large language models. Based on the scaling law, the performance of ANNs often improves with

larger network size and more training data samples, which can significantly increase the need of computational resources and memory size.

### 1.1.1 Edge AI

Recently, with the rapid proliferation of mobile computing and Internet of Things (IoT), big data is going through a radical shift of data source from the mega-scale cloud datacenters to the network edge, e.g., mobile devices and IoT devices, as illustrated in Fig. 1.1. It is anticipated that the data generated by connected devices would reach 175 zettabytes by 2025 [16], far greater than that the cloud datacenters could handle. As the personal data collected by IoT devices is sensitive in nature, there is a growing consensus that much of the personal data should be used for training the models locally and would never go beyond the network edge. Besides, many intelligent applications, such as autonomous driving and augmented reality, need to accomplish decision making with low latency, in order to meet the requirements for safety, accuracy, performance and user experience. Clearly, the conventional wisdom of transporting the data bulks from the IoT devices to the cloud datacenters for analytics would not work well, due to the extremely stringent requirements in cost, privacy and performance. As a result, it is anticipated that a high percentage of IoT data will be stored and processed locally, giving rise to a new research area, namely 'edge AI' as the marriage of edge computing and AI.

More specifically, edge AI refers to the AI model training or inference at the resource-limited network edge by leveraging available data and computational resources across edge devices and cloud datacenters. Recently, a lot of research studies have emerged to explore various aspects of edge AI:

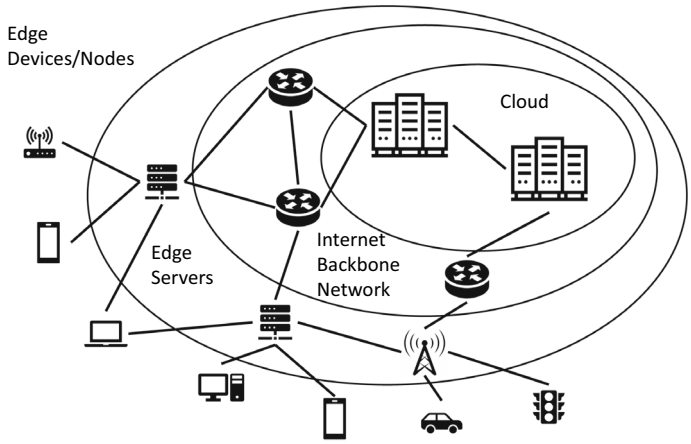


**Fig. 1.1** Illustration of Edge AI

- *Model training in edge AI.* Various ML techniques have been used to enable edge AI. For example, federated learning (FL) trains a global model in a distributed manner based on the local data across multiple devices, which is widely used for model training at the network edge, e.g., [3, 169, 226, 284, 288, 358]. Split learning splits a global model into multiple sections and each edge device trains one section by using its local data, e.g., [61, 106, 255, 285]. A pre-trained model in the cloud can be transferred to an edge device based on transfer learning for further fine-tuning with local data, e.g., [46, 62, 370, 372]. Various model compression techniques are also frequently used in edge AI to improve the efficiency of memory, computation, and communication for model training, including knowledge distillation (e.g., [10, 36, 361, 404]), quantization (e.g., [60, 184, 205, 251]), and model pruning (e.g., [130, 156, 222]).
- *Model inference in edge AI.* To make full use of the available data and computation resources in edge networks, the model inference in edge AI can be implemented in different frameworks. The trained model can be split into two parts, with one in the edge devices and the other in the cloud. The intermediate results calculated in the device based on local data will be sent to the remaining part of model in the cloud to complete the inference (e.g., [161, 172, 180]). To reduce the latency and communication cost, collaborations between edge serves and edge devices have been introduced for model inference. For example, [204] studied how to accelerate inference for edge-device collaboration by jointly optimizing the model partitioning and right-sizing. Moreover, the inference can be done based on the collaborations between edge devices, which can be very useful in scenarios with high mobility or harsh environments (e.g., [84, 405]). Recent studies have also investigated efficient on-device model inference. For instance, OnceNAS [397] was proposed in to jointly optimize the number of parameters and inference latency through neural architecture search, which has significantly reduced the model size and increased the inference speed.
- *Applications of edge AI.* Due to the advantages of privacy protection, low latency, low cost, and high flexibility in personalization, edge AI can find applications in a lot of different domains, including smart healthcare [53, 123, 160, 275], smart cities [9, 66, 148, 324], smart agriculture [90, 269, 308], autonomous vehicles [54, 75, 369], recommendation systems [101], etc.

### 1.1.2 Continual Learning and Reinforcement Learning for Edge AI

Despite the great potential to create many novel intelligent applications and fuel the continuous prosperity of AI, pushing the AI frontier to the network edge for achieving edge AI is highly nontrivial. Specifically, running AI applications directly on edge devices to process the IoT data locally, if not designed intelligently, would suffer from poor performance and energy inefficiency, simply because many AI applications typically require high computa-

tional power that greatly outweighs the capacity of resource- and energy-constrained IoT devices. Tackling these unique challenges in edge networks successfully is vital to sustain the rapid progress of this field. To this end, an appropriate and efficient edge learning framework should possess the following properties:

- *Guarantee performance under small datasets.* While tremendous data will be generated by the ensemble of IoT devices, the amount of personal data at every edge node is limited. Therefore, to facilitate various edge-AI applications, especially on-device AI applications, the edge learning algorithms should be able to effectively extract useful knowledge from small local datasets.
- *Work efficiently with simple learning algorithms.* In contrast to more capable cloud datacenters, the edge node only has limited computational resources, which significantly restrict its capability to run complicated learning algorithms on complex models. Consequently, it is more desirable to just deploy simple algorithms, e.g., gradient descent, on a more compact model.
- *Learn quickly with high communication efficiency.* The strict requirement on performance and latency for many AI applications clearly precludes the approaches that need extensive training time, calling for a fast learning algorithm that can quickly train ML models at each edge node with performance guarantee. In the meanwhile, the communication efficiency should be taken into consideration for edge learning frameworks based on collaborations among the cloud and edge nodes.
- *Adapt continuously.* Distributions of locally generated data can shift in a lifelong manner due to the environment change. For instance, autonomous vehicles can experience different terrain and weather conditions when collecting data. The online and non-stationary nature of local data at edge nodes requires model training in edge AI to adapt continuously, such that the knowledge learned previously will not only be maintained but also be leveraged to facilitate better learning of new data.

While a lot of existing studies on edge AI have proposed various edge learning frameworks that meet the first three properties, much less attention has been paid to frameworks that can enable continuous adaptation of AI models at the network edge [350, 406]. Clearly, the standard ML paradigms that focus on learning with a fixed dataset and stationary data distributions would not work well. On one hand, directly applying the model learned for previous data distributions can fail in solving new tasks due to the data distribution shift. On the other hand, simply adapting the learned model with the new data can result in the forgetting of the knowledge of old data and sometimes may even hinder the new task learning. Note that continual learning (CL) [215] is a learning paradigm in ML which continuously adapts a single model to learn a sequence of tasks without forgetting the learned knowledge of old tasks. In particular, by leveraging the knowledge transfer across different tasks, continuoual learning (CL) can potentially solve new tasks more easily and also improve the model performance on old tasks. Therefore, a new formulation of edge

learning from the perspective of CL is needed towards building edge AI that possess the four properties above.

Many edge AI applications involve in-time decision making to meet the requirements for safety and performance [80]. The goal is to find optimal control policies to maximize long-term returns. For example, autonomous vehicles (AV) should learn expert control policies to drive safely and efficiently; a virtual health assistant seeks to create a personalized wellness plan based on a patient's unique genetic profile and health history, in order to maximize his/her health. RL [32, 321] provides a promising solution to these problems by learning through interactions with the environment. However, one of the critical challenges to prevent RL from being directly used for edge AI is that RL algorithms typically require extensive online interactions with the environment for policy learning, which can be costly for resource-limited edge nodes and dangerous for safety-critical applications such as AV [170, 388] and healthcare [67, 383]. Another challenge is that most RL algorithms focus on the policy learning for stationary environments, whereas the system dynamics for edge AI applications can continuously change due to various reasons such as the physical environment changes in self-driving [253]. To address these challenges and put the great promise of RL on the ground for edge AI, continual RL recently emerged by introducing the idea of CL into RL, which can be carried out from two perspectives:

- *Warm-start RL.* To eliminate the need of online interactions, offline RL has recently attracted extensive attention by learning from offline datasets previously collected via some behavior policy [7, 200]. It is usually guaranteed that the learned policy is a better one than the behavior policy. However, the performance of pure offline RL highly depends on the quality of the offline dataset, and the learned policy from a low-quality dataset can not be directly used in real applications, even with improved performance over the behavior policy. To address this, warm-start RL combines offline RL and online RL, which continuously finetunes the learned offline policy using a few online interactions with the environment. From a different perspective, warm-start RL also enables fast online policy learning by leveraging the offline policy learned from a fixed dataset, significantly improving the usability of RL in edge AI.
- *Continual RL.* Warm-start RL typically assumes that the underlying MDP stays the same for both offline and online learning, whereas the online environment can be nonstationary in edge AI. The control problem in a stationary environment can be treated as an MDP and the MDP formulation changes when the system dynamics change. To handle the distribution shift across different MDPs, continual RL [1, 164] leverages the idea of CL to continuously finetune the online policy, such that new MDPs can be quickly solved without forgetting the policies learned for previous MDPs.

As alluded to earlier on, the research in CL and RL for edge AI is still in its infancy stage, and there is an urgent need to build a dedicated venue for exchanging the recent advances. In the rest of this chapter, we will discuss the frameworks for building efficient edge AI with CL and RL and also the important design considerations.

## 1.2 Framework

To fully leverage the available data and resources across the edge networks, there are various frameworks to enable CL for edge AI depending on different application scenarios.

- *Edge-only.* The learning problem locally at a single edge node can be formulated as an online continual learning problem, because of the online and nonstationary nature of local data. In this case, due to the limited data samples and constrained computation resources at the edge node, the on-edge CL algorithms need to be able to extract useful knowledge from the local data in an effective and efficient manner. Particularly, the CL algorithms should continuously adapt the local ML model to solve the new task, corresponding to the newly collected data, quickly with only a few data samples, whereas the knowledge of the learned tasks should be preserved as well in a computationally efficient way. Towards this end, a key idea is to enhance the knowledge transfer between old tasks and new tasks across the time. More specifically, appropriately leveraging the accumulated knowledge of old tasks can significantly reduce the number of data samples required for learning similar new tasks, following the same spirit as in transfer learning and meta-learning. On the other hand, by facilitating the positive knowledge transfer for the new task to similar old tasks, the model performance on these old tasks can also be potentially improved. This edge-only framework is particular important for on-device intelligent applications, such as autonomous vehicles, personalized healthcare, and audio/video surveillance.
- *Cloud-edge collaboration.* While sending a large amount of local data to the cloud for processing is not feasible for edge AI, the abundant history data and computation resources in the cloud are clearly beneficial for edge learning if leveraged through appropriate cloud-edge collaborations. In particular, there are two commonly used frameworks for cloud-edge collaboration: (1) *Cloud as a priori.* The model generalization capability closely hinges upon the number of available training data samples. With the huge amount of history data and computing resources in the cloud, a powerful ML model can be pretrained with the capability of grasping general knowledge and features from data, such as the foundation models. These pretrained models in the cloud provide a very helpful priori for local edge learning. In particular, finetuning from (or learning an adapter with) the pretrained model using local data can not only solve edge tasks more easily by building on the learned general knowledge in the pretrained model, but also solve edge tasks better by leveraging the excellent knowledge extracting capability of the pretrained model. Based on this pretrained-finetuning paradigm, the CL at the edge can be substantially improved

by starting from the pretrained models in the cloud. As the pretrained models are typically very large in size to guarantee the generalization capability, how to efficiently finetune these model at the computation-constrained edge nodes is one key challenge herein. (2) *Cloud as a platform.* On the other hand, the cloud can serve as a platform to exchange and aggregate the information among different edge nodes, such that the CL on one edge node can potentially leverage the useful knowledge from other edge nodes with similar learning tasks. More specifically, each participating edge node has a local CL problem, and seeks to solve the new tasks based on the knowledge extracted from its past tasks and also the information from other edge nodes shared through the cloud. Following the same idea as in federated learning, the information from edge nodes should be shared in such a way that the privacy of edge nodes is protected and the communication cost is minimized. However, if not designed carefully, aggregating information from other nodes may not only hinder the new task learning but also exacerbate the forgetting of old tasks at one edge node, due to the potential interference between different edge nodes.

- *Edge-edge collaboration.* Communicating with a central cloud platform may not be always feasible, or trustworthy even if feasible. In this case, the information sharing between different edge nodes in an edge network can be done in a decentralized manner, where each edge node can only communicate with its neighbors in a certain range, e.g., self-driving cars in a vehicle-to-vehicle network. This leads to a framework of decentralized CL for which each node has a local CL problem. Similar to cloud-edge collaboration where the cloud serves as a platform, how to appropriately share information across different edge nodes is a central problem to guarantee the performance of this learning framework for edge-edge collaboration. Moreover, the communication protocol in this case is particularly important because the edge nodes share limited communication bandwidth and the information propagation delay across the edge network can significantly affect the learning performance.

## 1.3 Performance Measures and Efficiency

To provide guidance for algorithm design and performance evaluation of the designed algorithms for edge AI with CL and RL, it is essential to define various metrics for evaluating the learning performance and also the computational costs. In this monograph, we introduce the following metrics, which can be used to evaluate how the edge AI system performs in specific aspects:

- *Task learning performance.* Needless to say, different types of tasks may use different performance metrics. For example, test accuracy is usually used for prediction tasks such as image classification, whereas the Frechet-Inception Distance score is widely adopted for evaluating the performance of GAN models in generative tasks.

  - **New task learning performance**. The metric measures how well the ML model performs after adapting with the new coming data at the edge. In general, the learning performance of a good edge CL algorithm is expected to be better than learning from scratch with the new data only, because of the forward knowledge transfer from similar tasks that the edge node has experienced in the past or useful knowledge shared by the cloud and other edge nodes.
  - **Task average learning performance**. In many real applications, not only the new task learning performance matters, but also the average learning performance over all seen tasks. For example, an autonomous vehicle should be able to drive safely in all environments it has experienced. A good task average learning performance indicates that the edge AI system demonstrates competitive performance overall, which is particularly attractive when all tasks are equally important.
  - **Task average forgetting**. The forgetting for a particular task evaluates the performance change between the ML model after learning this task and the ML model after learning the current task. A positive value of forgetting means that the knowledge of old tasks has been forgotten after learning new tasks, whereas a negative value implies that the knowledge gained from new tasks indeed improves the model performance on old tasks.
  - **Task sample complexity**. Task sample complexity typically evaluates the number of data samples required to train a good ML model for solving a particular task. Since the amount of available data samples at a single edge node is often small, a practical edge learning algorithm should be capable of effectively extracting knowledge from limited data, leading to a low sample complexity. This metric is particularly important to evaluate the learning efficiency of RL algorithms, in terms of the number of interactions with the environment for policy learning.

- *Computational efficiency.* Unlike most standard ML algorithms, the computational efficiency is a very critical factor, which could be even more critical than the learning performance, in performance evaluation for edge learning algorithms, considering the constrained computational resources at the network edge.

  - **Model training time**. Model training time for a particular task is the most direct metric to evaluate the computational efficiency of an edge learning algorithm, which evaluates the time taken from the start of the training to when a good quality ML model is obtained.
  - **Number of training steps**. Computational capability of different edge nodes can be heterogeneous, which highly affects the model training time. In contrast, the number of training steps is a metric that does not depend on the computing power at the network edge but directly evaluate the convergence speed of an edge learning algorithm.

  - **Memory usage**. The memory usage of a learning algorithm typically depends on various factors, including the ML model size, intermediate training results required by the algorithm, additional data points, etc. Considering that edge nodes often have a small memory size, edge learning algorithms should leverage the limited memory in a highly efficient manner.

- *Communication efficiency.* Communication efficiency is also an important factor in characterizing the performance of edge learning algorithms, especially for cloud-edge and edge-edge collaboration frameworks. Typically, edge nodes communicate with each other and the cloud through wireless communication channels with limited spectrum. Both communication cost and communication frequency will affect the communication efficiency of the edge AI system.

  - **Communication cost**. The communication cost is usually captured by the communication bandwidth and the transmission time needed for each communication round. It highly depends on what information is shared between the cloud and the edge nodes, e.g., data, gradient, ML model. To ensure the success rate of the communication and also the overall efficiency of the edge AI system, it is important to reduce the communication cost for edge learning algorithms.
  - **Communication frequency**. The communication frequency can also be evaluated as the number of communication rounds during the entire model training process. Reducing the communication frequency in the edge learning algorithms will not only minimize the impact of unreliable communication channels but also mitigate the potential interference among different communication links in the wireless edge networks, therefore improving the communication efficiency.

In this monograph, we provide an overview of recent advances in continual learning and reinforcement learning in edge AI, with 'bias' towards our own research efforts in this area, and offer our subjective perspective on recent trends and potential cross-disciplinary developments.

# Part I
# Algorithmic and Theoretical Foundations

In Part I, we provide an overview of key machine learning techniques in edge AI, for enabling continuous model training, including continual learning, reinforcement learning, and meta-learning, and present a few recent research results of ours in these areas.


# Continual Learning for Edge AI

# 2

## 2.1 Introduction

Continual learning (CL) [258] is a learning paradigm where an agent needs to continuously learn a sequence of tasks. To emulate the remarkable lifelong learning ability of humans, the agent is expected to leverage accumulated knowledge from previous tasks to more easily learn new ones, and further improve the learning performance of old tasks by leveraging the knowledge of new tasks. The former is referred to as forward knowledge transfer and the latter as backward knowledge transfer. One major challenge herein is the so-called *catastrophic forgetting* [231], i.e., the agent easily forgets the knowledge of old tasks when learning new tasks.

***General setup*** In CL, different tasks arrive in a sequential manner, and each task has its own training dataset and task identity, where each data sample consists of the input feature and the corresponding label. The objective here is to train a model sequentially for learning each new task with no or limited access to old task data, such that the model can perform well on the test datasets for all seen tasks. Depending on the characterization of task datasets and availability of task identities, there are several typical CL setups:

- *Domain incremental learning*: Tasks share the same label space but have different input feature distributions.
- *Task incremental learning*: Tasks may have different input distributions and label spaces, and task identities need to be provided during both training and testing.
- *Class incremental learning*: Tasks may have different input distributions and label spaces, and task identities are known during training but not testing.
- *Online CL*: The task training data arrive as an online data stream and can only be used to update the model once.



H. Wang et al., *Continual and Reinforcement Learning for Edge AI*,
Synthesis Lectures on Learning, Networks, and Algorithms,
https://doi.org/10.1007/978-3-031-84363-1_2

***Evaluation metrics*** Let $A_{j,i}$ is the accuracy of the model on $i$-th task after learning the $j$-th task sequentially where $j \geq i$. To evaluate the performance of the learned model in CL, there are three metrics widely used in the literature:

- Overall accuracy $AA_j$: measures the average test accuracy of the model learned after task $j$ on all seen tasks

$$AA_j = \frac{1}{j} \sum_{i=1}^{j} A_{j,i}.$$

- Backward knowledge transfer $BWT_j$: measures the average forgetting (if negative) or the performance improvement (if positive) of old tasks after learning task $j \geq 2$

$$BWT_j = \frac{1}{j-1} \sum_{i=1}^{j-1} A_{j,i} - A_{i,i}.$$

- Forward knowledge transfer $FWT_j$: measures the impact of all old tasks on the learning performance of the CL model in the current task $i$

$$FWT_j = \frac{1}{j-1} \sum_{i=2}^{j} A_{i,i} - A_{i,*}.$$

  Here $A_{i,*}$ is the performance of some reference model, e.g., a randomly-initialized model trained using the data from task $i$, in task $i$.

## 2.2 Theoretical Studies on Continual Learning

The theoretical understanding of CL is still in the early stage, where only a few attempts have emerged recently. Specifically, Bennani et al. [31] and Doan et al. [87] analyzed generalization error and forgetting for the orthogonal gradient descent (OGD) approach [100] based on NTK models, and further proposed variants of OGD to address forgetting. Yin et al. [377] proposed a unified framework for the performance analysis of regularization-based CL methods, by formulating them as a second-order Taylor approximation of the loss function for each task. Asanuma et al. [22] and Lee et al. [195] studied CL in the teacher-student setup to characterize the impact of task similarity on forgetting performance. Cao et al. [47] and Li et al. [210] investigated continual representation learning with dynamically expanding feature spaces, and developed provably efficient CL methods with a characterization of the sample complexity. Chen et al. [59] characterized the lower bound of memory in CL using the PAC framework. By investigating the information flow between neural network layers,

Andle and Yasaei Sekeh [15] analyzed the selection of frozen filters based on layer sensitivity to maximize the performance of CL. Goldfarb and Hand [115] investigated the impact of overparameterization for linear models in a two-task setup. Evron et al. [94] studied CL in overparameterized linear models by analyzing the forgetting based on the training data. Li et al. [206] explored the theory in the case of applying the mixture-of-experts in CL. Nevertheless, none of these existing works show an explicit form of forgetting and generalization error, that only depends on fundamental system parameters/setups (e.g., number of tasks/samples/parameters, noise level, task similarity/order). In this chapter, we provide the first-known explicit theoretical result in a more general CL setup with an arbitrary number of tasks, which enables us to comprehensively understand which factors are relevant and how they (precisely) affect forgetting and generalization error of CL.

### 2.2.1 CL in Linear Models

Consider the standard CL setup where a sequence of tasks $\mathbb{T} = \{1, \ldots, T\}$ arrives sequentially in time.

**Ground truth**. We consider a linear ground truth [26, 94] for each task. Specifically, for task $t$, the output $y \in \mathbb{R}$ is given by

$$y_t = \hat{\boldsymbol{x}}_t^\top \hat{\boldsymbol{w}}_t^* + z_t, \tag{2.1}$$

where $\hat{\boldsymbol{x}}_t \in \mathbb{R}^{s_t}$ denotes the feature vector, $\hat{\boldsymbol{w}}_t^* \in \mathbb{R}^{s_t}$ denotes the model parameters, and $z_t$ is the random noise. Here $s_t$ denotes the number of features of ground truth (i.e., the number of true features). In practice, true features are unknown in advance. Therefore, when choosing a model to learn a certain task, people usually choose more features than enough such that all possible features are included. We write this formally into the following assumption.[1]

**Assumption 2.1** We index all possible features by $1, 2, \cdots$. Let $\mathcal{W}$ denote the set of indices of all the chosen features in the model to be trained, with cardinality $|\mathcal{W}| = p$. Let $\mathcal{S}_t$ denote the set of indices of $t$-th task's true features, with cardinality $|\mathcal{S}_t| = s_t$. We assume that $\bigcup_{t\in\mathbb{T}} \mathcal{S}_t \subseteq \mathcal{W}$.

We next define an expanded ground-truth vector $\boldsymbol{w}_t^* \in \mathbb{R}^p$ that expands the original ground-truth vector $\hat{\boldsymbol{w}}_t^*$ from dimension $s_t$ to dimension $p$ by filling zeros in the positions $\mathcal{W} \setminus \mathcal{S}_t$. Let $\boldsymbol{x}_t$ be the corresponding features for $\boldsymbol{w}_t^*$. Therefore, the ground truth Eq. (2.1) can be rewritten as

$$y_t = \boldsymbol{x}_t^\top \boldsymbol{w}_t^* + z_t. \tag{2.2}$$

[1] When Assumption 2.1 does not hold, the derivation techniques for Theorem 2.1 in the next section still hold with a minor modification that treats the missing features as noise.