Scalable and Continuous Federated Learning on Edge-Cloud Computing

Rui Sun

School of Computing
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

School of Computing March 2025

I would like to dedicate this thesis to my loving parents and my wife.

Declaration

I declare that this thesis was composed by myself and is my own work except where explicitly stated otherwise in the text. No part of this thesis has previously been submitted for a degree or any other qualification at Newcastle University or any other institution, except as specified.

Rui Sun
March 2025

Publications

Part of the works within this thesis have been documented in the following publications:

Published

Note: (* These authors contributed equally to this work.)

1. Y. Zhang, H. Duan, **R. Sun**, Y. Cheng, T. Shah, R. Ranjan, B. Wei, *"LAGD: Local topological-Alignment and Global semantic-Deconstruction for Incremental 3D Semantic Segmentation."* Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI), 2025.

2. D. Alqattan, **R. Sun**, H. Liang, G. Nicosia, V. Snasel, R. Ranjan, V. Ojha, *"Security Assessment of Hierarchical Federated Deep Learning."* Internet Corporation for Assigned Names and Numbers (ICANN), 2024.

3. H. Duan*, **R. Sun***, V. Ojha, T. Shah, Z. Huang, Z. Ouyang, Y. Huang, Y. Long, R. Ranjan, *"Dual Variational Knowledge Attention for Class Incremental Vision Transformer."* In 2024 International Joint Conference on Neural Networks (IJCNN), pp. 1-8. IEEE, 2024.

    - Chapter 4 is based on this paper.

4. Y. Ma, K. Liu, M. Chen, Y. Li, **R. Sun**, R. Ranjan, *"Rapid Crowd Evacuation for Passenger Ships Using LPWAN."* IEEE Transactions on Intelligent Transportation Systems, 2023.

5. Z. Wen, H. Hu, R. Yang, B. Qian, R. W. H. Sham, **R. Sun**, J. Xu, P. Patel, O. Rana, S. Dustdar, R. Ranjan, *"Orchestrating Networked Machine Learning Applications Using Autosteer."* IEEE Internet Computing, 26(6), 51-58, 2022.

6. **R. Sun**, Y. Li, T. Shah, R.W. Sham, T. Szydlo, B. Qian, D. Thakker, R. Ranjan, *"FedMSA: A Model Selection and Adaptation System for Federated Learning."* Sensors, 22(19), 7244, 2022.

- Chapter 3 is based on this paper.

7. J. Phengsuwan, T. Shah, **R. Sun**, P. James, D. Thakker, R. Ranjan, *"An Ontology-Based System for Discovering Landslide-Induced Emergencies in Electrical Grid."* Wiley Transactions on Emerging Telecommunications Technologies (ETT), 2020.

8. J. Phengsuwan, T. Shah, N. B. Thekkummal, Z. Wen, **R. Sun**, D. Pullarkatt, H. Thirugnanam, M. V. Ramesh, G. Morgan, P. James, R. Ranjan, *"Use of Social Media Data in Disaster Management: A Survey."* Future Internet, 2021.

9. Z. Wen, J. Phengsuwan, N. B. Thekkummal, **R. Sun**, P. J. Chidananda, T. Shah, P. James, R. Ranjan, *"Active Hazard Observation via Human-in-the-Loop Social Media Analytics System."* In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM), pp. 3469-3472, 2020.

Under Review or Preprinted

Note: (* These authors contributed equally to this work.)

1. **R. Sun**, Y. Zhang, H. Duan, V. Ojha, T. Shah, B. Wei, R. Ranjan, *"Exemplar-condensed Federated Class-incremental Learning."* International Joint Conference on Artificial Intelligence (IJCAI), 2025.

   - Chapter 5 is based on this paper.

2. **R. Sun**, H. Duan, J. Dong, V. Ojha, T. Shah, R. Ranjan, *"Rehearsal-free Federated Domain-incremental Learning."* IEEE Transactions on Neural Networks and Learning Systems.

   - Chapter 6 is based on this paper.

3. **R. Sun***, Y. Zhang*, T. Shah, J. Sun, S. Zhang, W. Li, H. Duan, B. Wei, R. Ranjan, *"From Sora What We Can See: A Survey of Text-to-Video Generation."* arXiv preprint arXiv:2405.10674, 2024.

4. Y. Ma, K. Liu, **R. Sun**, T. Shah, Z. Ouyang, Y. Li, R. Ranjan, *"Adaptive Path Planning and Charging Arrangement for Electric Vehicle Navigation."* Knowledge-Based Systems, 2024.

Acknowledgements

I am heartily indebted to my PhD co-advisors, Dr. Tejal Shah, Professor Rajiv Ranjan, and Professor Graham Morgan, for their unwavering belief in my abilities and their steadfast support throughout my PhD journey. I feel incredibly privileged to have worked with Dr. Shah and Professor Ranjan, who began supervising me during my MSc studies. I owe them both a profound debt of gratitude for their patience and inspiration. Dr. Shah played a crucial role in honing my drafting and writing skills, and her expertise was instrumental in guiding my exploration of federated learning. Professor Ranjan, meanwhile, introduced me to IoT-Edge-Cloud Computing and Distributed Systems, which opened the door to the exciting research areas that form the foundation of this thesis.

I extend my heartfelt thanks to the members of the NUSE group, including Dr. Yinhao Li, Yumin Zhang, Stanly Wilson Palathingal, Kwabena Adu-Duodu, Dr. Jedsada Phengsuwan, Professor Zhenyu Wen, Shamil Al-Ameen, Sultan Altarrazi, Dr. Fawzy Habeeb, Duaa Alqattan, Dr. Nipun Balan TH, Dr. Trinadh Pamulapati, Ringo Sham, Marc Walker, Dr. Bo Wei, Dr. Tomasz Szydlo, Dr. Varun Ojha, and Dr. Wei Sun, for their invaluable research advice and support. The regular feedback provided during our weekly NUSE group meetings significantly enhanced the ideas presented in this thesis. I am particularly grateful to Dr. Szydlo and Dr. Ojha for their guidance and meticulous proofreading of my work on numerous occasions. A special thanks to Dr. Phengsuwan, who initiated my research journey, inspired many of the ideas in this thesis, and provided unwavering support throughout my research endeavors.

I would like to express my deepest appreciation to my wife, Wenqi Li, for her unwavering support, care, and companionship, which have been my greatest source of strength throughout this long and challenging journey, from my MSc to my PhD, and as we look forward to the future together. During moments when I felt overwhelmed or stuck in my work, she knew exactly how to lift my spirits, whether by taking me out for a perfect meal or planning a getaway to help me relax and recharge. Her thoughtful care and constant encouragement allowed me to pursue my PhD without worry, enabling me to focus fully on my research. Wenqi has been my rock, providing

not only practical support in our daily life but also the emotional strength I needed to persevere through the challenges of this journey. I am truly grateful for her love, patience, and belief in me.

This thesis would not have been possible without the unwavering support of my family, particularly my father, Zhongli Sun, and my mother, Fang Gao. I am deeply grateful for their selfless dedication and unconditional support, even when I made mistakes or did not perform well in exams. My heartfelt thanks also go to my father-in-law, Gaowu Li, my mother-in-law, Peisheng Luo, and my brother-in-law, Ziliang Li, whose encouragement has been a constant source of strength. I am also profoundly grateful to my friends, including Dr. Fan Wan, Dr. Haoran Duan, Dr. Bing Zhai, Xin Liu, Zizhou Ouyang, Jiawei Ji, Zhen Wang, Yuting Ma, Jiahao Sun, Shuoying Zhang, and Yejun Wang, for their companionship and support throughout this journey. A special thanks to Zhen Wang, who provided invaluable support during my MSc and PhD studies, and to Dr. Haoran Duan, who not only shared his expertise in AI and Computer Vision but also supported me in both academic and personal matters. I would also like to express my deep appreciation to Yejun Wang, my former line manager in Shenzhen, for teaching me invaluable skills in project management, enterprise-level coding, and problem-solving, which were crucial in completing my PhD.

Abstract

A critical challenge in developing intelligent systems that operate efficiently in distributed and dynamic environments is ensuring their ability to continually learn from new data while maintaining data privacy. Federated Learning (FL) offers a decentralized solution that enables intelligent systems to collaboratively adapt to new information across diverse environments without the need for centralizing data. However, existing FL systems face significant challenges in scalability, efficiency, and the ability to retain and integrate new knowledge, particularly when dealing with varied hardware, network conditions, and evolving data streams. Relying on traditional, centralized methods for data aggregation and model training can lead to bottlenecks, reduced operational efficiency, and increased vulnerability to single-point failures. This thesis addresses these challenges by proposing enhancements to FL systems that not only preserve privacy but also improve scalability, adaptability, continual learning capabilities, and robustness, making them better suited for real-world deployment.

To address hardware heterogeneity in FL, this thesis introduces **FedMSA**, a hardware-adaptive model selection method that balances performance and efficiency using hardware-aware Neural Architecture Search (HW-NAS). This approach simplifies model selection and adapts architectures dynamically, catering to the diverse hardware in large-scale FL deployments.

Building on this, **FedDAS** is introduced to enhance the robustness of FL through runtime system monitoring and diagnostics, detecting issues like device failures or network delays and providing real-time adaptation suggestions. This system ensures FL applications remain stable and efficient in dynamic environments by combining automated adaptation with human-in-the-loop diagnostics.

In FL systems, continual learning is crucial as clients generate new data that must be incorporated without forgetting previous knowledge. To address this, the thesis introduces the **Dual Variational Knowledge Attention (DVKA)** mechanism, which balances the retention of old knowledge with the integration of new information, mitigating catastrophic forgetting and ensuring robust learning in FL contexts requiring continual adaptation.

Further advancing Federated Continual Learning (FCL), the thesis presents **ECoral**, a novel approach that improves the effectiveness of replay-based methods in class-incremental learning by condensing exemplars and enhancing knowledge retention in non-IID environments.

Finally, the thesis extends continual learning principles into the federated domain with **RefFil**, a rehearsal-free Federated Domain-Incremental Learning (FDIL) framework. RefFil utilizes global prompt-sharing and domain-specific contrastive learning to tackle domain shifts in FL, ensuring robust generalization across diverse and evolving data distributions, which is crucial for performance in resource-constrained and privacy-sensitive environments.

These contributions collectively advance FL systems by combining adaptive model selection, robust continual learning strategies, and efficient data management, offering a comprehensive solution for deploying scalable and privacy-preserving machine learning in complex, distributed environments.

Table of contents

List of figures

List of tables

Acronym List

Acronym	Description	Acronym	Description
AI	Artificial Intelligence	IoT	Internet of Things
FL	Federated Learning	GDPR	General Data Protection Regulation
CPU	Central Processing Unit	GPU	Graphics Processing Unit
5G	Fifth Generation (mobile network)	Wi-Fi	Wireless Fidelity
CIL	Class Incremental Learning	NAS	Neural Architecture Search
FedMSA	Federated Model Selection and Adaptation	FedDAS	Federated Diagnosis and Adaptation System
ECoral	Exemplar-condensed Federated Class-incremental Learning	RefFil	Rehearsal-free Federated Domain-Incremental Learning
FDIL	Federated Domain-Incremental Learning	ML	Machine Learning
DL	Deep Learning	ANN	Artificial Neural Networks
SVM	Support Vector Machines	CNN	Convolutional Neural Network
RNN	Recurrent Neural Network	MLP	Multi-Layer Perceptron
DBN	Deep Belief Networks	NLP	Natural Language Processing
ResNet	Residual Network	DML	Distributed Machine Learning
SMPC	Secure Multi-party Computation	HFL	Horizontal Federated Learning
VFL	Vertical Federated Learning	FedAvg	Federated Averaging
DARTS	Differentiable Architecture Search	AutoML	Automated Machine Learning
HPO	Hyperparameter Optimization	FedNAS	Federated Neural Architecture Search
ER	Experience Replay	LwF	Learning without Forgetting
EWC	Elastic Weight Consolidation	BO	Bayesian Optimization
MS	Model Selection	HW-NAS-Bench	Hardware-aware Neural Architecture Search Benchmark
EffPT	Efficient and Performance Trade-off	IB	Information Bottleneck
KD	Knowledge Distillation	ViT	Vision Transformer
FVKA	Feature-level Variational Knowledge Attention	TVKA	Token-level Variational Knowledge Attention
CE	Cross Entropy	FCIL	Federated Class-Incremental Learning
FCL	Federated Continual Learning	DC	Dataset Condensation
SSD	Sample Summarization Distillation	BwT	Backward Transfer
FwT	Forward Transfer	MKCL	Meta-Knowledge Contrastive Learning
VAE	Variational Auto-Encoder	FINCH	Fast, Incremental Clustering
CDAP	Client-wise Domain Adaptive Prompt	DPCL	Domain-specific Prompt Contrastive Learning
GPL	Global Prompt Learning	RefFiL	Rehearsal-free Federated Domain-Incremental Learning
FDIL	Federated Domain-Incremental Learning	DIG	Digits-five
PACS	Photo, Art painting, Cartoon, Sketch	FedDomainNet	Federated DomainNet
LPG	Local Prompts Group	CCDA	Cross-Client Domain Adaptation
LT	Linear Transformation	L2P	Learning to Prompt
DualPrompt	Dual Prompting for Continual Learning		

Chapter 1

Introduction

As artificial intelligence (AI) continues to permeate various aspects of modern life, from healthcare [107] to finance [94] and autonomous vehicles [118], the need for more robust, adaptable, and privacy-preserving machine learning models has become increasingly critical. The deployment of AI in real-world applications frequently encounters challenges such as data privacy concerns, the necessity for continual adaptation to new information, and the efficient management of increasingly large and complex datasets. To address these challenges, it is imperative to develop advanced techniques that enhance the capability of machine learning models to learn continuously, adapt to dynamic environments, and do so in a distributed manner while safeguarding privacy.

These challenges are further compounded by the rapid expansion of cloud computing and the Internet of Things (IoT) industry. Since the advent of cloud computing around 2005 [4], it has profoundly impacted daily life and industrial production, enabling services such as social media, cloud storage, and Industry 4.0. Simultaneously, the number of IoT devices is forecasted to reach over 41 billion by 2027 [64], generating vast amounts of data that are transferred to cloud servers. However, the centralized nature of cloud computing, coupled with stringent data protection regulations like the EU/UK GDPR [39], poses significant challenges in data collection across heterogeneous sources. Additionally, network bandwidth constraints and privacy concerns further complicate centralized data storage and processing. As a result, there is a growing shift towards on-device machine learning, which addresses these issues by enabling devices to locally train models and make predictions, thereby reducing bandwidth usage and enhancing data privacy [24].

Existing approaches, such as homomorphic encryption [176] and differential privacy [160] techniques, offer robust methods to protect data during transfer or enable model training within a constrained number of computation steps. However, these

methods often demand significant computational resources, making them less feasible for large-scale machine learning applications. While Distributed Machine Learning [148] approach provides the ability to train models across multiple devices efficiently, it typically requires all compute nodes to be able to access raw data or the exchange of intermediate states of the learning model. This dependency exposes the system to potential attacks and privacy breaches. On the other hand, techniques like continual learning [123, 84, 162, 21] are designed to enable models to incorporate new knowledge incrementally from new data. However, these methods often encounter limitations when applied on a single device, where the data may be drawn from the same distribution or possess similar features. This homogeneity in data fails to enhance the model's robustness, transferability, and generalization capabilities when it is exposed to entirely new or diverse environments. Consequently, these methods, while effective in specific scenarios, fall short of providing a comprehensive solution to the combined challenges of privacy, adaptability, and efficiency in machine learning.

In contrast, Federated Learning (FL) [100], first proposed in 2016, emerges as a powerful and holistic approach that inherently addresses these challenges. FL is a distributed machine learning paradigm that enables multiple devices or institutions to collaboratively train a shared model without the need to exchange their raw data. By keeping the data localized and only sharing model updates (such as gradients), FL significantly reduces the risk of data breaches and ensures a higher level of privacy. Furthermore, FL supports the integration of continual learning principles by allowing the model to be updated incrementally as new data becomes available from diverse sources, leading to a more adaptable and robust model. This diversity in data sources enhances the model's ability to generalize across different environments, overcoming the limitations of training on homogeneous data. Additionally, FL optimizes the use of distributed computational resources, making it more scalable and efficient for large-scale machine learning tasks.

1.1 Motivation

Federated Learning has emerged as a powerful solution for enabling collaborative machine learning across multiple devices while maintaining user privacy. In FL, models are trained locally on devices, and only model updates, not raw data, are sent to a central server for aggregation. This decentralized approach is essential for protecting sensitive data, especially in an era of increasing privacy concerns. However, deploying FL efficiently on edge devices, which are typically resource-constrained, presents several

unique challenges that need to be addressed to ensure the system's scalability and robustness.

One of the primary challenges in FL is system heterogeneity. This issue arises due to the wide range of devices participating in the training process, which vary in computational power (e.g., CPU, GPU), memory, storage, network connectivity (e.g., 5G, Wi-Fi), and power levels (e.g., battery life). Some devices might be high-end servers, while others could be resource-limited mobile phones or IoT devices. This disparity makes it difficult to design a uniform training process that works efficiently across all devices. A single, uniform approach risks either overwhelming weaker devices or underutilizing more powerful ones. As a result, it is essential to develop adaptive mechanisms that can tailor the training process to each device's capabilities, balancing computational load and optimizing resource utilization across the network.

Beyond hardware differences, another critical challenge in FL is statistical heterogeneity. Since FL operates on data locally generated by individual clients, rather than centrally pooled, the data collected by different devices can vary significantly, leading to substantial variability. For instance, data collected by a mobile app or an autonomous vehicle might vary greatly due to different user behaviors or environmental conditions, such as varying weather patterns. This variability introduces fundamental challenges in the learning process, requiring models to adapt to diverse data distributions without sacrificing performance.

A more advanced manifestation of statistical heterogeneity arises when models need to continuously learn from new data. For example, in class-incremental scenarios, a user's behavior and preferences may change over time, such as shifting from taking photos of cats to taking photos of dogs. In domain-incremental scenarios, a surveillance camera or an autonomous vehicle might need to recognize the same objects under different environmental conditions, such as when the weather changes from sunny to snowy. These situations exacerbate the variability in data distributions and often lead to the severe issue of catastrophic forgetting, where the model, while learning new tasks, may overwrite previously acquired knowledge, resulting in a decline in performance on older tasks. Addressing these complexities requires advanced algorithms capable of not only harmonizing differences in data distributions but also preventing catastrophic forgetting, ensuring robust model performance across all clients.

Effective global model aggregation in FL thus depends on the ability to continuously learn from new data while preserving previously acquired knowledge. Advanced strategies must ensure that models can incorporate new information without compro-

mising the accuracy of existing knowledge, all while adhering to the stringent privacy constraints inherent in FL.

In summary, addressing the challenges of system heterogeneity, statistical variability, and catastrophic forgetting is essential for the success of FL in dynamic and resource-constrained environments. This thesis seeks to tackle these challenges through the development of new methods and architectures that enhance the scalability, robustness, and adaptability of FL systems in real-world scenarios.

1.2 Research Questions

Among the numerous challenges inherent in FL, system heterogeneity, particularly in the context of large-scale deployments across heterogeneous devices, and the need for continuous learning from novel data stand out as particularly critical in real-world FL deployments. Addressing these challenges is essential for ensuring that FL systems are robust, adaptable, and efficient, particularly in dynamic and resource-constrained edge-cloud computing environments. This thesis focuses on enhancing the performance of FL systems by addressing these challenges through four interrelated studies. Each study in this thesis is designed to improve the system's capability in large-scale deployment, continual learning and data efficiency, ultimately contributing to the broader goal of making FL more scalable and effective in diverse real-world scenarios. In particular, this PhD thesis is guided by the following research questions:

1. *How can an adaptive model selection system in FL be designed to balance training efficiency and model performance in hardware-heterogeneous environments?* In the context of FL, diverse hardware among participating devices creates challenges in achieving both high model performance (accuracy, inference latency) and training efficiency (training time, resource utilization). As FL developers often have different expectations, from prioritizing fast training to focusing on accuracy, it is essential to automate model selection and adaptation based on these varying needs.

2. *How can a model's ability to retain knowledge be enhanced during the local model updating process for a single FL client, enabling continuous learning on class-incremental data while minimizing catastrophic forgetting?* FL involves both local training and global aggregation. Each client operates in its own environment, where new data may arise during local training. To ensure effective global aggregation, it is crucial that clients perform continuous learning while

minimizing catastrophic forgetting, so that the updated local model does not negatively affect the global model in subsequent aggregation rounds.

3. *How can knowledge retention techniques be optimized to make efficient use of limited exemplar memory, enhancing information storage while mitigating catastrophic forgetting, without compromising data privacy in FL?* In FL, memory constraints make it difficult to store representative data from previous tasks. This challenge is particularly pronounced in edge-cloud environments, where clients are often resource-limited edge devices. Developing methods that maximize the utility of limited memory space, while ensuring data privacy, is key to preventing catastrophic forgetting in these resource-constrained settings.

4. *What effective strategies can be developed for handling domain-incremental data in FL, focusing on enhancing cross-domain prediction robustness and minimizing catastrophic forgetting in resource-constrained environments?* In FL, particularly in scenarios involving domain-incremental learning, clients often face evolving data distributions across diverse domains. This challenge is further amplified in resource-constrained edge-cloud environments where memory and computational power are limited. Developing strategies that can maintain model performance across domains without relying on rehearsal techniques is critical to improving the overall robustness and efficiency of FL systems.

1.3 Overview of contributions

The central theme of this thesis is the enhancement of FL systems through novel approaches, including adaptive model selection techniques to address system heterogeneity, innovative mechanisms like DVKA to prevent catastrophic forgetting and support continual learning, pioneering information management frameworks such as ECoral to optimize memory usage and maintain privacy, and groundbreaking strategies like RefFil to handle domain shifts and enhance cross-domain robustness. These novel contributions ensure that FL systems can efficiently learn, adapt, and generalize across a wide range of diverse and dynamic environments.

(Chapter 3) Enhancing FL Through Adaptive Model Selection. The first study introduces a novel approach to Research Question 1 by addressing the challenges of FL in environments with heterogeneous devices and varying computational resources. Traditional FL systems often rely on predefined model architectures that may not be well-suited for diverse hardware, leading to inefficiencies in both training and

performance. To tackle this, FedMSA (Model Selection and Adaptation) is introduced, a system that leverages Neural Architecture Search (NAS) to dynamically select and adapt model architectures for different devices. FedMSA automates the model selection process by balancing training efficiency and model performance based on hardware capabilities and developer preferences. This ensures that models can be efficiently deployed and trained across a wide range of devices, making FL systems more scalable and adaptable in real-world applications.

Next, FedDAS is developed upon the foundation of FedMSA, where the former extends the automating model selection and adaptation capabilities of FedMSA by adding real-time diagnostic and anomaly detection functionalities. This novel extension allows for continuously monitoring system-level metrics, providing alerts for issues such as device failures or communication delays. This human-in-the-loop system enables users to intervene and make adjustments on the fly, ensuring the smooth operation of FL tasks even in large-scale, dynamic environments, significantly enhancing the robustness and efficiency of FL systems. Together, FedMSA and FedDAS offer a comprehensive solution that not only simplifies model deployment and adaptation but also enhances the robustness and efficiency of FL systems through continuous monitoring and real-time adaptation.

(Chapter 4) Enhancing model retention of learned information during local updates in FL. This study addresses Research Question 2, which focuses on enabling clients to perform continual learning within FL environments without suffering from catastrophic forgetting. This challenge is addressed by introducing a novel mechanism, Dual Variational Knowledge Attention (DVKA), an approach designed to optimize the attention processes in transformers, a model architecture well-suited for sequential data processing. DVKA enhances the balance between retaining existing knowledge and integrating new information, ensuring that the model remains effective over time.

By equipping transformers with DVKA, the study shows that the model's capacity to handle the complexities of continual learning in federated settings is significantly improved. This advancement is crucial for FL systems, as it allows them to incrementally learn new tasks while maintaining a strong grasp of previously acquired knowledge, which is essential for their reliable deployment in dynamic and resource-constrained environments.

(Chapter 5) Optimizing Data Efficiency in Federated Continual Learning. Building on the advancements in information retention from Chapter 4, this study addresses Research Question 3, focusing on improving data efficiency in federated continual learning, particularly in class-incremental scenarios where clients encounter streaming

data with new classes. The study introduces ECoral, a novel Exemplar-Condensed federated class-incremental learning framework that enhances the information density of rehearsal exemplars, thereby increasing data management efficiency and preserving privacy in FL environments.

By distilling the knowledge from streaming data into more informative rehearsal exemplars, ECoral ensures that FL systems can manage data more effectively, reducing the computational burden and addressing privacy concerns. This contribution ties directly into the central theme of this thesis by optimizing the way data is utilized within FL systems, ensuring that these systems can operate efficiently even in resource-constrained environments.

(Chapter 6) Addressing Domain Shifts in FL. Building on the prompt-based model structure introduced in Chapter 4 and the efficient model selection framework established in Chapter 3, this chapter tackles Research Question 4 by focusing on the challenge of domain shifts within FL environments. In previous chapters, we addressed scalable deployment and continual learning in resource-constrained settings by dynamically selecting model architectures (Chapter 3) and mitigating catastrophic forgetting using limited memory and exemplar condensation (Chapters 4 and 5). In contrast, this chapter extends these principles to more extreme scenarios where models must continue training without requiring a full-scale redeployment. The study introduces novel RefFil, a Federated Domain-Incremental Learning (FDIL) framework that enhances the robustness of local models by leveraging a combination of global prompt-sharing and domain-specific contrastive learning. These techniques collectively strengthen the acquisition of domain-invariant knowledge, allowing models to generalize more effectively across diverse and evolving data distributions.

By focusing on domain-incremental scenarios, where task distinctions are subtle, and the risk of catastrophic forgetting is higher, RefFil directly contributes to the central theme of this thesis by ensuring that FL systems can maintain high performance even as they encounter new and diverse data distributions. Thus, this work emphasizes the importance of developing systems for FL that can be generalized across a wide range of domains and environments.

In combination, these novel contributions form a robust and unified framework that supports both the continuous deployment and learning of federated systems in dynamic and resource-constrained environments. This thesis advances state-of-the-art in FL, continual learning, neural architecture search, and dataset distillation and also provides a unified solution for some of the most pressing challenges in modern machine learning.

1.4 Thesis outline

The remainder of this thesis is structured as shown in Figure 1.1. The arrows indicate the logical flow of chapters, where earlier chapters establish foundational knowledge and methods that are expanded upon in later chapters.

Chapter 2 provides a comprehensive literature review, setting the stage by discussing the current state of FL, model selection, and continual learning techniques.

Chapter 3 introduces a system designed for large-scale FL deployment, incorporating model selection algorithms and real-time diagnostics to enhance robustness in heterogeneous FL environments.

Chapter 4 presents a mechanism focused on improving knowledge retention in FL, addressing catastrophic forgetting during local updates of individual clients when new class data is continually introduced.

Chapter 5 introduces an information-centric framework, building on the retention strategies from Chapter 4, to optimize data efficiency in class-incremental FL scenarios by enhancing the utility of exemplar memory while preserving data privacy.

Chapter 6 proposes a model-centric framework to address domain shifts in FL, further building on Chapter 4 strategies to enhance cross-domain robustness and mitigate catastrophic forgetting without relying on data replay.

Finally, **Chapter 7** concludes the thesis by summarizing the key contributions, discussing their implications for FL, and outlining future research directions.

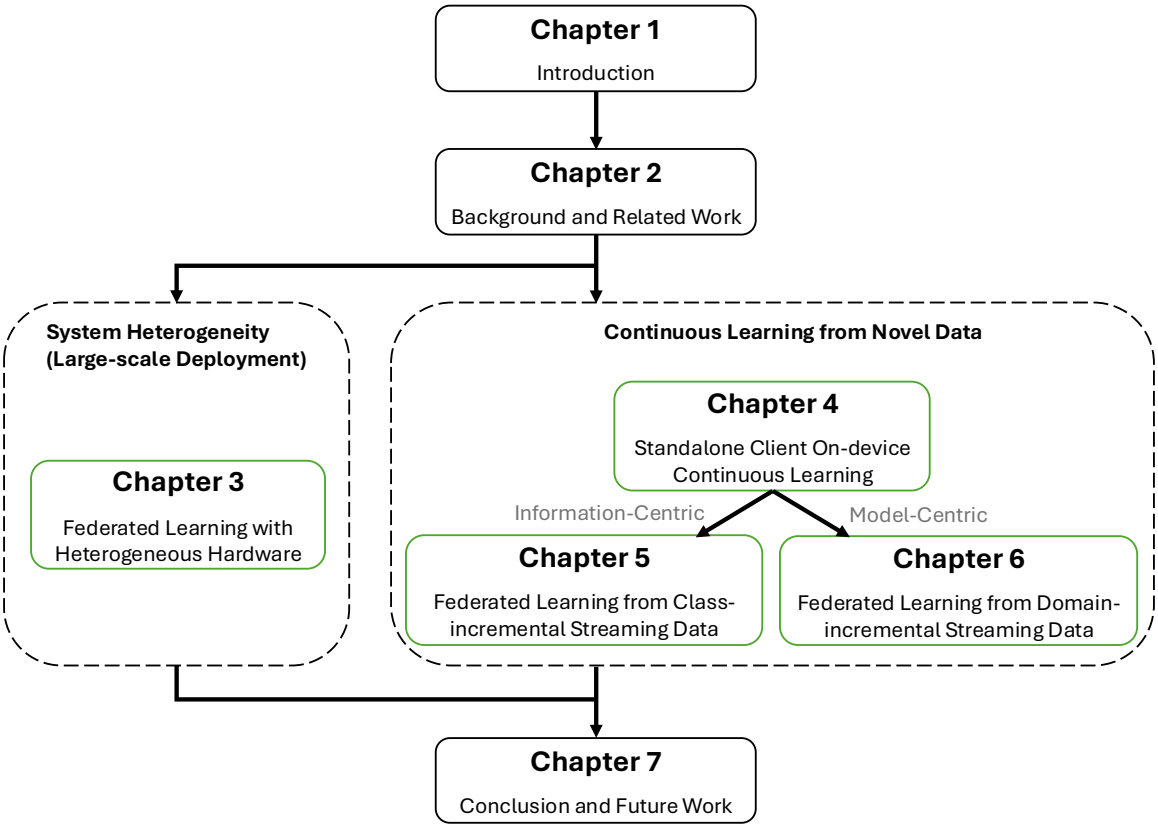

Fig. 1.1 Thesis organization.

Chapter 2

Background and literature review

This chapter provides an overview of the background and related work relevant to this thesis. Given the rapid expansion of the fields of federated learning and machine learning, this section is not intended to offer a comprehensive survey of the entire body of literature. Instead, the focus is on examining key studies that are closely aligned with the research presented in this thesis, emphasizing their relevance and contributions.

2.1 Machine Learning and Deep Learning

Machine Learning (ML), a subfield of artificial intelligence (AI), was proposed by Arthur Samuel in 1959 [159, 67], marking a significant shift from rule-based systems to data-driven approaches where algorithms improve their performance by processing more data. Initially, ML relied on symbolic AI methods like decision trees [136], but these had limitations in handling complex data. As the field evolved, statistical approaches were introduced, leading to foundational algorithms such as the Perceptron [127] in 1958, and more advanced techniques like support vector machines (SVMs) [55].

Over the past two decades, ML has made significant progress, finding successful applications in many areas of technology and science, such as computer vision, natural language processing, autonomous vehicles, and robotics. Traditional ML tasks, including classification or linear regression, typically require only several minutes or hours of training on a server or personal computer. However, the rise of Deep Learning (DL), a subset of ML based on artificial neural networks (ANNs) [172], has led to numerous technological breakthroughs. Deep learning gained prominence with the introduction of deep convolutional neural networks in 2012 [76], which significantly outperformed traditional ML approaches.

The resurgence of deep learning in the 2000s was driven by the availability of large-scale datasets, advances in computational power, particularly through the use of GPUs, and the development of novel architectures like deep belief networks (DBNs) [58]. The complexity of models designed by ML engineers continues to grow. Architectures like ResNet [53], which allows neural network layers to be continuously superimposed without degrading the model's accuracy, have pushed the envelope by expanding to 110 layers with 1.7 million parameters. Additionally, the Transformer model [147] introduced in 2017, further advanced natural language processing (NLP) by leveraging attention mechanisms to capture long-range dependencies more effectively. The basic Transformer model contains 67 million parameters, while larger versions can reach 213 million, significantly increasing computational complexity and resource demands compared to the three-layer multi-layer perceptron (MLP) models widely used in classic ML tasks.

Simultaneously, the proliferation of edge and IoT devices connected to the Internet has led to high volumes of data being generated and transmitted to cloud servers for model training. However, this simultaneous and high-frequency transmission of large amounts of data can cause network overload and risk user privacy. Furthermore, stringent data regulations, such as the General Data Protection Regulation (GDPR) [39], complicate access to cloud service providers and the collection of data from terminal devices like mobiles and sensors. These regulations have significantly impacted traditional machine learning approaches, which often require collecting all user data in a centralized location for training. This centralization not only raises privacy risks but also introduces challenges in compliance with data protection laws. The increased focus on safeguarding personal information has made it difficult for organizations to freely aggregate and use large datasets for model training, as this could expose sensitive user data and violate privacy regulations. In response, research has increasingly focused on decentralized model training methods that preserve privacy, which has become a major challenge and an emerging direction in the field. This forms the foundation of this thesis.

2.2 Federated and Distributed Machine Learning

Distributed machine learning (DML) [82] is an approach to training machine learning models across multiple computational nodes, often in geographically dispersed locations. This paradigm is essential for handling large-scale datasets and complex models that exceed the capacity of a single machine. In DML, data and computation are distributed

across multiple nodes or devices, allowing for parallel processing and reducing the overall training time. DML is particularly useful in scenarios where data is too large to be stored in one place or when computational resources need to be shared among different machines to accelerate learning.

There are two primary strategies in distributed machine learning: data parallelism and model parallelism. In data parallelism, the dataset is divided across multiple nodes, each of which trains a separate copy of the model on its portion of the data, and periodically shares model updates with a central server for aggregation. In contrast, model parallelism involves splitting the model itself across different nodes, with each node responsible for computing a subset of the model parameters. This approach is beneficial when the model is too large to fit in the memory of a single device.

However, DML alone does not address data privacy concerns, especially when data is distributed across multiple organizations. This limitation led to the development of federated learning, a method introduced by Google in 2016 [100]. FL shares similarities with the data parallelism approach in DML, where all participants train the same model or the same part of the model, and a central server aggregates the updates. The key difference is that, in DML, data is typically split from a unified dataset across nodes, whereas in FL, each participant trains the model using its own local data, which remains on the participant's device. Rather than transferring the data, only model updates are shared, ensuring privacy-preserving model training.

Typically, according to [169], there are two main types of learning paradigms in FL based on data partitioning: **horizontal (sample-based)** and **vertical (feature-based)**.

Horizontal FL (HFL): In horizontal FL, participants share the same feature space but have different data samples. For example, several hospitals in different regions may collect health records with the same types of features (e.g., age, gender, blood pressure) but from different patients. The goal is to collaboratively train a model using these similar features across distinct datasets. Mathematically, if D_i represents the dataset held by client i, then:

$$D_1 = \{(x_1^1, y_1^1), (x_2^1, y_2^1), \dots\}, \quad D_2 = \{(x_1^2, y_1^2), (x_2^2, y_2^2), \dots\} \tag{2.1}$$

where x represents features and y represents labels. In horizontal FL, $x_1^1 \approx x_1^2$, but $D_1 \neq D_2$, meaning the feature sets are similar, but the data samples are different.

Vertical FL (VFL): In vertical FL, participants share the same data samples but with different features. For example, a bank and an insurance company may collaborate to build a model where they both have information about the same set of individuals,

but the bank has financial data (e.g., credit score) and the insurance company has health-related data (e.g., medical claims). The data is aligned by the shared set of users. In this case, the feature sets differ across clients, but the sample identifiers are the same. Mathematically, if client 1 holds feature set X_1 and client 2 holds feature set X_2, then:

$$D_1 = \{(x_1^1, x_2^1, \dots)\}, \quad D_2 = \{(x_1^2, x_2^2, \dots)\} \tag{2.2}$$

where D_1 and D_2 represent different features for the same sample x_1.

In both cases, the **Federated Averaging (FedAvg)** algorithm, introduced in [100], is commonly used. Each client i independently trains a local model w_i on their local data D_i, and the global model is updated by aggregating the weights of each client:

$$w_{\text{global}} = \frac{1}{c} \sum_{i=1}^{c} w_i \tag{2.3}$$

Here, c represents the number of clients, and w_{global} is the aggregated global model after averaging the weights of all participants.

Compared to HFL, VFL places a stronger emphasis on privacy-preserving techniques because clients in VFL need to securely combine their features without directly revealing raw data to each other. Methods such as secure multi-party computation (SMPC) or homomorphic encryption are commonly employed in VFL to ensure that sensitive information remains private during model training. In contrast, HFL participants share the same feature space but have different data samples, leading to a focus more on system-level optimization, such as communication efficiency and scalability. This thesis primarily focuses on HFL, where the main challenges lie in optimizing the system to handle distributed data efficiently across various clients while ensuring model performance and scalability.

2.2.1 Federated Learning on edge-cloud computing

FL has been applied to various privacy-critical scenarios such as hospitals [165], banks [170] and autonomous vehicles [116]. One round model training process of horizontal FL within edge-cloud computing environment is shown in Figure 2.1. The process involves the following steps: ① The central server first broadcasts the global model to all participating clients; ② Each client then trains this global model locally using its own data to generate an updated local model; ③ These updated local models

are subsequently uploaded back to the central server; ④ Finally, the central server aggregates the local models to update the global model, completing one round of the training cycle."

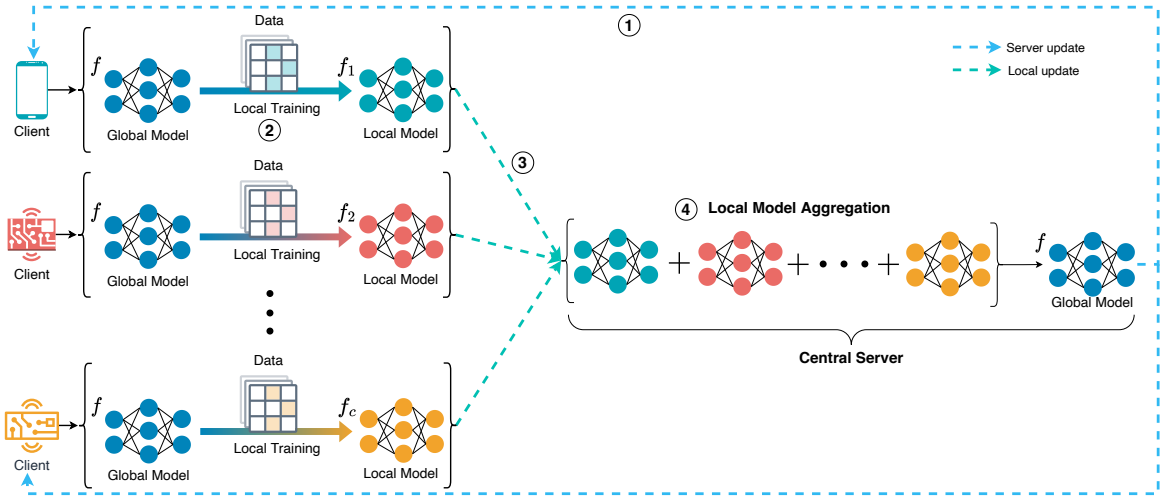

Fig. 2.1 Vanilla Federated Learning Workflow.

According to [69], FL is commonly categorized into two main scenarios: **cross-silo** and **cross-device**. In cross-silo FL, the collaboration typically occurs between a small number of reliable and powerful institutions or organizations, such as hospitals, banks, or research centers. These entities, or "silos," each manage large datasets and robust computational infrastructures. The computation in cross-silo FL is primarily cloud-to-cloud, where the training process happens across different data centers or cloud environments. This setup ensures stable communication, allowing institutions to securely train a global model while addressing heightened privacy concerns due to the sensitive nature of the data. The focus in cross-silo FL is on privacy-preserving techniques and secure multi-party computation, given that data remains siloed within each organization.

In contrast, cross-device FL is more akin to training models directly from decentralized data sources, presenting significantly greater challenges. It operates in a highly decentralized and large-scale environment, typically involving edge computing across millions of devices, such as smartphones or IoT devices. Each device holds a small, local dataset and has limited computational power, storage, and intermittent network connectivity. Computation occurs directly on these edge devices, with model updates periodically sent back to a central server for aggregation. The key challenges in cross-device FL include handling device heterogeneity, learning from streaming data with new instances or categories, optimizing communication under constrained bandwidth,

and ensuring scalability across a vast number of distributed devices. This decentralized structure, coupled with limited resources on edge devices, makes cross-device FL far more complex than the relatively stable and resource-rich cloud-to-cloud setup of cross-silo FL.

One of the most significant challenges in cross-device FL is the selection and adaptation of task model structures to accommodate the diverse hardware of these edge devices. Historically, most FL research has either directly adopted model structures used in centralized machine learning [101] or manually designed structures based on the specific training task [96]. More recently, studies have explored the use of Neural Architecture Search (NAS) [52] to automatically generate optimal model structures. However, these efforts have primarily focused on optimizing model performance at the reference stage—such as achieving higher accuracy and lower inference latency—without fully addressing the need for efficient model training and adaptation across heterogeneous hardware. This gap can hinder the applicability of FL, as the efficiency and performance of the entire system are also impacted by these factors. Without effective cross-platform model adaptation and simultaneous training and inference, FL developers may struggle to achieve the desired outcomes in cross-device FL. Therefore, this thesis focuses on addressing the challenges of cross-device FL within an edge-cloud computing framework. Therefore, this thesis focuses on addressing the challenges of cross-device FL within an edge-cloud computing framework.

2.2.2 Challenges of Federated Learning on edge-cloud computing

Challenge 1: System Heterogeneity. In FL, system heterogeneity arises from the involvement of diverse devices, each with varying computational capabilities (e.g., CPU, GPU), memory, storage, network connectivity (e.g., 5G, Wi-Fi), and power levels (e.g., battery life). These devices range from high-end servers to resource-constrained mobile phones or IoT devices. The challenge lies in designing a FL system that can accommodate this diversity while ensuring efficient training across all devices. A one-size-fits-all approach to model training and deployment is impractical because it may overwhelm less capable devices or underutilize more powerful ones. Therefore, it is crucial to develop adaptive mechanisms that can tailor the training process to the specific capabilities of each device, balancing load and optimizing resource utilization across the network.

Challenge 2: Statistical Heterogeneity. This challenge refers to the non-identically independently distributed (non-IID) nature of data across different clients in a FL system. For example, in a next-word prediction task, mobile phone users may exhibit varying language usage patterns. Unlike traditional centralized machine learning, where data is assumed to be IID, FL often deals with data that is skewed, unbalanced, or distributed differently among clients. This heterogeneity can significantly complicate model training, as the global model may struggle to generalize well across all clients.

Challenge 3: Continuous Learning from Novel Data. Moreover, the challenge of statistical heterogeneity is exacerbated when clients encounter new data types. For instance, a mobile user may shift their interest from taking pictures of cats to dogs, or an autonomous vehicle may enter a new environment. These changes increase overall statistical heterogeneity and introduce two additional research challenges. The first is *Learning from Domain-Incremental Data*, where each client might have data from entirely different domains, requiring the model to adapt to diverse and evolving contexts. The second is *Learning from Class-Incremental Data*, which arises when different clients possess data representing different classes, making it challenging for the model to maintain consistency across the entire dataset. Addressing these challenges necessitates the development of advanced algorithms capable of reconciling differences in data distributions and ensuring robust model performance across all clients.

Challenge 4: Model Heterogeneity. In FL systems, clients may use different model architectures or require varying model configurations due to their specific application needs or resource constraints. This diversity can complicate the process of aggregating local models into a single global model, as traditional aggregation methods assume homogeneity in model structure. When clients use models with different architectures, parameters, or hyperparameters, it becomes challenging to reconcile these differences during the aggregation phase. Ensuring that the global model benefits from the diverse contributions of all clients, while maintaining its overall performance and consistency, requires innovative solutions that can handle such heterogeneity effectively. Techniques like knowledge distillation, model ensembling, or hybrid aggregation methods may be necessary to address this challenge.

Challenge 5: Expensive Communication. Communication overhead is a critical concern in FL, where frequent model updates must be exchanged between clients and the central server. This process can be expensive, especially in environments with limited bandwidth, high latency, or costly data transmission. The primary challenge is to minimize the communication overhead without compromising the quality of the model updates. Techniques such as gradient compression, periodic model

updates, and communication-efficient algorithms have been proposed as potential solutions to decrease the amount of data exchanged. However, balancing the trade-off between communication frequency and model accuracy remains a significant challenge, particularly in large-scale deployments where the cost of communication can quickly become prohibitive.

Challenge 6: Client Participation Variability. This challenge refers to the unpredictable and often intermittent participation of clients in the FL process. Clients may drop out of training due to connectivity issues, battery limitations, or user behavior, resulting in inconsistent contributions to model training. Such variability can destabilize the global model and hinder convergence. When clients drop out and later rejoin the training process, these disruptions can mislead the global optimization direction by altering the overall data distribution, significantly degrading the model's performance. Moreover, non-uniform client participation can exacerbate existing challenges related to data and system heterogeneity, further complicating the learning process. To effectively address this challenge, FL systems need to be resilient to client dropouts and capable of dynamically adapting to varying participation levels. This may involve implementing flexible aggregation strategies and designing models that are robust to partial updates.

2.3 Model Selection

Model selection is a critical process in machine learning (ML) and deep learning (DL) that aims to find the optimal model architecture and hyperparameter configuration for a specific task. As data scales increase and models become more complex, the importance of model selection has become increasingly prominent. It not only affects the performance of the model but also determines its generalization ability and computational efficiency.

In traditional machine learning, model selection often involves methods like cross-validation to evaluate different algorithms and hyperparameter settings [10]. For instance, in support vector machines, choosing the appropriate kernel function and regularization parameter is crucial for the model's performance. However, with the rise of deep learning, the parameter space of neural networks has become extremely large, making manual adjustment of model architectures and hyperparameters both time-consuming and inefficient.

To address this issue, researchers have proposed various automated model selection methods aimed at reducing human intervention by utilizing algorithms to automatically search for the best model configurations.

Hyperparameter Optimization (HPO): HPO seeks to find hyperparameter combinations that minimize a predefined loss function. Traditional methods include grid search and random search [8], but these become inefficient in high-dimensional spaces. Bayesian optimization [134] achieves more efficient searches by constructing a probabilistic model of the relationship between hyperparameters and model performance.
Neural Architecture Search (NAS): NAS automates the design of neural network architectures, addressing the complexity of manual design. Early NAS methods used reinforcement learning [183] and evolutionary algorithms [122] to explore the architecture space, but these approaches were computationally intensive. To improve efficiency, methods like Differentiable Architecture Search (DARTS) [89] introduced gradient-based optimization, significantly reducing search time.
Automated Machine Learning (AutoML): AutoML integrates HPO and NAS to provide end-to-end solutions for model selection [54]. It enables non-experts to build high-performance machine learning models, promoting the widespread adoption and application of ML.

2.3.1 Model Selection for Federated Learning

Federated learning enables multiple clients to collaboratively train a shared global model without exchanging their local data, thereby preserving privacy [100]. However, FL introduces unique challenges in model selection due to the decentralized nature of data and the heterogeneity of participating devices. One significant challenge is **hardware heterogeneity**, where clients possess diverse computational resources, storage capacities, and energy constraints. This variability complicates the deployment and training of a uniform model across all clients and necessitates specialized model selection methods.

In FL, model selection must account for multiple factors influenced by hardware heterogeneity:

- **Computational Constraints**: Resource-limited devices may struggle to train or even store large, complex models [83].
- **Training Efficiency**: Disparities in device capabilities can lead to stragglers that slow down the overall training process [108].
- **Inference Performance**: Models must deliver acceptable inference latency across devices with varying performance profiles, especially for real-time applications [25].

To address these challenges, researchers have proposed several model selection methods that adapt to varying hardware configurations:

Resource-Adaptive Models: Designing models that can adjust their complexity based on the client's hardware capabilities is a practical approach. *Slimmable Neural Networks* [175] allow a single model to operate at different widths, enabling each client to select a model variant that fits its computational and memory constraints. However, integrating slimmable networks into FL requires careful coordination to ensure consistent model updates during aggregation.

Model Compression and Pruning: Techniques like pruning and quantization reduce model size and computational demands. *Federated Pruning* [66] enables clients to collaboratively prune the global model based on local hardware constraints, reducing both computation and communication overhead. Nonetheless, excessive pruning may degrade model performance, and determining the optimal pruning level per client remains challenging.

Personalized Federated Learning: Personalization aims to tailor the global model to better suit individual client characteristics, including hardware capabilities. Methods like *FedAvgM* [60] modify the aggregation process to account for client-specific factors. While personalization can improve local performance, it may introduce additional complexity in model management and does not inherently solve training efficiency issues due to hardware heterogeneity.

Federated Neural Architecture Search (FedNAS): Extending NAS to the federated setting, FedNAS allows clients to jointly search for an optimal model architecture that balances performance and resource usage [52]. By considering hardware constraints during architecture search, FedNAS generates models tailored to the computational capabilities of participating clients. However, NAS processes are computationally intensive and may not be practical for resource-constrained devices without significant optimization.

Asynchronous and Adaptive Training Protocols: Adjusting the FL training protocols can mitigate the impact of hardware heterogeneity. *Asynchronous Federated Learning* [163] allows clients to perform local updates and communicate with the server independently, accommodating devices with slower processing speeds or intermittent connectivity. Additionally, *Adaptive Client Selection* [108] strategically selects clients based on their resources to optimize training efficiency, though this may reduce the representativeness of the aggregated model.

Despite these efforts, existing methods often address only specific aspects of the hardware heterogeneity problem and may not provide a comprehensive solution that

balances training efficiency and model performance based on developer expectations. For instance, resource-adaptive models and model compression techniques focus on making the model feasible for all devices but may sacrifice performance or complicate the aggregation process. Personalized FL improves local performance but can increase system complexity and does not inherently enhance training efficiency.

Moreover, most approaches do not offer a systematic way for developers to specify their preferences or expectations regarding the trade-offs between training efficiency and model performance. This gap makes it challenging to select or design models that meet the specific needs of different FL applications, especially in large-scale, cross-device deployments with significant hardware diversity.

2.4 Continual Learning

Continual Learning (CL), also referred to as Lifelong Learning, is an evolving area in machine learning that focuses on enabling models to learn continuously from a stream of data over time, adapting to new tasks without forgetting previously learned knowledge. This stands in contrast to traditional machine learning paradigms, where models are trained on static datasets and then deployed without further adaptation. CL seeks to build systems that mimic the human ability to accumulate knowledge and skills incrementally, adapting to new experiences while retaining past knowledge.

The central challenge in CL is catastrophic forgetting the tendency of neural networks to lose previously acquired knowledge when trained on new tasks. Addressing this challenge has led to the development of various techniques designed to allow models to learn sequentially without forgetting, thereby enabling them to operate effectively in dynamic and evolving environments.

According to [146], CL can be categorised into three main paradigms based on how tasks are introduced and how the model manages new information:

Task-Incremental Learning (Task-IL): In Task-IL, the model learns a series of distinct tasks, with the identity of each task provided during both training and testing. The model only needs to differentiate between classes within the current task, making it somewhat easier to manage catastrophic forgetting. Techniques like regularization-based methods are commonly used in this setting to protect task-specific knowledge. Mathematically, this can be expressed as:

$$f_i(x) = \arg\max_j p(y_j \mid x, T_i)$$

where $p(y_j \mid x, T_i)$ denotes the probability that input x belongs to class y_j under task T_i.

Domain-Incremental Learning (Domain-IL): Domain-IL involves learning the same task across different domains, where the input distribution varies. The model is not informed about the domain during inference, making generalization across domains crucial. This paradigm is relevant when the same task must be performed under varying conditions, such as different environments. The objective is to learn a function $f(x)$ that generalizes across domains, defined as:

$$f(x) = \arg\max_j p(y_j \mid x)$$

where $p(y_j \mid x)$ represents the probability that input x belongs to class y_j without needing to know the domain D_i.

Class-Incremental Learning (Class-IL): Class-IL is the most challenging paradigm, where the model must learn to classify new classes as they appear while retaining the ability to classify previously learned classes. This requires balancing the acquisition of new knowledge with the retention of existing knowledge. The model learns a function $f(x)$ to classify across all learned classes, represented as:

$$f(x) = \arg\max_j p(y_j \mid x, \theta)$$

where θ encompasses the parameters and knowledge learned from prior tasks, and $p(y_j \mid x, \theta)$ denotes the probability of classifying x as y_j considering the accumulated knowledge.

Key approaches to CL include rehearsal-based methods, knowledge distillation, and regularization-based techniques. Rehearsal methods, like Experience Replay (ER) [125], store and replay subsets of previous data to mitigate forgetting. Knowledge distillation methods, such as Learning without Forgetting (LwF) [84], transfer knowledge from older to newer models to retain past information. Regularization-based methods, like Elastic Weight Consolidation (EWC) [162], add constraints to model updates to protect important parameters, preventing forgetting.

Despite these advances, CL still faces significant limitations. The primary challenge remains the balance between model stability and plasticity. While rehearsal-based methods help retain information by storing and replaying previous examples, they are often constrained by memory limitations and can become computationally expensive, particularly on edge devices and other resource-limited platforms. Regularization methods, though effective in some cases, may not always provide sufficient protection

against catastrophic forgetting, especially when the new tasks are significantly different from previous ones. Additionally, dynamic network-based methods, which expand the network to accommodate new tasks, can lead to increased model complexity and larger network sizes, making them less practical for deployment on devices with limited computational resources.

Another critical limitation is the potential for information dilution, particularly in transformer-based models where attention mechanisms can become overloaded as new tasks are introduced. As these models continuously integrate new information, there is a risk that important knowledge from previous tasks may be overshadowed or lost. This issue is exacerbated in scenarios with large shifts in data distribution, where the attention mechanism might overly focus on new information, neglecting the retention of crucial features from past tasks. Balancing the representation of old and new knowledge within the model's limited capacity, especially on resource-constrained edge devices, remains a significant challenge.

Moreover, most existing CL methods are designed with single-client settings in mind, focusing on minimizing catastrophic forgetting at the individual client level. This approach may not be directly applicable to FL environments, where the primary goal is to ensure that the global model, aggregated from multiple clients, can effectively mitigate catastrophic forgetting across all participating clients. This distinction introduces unique challenges in FL that traditional CL approaches do not fully address. In FL, it is essential to optimize knowledge retention not only within each client but also across the entire federated system, ensuring that the global model remains robust and effective over time, even when deployed on resource-limited edge devices.

2.4.1 Federated Continual Learning

Federated Continual Learning (FCL) emerges as a vital extension of FL, making it more practical for real-world applications by addressing the dynamic and evolving nature of data in decentralized environments. Unlike traditional FL, which assumes a static dataset and client participation, FCL is designed to handle dynamic situations where new data and tasks are continuously generated, clients may join or leave the system at any time, and the distribution of data can change across clients. By enabling models to learn from these evolving data streams while maintaining privacy, FCL allows FL to adapt to real-world challenges such as varying client availability, non-IID data distributions, and continuous learning needs, making it more suitable for practical deployments in fields like healthcare, IoT, and mobile applications.

So far, a limited number of studies have delved into the realm of FCL. These studies have primarily explored FCL through three scenarios: class-incremental [98, 28, 27, 119], domain-incremental [29, 62, 137], and task-incremental learning [173]. Among these, the class-incremental and domain-incremental scenarios pose the most significant challenges and have been the focus of the majority of existing FCL studies, including this thesis.

In the **class-incremental setting**, models must continuously learn new classes while retaining knowledge of previously learned ones, where catastrophic forgetting becomes a central issue. In this scenario, each local client c learns a model $f_{\text{local},c}(x) = \arg\max_j p(y_j \mid x, \theta_c)$, where θ_c represents the model parameters learned by the client. The global model $f_{\text{global}}(x)$, which combines knowledge from multiple clients, is updated by aggregating these local models as follows:

$$f_{\text{global}}(x) = \arg\max_j \frac{1}{C} \sum_{c=1}^{C} p(y_j \mid x, \theta_c) \tag{2.4}$$

Here, C is the total number of clients, and the aggregated global model generalizes the knowledge learned across all participants. This approach aims to balance learning new classes while retaining performance on previously learned classes.

GLFC, proposed in [28], tackle class-incremental learning challenges in federated settings with their Global-Local Forgetting Compensation (GLFC) model. It employs class-aware gradient compensation and class-semantic relation distillation to balance learning new and old classes. A proxy server helps mitigate global forgetting by selecting an optimal global model to support local training, improving overall model retention and performance. Other studies have focused on knowledge distillation (KD)-based approaches to mitigate catastrophic forgetting. For example, [98] introduced a KD strategy using surrogate datasets collected from public data, allowing clients to balance learning new tasks while reviewing older ones.

In **domain-incremental learning**, clients face shifts in data distribution across different domains. The local model on client c learns to generalize over its specific domain D_c, where the task remains the same, but the input distribution varies. The local model can be expressed as:

$$f_{\text{local},c}(x) = \arg\max_j p(y_j \mid x, D_c) \tag{2.5}$$

The global model, which aggregates the domain-specific knowledge from all clients, is updated by combining local models:

$$f_{\text{global}}(x) = \arg\max_{j} \frac{1}{C} \sum_{c=1}^{C} p(y_j \mid x, D_c) \tag{2.6}$$

This allows the global model to adapt to different domains while generalizing across varied data distributions. To address the challenges posed by Non-IID data in this setting, [62] proposed an inter-domain and intra-domain knowledge distillation method that combines insights from both updated and pre-trained models to balance knowledge integration and retention.

Moreover, recent studies have explored dynamic scenarios in real-world FL applications, where servers cannot predict the nature or timing of new client data. For instance, [28] introduced an FCL setting where new clients may join at any time, bringing in new data categories. In this scenario, some clients transfer tasks, while others focus on previous tasks. To detect task transitions, the authors developed an entropy-based method, complemented by a class-aware gradient compensation loss and class-semantic relation distillation loss to balance the retention of old classes while learning new ones. Furthermore, they proposed using a proxy server to select the most suitable old global model, mitigating global forgetting caused by class imbalances across clients.

However, FCL still faces significant challenges, particularly in managing catastrophic forgetting and optimizing data efficiency across distributed environments. Traditional rehearsal-based approaches, which rely on storing and replaying raw data from previous tasks, are not only impractical in federated settings due to privacy constraints and the limited resources of edge devices but also suboptimal in terms of data efficiency. These methods often involve simple sampling of raw data, which may not capture the most informative aspects of the data needed for effective learning. This results in an inefficient use of the limited memory available on resource-constrained devices, leading to poor retention of past knowledge and a higher risk of forgetting. Additionally, the issue of meta-information heterogeneity, where data from non-IID distributions across clients introduces inconsistencies, further complicates the model's ability to maintain and integrate knowledge across tasks, ultimately weakening overall model performance.

Beyond the limitations of rehearsal-based methods, FCL also struggles with domain-incremental learning, where models must adapt to new domains introduced sequentially while retaining performance across previously encountered domains. Existing approaches often depend on either storing large amounts of rehearsal data or expanding the model architecture, both of which are impractical in federated environments. These methods typically fail to enhance the model's ability to generalize across different

domains effectively, as they do not fully leverage the diverse domain-specific information available across clients. The lack of strategies to increase the richness of stored data, beyond simple raw data sampling, means that models are less robust in dealing with the variability and complexity of domain-incremental scenarios. Furthermore, the reliance on raw data or network expansions does not address the need for maintaining privacy and reducing computational overhead, which are critical in resource-constrained, privacy-sensitive FL environments.

This thesis builds upon the existing body of work by focusing on both federated class-incremental and domain-incremental learning scenarios, addressing these unique challenges associated with continual learning in decentralized and distributed environments.

2.5 Thesis scope

This thesis focuses on addressing critical challenges in FL systems, particularly in environments with heterogeneous devices and continuous data streams, as outlined in the scenario structure shown in Figure 2.3. The research aims to enhance scalability, adaptability, and efficiency in FL systems, as depicted in the taxonomy (Figure 2.2), with all the work situated within the **Cross-device (Edge-Cloud Computing)** architecture and **Horizontal Federated Learning** usecase.

Each chapter in this thesis contributes to solving specific challenges, summarized as follows:

1. **Adaptive Model Selection and System Heterogeneity (FedMSA):** One of the central challenges in FL is the heterogeneity of devices participating in the learning process, which vary greatly in computational power, memory, and network capabilities. FedMSA (Model Selection and Adaptation) is introduced to address **Challenge 1: System Heterogeneity**. By employing NAS, FedMSA allows the system to dynamically select model architectures that are best suited to the hardware constraints of each participating device. This ensures that models are not only computationally efficient but also adaptable to the capabilities of a wide range of devices, from powerful servers to resource-constrained mobile devices. This contribution lays the foundation for scalable FL systems that can function effectively across diverse hardware configurations.

2. **Real-Time Diagnostics and System Adaptation (FedDAS):** Extending the capabilities of FedMSA, this thesis introduces FedDAS, a real-time diagnostic

and adaptation system designed to enhance large-scale FL deployment. While FedMSA focuses on model selection and hardware adaptation, FedDAS addresses the challenges of dynamic environments where system instability, device failures, and changing network conditions can degrade performance. By continuously monitoring system-level metrics such as device memory usage, network bandwidth, and training latency, FedDAS ensures that anomalies are detected early and addressed through human-in-the-loop interventions. This framework enhances the robustness and scalability of FL systems, particularly in highly heterogeneous environments, ensuring that FL can continue operating smoothly in practical deployments.

3. **Enhancing Model Knowledge Retention in Class-Incremental Learning (DVKA):** In FL, the ability to continuously learn from new data while preserving previously acquired knowledge is critical for every client during local model updating, especially in class-incremental scenarios where new classes of data are introduced incrementally. This thesis addresses **Challenge 3: Continuous Learning from Novel Data** by proposing the Dual Variational Knowledge Attention (DVKA) mechanism. DVKA strengthens the transformers' capacity to manage Class-Incremental Learning by mitigating catastrophic forgetting, a common issue where new data overwrites prior knowledge. Grounded in principles inspired by the information bottleneck theory, DVKA selectively filters relevant information through feature-level and token-level attention mechanisms, ensuring that only the most pertinent knowledge from new tasks is integrated while preserving critical information from previous tasks. This approach optimizes the trade-off between retaining old knowledge and learning new information, ensuring that the model remains robust and adaptable in dynamic FL environments.

4. **Optimizing Memory Efficiency and Addressing Meta-Information Heterogeneity (ECoral):** Another significant challenge in FL is managing the trade-off between data efficiency and knowledge retention, particularly when handling class-incremental data. The ECoral framework, introduced in this thesis, focuses on **Challenge 2: Statistical Heterogeneity** and **Challenge 3: Continuous Learning from Novel Data**, specifically optimizing memory efficiency through exemplar-condensed learning strategies. In FL, where privacy constraints limit the ability to share raw data between clients, ECoral improves the density of stored exemplars, allowing the system to retain essential information from past tasks while using minimal memory. This contribution is particularly valu-

able in resource-constrained environments, where maintaining large datasets is impractical. ECoral's ability to manage meta-information heterogeneity ensures that the system can adapt to new classes of data while preserving the integrity of previously learned information.

5. **Handling Domain Shifts and Reducing Rehearsal Memory Costs (RefFil):** Domain shifts, where data distributions change over time or vary across clients, pose a significant challenge to maintaining robust model performance in FL. RefFil (Federated Domain-Incremental Learning) is designed to address this issue, focusing on **Challenge 3: Continuous Learning from Novel Data** in the context of domain-incremental learning. In FL, where clients often work with data from different domains, the ability to generalize across these domains is critical. RefFil ensures that local models can adapt to evolving data distributions by leveraging global domain information, allowing the FL system to maintain high performance even as new and diverse data is introduced. This makes RefFil an essential component for FL applications in real-world environments where data is dynamic and non-stationary.

In summary, this thesis addresses key challenges in FL related to system heterogeneity, real-time adaptation, continual learning, data efficiency, and domain shifts. The proposed solutions, including FedMSA, FedDAS, DVKA, ECoral, and RefFil, collectively form a comprehensive framework for building scalable, adaptable, and efficient FL systems that can be deployed in dynamic, resource-constrained environments.

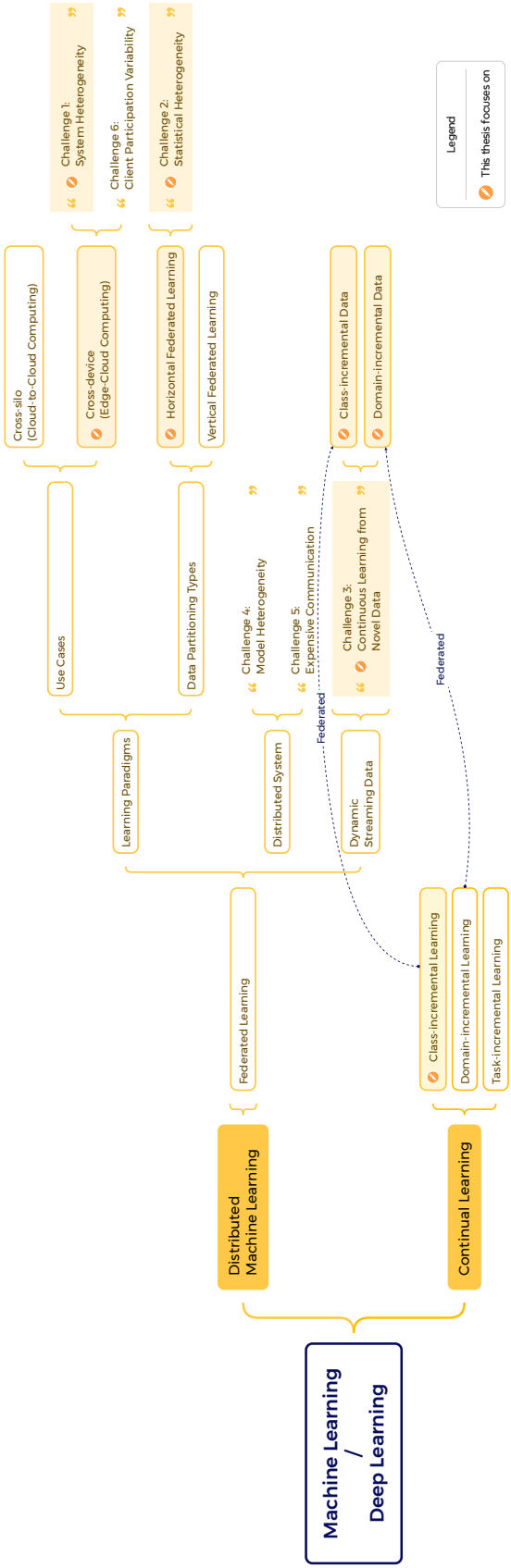

Fig. 2.2 ML taxonomy and focus of research presented in this thesis.

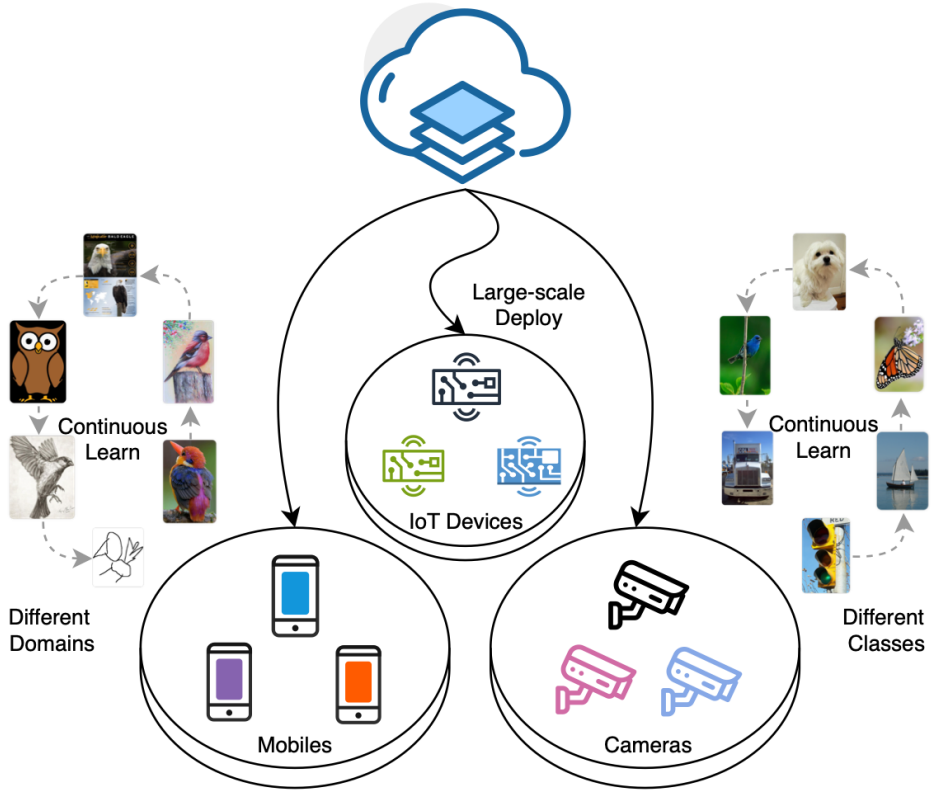

Fig. 2.3 Overview of the targeted scenario structure in a federated learning application within an edge-cloud computing environment, involving a central cloud server and numerous edge devices, which serve as clients, with varying system configurations, such as mobile phones, IoT devices, and cameras. The process begins with the large-scale deployment of a model that effectively operates in this heterogeneous hardware environment. Once deployed, the model and all clients continuously learn from new data, adapting to novel domains and classes introduced sequentially.

Chapter 3

Federated Learning with Heterogeneous Hardware

As we delve deeper into the complexities of federated learning (FL) systems, the challenge of accommodating hardware heterogeneity becomes increasingly apparent. Following the foundational principles in the previous chapters, where we discussed the general architecture and challenges of FL, this chapter focuses on the specific issue of hardware heterogeneity in FL environments. By introducing adaptive model selection techniques, we aim to ensure that FL can be effectively scaled across a wide range of devices with varying computational capabilities. This sets the stage for developing robust FL systems that can operate efficiently, regardless of the hardware disparities among participating clients.

FL enables multiple clients to collaboratively train a shared model without sharing personal data. However, in large-scale FL applications with heterogeneous devices, selecting and quickly adapting a model to meet user expectations is challenging. In this chapter, we propose a model selection and adaptation system for FL (FedMSA), which includes a hardware-aware model selection algorithm that balances model training efficiency and performance based on FL developers' expectations. FedMSA supports full automation in the building and deployment of FL tasks across different hardware at scale, ensuring that the expected model performance is achieved through dynamic adaptation. Experiments on benchmark and real-world datasets demonstrate the effectiveness of FedMSA's model selection algorithm on real devices such as Raspberry Pi and Jetson Nano.

Extending FedMSA, we introduce **FedDAS**: a **D**iagnosis and **A**daptation **S**ystem for *Fed*erated Learning. FedDAS primarily provides real-time diagnostic feedback, enabling users to proactively adjust the FL system. By monitoring system-level metrics

and diagnosing issues such as improper hyper-parameter settings or communication delays, FedDAS alerts users to potential anomalies that may affect performance. This human-in-the-loop system adaptation allows users to make informed adjustments on the fly, ensuring smooth operation and optimal performance of FL applications despite the challenges posed by device heterogeneity and system complexities in large-scale deployments.

3.1 FedMSA: A Model Selection and Adaptation System for Federated Learning

To improve the applicability of practical FL, we consider a realistic setting in which FL system contains heterogeneous hardware where some clients contribute to the central model and use the local model to make predictions simultaneously. Our goal is to select an expected model structure that trades-off the training efficiency (e.g., training time, network utility) and model performance (e.g., accuracy, inference latency), while considering the developer's choices. For example, some developers are more sensitive to model training efficiency, and some want to get a model with optimal performance. Because NAS requires high computational demand, some methods need thousands of GPU days to search for the best architecture for some tasks, which can be an unacceptable and catastrophic situation for developers [124]. So our work is based on the pre-searched models' metrics from NAS-Bench-201 [80] and HW-NAS-Bench [30]. Nevertheless, although our model selection algorithm can accurately select the optimal model structure under the given parameter, we recognized that it is difficult to accurately select the optimal model structure that meets the developer's expectations through our model algorithm by one-time deployment as FL developers may have more personalized preferences. Therefore, we need a system that supports rapid task deployment at scale, iterating over the results to get the optimal result that developers expect.

To this end, we introduce *FedMSA, a **Model** **Selection** and **Adaptation** system for **Fed**erated Learning* , which can address the previously listed challenges. It reduces the complexity of FL system deployment by providing automation of adaptation and deployment of training tasks as microservices in a FL life-cycle along with *model selection* (MS) algorithm. Our MS algorithm could help FL developers search for an optimal model structure based on their expectations in training efficiency and final model performance. On comparing the results of our MS algorithm with manual design and direct selection of a model by one factor from NAS, the model selected by our

algorithm achieves good performance in training efficiency and model performance. Similar ideas have been proposed in network speed optimization; for example, the MARS method proposed in [135] maintains balance for comprehensive utilization, transmission efficiency, and monetary cost to optimize high-speed mobile networks. But, to the best of our knowledge, this work is the first attempt to tackle the challenges of training efficiency and model performance by identifying the optimal model structure that meets developer expectations.

Scenario

Consider a system of a real-world large-scale FL application shown in Figure 3.1. Edge-IoT client devices with various hardware architectures simultaneously perform local prediction tasks which contribute to a central ML model training in a cross-device FL framework. In a networked, distributed, and cross-device machine learning system, the primary challenge is model design or model selection, as many factors, such as local and global training efficiency, network pressure, and model performance, need to be considered in heterogeneous hardware architectures. As regards efficiency-sensitive model training tasks, such as recommendation systems, the cold-start phase requires the model training to achieve high accuracy within a certain time constraint when a new user or item joins the network. Moreover, some performance-sensitive model training tasks, such as CCTV anomaly detection systems, real-time temperature monitoring, and alert systems, require the model training to perform high accuracy along with low inference latency. These factors raise challenges in manual model design or model selection. Although there exists research that focuses on model selection method [143, 102], they fail to consider federated and distributed machine learning framework. Model structure selection automation is crucial in balancing model training efficiency and performance.

However, the generated model may not meet the FL developer's expectations. Therefore, a better way is to allow users to define the satisfying proportion of model performance and training efficiency. Additionally, several existing NAS studies [178, 142, 161, 80] provide model structure search space with pre-measured metrics and inference latency in various platforms (e.g., Raspberry Pi, Edge GPU device, and TPU). Nonetheless, no existing studies focus on searching for a ML model based on the developer's preference to balance training efficiency and model performance in FL. System that provides model selection and multi-platform adaptation service, and simplifies the deployment of FL tasks in real devices is a fundamental requirement.

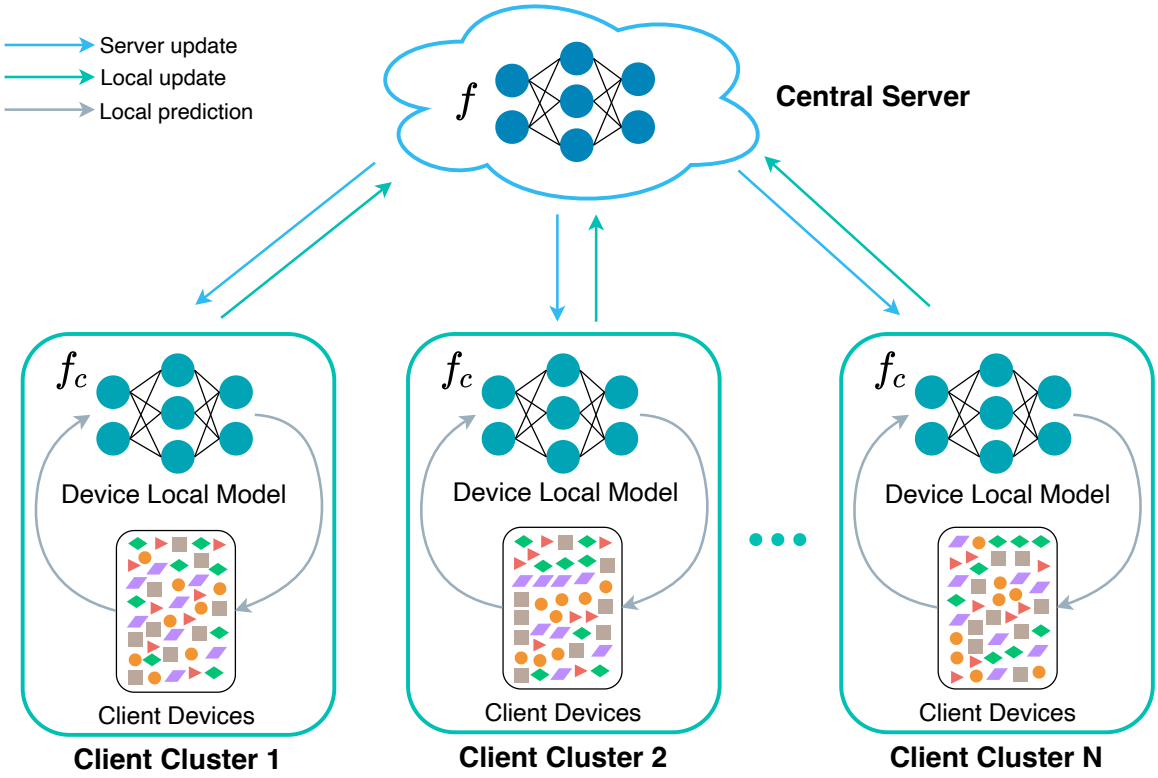

Fig. 3.1 Illustration of the scenario. The client clusters represent companies where each cluster contains many heterogeneous devices as FL clients. The local model of a device participates in FL training and task inference simultaneously.

3.1.1 Related Work

In centralized deep learning, developers typically design model structures manually for specific tasks, such as ResNet [53] for image classification and Transformer [147] and BERT [23] for machine translation. This manual design process is time-consuming and labor-intensive, often requiring numerous iterations to find optimal neural network configurations [183]. To streamline this, recent studies have focused on using neural architecture search (NAS) to automate model design, leading to architectures that outperform manually designed models in tasks like object detection [42, 112] and semantic segmentation [40, 87, 105]. While early NAS efforts prioritized accuracy [88], recent work has aimed to balance performance and efficiency [105, 41]. As centralized ML shifts toward on-device learning due to privacy concerns and network constraints, some NAS methods now consider target hardware [178, 142, 161, 80], including micro-controllers [41, 86]. However, on-device learning can suffer from reduced data diversity, potentially limiting model performance.

In FL, selecting an optimal model structure is challenging due to the need to balance training efficiency, system utility, and performance across heterogeneous hardware. Existing NAS methods typically focus on maximizing performance [105, 41, 40, 87, 42, 112, 86] or providing pre-measured metrics for various models [80, 161, 30], but they do not address the unique demands of FL systems. While some model selection methods have been proposed, such as adaptive inference models [143] and transfer learning model selection based on mean silhouette coefficient scores [102], these approaches do not fully consider training efficiency, performance, and hardware heterogeneity in FL environments. Furthermore, in real-world FL systems, rapid model adaptation and selection are crucial. FL developers often need to iterate on model selection, making a scalable orchestration system for rapid adaptation necessary. Although platforms like Flower [9], FATE [92], and FedML [51] support FL deployment and development, they lack consideration for heterogeneous hardware and model selection in distributed systems. To date, runtime model selection and adaptation systems have not been fully developed for FL applications.

3.1.2 System design

In response to the challenges of model selection for heterogeneous hardware, we developed *FedMSA*, a cross-platform model selection and adaptation system for FL applications. The architecture of *FedMSA* is presented in Figure 3.2, which consists of three main components: *Orchestration Infrastructure*, *Federated Learning* and

Visualization. *FedMSA* meets aforementioned requirements, providing flexibility and reliability for FL developers in personalized FL tasks deployment with automatic model selection and adaptation for a specific client's hardware platform.

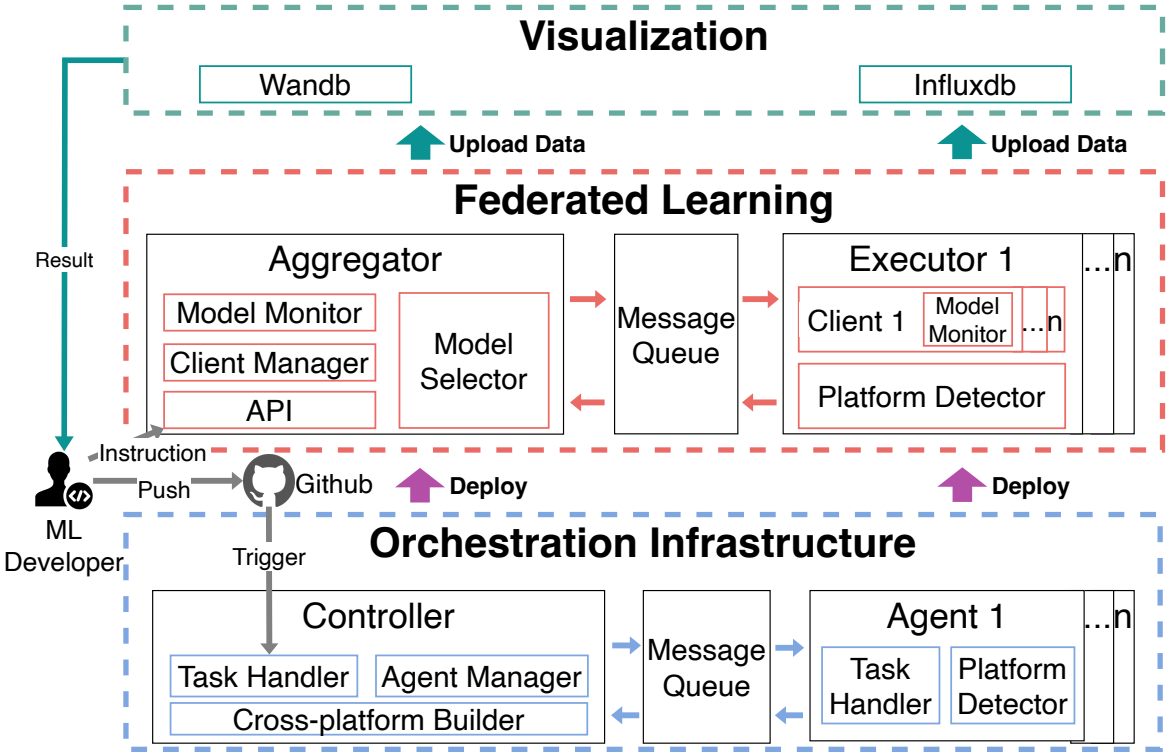

Fig. 3.2 FedMSA System Architecture.

Orchestration Infrastructure. Automatic, rapid, and agile deployment of FL tasks is the key to achieving effective model adaptation for a distributed system across different devices with heterogeneous hardware architecture, especially for a large-scale networked ML system. Therefore, we designed an orchestrator as the infrastructure of *FedMSA*, which supports all aforementioned functionalities from source code auto-building to auto-deployment. It consists of two main components: *controller* and *agents* running at the cloud tier and edge tier of the edge-cloud computing environment, respectively. Meanwhile, all messages transferring between components are handled by a message queue service, and all agents are organized by the agent manager. The training task starts after a machine learning developer pushes the latest code to Github and triggers the task handler to process the task code by calling a cross-platform builder to build FL task code into the microservices based on the hardware architecture information (e.g., x86_64, aarch64 and armv7l) of every agent host which is reported by platform detector.

Federated Learning. An FL system (e,g., FedAvg [101]) usually consists of thousands of participating heterogeneous devices. The scalability requirement of FL deployment necessitates a model selector to generate model structure automatically based on the developer's expectation. Accordingly, we design a Neural Architecture Search (NAS) based model selection algorithm that traded-off model training efficiency and model performance on the heterogeneous hardware, which we discuss in detail in Section 3.1.2. Further, aggregators and executors are deployed by orchestration infrastructure while the message queue handles communication between them, where each executor deploys one/multiple clients based on the number of computation units (e.g., CPU, GPU) in the target device. A global client manager manages all clients in the aggregator, including client registration, join or quit training requests, and client selection algorithm management.

Cost Mitigation for Deployment and Training. Although deploying models and initiating training for each new query might appear computationally expensive, our system effectively mitigates these costs through several strategies. First, we leverage pre-computed model metrics from NAS-Bench-201 and HW-NAS-Bench, caching these values in a model metrics file (see Algorithm 1) to enable rapid model scoring and selection without the need for full retraining. Second, our automated, containerized deployment process—facilitated by the cross-platform builder and orchestrator—ensures that redeployment occurs within seconds, significantly reducing overhead. Finally, in a large-scale federated learning environment, the marginal cost of redeployment is distributed across numerous devices, making the overall process cost-efficient.

Furthermore, we provide two ways for fast model adaptation: 1. developers can send an instruction to switch the current training model by the public API whereby the aggregator and all clients will stop current training tasks and load a new model to execute by received commands. 2. developers can modify a parameter in the configuration file to calculate new scores for all model candidates and once the latest codes are pushed, the orchestration infrastructure will redeploy all components automatically.

Visualization. All model training information and model performance is uploaded to visualization tools in real-time by the model monitor of FL of FedMSA. Our system supports the online monitoring tool Wandb (https://wandb.ai/site accessed on 1 July 2022) and local visualization tool Influxdb (https://www.influxdata.com/ accessed on 1 July 2022).

Model Selection Algorithm

As mentioned in Section 3.1, all model structures x that our algorithm uses are from the search space of NAS-201-Bench X (the i^{th} model structure is $x^{(i)} \in X$), essential metrics of every model are from NAS-201-Bench, and metrics about model inference latency in different hardware are from HW-NAS-Bench. We propose a method in Algorithm 1 to collect and organize the metrics from both NAS benchmark studies. The platform detector component in the FL collects platform information P and bandwidth B from all clients C Figure 3.2. In particular, our system allows FL developers to pre-filter out the model with unexpected accuracy according to a given set model accuracy expectation parameter $\alpha \in [0, 100]$ in the configuration file while abnormal model structures are filtered out by the system. Besides, to ensure that the impact of each metric on the final score calculation is similar, we adopt the data Min-max normalization [49] that reorganizes the distribution space of all metrics belonging to the values between 0 and 1, and is shown as:

$$f(metrics) = \frac{x_{i,j} - x_j^{min}}{x_j^{max} - x_j^{min}} \tag{3.1}$$

In the Equation (3.1), j is the index of attribute of models (e.g,. size of model, estimated accuracy of model), i is the index of a model, $x_{i,j}$ represents the exact value of attribute j of the model i, and x_j^{min} and x_j^{max} is the minimum and maximum value of attribute j respectively.

Algorithm 1: Pseudocode for model metrics collection

Input : Client information c, Model accuracy threshold α
Output : A set of normalized model metrics M_n

function *MODEL_METRICS_COLLECTION(c)*
  $P, B \leftarrow$ Platform Detector ; `// Collect platform information` P `and Device Network Bandwidth` B `from all clients`
  for *i in range(N)* **do**
    m *append* $HW_{lat} \leftarrow$HW-NAS-Bench(i) ; `// A row of metrics` m
    m *append* $T_{acc}, Acc_{eval}, Loss_{eval}, F_{model}, N_{params} \leftarrow$ NAS-Bench-201(i)
    m *append* E_{train} which is calculated by Equation (3.4)
    m *append* U which is calculated by Equation (3.5)
    M_c *append* data_clean(m,α) ; `// Filter out the abnormal model structure and the model accuracy less than` α
  $M_n \leftarrow$ *normalization*(M_c) base on Equation (3.1)
  Save M_n as a model metrics file
  return M_n

In the system design view, we consider the running memory requirements of NAS-201-Bench to be around 25G. Accordingly, in the Algorithm 1, we use one-time data collecting or pre-collecting and loading from NAS-201-Bench in the future method to save M_n as a model metrics file. This approach not only dramatically reduces the spatial complexity of our system, but also enables our system to support more devices.

Our model score calculation algorithm mainly follows the Equation (3.2), and working details shown in Algorithm 2. It aims to trade off model training efficiency and model performance based on a FL developer's input parameter $\gamma \in [0, 1]$, which indicates the extent to which the developer is concerned about the efficiency of model training.

$$Score = f(ModelTrainingEfficiency) \times \gamma + f(ModelPerformance) \times (1 - \gamma) \quad (3.2)$$

Algorithm 2: Pseudocode for score calculating

Input : Normalized model metrics set M_n
Output : A set of model metrics with scores M_s

```
function MODEL_SCORE_CALCULATE(M_n)
    if Not exist model metrics file then
        M_n ← Algorithm 1
    else
        M_n ← load model metrics file
    for i in range(NAS-201-Bench search space) do
        m ← M_n[i]
        Score ← Calculate m by Equation (3.2)
        M_s append Score
    return M_s
```

Based on the model metrics M_n from Algorithm 1, we found that five factors can form Equation (3.3) to calculate model training efficiency: training time to the model accuracy T_{train}, FLOPs of the model F_{model}, numbers of parameters of the model N_{params}, training efficiency in accuracy increasing E_{train}, and bandwidth utility of a device U. Since some metrics are negatively correlated with model training efficiency, we set their value as negative, and upon averaging, we get:

$$f(ModelTrainingEfficiency) = \overline{-(T_{train} + F_{model} + N_{params} + E_{train}) - U} \tag{3.3}$$

where E_{train} is calculated by T_{train} and expected final evaluation accuracy of the model Acc_{eval} which is:

$$E_{train} = \frac{T_{train}}{Acc_{eval}} \tag{3.4}$$

and U calculated by the average of all bandwidth of client devices B_c, and the S_{bit} which means the storage bit of the development platform. In this work, we use Pytorch (https://pytorch.org/ accessed on 1 July 2022) as our development platform, so this parameter is equal to 32 bit.

$$U = \frac{\overline{\sum_{c=1}^{C} B_c}}{N_{params} * S_{bit}} \tag{3.5}$$

We can then summarize the equation for the model training efficiency calculation as:

$$f(ModelTrainingEfficiency) = \overline{-(T_{train} + F_{model} + N_{params}) + \frac{T_{train}}{acc_{eval}} + \frac{\overline{\sum_{c=1}^{C} B_c}}{N_{params} * S_{bit}}} \tag{3.6}$$

In order to let every metric have the same impact for score calculating, we set the calculation of E_{train} and U before the data normalization process as shown in Algorithm 1.

As for the model performance calculating, we mainly focus on the impact of model evaluation accuracy and loss and the model inference latency in every FL client device, which is negatively correlated with model performance. It can be described as:

$$f(ModelPerformance) = \overline{acc_{eval} + loss_{eval} - \overline{\sum_{n=1}^{N} Lat_n}} \tag{3.7}$$

Overall, the model score calculation equation can be summarized as Equation (3.8). We calculate scores for all model structures and wrap them up into a set M_s for further model selection decision.

$$\begin{aligned} g(Score) = & \underbrace{\overline{-(T_{train} + F_{model} + N_{params}) + \frac{T_{train}}{acc_{eval}} + \frac{\overline{\sum_{c=1}^{C} B_c}}{N_{params} * S_{bit}}} \times \gamma}_{\text{ModelTrainingEfficiency}} \\ & + \underbrace{\overline{acc_{eval} + loss_{eval} - \overline{\sum_{n=1}^{N} Lat_n}} \times (1 - \gamma)}_{\text{ModelPerformance}} \end{aligned} \tag{3.8}$$

Since all scoring calculations rely only on pre-measured and stored model performance metrics (e.g., accuracy, latency, training time, and resource usage), this scoring process has a computational complexity of $\mathcal{O}(1)$ for each model candidate. The calculation only involves simple arithmetic operations such as addition, subtraction, and division on precomputed values, making it highly efficient even in large-scale federated learning deployments. Additionally, by filtering models that do not meet developer-defined constraints (e.g., accuracy threshold α), we further reduce the number of evaluated candidates, ensuring that the selection process remains scalable and computationally lightweight.

After defining the necessary functions, we start primary FL training as presented in Algorithm 3. The task starts from procedure D, and the system directly starts training if the FL developer has already assigned a specific model index idx in the configuration file. If not, the system will select the model index with the highest score from the ranked model metrics list M_r.

During model training, the public API allows the developer to decide whether the current training needs to be terminated and the model adapted from the results of the monitoring data visualization tool shown in Figure 3.2 by sending instructions at any time. In this way, the cost of task redeployment can be significantly reduced, and the developer can re-score all model structures by modifying the γ in the configuration file.

3.1.3 Experiments

All the system components, including the MS algorithm, are deployed on real-world testbeds as shown in Figure 3.3. In *FedDAS*, the cloud server is deployed on an Ubuntu server with 20 core (Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz), Gigabit Ethernet and 64 GB memory. The edge devices consist of five Jetson-Nanos (https://developer.nvidia.com/embedded/jetson-nano-developer-kit accessed on 1 July 2022) with 4GB RAM, 128-core Maxwell GPU, Gigabit Ethernet and ARM Cortex-A57 CPU each, and two Raspberry Pi 4 models with 4GB RAM, Gigabit Ethernet and 1.5 GHz 64-bit quad-core ARM Cortex-A72 CPU. The networking devices consist of a switcher with Gigabit Ethernet and a router with 10 Gigabit Ethernet.

For the proof-of-concept, all the experiments have been conducted on a widely adopted benchmark Cifar10 [75] in vanilla FL FedAvg [101] and the experimental models are trained for 140 communication rounds with 1 local epoch in total.

The solution is compared with three well-known neural network models in Table 3.1: (i) a manually designed, widely used, and influential model ResNet [53]. In order to let the size of the model close to the model searched from our system, we used ResNet20

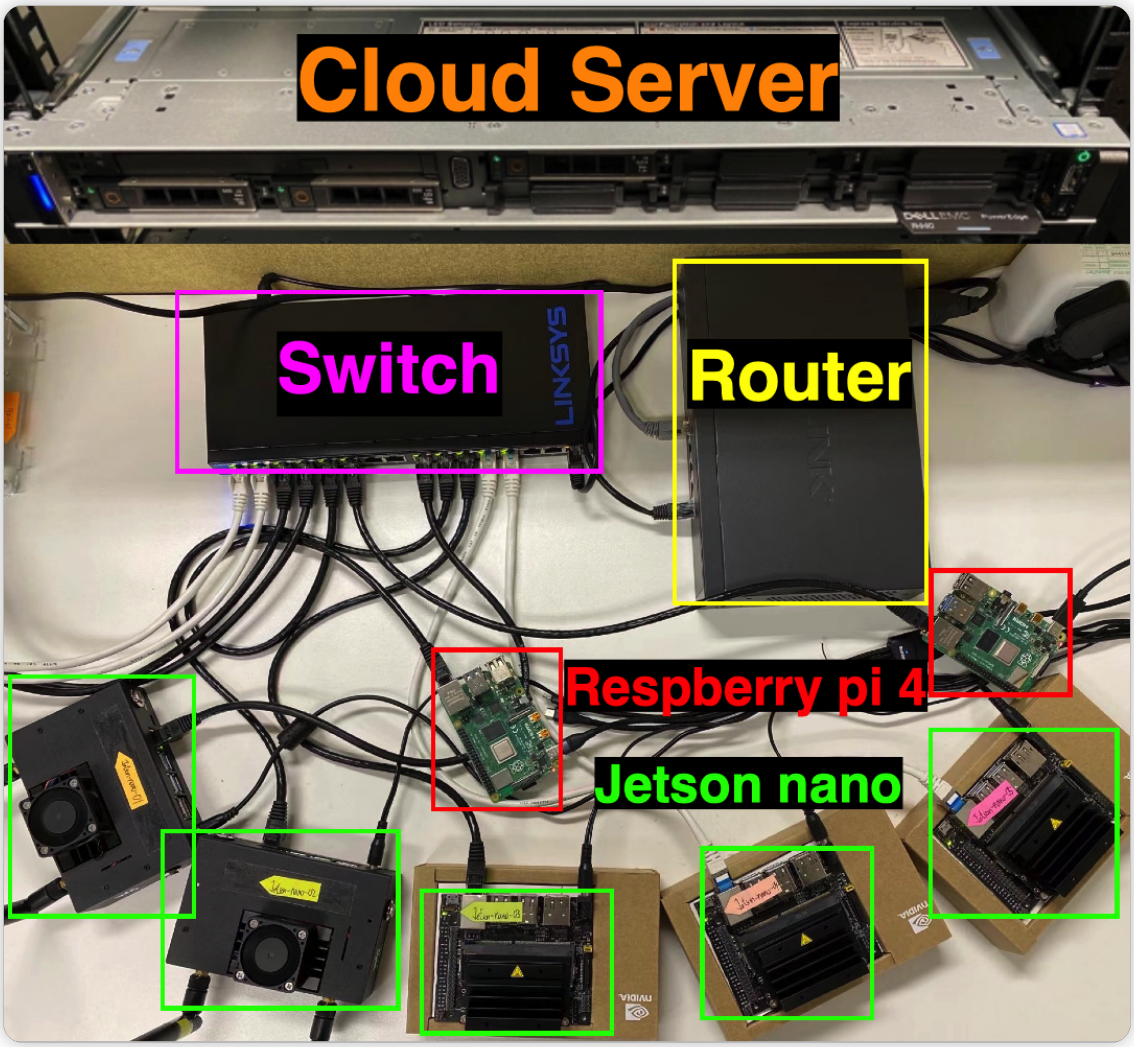

Fig. 3.3 Experiment Devices.

(20 layers ResNet); (ii) two models searched from NAS-201-Bench sorted by a single indicator. They were searched by maximizing evaluation accuracy and minimizing total training time to accuracy. Three models searched by our FedMSA system were defined by $\gamma = 0.1, 0.2, 0.8$.

In addition, FedMSA filters out models with less than 70% accuracy by setting $\alpha = 70$ in the configuration file. On the other hand, all models' latency measurement experiments are in batches to easily distinguish the gaps in the different experiments' results. The forward and backward propagation latency is measured based on the batch size of the training set (64), and inference latency is measured by the batch size of the test set (32)

The model selection algorithm is evaluated through a related study that is also trying to get the best model without training. Similar to FedMSA, NASWOT was proposed in [103], which is also based on the NAS-201-Bench search space. However, they did not consider training efficiency in the cross-device FL setting.

Algorithm 3: Algorithm of FedMSA system

Server input : training efficiency concern rate γ, global step-size α_g, global communication round R, and local updating iterations I, NAS-Bench-201 search space N, FL developer assigned model index *idx*

We start with **Procedure D**.
Then initializing clients with **Procedure A**.
For $r = 1 \rightarrow R$ rounds, we run **Procedure B** and **Procedure C** iteratively.
procedure A *CLIENT_INIT(c, index)*
    $f_c \leftarrow$ NAS-201-Bench(index) ; `// load model from NAS-201-Bench base on index`
procedure B *CLIENT_UPDATE(c)*
    $f_c \leftarrow f$; `// Receive updated model from SERVER_EXECUTE`
    for $l = 1 \rightarrow I$ **do**
        $f_c \leftarrow f_c - sgd$; `// Local model update`
    send $f_c - f$ to SERVER_EXECUTE
procedure C *SERVER_EXECUTE(index)*
    for $i \leftarrow 0$ **to** R **do**
        $f \leftarrow$ NAS-201-Bench(index) ; `// Load global model`
        Broadcast f to all clients ;
        Receive local models' updates $\{f_c - f\}_{c=1}^{C}$ from CLIENT_UPDATE
        $f \leftarrow f\text{-}\Sigma_{c=1}^{C}(f_c - f)$; `// FL aggregation broadcast` f `to CLIENT_UPDATE`
procedure D *main()*
    if *Developer pre-assigned idx* **then**
        Run **Procedure C** with input *idx*
    else
        $M_s \leftarrow$ Algorithm 2
        $M_r \leftarrow sort(M_s)$
        $i \leftarrow 0$
        $index \leftarrow M_r.index[i]$; `// Get the model index from highest score`
        Run **Procedure C** with input *index*
        while *end FL training* **do**
            if *request* $\leftarrow$ *User's instruction ;* `// System get user's instruction from API` **then**
                if *Users' assigned model index in request* **then**
                    $index \leftarrow$ *request*
                else
                    $i+ = 1$ $index \leftarrow M_r.index[i]$; `// Switch to next model in` M_r
                Broadcast message to the aggregator and all clients to stop current training, starting a new training with model *index*.

Table 3.1 Result Summary Total Communication Round 140.

Model	Numbers of Parameters (M)	FLOPs (M)	Training Time (m)/Round	Forward Propagation Latency (ms)	Back Propagation Latency (ms)	Model Inference Latency (ms)	Global Test Accuracy	Global Test Loss
NAS(Max Acc)	0.86 (×2.5)	15.61 (×2.6)	22.95 (×2.1)	2579.47 [1] (×2.1)135.42 [2] (×1.9)	4880.23 [1] (×2.4)160.49 [2] (×2.2)	2371.46 [1] (×2)105.91 [2] (×1.7)	**92.05%**	**0.247**
NASWOT	0.46 (×1.35)	8.3 (×1.37)	20.78 (×1.92)	2681.47 [1] (×2.16)148.65 [2] (×2)	3577.14 [1] (×1.78) 163.89 [2] (×2.30)	2438.59 [1] (×2.07)105.03 [2] (×1.68)	85.63% (−6.42%)	0.435 (+0.188)
FedMSA ($\gamma = 0.1$)	**0.34**	**6.06**	**10.8**	**1241.79 [1]72.31 [2]**	**2004.17 [1]71.35 [2]**	**1175.28 [1]62.66 [2]**	90.72% (−1.3%)	0.278 (−)
RestNet20	0.27 (×2.7)	5.16 (×3.3)	7.16 (×1.2)	**722.45 [1]29.55 [2]**	1560.13 [1] (×2) **41.21 [2]**	**675.93 [1]22.48 [2]**	**89.77%**	**0.308**
FedMSA ($\gamma = 0.2$)	**0.1**	**1.56**	**5.73**	754.07 [1] (−)59.35 [2] (×2)	**779.78 [1]**47.86 [2] (×1.2)	695.81 [1] (−)48.84 [2] (×2.2)	86.21% (−3.5%)	0.424 (+0.116)
NAS (Min T_{train})	**0.07**	**1**	**3.29**	410.5 [1] (−)35.53 [2] (−)	395.29 [1] (−)**21.78** [2]	393.10 [1] (−)31.20 [2] (−)	74.9% (−4.2%)	0.743 (+0.11)
FedMSA ($\gamma = 0.8$)	**0.07**	**1**	**3.29**	**409.03 [1]35.04 [2]**	**388.13** [1]22.50 [2] (−)	**389.31 [1] 30.82 [2]**	**79.10%**	**0.633**

[1] From Raspberry Pi 4; [2] From Jetson nano.

3.1.4 Results

We present a comprehensive investigation of the proposed FedMSA method on the Cifar10 benchmark. The experiment was divided into three control groups, and the experimental results of the proposed MS algorithm are reported in Table 3.1 from one Jetson nano and one Raspberry Pi 4.

In the first group, we use the selected model structure with a higher model performance by setting $\gamma = 0.1$ for FedMSA, a model structure searched by NASWOT and NAS (Max Acc) is the model structure with the highest accuracy directly searched in the NAS-201-Bench search space. As can be clearly seen although the global test accuracy of NAS (Max Acc) is 1.3% more than FedMSA$\gamma = 0.1$, it got two times more in FLOPs of the model, training time per round and all model latency metrics in client devices than FedMSA ($\gamma = 0.1$). Particularly noteworthy are some metrics with large base values, such as training time per round and model performance in latency where more than twice those metrics can be catastrophic for the tasks sensitive to training efficiency or inference delay. Besides, the model searched by NASWOT performs similar to NAS (Max Acc) in training efficiency, and model latencies matrix. But, the accuracy of the model was reduced 6.42% performance.

In the second group, we set $\gamma = 0.2$ for FedMSA to search a model structure and select a model structure as 20 layers ResNet, which is widely used and has good performance. FedMSA ($\gamma = 0.2$) has clear advantages in the number of parameters and FLOPs, which are important for resource-constrained and network congestion devices. Although the metrics in model forward propagation and inference latency of ResNet20 perform better on Jetson nano, many tasks are not sensitive to this small gap due to the relatively small base value. However, although the accuracy of ResNet is more than 3.5%, the model back propagation latency of FedMSA ($\gamma = 0.2$) is twice as high as on a ResNet20, and its base value is so large that it has a significant impact on the overall model training efficiency. Thus, our algorithm gets a smaller model and a faster training speed by sacrificing some accuracy.

In the final group, we get a model structure of the same model size as the model with the smallest model size in the NAS-201-Bench search space by setting $\gamma = 0.8$ for FedMSA. At the same time, these two models have a very intimate performance regarding model forward and backward propagation latency and inference latency. It is worth noting that the model structure selected by FedMSA ($\gamma = 0.8$) has an additional 4.3% accuracy over the model structure searched by NAS (Min T_{train}).

Training Time Complexity with Accuracy

The experimental result about training time complexity with accuracy is shown in Figure 3.4. For a distributed machine learning system, the training time complexity of the whole system decides the model training efficiency where a big training time complexity is unacceptable. The model structure search by NAS (Max Acc) has the highest accuracy in Figure 3.4a in 140 communication rounds, but the time spent on model training grows significantly rapidly in Figure 3.4b, and ends up being more than twice as high as the model structure searched by FedMSA ($\gamma = 0.1$). In the early stage of Figure 3.4c, the accuracy achieved of NAS (Max Acc) is much less than others. Besides, the model structure searched by FedMSA ($\gamma = 0.1$) can achieve similar accuracy with less training time growth rate, and it can be observed that through the complete training time of FedMSA ($\gamma = 0.1$), it's accuracy has always been significantly better than the NAS (Max Acc) in Figure 3.4c.

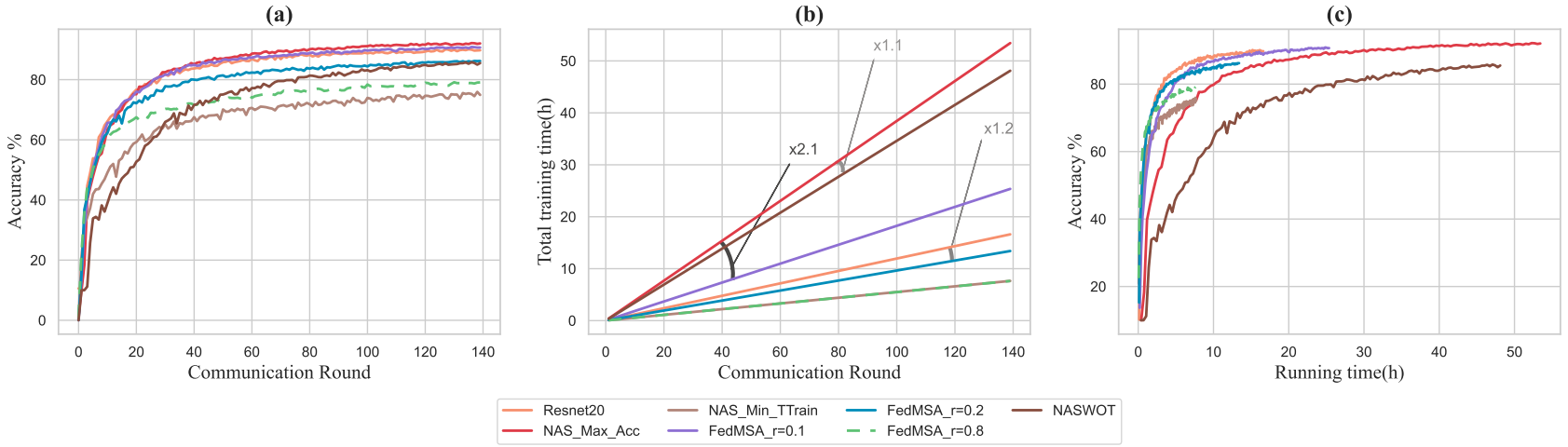

Fig. 3.4 Experiments on Cifar10 under FedMSA system. (**a**) Validation accuracy of models, (**b**) Total training time spent in every communication round of models, (**c**) Changes in model accuracy over time.

Furthermore, although FedMSA ($\gamma = 0.2$) has 3.5% less accuracy than ResNet20 as seen in Figure 3.4a, it has lower training time growth rate and finishes training earlier than ResNet20 as seen in Figures 3.4b,c. Furthermore, as seen in Figure 3.4b, although the two models searched by FedMSA ($\gamma = 0.8$) and NAS (Min Ttrain) have the same total training time and growth rate, FedMSA ($\gamma = 0.8$) has better performance during the whole training period with 4.2% additional accuracy than NAS(Min Ttrain) and can be seen in Figures 3.4a,c.

3.2 FedDAS: Diagnosis and Adaptation System for Federated Learning

FL is a privacy-preserving, distributed machine learning paradigm that enables multiple devices to collaboratively train a global model without exchanging raw data [101]. This edge-cloud paradigm, where a central server coordinates numerous distributed client devices, has gained traction in privacy-sensitive applications such as healthcare [166], banking [171], and autonomous driving [117]. However, FL's real-world deployment and development face significant challenges due to the diversity of devices and networks involved.

A major challenge in FL is the **complexity of the FL pipeline**. Training across multiple heterogeneous devices requires frequent communication between the server and clients, managing diverse hardware, and balancing the trade-off between local computation and global model updates. This complexity is exacerbated by the scale of FL systems, as manually managing and deploying FL tasks across thousands of devices becomes impractical. Therefore, automation of deployment and model adaptation is critical for large-scale FL applications.

Another significant challenge is **device adaptation**, especially in cross-device FL, where clients include mobile devices and IoT systems with diverse computational capabilities. Adaptation is necessary at multiple levels:

- *Hardware and model adaptation*: The FL system needs to adapt the model architecture and system configurations based on the target hardware to maintain efficient training and inference performance across heterogeneous devices.

- *Human-in-the-loop system adaptation*: While automated systems handle the majority of the operations, real-world FL applications often require manual intervention for system reconfigurations due to improper hyper-parameter settings, communication delays, or other system anomalies that affect performance. Thus,

it is essential to have a system that can detect such issues and allow users to intervene and adjust configurations in real-time.

Existing FL platforms such as Flower [9], FATE [92], and FedML [51] have provided valuable contributions to the field, but they fall short in addressing both the hardware heterogeneity and real-time anomaly detection required for large-scale FL applications.

FedMSA [138] was introduced to tackle part of these challenges by automating model selection and adaptation across heterogeneous devices, balancing model performance and training efficiency. However, even with FedMSA's automation, runtime issues such as device failures, communication bottlenecks, and system misconfigurations can still emerge during the training process.

To address these gaps, we present **FedDAS**: a *Federated Learning Diagnosis and Adaptation System*. FedDAS builds on the foundation laid by FedMSA by adding real-time system diagnostics and human-in-the-loop adaptation capabilities. While FedMSA focuses on selecting the optimal model and automating deployment across heterogeneous devices, FedDAS enhances the system's ability to detect and respond to runtime anomalies such as misconfigurations, inefficient resource usage, and communication delays.

FedDAS continuously monitors system-level metrics, including memory usage, network bandwidth, and latency, providing real-time diagnostic feedback to users. When an anomaly is detected, FedDAS alerts the user, enabling them to adjust system parameters or reallocate resources to optimize performance. For example, if a client device experiences excessive memory usage or abnormal latency, FedDAS will notify the user and recommend potential fixes, allowing for dynamic reconfiguration without halting the entire training process.

This combination of automated system adaptation and human-driven diagnostics ensures that FL systems maintain scalability and efficiency even in dynamic, heterogeneous environments. By enabling real-time feedback and manual adjustments, FedDAS addresses critical runtime issues that cannot be fully automated, thus providing a robust solution for real-world FL applications.

FedDAS contributions are summarized as follows:

- We propose an extendable *Federated Learning Orchestration Framework* that supports fault-tolerant and automated deployment of FL tasks, building upon the model selection and adaptation capabilities of FedMSA.

- We introduce a *Real-Time Diagnosis and Adaptation Tool*, which enables continuous monitoring of system-level metrics and human-in-the-loop adjustments to mitigate performance bottlenecks and address anomalies.
- We design a *Federated Learning Anomaly Detection System* that operates within a privacy-secure framework, allowing users to detect and diagnose issues such as device failure, network delays, and configuration errors during FL training.
- We demonstrate the efficacy of *FedDAS* through experiments on a distributed testbed, showcasing its ability to diagnose and adapt to real-world FL challenges in a dynamic and heterogeneous environment.

3.2.1 Related Work

FL has emerged as a promising approach for decentralized model training while preserving data privacy, but its large-scale deployment presents significant challenges, including device heterogeneity, system failures, and network bottlenecks. Numerous studies have addressed these issues through various strategies, including real-time system monitoring, anomaly detection, and adaptive model training. One notable approach is the work of Laptev et al. [78], which proposed a generic anomaly detection framework that dynamically adjusts thresholds to trigger alerts based on evolving system metrics. This method is particularly relevant in the context of FL, where continuous monitoring of system resources such as CPU, memory usage, and network bandwidth can prevent performance degradation during training by detecting and responding to system anomalies.

Another area of relevant research is the development of fault-tolerant systems for distributed learning environments. Ke et al. [71] introduced *LightGBM*, a decision tree-based model known for its efficiency in resource-constrained settings, which has been widely adopted for real-time classification tasks, including anomaly detection. This method's low computational overhead and high accuracy make it particularly suitable for edge devices in FL environments, where computing resources are often limited. Moreover, research on adaptive machine learning frameworks, such as *FedML* by Chaoyang et al. [51] and *Flower* by Beutel et al. [9], has laid important groundwork for scaling FL systems across heterogeneous devices. However, these frameworks primarily focus on model training and lack advanced runtime diagnostic and human-in-the-loop adaptation capabilities, which are essential for maintaining system robustness in real-world deployments.

In contrast, *FedDAS* builds on these foundations by combining real-time monitoring and anomaly detection with human-in-the-loop system adaptation. By leveraging tools like *HW-NAS-Bench* [30] and *NAS-Bench-201* [80], *FedDAS* enhances FL systems with automated model selection and system reconfiguration, addressing both training efficiency and performance bottlenecks in a distributed environment. This hybrid approach, which integrates diagnostic tools with FL, represents a novel solution to the challenges posed by dynamic, large-scale FL deployments.

3.2.2 System design

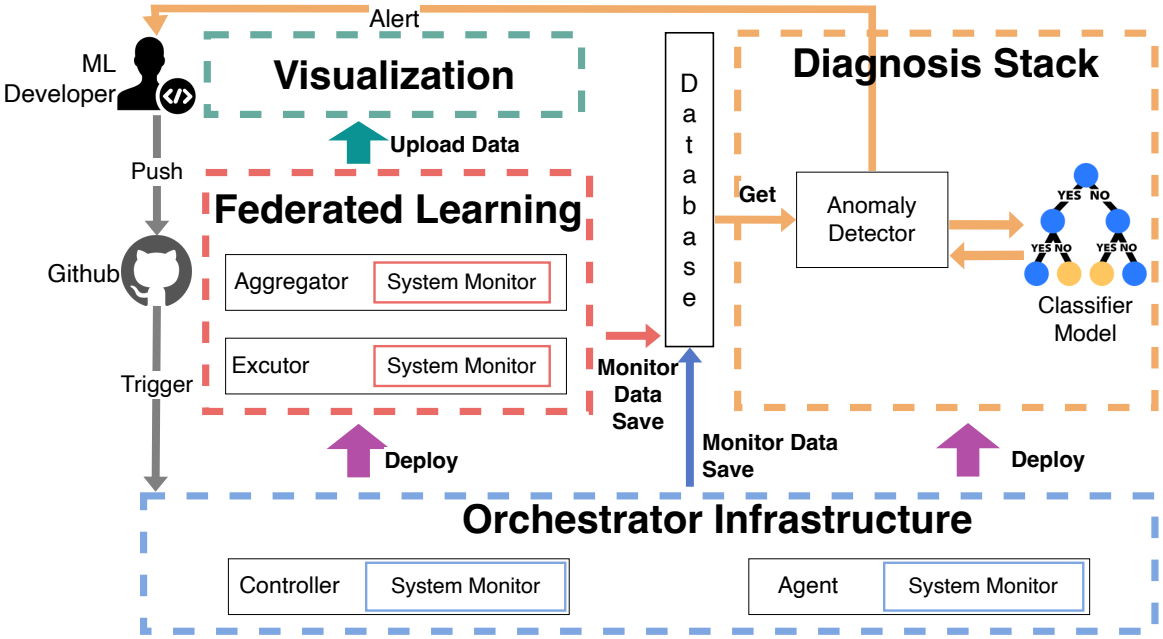

Fig. 3.5 System Overview.

In response to the challenges of real-time adaptation and anomaly detection in FL systems, we introduce *FedDAS*, a device adaptation and diagnostic system for FL. *FedDAS* builds upon some of the foundational automation provided by *FedMSA* but extends its functionality with real-time monitoring, anomaly detection, and human-in-the-loop system adaptation.

The architecture of *FedDAS* consists of three main components: *Orchestration Infrastructure*, *Federated Learning*, and the *Diagnosis Stack*, as depicted in Figure 3.5. *FedDAS* is designed with modularity and scalability in mind, ensuring flexibility for machine learning (ML) developers to personalize FL task deployment and system monitoring, even in large-scale deployments.

System Workflow

The general workflow of *FedDAS* is described below:

1. **Federated Learning Module**: This core module implements FL algorithms (e.g., FedAvg [101]) across an edge-cloud environment. Like *FedMSA*, it supports model adaptation and deployment across heterogeneous devices. However, FedDAS enhances this by allowing runtime monitoring and real-time feedback, which is critical in long-term and large-scale deployments where issues can emerge unpredictably. All aspects of model structure and data partitioning can be customized by developers, supporting highly flexible FL tasks.

2. **Diagnosis Stack**: The diagnosis stack continuously monitors and analyzes the system to detect anomalies in real-time, providing alerts when such anomalies are detected and diagnosing the root causes. It allows for flexible extension, so that new fault classes or diagnostic classifiers can easily be added by developers. This stack adds a critical layer to FL deployments by helping detect and address issues during training or inference that could degrade performance.

3. **Orchestration Infrastructure**: This is the backbone of the system, responsible for managing the deployment of FL tasks, handling system monitoring, and managing communication between components. Through containerization and orchestration tools, FedDAS can automatically deploy tasks across a variety of devices with different hardware architectures, making it highly scalable and adaptable to large-scale environments.

Federated Learning Module

In the context of *FedDAS*, the FL module shares some common features with *FedMSA*, particularly around the automation of model selection and deployment across heterogeneous devices. However, FedDAS goes beyond these functionalities by integrating dynamic, real-time system monitoring and adaptation capabilities.

The proposed FL module consists of one central aggregator deployed on a cloud server and several executors running on edge devices. Each executor is capable of deploying one or multiple clients, depending on the available computational resources (e.g., CPU, GPU) of the device. A client manager within the aggregator handles client registration, manages training requests, and orchestrates the client selection algorithms.

A critical challenge in FL, particularly in large-scale, cross-device settings, is the simultaneous requirement for efficient training and accurate inference. Unlike static

model designs, where developers must manually optimize the trade-off between accuracy, latency, and resource usage, FedDAS incorporates Neural Architecture Search (NAS) techniques such as HW-NAS-Bench [80] and NAS-Bench-201 [30] to automatically generate optimal model architectures. This integration allows FedDAS to adapt model selection dynamically based on user-specified parameters, such as desired accuracy or latency.

In addition to model architecture selection, FedDAS continuously monitors critical metrics during training, such as forward/backward propagation latency and system resource usage (CPU/GPU utilization, memory, and network traffic). This dynamic monitoring allows FedDAS to adjust model configurations in real-time to maintain performance across heterogeneous devices and networks.

Diagnosis Stack

The *Diagnosis Stack* in FedDAS is specifically designed to address system anomalies that can affect the performance of FL applications. The stack follows a *monitor-alert-diagnose* procedure:

- **Monitoring**: FedDAS continuously collects a wide array of system metrics, ranging from hardware resource utilization (e.g., CPU, GPU, memory) to network bandwidth, model performance, and training-related parameters. Both system and model performance data are recorded in real-time and stored in a database for further analysis.

- **Alerting**: The anomaly detection module continuously queries the collected data and triggers alerts if any metric surpasses predefined thresholds. These thresholds are empirically set and adjusted dynamically as the system evolves, following principles similar to those in [78]. For example, if a device shows unusually high memory usage or a sharp drop in model accuracy, an alert is raised to notify the user.

- **Diagnosing**: Upon an anomaly, FedDAS diagnoses the potential root causes using machine learning classifiers. Initially, anomalies are manually labeled to build a training dataset, after which a classifier (e.g., LightGBM [71]) is trained to automatically identify and classify new issues. The use of decision tree-based algorithms ensures interpretability, making it easier for developers to understand the causes of system failures or performance degradation.

In FL applications, transmitting sensitive data from devices to a centralized server for analysis is often impractical due to privacy concerns. To circumvent these issues, FedDAS performs diagnosis locally on edge devices, transmitting only the final diagnosis results to the central server. This distributed approach helps maintain privacy while still enabling accurate and timely anomaly detection.

Orchestration Infrastructure

Deploying and managing a distributed system across devices with heterogeneous hardware is a significant challenge, especially in large-scale FL environments. The *Orchestration Infrastructure* of FedDAS is designed to automate this process by managing the deployment, monitoring, and communication between components.

The infrastructure consists of two main components: the controller and the agents, operating at the cloud tier and edge tier, respectively. The controller manages the entire system, handling task orchestration, agent registration, and communication management via a message queue service. The agents, deployed on edge devices, are responsible for executing FL tasks and collecting system metrics.

FedDAS employs a cross-platform builder to automatically build and deploy FL tasks across different hardware architectures (e.g., x86_64, aarch64, armv7l). This ensures that even devices with diverse computational capabilities can participate in the FL system without manual intervention. The builder is based on the QEMU user-static emulator[1], which facilitates the cross-compilation of code for different architectures.

The infrastructure also includes a comprehensive monitoring and alerting system. If a microservice or component crashes, or if there is a loss of connection, the orchestrator immediately alerts the user and attempts to resolve the issue. The system also logs events for further investigation and diagnostics.

3.2.3 Experiments

Experiment Setup

The system components of *FedDAS* were deployed on real-world testbeds, as illustrated in Fig. 3.3. The cloud server runs on an Ubuntu machine equipped with a 20-core Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz and 64 GB of memory. For edge devices, we utilized five Jetson-Nano boards[2], each with 4GB of RAM and an ARM Cortex-A57

[1]https://github.com/multiarch/qemu-user-static
[2]https://developer.nvidia.com/embedded/jetson-nano-developer-kit

CPU, along with two Raspberry Pi 4 models, each with 4GB of RAM and a 1.5GHz 64-bit quad-core ARM Cortex-A72 CPU.

The orchestrator was implemented in Golang for efficient deployment, while both the Federated Learning and Diagnosis Stack components were developed in Python. For the experiments, we trained a Neural Network (NN) model using HW-NAS with the FedAvg algorithm [101] on the CIFAR-10 dataset[3].

As discussed in Section § 3.2.2, the diagnosis classifier was deployed on each edge device. We collected 15 key metrics locally on each device, including CPU and memory usage, disk I/O, network traffic, hyperparameter settings, communication delays, and epoch timings. The data was manually labeled into three categories: *normal*, *network congestion*, and *batch size too large*. In total, 40,000 rows of data were gathered for training and 10,000 rows for testing the classifier on the edge devices.

Execution Flow

This subsection details the execution sequence of *FedDAS*, demonstrating its ease of use and robust support for large-scale FL task deployment with high reliability (see Figure 3.6). The process begins when a machine learning developer pushes code to a GitHub repository, triggering the orchestrator to pull the code in Step 0. In Step 1, the cross-platform builder compiles the source code for the FL aggregator and executors based on the hardware architecture of the target devices. In Step 2, the FL aggregator is deployed locally, and the executor images are published to a cloud image repository. The orchestrator agents then retrieve these images in Step 4 after receiving the "docker image ready" notification from the controller in Step 3. Following this, all agents deploy the FL executors locally in Step 5. Once the FL aggregator receives "ready" notifications from all executors in Step 6, the model search process begins in Step 7. During Step 7, the system first lists the top 10 model architectures with minimized inference latency using HW-NAS-Bench, considering the hardware architecture of the target devices and the task dataset. The best architecture for the target task is then searched using NAS-Bench-201. The system checks if this architecture is among the top 10; if not, the Model Selection (MS) algorithm of FedMSA is used to find an optimal architecture based on user-defined parameters indicating the importance of accuracy versus inference latency. Once the optimal model is selected, all executors initialize the target model in Step 8 and begin training in Step 9. Throughout the training process, the system continuously monitors performance, detecting any anomalies. Detected anomalies are diagnosed, and notifications are sent to the ML developer to allow

[3]https://www.cs.toronto.edu/ kriz/cifar.html

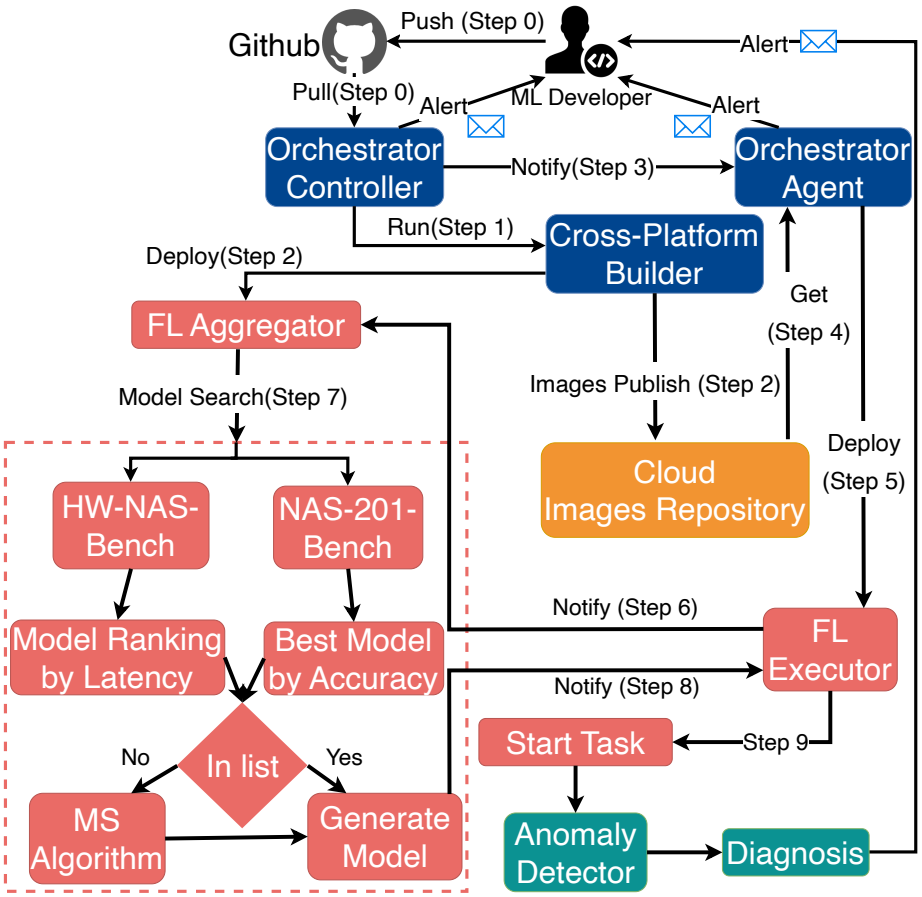

Fig. 3.6 Execution Flow.

code adjustments and job resubmission. Meanwhile, the orchestration infrastructure monitors the system and all microservices, alerting the user via email if any faults occur.

3.2.4 Results

LightGBM classifier models, known for their lightweight nature and low resource consumption, were implemented using Bayesian Optimization (BO) [134] to search for optimal hyperparameter settings. An example of a selected decision tree from the classifier model used by an executor is illustrated in Figure 3.7, showcasing the model's reasoning process through each node and the corresponding class outcomes at the leaves.

For instance, if the container's memory usage is below 1.59GB and the communication delay exceeds 14.5 seconds, the monitoring data would be classified as network

congestion. Approximately 12,000 rows of data were randomly selected to test the model's accuracy, achieving an accuracy of 0.999725 and an F1 score of 0.999744.

These preliminary results indicate a high level of accuracy. However, we are continuing to expand the anomaly database and retrain the model to diagnose a broader range of anomalies.

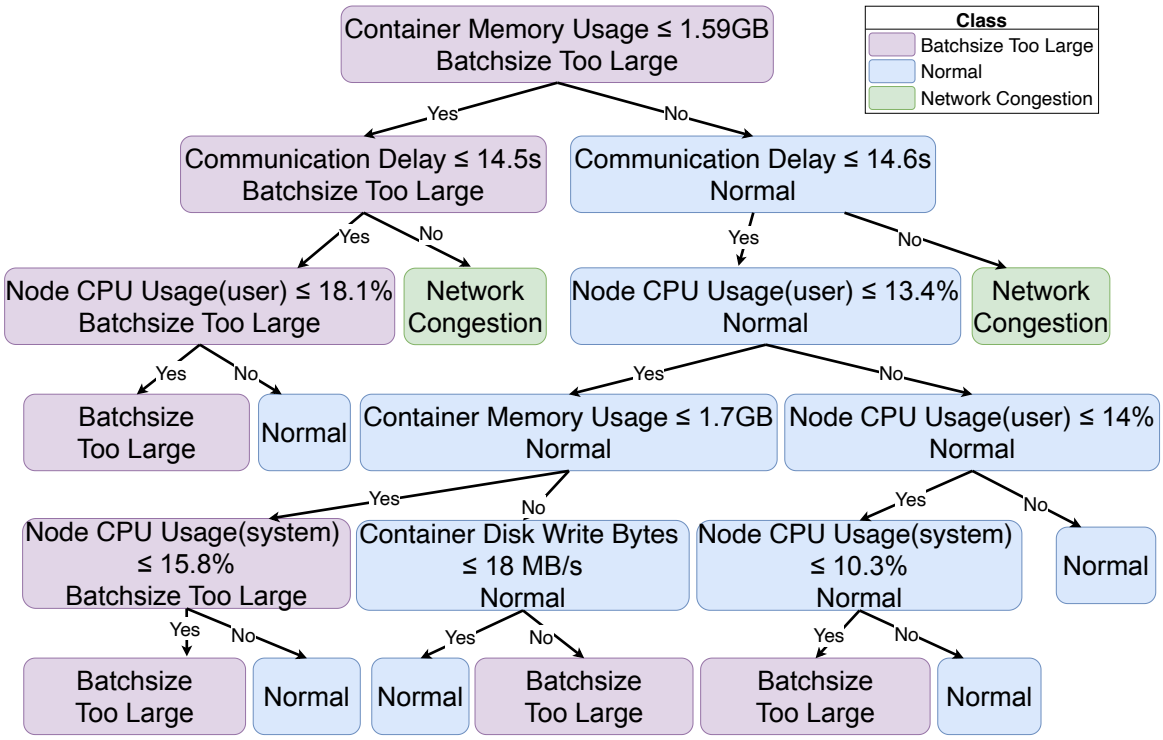

Fig. 3.7 Diagnosis Classifier Tree Structure.

3.3 Summary

In this chapter, we presented two significant contributions to the field of FL: **FedMSA** and **FedDAS**, both of which address different aspects of model selection, system adaptation, and deployment within FL environments.

FedMSA is a model selection and fast adaptation system designed to reduce the complexity of FL system deployment. It automates the adaptation and deployment of training tasks as microservices in the FL life cycle, while also providing an optimal model selection algorithm. The evaluation demonstrated that the model structures selected by FedMSA consistently outperformed manually selected models and widely used baselines. Without the benefit of a model selection algorithm, FL developers

may inadvertently choose models with higher training complexity and lower accuracy. By leveraging metrics from NAS search spaces, FedMSA ensures that developers can deploy models that strike a balance between training efficiency and performance.

FedDAS expands upon the automation provided by FedMSA, focusing on real-time diagnosis, adaptation, and large-scale system deployment. FedDAS is a scalable and reliable system that integrates real-time anomaly detection and system monitoring to adapt to diverse hardware environments in distributed FL systems. Through comprehensive experiments on real-world testbeds, FedDAS demonstrated its effectiveness in diagnosing and resolving system anomalies with impressive accuracy and F1 scores of 0.999725 and 0.999744, respectively. This showcases its robustness in managing and optimizing FL systems across heterogeneous devices and environments.

Together, *FedMSA* and *FedDAS* provide a complete framework for managing the challenges of FL. While FedMSA automates model selection and reduces deployment complexity, FedDAS ensures the reliability, and adaptability of these systems. These two systems, when used together, create a comprehensive solution that optimizes FL at both the model and system level, significantly improving performance, reducing complexity, and ensuring the efficient operation of FL systems in dynamic and large-scale environments.

Furthermore, while each new query in our system involves model deployment and training, the design choices (including pre-computation of metrics, automated containerized deployment, and cost distribution across a large-scale network) ensure that these costs remain low relative to the performance gains. This justification reinforces the efficiency of our approach in real-world federated learning scenarios.

For future work, we plan to extend the model selection algorithm of FedMSA to support a wider range of metrics, including more comprehensive metrics from distributed systems, and to integrate additional NAS benchmarks. We will also enhance the system's ability to fine-tune and adapt selected models to better align with developers' expectations by incorporating additional configuration parameters and potentially applying recommendation machine learning models. Furthermore, FedDAS will continue to be refined with enhanced anomaly detection capabilities, deeper integration with system-level metrics, and more advanced diagnostic classifiers, ensuring continued robustness in increasingly complex FL scenarios.

Chapter 4

Standalone Client On-device Continuous Learning

Building on the adaptive model selection strategies discussed in Chapter 3, this chapter shifts focus towards enhancing the learning capabilities of individual FL clients. Specifically, we address the challenge of continuous learning on edge devices, where the risk of catastrophic forgetting looms large. By introducing novel mechanisms designed to optimize knowledge retention during local updates, the aim is to ensure that models can continuously adapt to new data without compromising previously acquired knowledge. This approach is crucial for enabling FL systems to maintain high performance over time, particularly in dynamic environments.

Class incremental learning (CIL) strives to emulate the human cognitive process of continuously learning and adapting to new tasks while retaining knowledge from past experiences. Despite significant advancements in this field, Transformer-based models have not fully leveraged the potential of attention mechanisms to balance the transferable knowledge between tokens and the associated information. This chapter addresses this gap by using a dual variational knowledge attention (DVKA) mechanism within a Transformer-based encoder-decoder framework tailored for CIL. DVKA mechanism aims to manage the information flow through the attention maps, ensuring a balanced representation of all classes and mitigating the risk of information dilution as new classes are incrementally introduced. This method leverages the information bottleneck and mutual information principle and selectively filters less relevant information while directing the model's focus toward the most significant details for each class. DVKA is designed with two distinct attentions: one focused on the feature level and the other on the token dimension. The feature-focused attention aims to purify the complex nature of various classification tasks, ensuring a comprehensive

representation of both old and new tasks. The token-focused attention mechanism highlights specific tokens, facilitating local discrimination among disparate patches and fostering global coordination for a spectrum of task tokens. Our work is a major stride towards improving transformer models for class incremental learning, presenting a theoretical rationale and effective experimental results on three widely used datasets.

4.1 Dual Variational Knowledge Attention for Class Incremental Vision Transformer

The ability to continually learn and adapt to new information is a characteristic of human intelligence [48]. Class incremental learning (CIL) [7, 15, 33, 181, 162, 167, 97, 123, 182], has emerged as a critical area of research to mimic this cognitive process to prevent models from suffering catastrophic forgetting. Specifically, its goal is to incrementally learn and adapt to new tasks (i.e., classes), while preserving the knowledge of previously learned classes, without the need to revisit the entire dataset. However, catastrophic forgetting remains a major challenge for CIL models during sequential learning of new classes.

Various methods have been proposed to mitigate the catastrophic forgetting problem by striking a balance between model stability and plasticity [182]. Rehearsal-based approaches [123, 97, 93, 153] reduce knowledge forgetting by retaining a few samples of data from old tasks and replaying them during the training of new tasks. In contrast to directly reusing exemplars (historical data), regularization-based approaches, such as data regularization-based methods [95] and Knowledge Distillation (KD) [84, 59, 32], aim to regularize the model optimization direction of the current task using prior data or model. Furthermore, more recently, network expansion-based techniques [95, 74, 174, 128, 2, 167] have achieved a significant advance on alleviating the forgetting problem by optimizing existing or introducing new neurons or backbone structures, allowing the model to extract novel features incrementally. Apart from this, other approaches to maximizing the retention of old knowledge by reducing information dilution should not be overlooked.

Despite extensive research aimed at mitigating the loss of transferable knowledge within various model intermediaries, transformers-based approaches have yet to fully explore such knowledge between tokens and the associated information. Transformers, at their core, utilize attention mechanisms to model the dependencies between different parts of the input and have gradually been applied in CIL due to their exceptional performance across various tasks [37, 155, 111, 156, 33]. Most of them aim to balance

stability-plasticity with integrating extra learnable modules. For example, [155] tackles the issue of catastrophic forgetting in transformers using an exemplar replay strategy by introducing an inter-task attention mechanism. The strategy uses self-queries and an externally learnable key to create attention maps, effectively reducing the number of parameters and consolidating crucial attention weights. Concurrently, Dytox [33] mitigates forgetting by allocating a unique attention block for each task in the expanded network structure. Both approaches, through newly designed attention blocks, successfully conserve knowledge from earlier tasks.

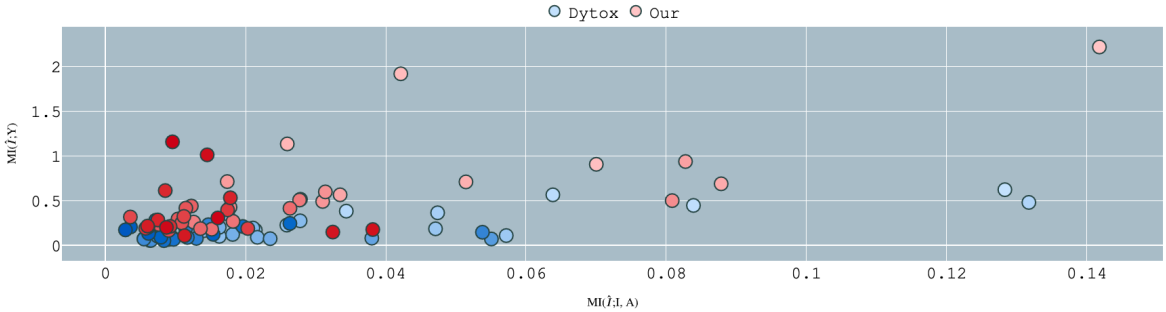

Fig. 4.1 Mutual information analysis of our approach and Dytox [33] on exemplar replay data from CIFAR100 dataset with 50 incremental task steps. The graph plots mutual information levels per task with Dytox in progressively darker shades of blue and our method in progressively darker shades of red as tasks increase. A smaller MI(I, A; $\hat{I}$) suggests diminished information retention by the decoder concerning the input, while a smaller MI($\hat{I}$; Y) indicates reduced mutual information between the decoder and model output. A wider point distribution from the coordinate center signifies enhanced information retention. Our method distinctively optimizes the decoder's ability to conserve information relative to the model output.

However, as new classes are introduced continuously, the relatively finite learning space of these attention maps in conventional transformers can become increasingly overloaded with information, potentially leading to a dilution of relevant information from earlier classes. Specifically, due to the huge training data quantity shift possibly happening when transferring to a new task, the attention mechanism will easily tilt the model's learning and assign biased correspondence and importance to the current task [11, 17, 152]. Although the rehearsal samples are accessible and useful, it is still challenging to fully capture the variability and complexity of the old classes. The attention mechanism becomes over-tailored to the representations of rehearsed samples and performs poorly on unseen examples from the old classes. To combat catastrophic forgetting, it is essential to instill a sense of "balance" into the attention mechanism as it is the key in transformers. This balance is not just about equal representation of

all classes, but also about managing the flow of information in such a way that the model retains a holistic understanding of all the classes it has seen so far. As shown in Figure 4.1 where, as the task increases, the model's mutual information about the old knowledge gradually decreases.

Herein lies the motivation for our proposed mechanism: maximizing representation retaining of classes of prior tasks by leveraging attention maps' information filtering approach, while balancing the overall amount of information expressed by current and old tasks. Drawing inspiration from the principles of the information bottleneck and mutual information [144], we propose a methodology designed to dynamically equilibrate the allocation of feature space dedicated to the information of both historical and novel knowledge, by continually optimizing the density of information within the constraints of the transformer network's information retention capacity. Our method starts by building a Transformer-based encoder-decoder framework where the encoder consists of the conventional transformer layers [31]. Most importantly, we proposed the *dual variational knowledge attentions* (DVKA) using the exiting theories [77] to construct the class incremental transformer decoder. The proposed mechanism generates the variational attention maps to provide a more nuanced control over the learning process, thus enabling efficient use of both current samples and rehearsal-based memory buffers and promoting better generalization across tasks. Specifically, one attention particularly focuses on the feature level of the self-attention process, effectively and meticulously addressing the escalating complexity inherent to various classification tasks, ensuring the holistic representation that benefits both old and new tasks. Another one aims to bolster the adaptability of attention along the token dimension, encouraging local discrimination among disparate patches, and facilitating global coordination for a spectrum of tasks interacting with task tokens, akin to class tokens [145]. The main contributions of this research work can be summarized as follows:

- Motivated by the information bottleneck and the mutual information theory, a dual variational knowledge attention mechanism is proposed to comprehensively improve the transformer that is specific for class incremental learning.

- In order to generate variational attention maps that strategically minimize redundant information while maximizing the retention of task-relevant knowledge, feature-level attention mechanism is introduced. This mechanism ensures that the model effectively incorporates new classes without compromising the stability of previously learned knowledge.

- Lastly, a token-focused attention mechanism is proposed that helps implicitly emphasize the importance of specific tokens associated with the output and which provide the most discriminative features for a new class in relation to existing ones.

4.1.1 Related Work

Class incremental learning (CIL) aims to enable models to learn new classes continuously while mitigating catastrophic forgetting of knowledge from previously learned classes [182]. To tackle the forgetting problem, existing studies could be divided into three categories: rehearsal-based, regularization-based, and dynamic network-based methods. **Rehearsal-based** methods retained a small subset sampled from prior training datasets as exemplars for continuous replay during new tasks, thus preserving older knowledge. Existing works [97, 93, 153] proposed memory optimization techniques to address the continuous accumulation of exemplars during task transfer, which are significant challenges in terms of computational and space complexity. They enhanced the quality of exemplars while preserving their discriminative capabilities. **Regularization-based** methods incorporate additional regularization terms into the loss function, fortifying pre-existing knowledge as the model processes new classes, an exemplification being the knowledge distillation (KD) approach [84, 123]. KD uses the old task model as a teacher model to regularize the current task model to prevent overfitting the knowledge on current data. **Dynamic network-based** methods aim to expand neuron [174] or new backbone network structure [128, 2, 167] to enhance the model's representation ability in capturing new class features incrementally. Although these dynamic architecture approaches, such as DER [167], have demonstrated remarkable success and powerful performance in mitigating knowledge forgetting, they face the challenge of an ever-growing model size as the number of classes increases.
Transformers, a groundbreaking model introduced in [147, 34, 35], has revolutionized the field of natural language processing with its innovative self-attention mechanism and ability to process input sequences in parallel. BERT [23] adapted the encoder and decoder layers of the original transformer architecture into a series of identical encoder blocks with attention mechanism [36]. Inspired by BERT, Dosovitskiy et al. (2020) proposed Vision Transformer (ViT), which extended the transformer model's capabilities to handle visual tasks [139, 149], by dividing images into small patches of pixels and using them as input for the model. ViT achieved remarkable success in learning paradigms and applications [72], exceptionally in class incremental learning [37, 155, 111, 156, 33].

Information bottleneck and mutual information was introduced by [144] as a theoretical framework to understand trade-offs in machine learning between compression and prediction, which is to extract the most relevant part of the data that is useful for prediction tasks and thereby facilitating a balance between data compression and prediction accuracy. In the context of deep learning, employed as an analytical instrument, the information bottleneck (IB) principle has been used to scrutinize deep neural networks' training dynamics [133]. Of late, the principle of IB has seen application in continual learning. Methods like [131] leverage it to alleviate catastrophic forgetting by preserving the vital information of previous tasks while learning new classes. However, despite its theoretical and practical importance, the application of the IB principle to certain domains, especially those domains involving transformers, remains a challenging open problem. This research tackles the above challenge by proposing a novel methodology that leverages the IB principle to balance the retention of knowledge from new and old data, thereby optimizing the performance of transformer networks in a class incremental setting.

4.1.2 Preliminaries

This chapter is focused on CIL, a particularly promising approach within the realm of continual learning. The proposed model (i.e., Transformer) is trained on a series of tasks indexed by $t \in \{1, 2, \ldots, T\}$, where each task contains a task-specific class set C_t and $t \neq t' \Rightarrow C_t \cap C_{t'} = \emptyset$. During each task, the model is trained with a dataset $D_t \sim \{(I_i, Y_i)\}_{i=1}^{n_t}$, where the I_i is the input image, Y_i is the corresponding labels $Y_i \in C_t$, and n_t indicating the number of such input pairs. The primary challenge lies in the fact that the data is only temporarily fully accessible. Aligning with previous studies, at task t only a limited number of samples from previous tasks (i.e., $t-1, t-2, ...$), are available for training as rehearsal data. Nevertheless, the model should maintain its ability to accurately classify test data originating from all previously encountered classes.

An initial pipeline is constructed for CIL, leveraging the capabilities of the transformer model [147, 31]. Consider an image $I_i \in \mathbb{R}^{C \times H \times W}$, where the C, H, and W denote the channel number, height, and width of an image, respectively. It is subsequently tokenized and encoded by a transformer encoder consisting of five self-attention modules into $I_i \in \mathbb{R}^{\mathrm{N} \times D}$ where N and D are the number of tokens and the feature dimension of each token respectively. A soft task-token is then added into I_i, which contains class-specific information about each task inspired by [145], resulting in $I_i \in \mathbb{R}^{(N+Z) \times D}$, where Z is the number of class in each task. The soft task token enhances the model to

assimilate and coordinate the global task information from the input for classification. Crucially, we introduce a dual variational knowledge attention mechanism (DVKA) to construct the transformer decoder, designed specifically for class incremental scenarios. This mechanism is capable of adaptively managing the information flow associated with different tasks. The I_i is subsequently processed through the decoder, which, in conjunction with a classifier, generates the final prediction. To ensure better attribution of the class incremental learning for different task-specific samples, each task executes with an independent classifier. By adopting the strategy of independent classifiers [132] in combination with sigmoid activation and binary cross-entropy loss $\mathcal{L}_{cls}$, we eliminate the possibility of late fusion [121] of task embeddings. This strategy inherently encourages the classifiers with greater specializations.

4.1.3 Methodology

The primary focus in the following sections is on the main contribution of of utilizing the dual variational knowledge for transformer attention mechanism in the scenario of class incremental learning. In this mechanism, our method is based on the information bottleneck [144, 1, 77] and mutual information [47], designed to manage both token-level and feature-level information flows. The variational operations serve the pivotal function of adaptively coordinating knowledge gleaned from past classes with recent learnings, which helps to reduce catastrophic forgetting. The notation used is as follows: $p(A_{\mathrm{D}} \mid I_i)$ and $p_N(A_N \mid I_i)$ denote the proposed attention modules, namely the feature-level knowledge guided attention and token-level knowledge guided attention respectively, which are used to generate the continuous variational attention map A_{D} and A_N. Then the dual attention-guided features can be denoted as $p(\hat{I}_i \mid I_i, A_{\mathrm{D}})$ and $p(\hat{I}_i \mid I_i, A_N)$, and the final prediction can be reflected as $p(Y_i \mid \hat{I}_i)$ along with a dynamic task decoder [33] which is the multiplication of two attentions.

Feature-level Variational Knowledge Attention

Along the feature dimension (i.e., D in I_i), the method generates the variational attention maps minimizing the mutual information between the attention-guided representation and the input, reducing the influence of redundant features, while maximizing the mutual information between the attention-guided representation and the class label in each task. This ensures the preservation and emphasis of task-specific knowledge crucial for adapting to new classes without forgetting previously learned knowledge. Consequently, we will have feature-level variational knowledge attention

starting with the help of kullback-leibler (KL) divergence along with estimating the $p(Y_i \mid \hat{I}_i)$ by variational approximation via $q_{\mathrm{D}}(Y_i \mid \hat{I}_i)$, which can be formed as:

$$\int p(Y_i \mid \hat{I}_i) \log \frac{p(Y_i \mid \hat{I}_i)}{q_{\mathrm{D}}(Y_i \mid \hat{I}_i)} dY_i - \int p(Y_i \mid \hat{I}_i) \log q_{\mathrm{D}}(Y_i \mid \hat{I}_i) dY_i. \tag{4.1}$$

Such divergence is always greater than or equal to zero, resulting in Jensen's inequality from information theory. The joint probability distribution of the output variable Y_i and the attention-guided feature $\hat{I}_i$ can be expressed as the product of the marginal distribution of the corresponding variables, and the calculation of entropy can be perceived as independent of the optimization process, and hence it can be disregarded here. We leverage the mutual information [144, 77] $M(\hat{I}; Y) - \beta M(I, A; \hat{I})$ to handle the attention's selectivity via information flow to guide the feature learning. Following this, the variational lower bound of $M(\hat{I}; Y)$ is maximized while the mutual information between the attention-guided representation and the input is minimized ensuring the attention mechanism can alleviate the redundant feature and select the important input information for the final prediction. Thus, the variational attention mechanism can be obtained as:

$$\begin{aligned} &\mathbb{E}_{I_i, A_{\mathrm{D}}, Y_i, \hat{I}_i} \left[p(I_i, y) p(A_{\mathrm{D}} \mid I_i) p(\hat{I}_i \mid I_i, A_{\mathrm{D}}) \log q(Y_i \mid \hat{I}_i) \right] \\ &- \beta \mathbb{E}_{I_i, A_{\mathrm{D}}, \hat{I}_i} \left[p(I_i) p(A_{\mathrm{D}} \mid I_i) p(\hat{I}_i \mid I_i, A_{\mathrm{D}}) \log \frac{p(\hat{I}_i \mid I_i, A_{\mathrm{D}})}{g(\hat{I}_i)} \right]. \end{aligned} \tag{4.2}$$

where $g(\cdot)$ is the Gaussian prior distribution. To enable the dynamic learning of the attention, the reparametrization trick [73] is leveraged with Gaussian random variables ρ and ζ, and regularization is calculated as follows:

$$\begin{aligned} \mathcal{L}_{FVKA} = &- \mathbb{E}_{\zeta \sim p(\zeta), \rho \sim p(\rho)} \left[\log q \left(Y_i \mid f(I_i, g(I_i, \rho), \zeta) \right) \right] \\ &+ \beta \mathbb{E}_{\rho \sim p(\rho \mid I_i)} \left[\log \frac{p(\hat{I}_i \mid I_i, g(I_i, \rho))}{g(\hat{I}_i)} \right] \end{aligned} \tag{4.3}$$

where the first term is negative log-likelihood for prediction, and the second term indicates KL divergence between the attention-guided feature and the prior.

Token-level Variational Knowledge Attention

The token-level variational knowledge attention is proposed to highlight the most informative and representative tokens. With the progressively increased number of tasks, this method can prevent the model from being overloaded with information of new class and alleviate the model overfitting due to the limited rehearsals.

Consider the attention map A_N derived from I_i. We propose token-level variational knowledge attention, which is formed as $A_N \odot I_i$. A_N is assumed to be conditionally independent given the input I_i, and $p(A_N \mid I_i) = \prod_{j,k} p\left(A_{N,j,k} \mid x_{i,j,k}\right)$. We set $p\left(A_N \mid I_i\right) = \text{Bernoulli}\left(\mathcal{F}(I_i)\right)$ and the $\mathcal{F}(\cdot)$ is leveraged to build a dynamic gating mechanism. The posterior distribution of A_N can be defined as a composition of Dirac-delta distributions. This intricate relationship can be succinctly represented as:

$$p\left(\hat{I}_i \mid I_i\right) = \left(1 - \mathcal{F}(I_i)\right) \delta\left(\hat{I}_i\right) + \mathcal{F}(I_i) \delta\left(\hat{I}_i - I_i\right). \tag{4.4}$$

In this equation, $\delta(\cdot)$ denotes the Dirac-delta probability distribution. This is a distinct type of distribution that yields a value of zero for all points except at the location denoted at the specifically defined points. Here we consider that our prior should be sparse for the output, where $g(A_N) = \text{Bernoulli}(\tau)$, and, the distributions over I_i and A_N on $A_N \odot I_i$ induces a distribution as $r\left(\hat{I}_i \mid I_i\right) = (1-\tau)\delta\left(I_i\right) + \tau\delta\left(\hat{I}_i - I_i\right)$. Then we constrain the $p\left(\hat{I}_i \mid I_i\right) = r\left(\hat{I}_i \mid I_i\right)$. Directly optimizing this target is challenging, while this can be alleviated by an upper bound for the mutual information [77], and the regularization can be obtained via leveraging the standard cross-entropy bound, which is formed as:

$$\begin{aligned} \mathcal{L}_{TVKA} = & \mathbb{E}_{A_N \sim p(A_N|I_i)}\left[-\log q(Y_i|A_N \odot I_i)\right] \\ & + \beta \mathbb{E}_{A_N \sim p(A_N|I_i)}\left[\log \frac{p(A_N|I_i)}{g(A_N)}\right], \end{aligned} \tag{4.5}$$

Finally, the entire transformer is optimized by the four losses as:

$$\mathcal{L} = \mathcal{L}_{cls} + \eta \mathcal{L}_{FVKA} + \gamma \mathcal{L}_{TVKA} + \mathcal{L}_{distill} \tag{4.6}$$

The knowledge distillation is widely used in existing work [162, 15, 123], and we applied the naive distillation loss on the probabilities [57].

4.1.4 Experiments

Experimental Setup

Datasets: The proposed method is evaluated in class incremental setting experiments on three widely used image recognition datasets namely, CIFAR100 [75], ImageNet100, and ImageNet1000 [22]. CIFAR100 consists of 60,000 color images of size 32×32 pixels for 100 classes. ImageNet1000 contains 1.2 million color images of size 224×224 pixels for 1000 classes. ImageNet100 is a subset of ImageNet1000, which involves

100 classes. All computations are carried out on a server with dual Nvidia A100 GPUs with a total of 2000 exemplars evenly distributed across classes.

Conventionally, in the context of continual learning scenarios, the evaluation typically involves 10 incremental tasks with 10 new classes per task for CIFAR100 and ImageNet100 and 100 new classes per task for ImageNet1000. To provide a comprehensive analysis aligning with the Dytox [33] framework, the evaluation is extended to include additional scenarios, specifically, a 20-task scenario for CIFAR100 is introduced, where each task consists of five new classes. Furthermore, the lifelong incremental learning setting is simulated with a 50-task scenario, where each task adds two new classes to the CIFAR100 dataset. The hyperparameter τ is set to 0.2 for CIFAR100, 0.25 for ImageNet100, and 0.4 for ImageNet1000. Also, η is set to 0.01 and γ to 0.01. During the experiments, the average accuracy after each step ($\mathcal{A}_{avg}$), the final accuracy after the last step ($\mathcal{A}_{last}$) and the numbers of parameters of the model #P are measured.

Table 4.1 Plural form on ImageNet100 and ImageNet1000.

Scenarios	ImageNet100 10 tasks					ImageNet1000 10 tasks				
		Top-1		Top-5			Top-1		Top-5	
	#P	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$	#P	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$
ResNet18 Joint	11.22	-	-	-	95.10	11.68	-	-	-	89.27
Transf. Joint	11.00	-	79.12	-	93.48	11.35	-	73.58	-	90.60
E2E [15]	11.22	-	-	89.92	80.29	11.68	-	-	72.09	52.29
iCaRL [123]	11.22	-	-	83.60	63.80	11.68	38.40	22.70	63.70	44.00
BiC [162]	11.22	-	-	90.60	84.40	11.68	-	-	84.00	73.20
WA [181]	11.22	-	-	91.00	84.10	11.68	65.67	55.60	86.60	81.10
RPSNet [120]	-	-	-	87.90	74.00	-	-	-	-	-
Simple-DER [85]	-	-	-	-	-	28.00	66.63	59.24	85.62	80.76
DER w/o P [167]	112.27	77.18	66.70	93.23	87.52	116.89	68.84	60.16	88.17	82.86
DER [167]	-	76.12	66.06	92.79	88.38	-	66.73	58.62	87.08	81.89
DyTox [33]	11.01	77.15	69.10	92.04	87.98	11.36	71.29	65.34	88.59	84.49
Ours (DVKA)	11.63	78.22	70.21	93.34	89.01	11.92	72.60	66.53	89.70	85.65

Table 4.2 Plural form CIFAR100.

	10 tasks			20 tasks			50 tasks		
Scenarios	#P	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$	#P	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$	#P	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$
ResNet18 Joint	11.22	-	80.41	11.22	-	81.49	11.22	-	81.74
Transf. Joint	10.72	--	76.12	10.72	-	76.12	10.72	-	76.12
iCaRL [123]	11.22	$65.27_{\pm 1.02}$	50.74	11.22	$61.20_{\pm 0.83}$	43.75	11.22	$56.08_{\pm 0.83}$	36.62
UCIR [59]	11.22	$58.66_{\pm 0.71}$	43.39	11.22	$58.17_{\pm 0.30}$	40.63	11.22	$56.86_{\pm 0.83}$	37.09
BiC [162]	11.22	$68.80_{\pm 1.20}$	53.54	11.22	$66.48_{\pm 0.32}$	47.02	11.22	$62.09_{\pm 0.85}$	41.04
WA [181]	11.22	$69.46_{\pm 0.29}$	53.78	11.22	$67.33_{\pm 0.15}$	47.31	11.22	$64.32_{\pm 0.28}$	42.14
PODNet [32]	11.22	$58.03_{\pm 1.27}$	41.05	11.22	$53.97_{\pm 0.36}$	35.02	11.22	$51.19_{\pm 1.02}$	32.99
RPSNet [120]	56.5	68.60	57.05	-	-	-	-	-	-
DER w/o P [167]	112.27	$75.36_{\pm 0.36}$	65.22	224.55	$74.09_{\pm 0.33}$	62.48	561.39	$72.41_{\pm 0.36}$	59.08
DER [167]	-	$74.64_{\pm 0.28}$	64.35	-	$73.98_{\pm 0.36}$	62.55	-	$72.05_{\pm 0.55}$	59.76
DyTox [33]	10.73	$73.66_{\pm 0.02}$	$60.67_{\pm 0.34}$	10.74	$72.27_{\pm 0.18}$	$56.32_{\pm 0.61}$	10.77	$70.20_{\pm 0.16}$	$52.34_{\pm 0.26}$
Ours (DVKA)	11.82	$74.70_{\pm 0.10}$	$61.92_{\pm 0.30}$	11.88	$73.40_{\pm 0.20}$	$57.52_{\pm 0.55}$	11.95	$71.21_{\pm 0.18}$	$53.41_{\pm 0.23}$

4.1.5 Results

Performance on ImageNet100 and ImageNet1000 (Table 4.1). Significant improvements are observed with the proposed method across all class incremental scenarios in ImageNet100 and ImageNet1000. Specifically, in the 10-task scenario on ImageNet100, an average accuracy (i.e., top-1 accuracy by default) of 78.22% and a final accuracy of 70.21% is achieved. This outperforms the second-best model, DyTox [33], which achieves an average accuracy of 77.15% and a final accuracy of 69.10%. Likewise, for the 10-task scenario on ImageNet1000, the model outperforms all other baselines with an average accuracy of 72.60% and a final accuracy of 66.53%. This superior performance in CIL tasks demonstrates the efficacy of the proposed model in this research work.

Regarding the top-5 accuracy, our method continues to excel. For ImageNet100, an average accuracy of 93.34% and a final accuracy of 89.01% is achieved, outperforming the best comparison method, DER [167], which scores an average accuracy of 92.79% and a final accuracy of 88.38%. For the more challenging ImageNet1000, an average accuracy of 89.70% and a final accuracy of 85.65% is achieved, surpassing DyTox, the second-best model, with an average accuracy of 88.59% and a final accuracy of 84.49%. This not only establishes our method's robustness to task complexity but also highlights its performance superiority.

Performance on CIFAR100 (Table 4.2). Under constrained model size expansion, our approach consistently excels against state-of-the-art methods on CIFAR100 for varying task complexities.

For the 10-task scenario, an average accuracy of 74.70% and a final accuracy of 61.92% is achieved. This is ahead of Dytox [33] with 73.66% average and 60.67% final accuracy.

In the 20-task scenario, our model continues to outperform others, marking an average of 73.40% and concluding at 57.52%. Dytox, the nearest rival, reports 72.27% on average and concludes at 56.32%. This gap in the final accuracy underscores our model's enhanced capability to preserve knowledge from prior tasks, a critical aspect of CIL.

In the 50-task scenario, our model's performance retains its superiority, yielding an average of 71.21% and a final accuracy of 53.41%. Dytox, in comparison, records 70.20% on average and concludes similarly at 53.41%. The substantial margin in the final accuracy highlights our model's adaptability and prowess in incremental task learning.

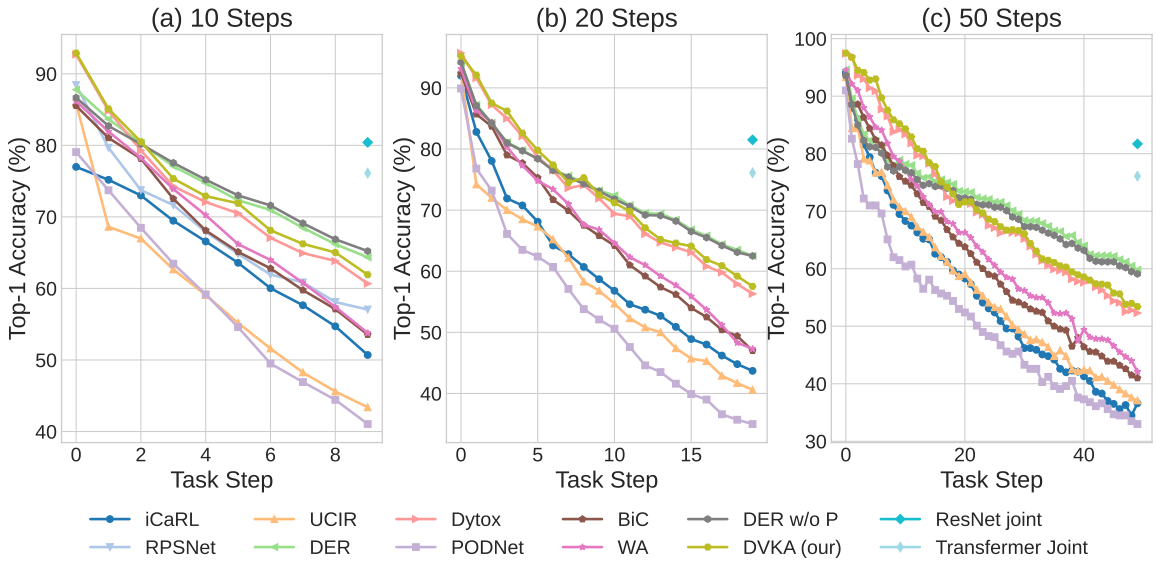

Fig. 4.2 Model performance evaluation on CIFAR100 in three class incremental learning scenarios: (a) 10 tasks (10 classes per task), (b) 20 tasks (5 classes per task), (c) 50 tasks (2 classes per task).

In addition, while DER w/o P [167] tops the charts without model size constraints, this huge model growth rate is often unacceptable, especially in real-world continual learning settings where task totals are uncertain.

Ablation Study (Table 4.3). In this section, we present the results of an ablation study undertaken to verify the effectiveness of our proposed components in the DVKA method for CIL. The first row of the table shows the baseline model that only applies the knowledge distillation (KD) technique. The average accuracy across the 50 tasks is 66.32%, while the final accuracy is 48.75%. This reflects the effect of KD alone, which serves as an acceptable baseline for the ablation study. In the second row, feature-level variational knowledge attention (FVKA) is introduced to the baseline model. It is evident that by incorporating the FVKA component, the average accuracy and the final task accuracy improve to 70.44% and 52.48%, respectively. This indicates the positive impact of FVKA, revealing that feature-level attention to the previously learned knowledge can significantly enhance the performance of the model. Finally, in the third row, token-level variational knowledge attention (TVKA) is included in the DVKA method. As a result, the model achieves an average accuracy of 71.21% and a last task accuracy of 53.41%. The improvement gained by adding the TVKA mechanism reveals the importance of token-level knowledge attention in boosting the model's performance. In summary, the ablation study demonstrates that both the

Table 4.3 Ablation study on 50-task incremental learning on the CIFAR100 dataset. KD denotes the knowledge distillation loss, FVKA denotes feature-level variational knowledge attention, and TVKA denotes the token-level variational knowledge distillation.

Methods	KD	FVKA	TVKA	avg	last
ours(DVKA)	✓			66.32	48.75
	✓	✓		70.44	52.48
	✓	✓	✓	71.21	53.41

FVKA and TVKA components contribute significantly to the effectiveness of the DVKA method for CIL.

Performance Across Tasks (Figure 4.2). As the number of tasks (as well as classes) increases, a general trend of decreasing accuracy across most methods can be observed, indicating it may solely be due to catastrophic forgetting. While the increasing number of classes makes the problem more challenging and exacerbates catastrophic forgetting, the DVKA method maintains a notably higher level of accuracy in such conditions, often surpassing the performance of other models. This highlights the effectiveness of managing information flow in the class-incremental transformer. While it can be observed that methods based on DER sometimes outperform ours, it is important to note that such methods add an entire model for each task during training, resulting in a significant increase in resource requirements due to numerous parameters. In contrast, our approach only employs the decoder, thereby enhancing the efficiency of the training process.

4.2 Summary

In this work, we introduced the dual variational knowledge attention (DVKA) mechanism for class incremental learning (CIL) within a Transformer-based framework. The DVKA, operating at both feature and token levels, balances the representation of all classes, mitigating catastrophic forgetting. Experimental results demonstrated significant improvements over traditional transformer models in CIL. Also, uniquely, in this research, mutual information and information bottleneck theory is applied to class incremental transformers. Furthermore, our approach reduces knowledge overfitting and catastrophic forgetting by utilizing the information filtering capabilities of attention maps to optimize the preservation of class representations from previous tasks. This

is achieved while maintaining a balance in the information conveyed by both current and former tasks. This research thus provides a promising foundation for further exploration and refinement of attention strategies in CIL models as well as opens up avenues for future research in understanding the mechanics of the transformer model in the context of CIL. It underscores the criticality of information flow in the process of model design to provide deeper insight into the architecture of effective learning systems.

Chapter 5

Federated Learning from Class-incremental Streaming Data

Having established a strong foundation in single-client continual learning advancements in the previous chapter, this chapter shifts focus towards enhancing data efficiency and mitigating catastrophic forgetting from a system-wide perspective in federated learning (FL). Motivated by the ideas introduced in Chapter 4 for enhancing information retention, this chapter explores how to further address the challenges posed by class-incremental learning (CIL) in FL environments. This chapter takes an information-centric approach, aiming to increase the information density of replay data by leveraging exemplar-condensed frameworks. By optimizing the management of streaming data, we ensure that FL systems can effectively handle class-incremental scenarios while preserving data privacy and computational efficiency.

Replay-based approaches have proven effective in mitigating catastrophic forgetting in federated continual learning (FCL) by rehearsing past samples stored in auxiliary memory, especially when clients encounter novel classes from streaming data. While several works have aimed to enhance exemplar selection strategies, the meta-information content of each sample in memory remains underexplored. This issue is particularly critical in FCL, where privacy concerns and resource constraints edge devices present additional challenges. Moreover, some methods attempt to increase the information density of each sample by summarizing data, but the heterogeneity of data in FCL can lead to heterogeneity at the information level of summarized data, thereby impacting the effectiveness of exemplar replay. To address these challenges, we propose **E**xemplar-**Co**ndensed fede**r**ated class-incremen**a**l **l**earning (ECoral) to distillate the knowledge from streaming data into more informative rehearsal exemplars by distilling the training characteristics of real images. By maintaining the consistency of training gradients

and the relationship to past tasks, these summarized exemplars better represent the streaming data than the original images. Additionally, by inter-client sharing of the disentanglement generative model, information-level heterogeneity of the summarized data is effectively reduced. Extensive experiments show that our ECoral outperforms several state-of-the-art methods and can be seamlessly integrated with many existing approaches to enhance their performance.

5.1 Exemplar-condensed Federated Class-incremental Learning

FL [100] enables the collaborative training of a global model across thousands of clients while keeping data decentralized. It effectively addresses the challenges presented by data silos, facilitating a cooperative learning environment without compromising individual privacy, and has been applied in various areas such as smart healthcare [107] and Internet-of-Things applications [106, 65]. However, traditional FL methods assume the application scenario is static, which conflicts with the realistic environment where the data of novel classes can emerge at any time [173, 98]. Simply fine-tuning the pre-trained model to the novel data results in *catastrophic forgetting* [84], where the model's performance significantly deteriorates on previously learned tasks. Moreover, the constraints of limited computational resources and data privacy concerns, which allow only a limited selection of previously learned data to be stored, make it impractical to retrain a model from scratch.

To address these challenges, recent work [28, 27, 177] has enabled the FL framework the ability to incrementally learn novel classes, known as Federated Class-Incremental Learning (FCIL). Among various strategies proposed for FCIL, rehearsal-based methods [28, 119] store and replay exemplars from prior tasks, stand out as effective approaches in mitigating the forgetting problem. However, these methods are limited by the constrained storage resources of edge devices and privacy concerns, allowing only a limited selection of data. This raises a crucial **question**: *How can a restricted dataset be utilized to encapsulate more information and effectively counteract the forgetting problem without compromising data privacy?*

A straightforward approach involves directly compressing data into a compact format. While this method efficiently condenses information, it is not necessarily ideal for training machine learning models. Even over-compression can lead to excessively complex information, hindering the model's ability to learn to distinguish decision boundaries effectively. Data condensation (DC) [154, 180] has emerged as a

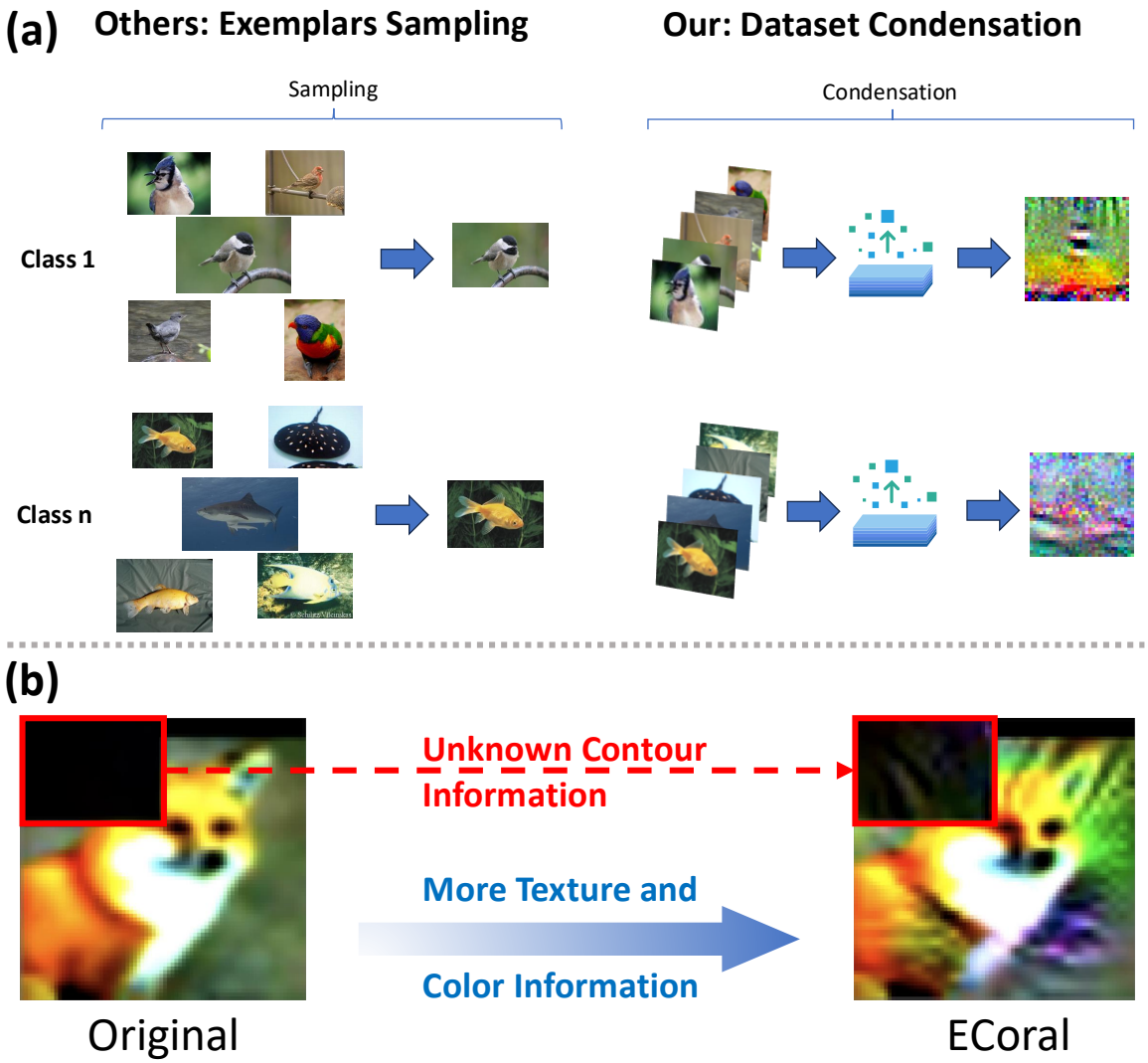

Fig. 5.1 Comparison between our approach and others. (a) Most Federated Continual Learning (FCL) methods use exemplars sampled from training data, whereas our method, ECoral, extracts and summarizes more comprehensive information, producing more informative exemplars. (b) ECoral can capture hidden contour information and enrich class-specific features such as texture and color, making the exemplars more representative of the overall data.

prominent solution that employs sophisticated approaches, such as distribution/feature matching [151, 179] or gradient/trajectory matching surrogate objectives [180, 16], to synthesize compact datasets. These condensed datasets are meticulously crafted to encapsulate the quintessential characteristics of the original, larger datasets. These condensed datasets are carefully crafted to capture the core characteristics of larger datasets and enable models trained on them to achieve performance comparable to that of models trained on the full datasets.

Most recently, various works [44, 61, 90, 164] have introduced DC into the FL framework, replacing the traditional exchange of model parameters with only a small portion of synthetic data. In contrast, in this work, we focus on leveraging DC to improve exemplar replay efficiency by maximizing the information stored in condensed datasets, thus better mitigating catastrophic forgetting. However, simply implementing such methods [99, 126, 168] into FL can suffer from a significant challenge we refer to as *meta-information heterogeneity*, where summarizing data from non-IID distributions across clients can disrupt optimization and degrade performance. Without proper handling, such heterogeneity can deviate from the optimization direction, limiting the ability to reduce catastrophic forgetting effectively.

To address these challenges, we propose the Exemplar-**Co**ndensed federated class-incremental **l**earning (ECoral) framework. ECoral enables the FL model to retain its performance on previously learned knowledge while incrementally learning new classes. It leverages a dual-distillation approach, where exemplars are distilled from the training dataset to capture more informative features, such as detailed contour information, enhanced texture, and richer color attributes, rather than relying on conventional sampling methods, as illustrated in Figure 5.1. Additionally, the model's prior knowledge is distilled from the model trained on previous tasks, ensuring that new learning does not compromise previously acquired knowledge. The main contributions of this work can be summarized as follows:

- A clear definition of the meta-information heterogeneity problem in federated continual learning (FCL) is proposed, identifying it as a significant challenge that affects model performance across different data sources.
- A novel method called ECoral is proposed that features a dual-distillation structure, specifically designed to address this issue by mitigating catastrophic forgetting in FCL.
- The proposed method in ECoral shifts from storing raw data to using condensed exemplars in memory, significantly enhancing privacy protection by abstracting stored information into less recognizable forms and addressing key privacy concerns in federated learning.

5.1.1 Related Work

Dataset Condensation

Dataset Condensation (DC) aims to reduce the size of a dataset while retaining enough representative information to allow machine learning models to perform comparably to training on the full dataset. Early works, such as those by Wang et al. [154] and Zhao et al. [180], framed this as a bi-level optimization problem. The objective is to create a smaller, condensed dataset that preserves the characteristics of the original data, which allows a model trained on this smaller dataset to achieve high performance. These techniques often focus on matching distributions, features, or gradients between the original and condensed data. This concept has been successfully integrated into the field of continual learning, where DC methods help mitigate catastrophic forgetting by compressing the data of previous tasks into small but representative memory sets [99, 46].

In the context of FL, several studies have explored the application of DC to reduce communication overhead by transmitting condensed datasets rather than full models or gradient updates between clients and the server [44, 61]. For example, Liu et al. [90] employed DC to address the challenges posed by heterogeneous data in FL environments, reducing both communication costs and data bias. These efforts suggest that DC is well-suited for handling the resource constraints and privacy considerations inherent in FL, as synthetic data can be shared while preserving local client privacy. Our approach builds on these foundations by introducing DC into federated class-incremental learning (FCIL) to enhance the informativeness of memory exemplars and mitigate catastrophic forgetting in non-IID data scenarios.

Federated Continual Learning

Federated Continual Learning (FCL) extends the traditional FL framework to dynamic environments, where new tasks or classes emerge incrementally over time. Unlike conventional FL, which assumes static data distributions, FCL must deal with continually evolving data and the challenges of catastrophic forgetting, non-IID data distribution, and communication constraints between clients and the server [173].

To address catastrophic forgetting, one of the most common approaches in FCL is rehearsal-based methods. These methods store a limited number of examples (exemplars) from previous tasks in memory and replay them to the model during training on new tasks, thereby helping to maintain the model's knowledge of older classes [28, 119]. For instance, GLFC [28] employs sample reconstruction techniques

to retain knowledge at the global level, while FedCIL [119] utilizes generative models to reconstruct past samples for rehearsal. Although effective in mitigating forgetting, these methods face significant challenges, especially in the context of federated settings where privacy and storage constraints are critical. TATGET [177], an exemplar-free distillation method, offers a privacy-preserving alternative by leveraging knowledge distillation from previously trained global models. It uses a generator to produce synthetic data that simulates the global distribution, reducing the reliance on real data storage while addressing catastrophic forgetting in highly non-IID settings.

A major limitation of existing rehearsal-based methods is the issue of class imbalance at the exemplar memory level. Since most clients in FL do not possess data for all classes, the memory stored at each client is often biased towards the locally available classes. This imbalance exacerbates catastrophic forgetting during memory replay, as the model tends to overfit to the classes that are well-represented in memory and underperform on underrepresented or unseen classes. Such imbalances are especially problematic in non-IID data settings, where the data distribution across clients can be highly skewed, further diminishing the effectiveness of memory replay. Even TARGET, despite its exemplar-free approach, relies on global knowledge distillation, which may struggle to capture the full diversity of class distributions across clients.

5.1.2 Preliminaries

Federated Class-incremental Learning. Federated Class-incremental Learning (FCIL) aims to collaboratively train a global model using streaming data that sequentially introduces new classes. In this context, a model training process consists of a series of sequential tasks $\mathcal{T} = \{\mathcal{T}^t\}_{t=1}^{T}$, where T denotes the total number of tasks. The system involves C local clients and a central server S_g. Each task comprises R global communication rounds (where $r = 1, \ldots, R$), and in each round r, a subset of the local participants is randomly selected for gradient aggregation. When the l-th client C_l is selected for a given global round in the t-th incremental task, it receives the latest global model $\theta^{r,t}$.

Drawing inspiration from online learning, each client maintains a fixed-size local memory $\mathcal{M}_l = \{(\mathbf{x}_{l,m}, \mathbf{y}_{l,m})\}_{m=1}^{M}$ of size M, storing examples from prior tasks for knowledge replay. In this work, we divide this memory into three parts: $\mathcal{M}_{\text{orig}}$, which holds original data sampled from the current task's training set; $\mathcal{M}_{\text{cond}}$, which stores condensed exemplars from prior tasks; and $\mathcal{M}_{\text{sum}}$, which saves summarizing data from the current task. At each iteration of the current task, a batch of samples $\mathbf{B}_m = \{(\mathbf{x}_{i,m}, \mathbf{y}^{i,m})\}_{i=1}^{B_m}$ is randomly drawn from the memory and jointly trained

alongside the current task data $\mathbf{B}_n = \{(\mathbf{x}_i^t, \mathbf{y}_i^t)\}_{i=1}^{B_n}$. Here, $B_m \leq M$ and B_n represent the mini-batch sizes of the replayed data and the current task data, respectively. The joint training objective is expressed as:

$$\theta_l^{r,t} = \arg\min_{\theta^{r,t}} \mathcal{L}(\theta^{r,t}; \mathbf{B}_n) + \lambda \mathcal{L}_m(\theta^{r,t}; \mathbf{B}_m), \tag{5.1}$$

where $\mathcal{L}$ and $\mathcal{L}_m$ are the loss functions for the current task data and memory data, respectively. λ is a hyper-parameter for regulation.

The client trains the global model $\theta^{r,t}$ on its own t-th incremental task data $\mathcal{D}_l^t \cup \mathcal{M}_l$, where $\mathcal{D}_l^t = \left\{\left(\mathbf{x}_{l,i}^t, \mathbf{y}_{l,i}^t\right)\right\}_{i=1}^{N_l^t} \subset \mathcal{T}^t$ represents the training data for new categories specific to the l-th client. The category distribution for the l-th client is denoted by $\mathbf{P}_l$. The distributions $\{\mathbf{P}_l\}_{l=1}^C$ are non-independent and identically distributed (non-IID). At the t-th incremental task, the label space $\mathcal{Y}_l^t \subseteq \mathcal{Y}^t$ for the l-th local client is a subset of $\mathcal{Y}^t = \bigcup_{l=1}^C \mathcal{Y}_l^t$, which includes $\mathcal{K}_l^t$ new categories ($\mathcal{K}_l^t \leq \mathcal{K}^t$), distinct from the previous categories $\mathcal{K}_l^p = \sum_{i=1}^{t-1} \mathcal{K}_l^i \subseteq \bigcup_{j=1}^{t-1} \mathcal{Y}_l^j$. After receiving $\theta^{r,t}$ and performing local training on the t-th incremental task, the l-th client obtains an updated model $\theta_l^{r,t}$. These locally updated models from selected clients are then uploaded to the central server S_g, where they are aggregated to form the new global model $\theta^{r+1,t}$ for the next round. The central server S_g subsequently distributes the updated parameters $\theta^{r+1,t}$ to the local clients for the following global round.

Client increment strategy. To better simulate a real-world federated continual learning application, we adopt the client increment strategy introduced in GLFC [28]. This strategy divides local participants into three dynamic groups for each incremental task: Old ($\mathcal{G}_o$), In-between ($\mathcal{G}_b$), and New ($\mathcal{G}_n$). The Old group ($\mathcal{G}_o$), consisting of G_o participants, only has access to data from classes introduced in previous tasks and does not receive any data for the new task. The In-between group ($\mathcal{G}_b$), with G_b members, works with both the new classes from the current task and the classes from the previous task. Finally, the New group ($\mathcal{G}_n$), comprising G_n newly added participants, focuses exclusively on data containing new classes from the current task.

The group compositions are dynamically updated with the progression of tasks. Specifically, the membership of the groups $\mathcal{G}_o, \mathcal{G}_b, \mathcal{G}_n$ is redefined randomly at each global round, and new participants are irregularly added to $\mathcal{G}_n$ as incremental tasks arrive. This incremental process gradually increases the total number of participants, $G = G_o + G_b + G_n$, as more tasks are introduced, closely mimicking the nature of streaming data in real-world FL applications.

Problem Definition

Forgetting in FCIL The primary objective of global model optimization at the t-th incremental task is to minimize the classification error across the current category set $\mathcal{K}_t$. However, when a new task arises, clients are often constrained by privacy restrictions and limited resources, allowing only restricted access to data from previous tasks. The category imbalance between old and new categories ($\mathcal{T}_l^t$ and $\mathcal{M}_l$) at the local level exacerbates this issue, leading to significant performance degradation during local training. This limitation frequently results in a notable decline in performance on earlier tasks, a phenomenon known as catastrophic forgetting. To mitigate catastrophic forgetting in the global model, our goal is to minimize the classification error on the current category set $\mathcal{K}_t$ while simultaneously preserving the knowledge of previously learned categories. The objective function is formally defined as:

$$\min_{\theta_t} \sum_{k \in \mathcal{K}_t} \sum_{i=1}^{N_k} \mathcal{L}\left(\mathbf{P}_l^t\left(\mathbf{x}_{l,i}^t; \theta_{r,t}\right), \mathbf{y}_{l,i}^t\right) \tag{5.2}$$

where $\mathcal{L}$ is a loss function that measures the classification error, and N_k is the number of samples in class k.

Meta-information Heterogeneity The condensation of data from non-IID sources inherently retains the non-IID characteristics at an information level, leading to what we define as the meta-information heterogeneity problem. GGiven that each client's original dataset $\mathcal{D}_l^t$ on task t-th is drawn from a unique distribution $\mathbf{P}_l(X, Y)$, the resulting condensed exemplar dataset $\mathcal{M}_{\text{cond}}$, optimized to represent $\mathcal{D}_l^t$, will also reflect this distinct distribution. Mathematically, this is expressed as $\mathbf{P}_l^{\text{cond}}(X, Y) \neq \mathbf{P}_{l'}^{\text{cond}}(X, Y)$ for some clients $l \neq l'$. The divergence in information content between these condensed datasets can be quantified using measures such as Kullback-Leibler (KL) divergence, where $\text{KL}(\mathcal{I}(\mathcal{T}_l^{\text{cond}}) \parallel \mathcal{I}(\mathcal{T}_{l''^{\text{cond}}})) > 0$ indicates non-identical information content across clients, thus confirming meta-information heterogeneity. Non-IID data has been shown to exacerbate catastrophic forgetting, as explored in [177], further complicating federated continual learning. Similarly, when these heterogeneous condensed datasets are used to train a global model $\theta^{r,t}$, the model's updates from different clients may conflict due to the diverse information content, leading to suboptimal performance. This degradation is reflected in the global loss function $\mathcal{L}(\theta)$, which generally increases compared to an IID scenario, expressed as $\Delta\mathcal{L} = \mathcal{L}_{\text{non-iid}}(\theta^{r,t}) - \mathcal{L}_{\text{iid}}(\theta^{r,t}) > 0$. Therefore, condensed datasets from non-IID sources introduce a meta-information heterogeneity

problem that adversely affects the global model's performance, mirroring the challenges posed by non-IID data in traditional FL.

5.1.3 Methodology

Online Exemplars Condensation

In edge devices within FCIL, where memory space is highly restricted, most existing approaches focus on efficient exemplar sampling strategies. Compared to these, our approach enhances the meta-knowledge capacity of each individual image, thereby increasing its information level, and also balances these improvements with memory efficiency. The balance and trade-offs are further explained in the following sections, highlighting comparisons and improvements over existing methods.

Adjustable Memory. Efficiently managing a fixed memory space for rehearsal typically requires sophisticated selection algorithms that continually update stored examples, as proposed in methods like [123, 3]. However, these approaches are not well-suited for our objective, which involves distilling meta-knowledge into exemplars from the entire local training dataset.

In contrast to conventional online learning methods like SSD [46], which employ a fixed exemplar position strategy for summarizing information, our approach addresses several key limitations. SSD assumes balanced class data in each batch and prior knowledge of the total number of classes, both of which are often unrealistic in real-world FL scenarios. Moreover, SSD allocates only a small fraction of memory to old exemplars. For instance, saving just one exemplar per class when training on a task with 100 total classes within a 100-exemplar space. This inefficient use of memory dedicates the majority of space to current data, hindering effective rehearsal. In an FL environment, this issue is further exacerbated by the non-IID nature of the data and class imbalance, as each client typically holds only a subset of the total class data. Additionally, in our scenario, since many clients do not participate in all tasks, SSD's strategy often results in memory slots being predominantly occupied by current task data by the end of the whole training process.

To overcome these challenges, we propose a dynamic memory allocation strategy that adjusts the exemplar space by calculating the required number of samples for each class at the beginning of each task. This approach reduces the number of prior exemplars to free up space, ensuring balanced storage of prior task exemplars while incorporating current task data samples. For example, if the first task includes 10 classes and the total memory space is 100 exemplars, 10 samples are initially stored

for each class. When the next task introduces 10 new classes, we reduce the number of exemplars for each previous class to 5 and allocate the remaining 50 slots to the new classes. This ensures a balanced distribution of memory across tasks and efficient rehearsal.

Meta-knowledge Condensation. In Federated Class Incremental Learning (FCIL), where each client is limited by constrained exemplar memory, our approach extends beyond simply managing exemplar selection. The aim is to enhance the information capacity of each image, thereby maximizing meta-knowledge to improve the effectiveness of rehearsal. The primary objective of Meta-Knowledge Condensation is to minimize the divergence between the memory set and the client's local task $\mathcal{T}_l$ training data distribution, resulting in an optimized memory set, $\hat{\mathcal{M}}_l$. We hypothesize that the global model trained on exemplars with condensed meta-knowledge can perform similarly as it trained on the whole local training dataset in t-th task. Drawing inspiration from dataset condensation methods [180], the process is implemented by sequentially distilling knowledge from mini-batches of real data into summarized exemplars corresponding to each class.

Considering the t-th task, let the condensed exemplars for a class k be denoted by $\mathcal{M}_k$, with a predefined size of m. The data points of the same class within the current mini-batch are represented by $\mathbf{B}_k$. The primary objective of the memory condensation process is to reduce the divergence between $\mathcal{M}_k$ and $\mathbf{B}_k$. In the DC works by [180], a novel and efficient metric is introduced, which quantifies the distance between the gradients of the training process on the same neural network when comparing condensed samples with original data points. By aligning the gradient-based model update metrics across the stream of data, the condensed samples are progressively refined to approximate the training performance of the entire dataset. The objective function for gradient matching is defined as:

$$\mathcal{L}_{\text{cond}} = f_{\text{dist}}\left(\nabla_{\theta^{r,t}}\mathcal{L}_{ce}(\theta^{r,t};\mathcal{M}_k), \nabla_{\theta^{r,t}}\mathcal{L}_{ce}(\theta^{r,t};\mathbf{B}_k)\right) \tag{5.3}$$

, where f_{dist} is a distance function.

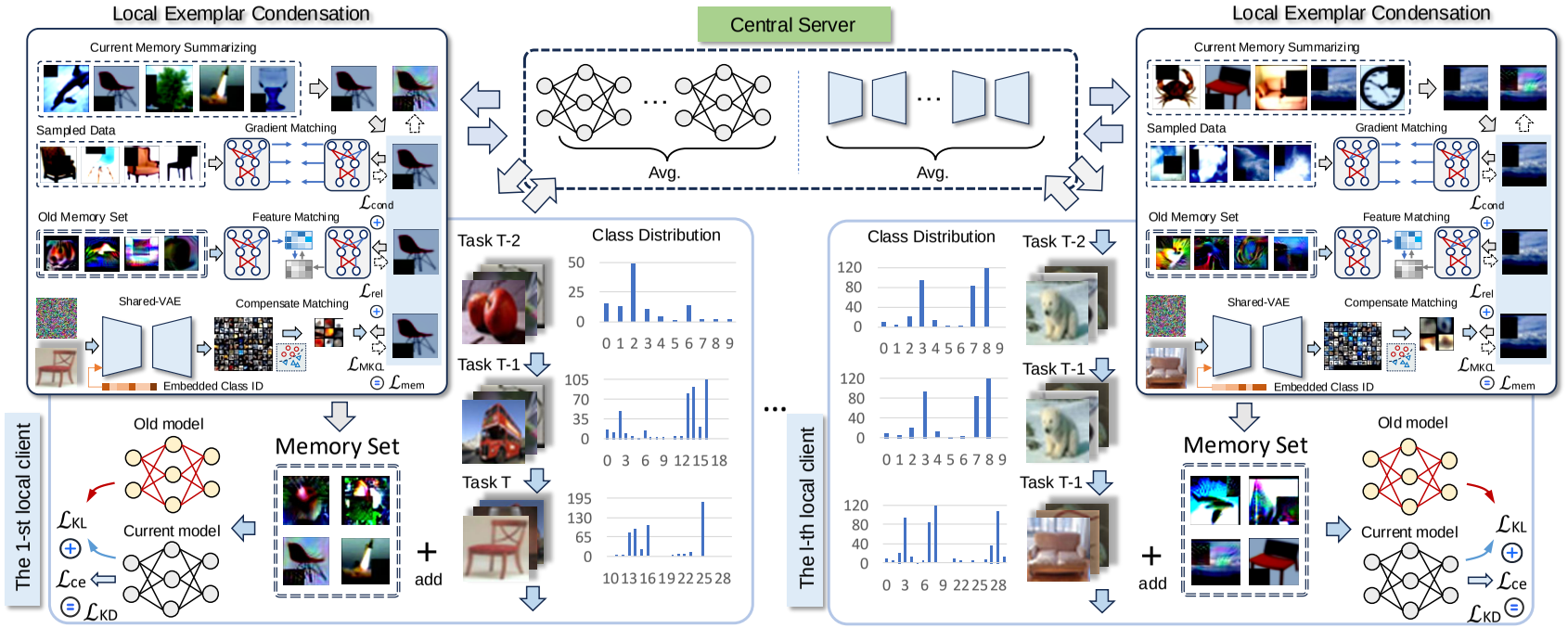

Fig. 5.2 Overview of our ECoral framework. Clients (from the 1st to the l-th) continuously learn from new class data sequences, each using a dual-distillation structure to mitigate catastrophic forgetting during local training. The local exemplar condensation process involves three key components: a gradient matching loss ($\mathcal{L}_{\text{cond}}$) to distill meta-information from the training dataset, a feature matching loss ($\mathcal{L}_{\text{rel}}$) to ensure consistency between the mean features of condensed samples and real images, and a compensation matching loss ($\mathcal{L}_{\text{MKCL}}$) to reduce the impact of meta-information heterogeneity problem, using disentangled features generated by a globally shared model (Shared-VAE). Additionally, a knowledge distillation loss ($\mathcal{L}_{\text{KD}}$) helps the model retain previously learned knowledge.

Current Knowledge Condensation. In conventional dataset condensation [180], gradient matching is often alternated with model updates. The primary goal is to simulate a comprehensive training process that allows for the matching of gradients across various training stages. During each training iteration, when a new batch of stream data $\mathbf{B}_n$ is introduced, the parameters ω of the condensation model are first randomly initialized and updated as follows:

$$\omega \leftarrow \omega - \eta \nabla_{\omega} \mathcal{L}_{ce}(\omega; \mathbf{B}_n) \tag{5.4}$$

, where η is the learning rate for the condensation model. At this stage, only real images participate in the model summarization update, which prevents knowledge leakage from the summarized samples.

However, in the context of continual learning, the number of classes is not fixed throughout the training process. To address this challenge, we draw inspiration from the SSD approach [46], which suggests re-initializing the model when new classes are introduced. To prevent the loss of valuable gradient information when the model's decision boundaries are constructed only for current classes, we employ dataset condensation across multiple dataset batches. This approach updates the model using both current stream data and stored real images in memory $\mathcal{M}_{\text{orig}}$:

$$\omega \leftarrow \omega - \eta \nabla_{\omega} \left(\mathcal{L}_{ce}(\omega; \mathbf{B}_n) + \mathcal{L}_{ce}(\omega; \mathcal{M}_{\text{orig}}) \right) \tag{5.5}$$

Building on the SSD approach, a relationship-matching strategy is implemented to further refine the condensation process. By using the extracted features of previously summarized samples as anchors, consistency is explicitly enforced between the mean features of condensed samples and real images, ensuring they maintain a consistent relationship with these anchors. The objective for relationship matching is defined as:

$$\begin{aligned} \mathcal{L}_{\text{rel}} = f_{\text{dist}} \left(\rho(\mathcal{M}_k, \mathcal{M}_{\text{cond}} \setminus \mathcal{M}_k, \omega), \rho(B_k, \mathcal{M}_{\text{cond}} \setminus \mathcal{M}_k, \omega) \right) \\ \rho(\mathcal{X}_1, \mathcal{X}_2, \omega) = f_{\text{dist}} \left(\frac{1}{|\mathcal{X}_1|} \sum_{x \in \mathcal{X}_1} \Phi(\omega; x), \Phi(\omega; \mathcal{X}_2) \right) \end{aligned} \tag{5.6}$$

where ρ represents the relationship calculation, $\mathcal{M}_{\text{cond}} \setminus \mathcal{M}_k$ refers to the condensed samples excluding $\mathcal{M}_k$, Φ denotes the feature extraction function. This relationship consistency helps establish a more balanced distribution of condensed samples within the memory.

As defined in Section 5.1.2, the condensed exemplars from non-iid data can still leave the meta-information heterogeneous which affect replay efficiency. Thus, this problem must be addressed from both data quantity shift and class shift problems in a privacy-preserving way.

Client-wise Feature Disentanglement. To address the feature and class skew problem, our goal is to enable each client not only to extract and generate features from the local dataset but also to generate features that are not visible in the local data but present in other clients' datasets, such as color, structure, and texture characteristics. By generating these features from random noise for classes unseen locally but present in other clients, the label skew problem can be reduced. The disentanglement approach [13, 56] is an efficient method for low-level feature extraction. As previously discussed, the heterogeneity of meta-knowledge arises from two main issues: feature skew and class skew. To tackle both challenges, we proposed the Client-wise Shared Conditional Variational Auto-Encoder (Shared-VAE) model. In this model, both the encoder $E_\phi(x) = q_\phi(z|x)$ and the decoder $D_\theta(z) = p_\theta(x|z)$ are updated through FL in each round. This allows the encoder to improve its ability to extract and refine hidden latent based on global knowledge while the decoder generates information that extends beyond the local data distribution by leveraging local latent information. At the beginning of the local model update, both the local encoder E_ϕ^l and decoder D_θ^l of the l-th client are updated with the latest global parameters E_ϕ^g and D_θ^g, respectively. The local training dataset is then directly used to generate a set of disentangled features if the class data is available locally, addressing the feature skew problem. Otherwise, for classes unseen locally, features are generated from random noise to address the label skew problem. The class ID is provided as a condition during feature generation, ensuring that the model generates class-specific features. The resulting feature set is represented as $\mathbf{H}$. After each round of local model updates, the Shared-VAE model is further refined using only the local training data.

Unbiased Representative Feature Prototypes However, a globally updated Shared-VAE tends to generate features that predominantly represent the majority distribution across all clients. For example, while most cats have fur, a smaller subset, like the Sphynx cat, do not. As a result, when these representations are used to guide data condensation, the condensed data may become biased towards this majority distribution, leading to skewed optimization. To address this limitation, we propose the use of unbiased representative feature prototypes. Specifically, each class k can be represented by multiple characteristic features, denoted as $\mathbf{H}_k = \{\mathbf{h}_i \mid y_i = k\}$, where y_i is the class label for feature $\mathbf{h}_i$. These features are grouped using an unsupervised

clustering method, FINCH [129], which is parameter-free and suitable for scenarios where the number of clients is uncertain.

First, the data are divided into groups by category, with one group of data for each class, denoted as G_k for class k. Then, FINCH is applied to cluster each group G_k to extract characteristic features for each class. After applying FINCH clustering, each group G_k is divided into V_k clusters, represented as $\mathcal{Q}_k = \{\mathcal{Q}_{k,j}\}_{j=1}^{V_k}$, where $\mathcal{Q}_{k,j}$ represents the j-th cluster for class k. Next, the average of all feature vectors in each cluster is directly computed to obtain the representative feature prototype for that cluster:

$$\mathbf{u}_{k,j} = \frac{1}{|\mathcal{Q}_{k,j}|} \sum_{\mathbf{h}_i \in \mathcal{Q}_{k,j}} \mathbf{h}_i \tag{5.7}$$

Here, $\mathbf{u}_{k,j}$ is the prototype for cluster j in class k, calculated by averaging all the features $\mathbf{h}_i$ within that cluster. Each class k is represented by the set of prototypes from all its clusters:

$$\mathbf{U}_k = \{\mathbf{u}_{k,j}\}_{j=1}^{V_k} \tag{5.8}$$

Finally, the overall set of representative prototypes for all classes is given by:

$$\mathcal{U} = \{\mathbf{U}_k\}_{k=1}^{C} \tag{5.9}$$

This ensures that each class k is represented by the averaged prototypes of its clusters, providing a balanced and representative set of features for further processing.

Meta-knowledge Contrastive Learning. By incorporating more class-specific characteristic features in the condensed exemplars while minimizing class-irrelevant features, we hypothesize that more discriminative representations can be created, resulting in clearer decision boundaries between different classes. To achieve this, for the current task's condensed data $\mathcal{M}_{\text{cond}}$, we ensure that the condensed data is similar to its corresponding class prototypes $\mathcal{P}^k$, and dissimilar to prototypes of other classes, represented as $\mathcal{N}^k = \bar{\mathbf{U}} - \mathcal{P}^k$.

The similarity between the embedding of a query sample $\mathbf{z}_i$ and the corresponding cluster prototypes $\mathbf{u} \in \bar{\mathbf{U}}$ is calculated using cosine similarity. For two feature vectors $\mathbf{z}_i$ and $\mathbf{u}$, the cosine similarity is defined as:

$$\text{sim}(\mathbf{z}_i, \mathbf{u}) = \frac{\mathbf{z}_i \cdot \mathbf{u}}{\|\mathbf{z}_i\| \times \|\mathbf{u}\| / \tau} \tag{5.10}$$

where $\mathbf{z}_i = f(x_i)$ is the embedding of sample x_i, and τ is a temperature parameter that controls sensitivity to similarities.

Therefore, the aim is to optimize the characters of each data sample to bring the local features of the current class closer to the global set of features for that class while distancing them from the characters of other classes. This optimization is intended to maintain a clear decision boundary, allowing the model to perform replay efficiently. By doing so, the feature distribution of this class can remain balanced across all clients in FL at the character's level. Thus, we propose Meta-knowledge Contrastive Learning (MKCL) for compensating matching, which contrasts cluster prototypes of the same class for each query sample against those of other classes with differing semantics. This approach naturally results in the following optimization objective term:

$$\mathcal{L}_{\mathrm{MKCL}} = -\log \frac{\sum_{\mathbf{u}\in\mathcal{P}^k} \mathbb{E}\left(\mathrm{sim}\left(\mathbf{z_i}, \mathbf{u}\right)\right)}{\sum_{\mathbf{u}\in\mathcal{P}^k} \mathbb{E}\left(\mathrm{sim}\left(\mathbf{z_i}, c\right)\right) + \sum_{\mathbf{u}\in\mathcal{N}^k} \mathbb{E}\left(\mathrm{sim}\left(\mathbf{z_i}, \mathbf{u}\right)\right)} \tag{5.11}$$

, finally, we can define as

$$\log\left(\sum_{\mathbf{u}\in\mathcal{N}^k} \mathbb{E}\left(\mathrm{sim}\left(\mathbf{z_i}, \mathbf{u}\right)\right)\right) - \log\left(\sum_{\mathbf{u}\in\mathcal{P}^k} \mathbb{E}\left(\mathrm{sim}\left(\mathbf{z_i}, \mathbf{u}\right)\right)\right) \tag{5.12}$$

.

Here, we can summarise the total objective of exemplar condensation as:

$$\mathcal{L}_{\mathrm{mem}} = \mathcal{L}_{\mathrm{cond}} + \mathcal{L}_{\mathrm{rel}} + \beta\mathcal{L}_{\mathrm{MKCL}} \tag{5.13}$$

, where β is weighting coefficients for meta-knowledge contrastive learning.

Prior Knowledge Supervision.

Another important part of the dual-distillation structure is knowledge distillation, which is used to transfer knowledge from previous tasks to new tasks, thereby mitigating the problem of catastrophic forgetting. We directly implement Knowledge Distillation [123, 84, 162] widely used in continual learning, which aims to leverage the soft output of a previously trained global model (named teacher model) as a regularization term for the training of the current task global model (named student model).

Mathematically, let $p_{t-1}(x)$ denote the probability distribution (softmax output) of the teacher model after training on the previous task $t-1$, and $p_t(x)$ denote the distribution of the student model being trained on the current task t. The goal is to minimize the following objective function:

$$\mathcal{L}_{\mathrm{KD}} = \mathcal{L}_{\mathrm{ce}} + \lambda \cdot \mathcal{L}_{\mathrm{KL}} \tag{5.14}$$
$$\text{, where } \mathcal{L}_{\mathrm{KL}} = KL(p_{t-1}(x) \parallel p_t(x))$$

Here, $\mathcal{L}_{ce}$ represents the task-specific cross-entropy loss for the current task t, such as cross-entropy loss, and $\mathcal{L}_{\mathrm{KL}}$ is the loss function of Kullback-Leibler divergence $\mathrm{KL}(p_{t-1}(x) \parallel p_t(x))$ between the teacher's and student's output distributions, which serves as the distillation loss. The parameter λ controls the balance between the task loss and the distillation loss. By minimizing this objective, the student model learns to perform well on the new task while retaining knowledge from previous tasks, thus reducing catastrophic forgetting.

5.1.4 Experiments

Implementation details.

In this work, all methods were implemented using PyTorch [109] and executed on a single NVIDIA RTX 4090 GPU paired with an AMD 7950X CPU, utilizing ResNet18 [53] as the backbone for feature extraction in our classification models. The FedAvg algorithm [100] was employed for global model aggregation. Each task involved training the model over $R = 50$ communication rounds, with each client performing $E = 30$ local epochs per round. A learning rate of 0.003 was used across all datasets to achieve optimal performance, and Stochastic Gradient Descent (SGD) served as the optimizer in all experiments. Unless specified otherwise, the constraint factor λ in the Elastic Weight Consolidation (EWC) method was set to 300. For knowledge distillation, the temperature parameter was set to 2, and the distillation loss weight λ was set to 3. Based on our experimental results and a thorough grid search, we set $\beta = 0.5$ for all experiments. To simulate non-IID data distributions, we partitioned the datasets among clients using the Latent Dirichlet Allocation (LDA) method, where the concentration parameter σ controls the degree of data skew; we varied σ to simulate different levels of non-IID data. For experiments on the CIFAR-100 and TinyImageNet datasets, we started with 20 clients and selected 10 clients per round, incrementing the total number of clients by 5 with each new task for the CIFAR-100 experiments with 10 and 20 tasks. However, due to data quantity limitations, only one client was incremented per new task in the 50-task CIFAR-100 experiment. Conversely, for the Caltech256 dataset, due to the limited number of samples per class, we started with five clients, selected 50% of the total clients in each round, and incremented the total number of clients by 1 with each new task. In all experiments, for each new task, 90%

of the existing clients $\mathcal{U}_o$ transitioned to the new task. To clearly illustrate the data distributions across clients under varying degrees of non-IID settings (controlled by σ), Figure 5.3 presents the data distribution for the final task in the CIFAR-100 dataset experiment with a total of 10 tasks.

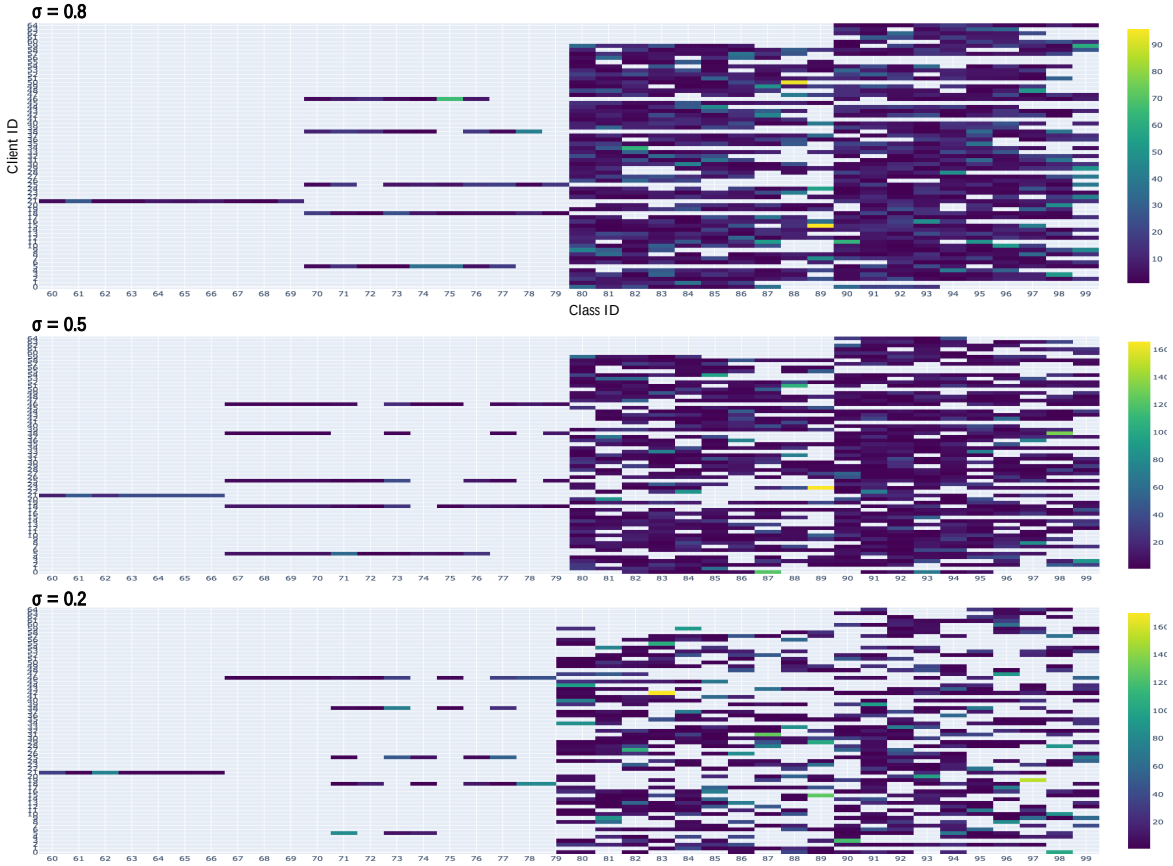

Fig. 5.3 Training data distribution of every client for CIFAR-100 on the final task, with non-IID levels σ of 0.2, 0.5, and 0.8 across a total of 10 tasks (each task containing 10 classes).

Datasets

We evaluated the framework on three widely used datasets for image classification tasks: CIFAR-100, TinyImageNet, and Caltech-256. The CIFAR-100 dataset [75] consists of 60,000 color images of size 32×32 pixels, distributed across 100 classes with 600 images per class. We allocated an exemplar memory space of 100 for each client and partitioned the dataset into tasks of varying sizes: 10 tasks with 10 classes per task, 20 tasks with five classes per task, and 50 tasks with two classes per task. The TinyImageNet dataset [79] contains 100,000 images of size 64×64 pixels, distributed

over 200 classes with 500 images per class. We allocated an exemplar memory space of 200 for each client and evaluated our method using 10 tasks, each comprising 20 classes. The Caltech-256 dataset [45] includes 30,607 images across 257 classes; we removed the "background" class, originally used for validation, resulting in a total of 256 classes. Each image was resized to 64×64 pixels due to computational resource constraints. We allocated an exemplar memory space of 256 for each client and evaluated our method using 16 tasks with 16 classes per task.

Baselines

Replay maintains an exemplar memory at each client to store and replay a subset of previous data, mitigating catastrophic forgetting in FL settings by randomly selecting samples from the training data and incrementally adding new classes with each new task.

iCaRL [123] proposes an incremental learning method that integrates representation learning with a nearest-mean-of-exemplars classifier, utilizing a fixed memory budget to store exemplars from previous classes, thereby mitigating catastrophic forgetting while learning new classes.

LwF [84] enables a neural network to learn new tasks without forgetting previously learned tasks by using knowledge distillation to preserve the model's responses on old tasks during training, all without requiring access to the original data from the old tasks.

EWC [74] mitigates catastrophic forgetting in neural networks by adding a regularization term that penalizes changes to important weights, identified using the Fisher information matrix. This approach allows the model to learn new tasks while preserving performance on previously learned tasks without requiring access to old data.

BiC [162] tackles the bias toward new classes in class-incremental learning by introducing a two-stage training framework that adds a bias correction layer, which is fine-tuned using a small validation set to adjust the decision boundary between old and new classes, effectively reducing bias and improving classification accuracy.

TARGET [177] addresses federated class-continual learning by introducing an exemplar-free distillation method that utilizes global prototypes to preserve knowledge of previous classes without storing or generating data, effectively mitigating catastrophic forgetting in a privacy-preserving manner.

FedCIL [119] addresses federated class-incremental learning by introducing a global knowledge distillation method to preserve knowledge of old classes and a class-balanced

sampling strategy to mitigate class imbalance, enabling clients to incrementally learn new classes while reducing catastrophic forgetting.

Evaluation Metrics

Accuracy ($\mathcal{A}$): This metric computes the accuracy for a given task. We report the final accuracy after all tasks have been trained as $\mathcal{A}_{last}$, and the average accuracy across the last round of every task as $\mathcal{A}_{avg}$.
Averaged Incremental Accuracy $\mathcal{A}^{incre}$ [123]: This metric calculates the average accuracy after the completion of each task, emphasizing the model's performance throughout the incremental learning process. We denote the overall averaged accuracy across all tasks as $\mathcal{A}^{incre}_{avg}$, and the accuracy after the last task as $\mathcal{A}^{incre}_{last}$.
Accuracy A ($\mathcal{A}^a$) [26]: As defined by Díaz-Rodríguez et al., this metric differs from standard accuracy by assigning equal weight to the accuracy of each task, regardless of the number of samples. For instance, in a scenario where Task 1 has 50,000 images and Task 2 has 1,000 images, standard accuracy would give more weight to Task 1, whereas Accuracy A treats both tasks equally. We denote the overall averaged Accuracy A as $\mathcal{A}^a_{avg}$ and the Accuracy A after the last task as $\mathcal{A}^a_{last}$.
Backward Transfer (BwT) [95]: This metric measures the influence that learning a new task has on the performance of previously learned tasks. A positive BwT indicates an improvement in past tasks after learning new ones, while a negative BwT signifies forgetting. It is denoted as BwT.
Forward Transfer (FwT) [95]: This metric assesses the influence that learning a new task has on the performance of future tasks. Positive forward transfer implies that learning prior tasks benefits future tasks, enhancing initial performance. It is denoted as FwT.
Remembering [26]: This metric calculates the degree of retention for previous tasks as part of the backward transfer process. It quantifies how well the model remembers earlier tasks after learning new ones.
Forgetting [18]: This metric measures the average amount of forgetting across all tasks, helping to quantify how much information is lost as new tasks are learned. It is calculated by comparing the maximum performance on a task with its performance after learning subsequent tasks.

5.1.5 Results

ECoral efficiently mitigates forgetting. As shown in Table 5.1 with $\sigma = 0.5$, and extended further in Table 5.2 with more complex datasets, ECoral consistently outperforms other baseline methods across multiple evaluation metrics. For the CIFAR-100 dataset at $\sigma = 0.5$, ECoral achieves an impressive average accuracy ($\mathcal{A}_{avg}$) of 49.17% and a last-task accuracy ($\mathcal{A}_{last}$) of 27.97%, significantly surpassing iCaRL, the closest baseline, which only reaches an $\mathcal{A}_{avg}$ of 43.85% and $\mathcal{A}_{last}$ of 21.76%. This underscores ECoral's superior ability to mitigate catastrophic forgetting and maintain strong performance across tasks compared to other methods. In addition, ECoral demonstrates remarkable storage efficiency. For the CIFAR-100 dataset, it stores 100 condensed exemplars per client (one per class), with each exemplar represented as a 32×32 RGB image (approximately 1 KB). This results in a total memory footprint of only 100 KB per client, which is negligible for edge devices (e.g., a Raspberry Pi 4 has 4–8 GB RAM). Compared to raw data storage (600 images per class), ECoral reduces storage costs by 600× while simultaneously improving replay efficiency.

When examining more complex datasets like Tiny-ImageNet and Caltech-256, ECoral continues to demonstrate its effectiveness. On Tiny-ImageNet, ECoral achieves the best average accuracy ($\mathcal{A}_{avg}$ of 38.78%), outperforming iCaRL, which records 36.46%. Though iCaRL slightly surpasses ECoral in last-task accuracy ($\mathcal{A}_{last}$ 22.80% vs. 20.88%), EECoral'soverall performance balance across tasks highlights its ability to manage incremental learning effectively. A similar trend is seen with the Caltech-256 dataset, where ECoral achieves the highest average accuracy ($\mathcal{A}_{avg}$ of 31.06%) but slightly trails iCaRL in last-task accuracy (21.66% vs. 23.52%). This small gap in last-task accuracy can be explained by iCaRL's emphasis on incremental learning for recent tasks, whereas ECoral focuses on holistic task balance, trading off minor performance losses on the last task for superior average accuracy across all tasks.

Despite these few instances where iCaRL outperforms ECoral in last-task accuracy, ECoral consistently excels in overall task performance, demonstrating its strength in mitigating forgetting across all tasks. These results highlight ECoral as an effective approach for continual learning, particularly in scenarios where sustained performance across many tasks is critical.

ECoral balance the knowledge learned in each task. As illustrated in Figure 5.4, the CIFAR-100 experiments with $\sigma = 0.5$ provide a detailed analysis of ECoral's performance as tasks are progressively introduced, reflecting a typical continual learning scenario. In the early stage, for the first task (T0), all methods show strong performance, with Target and iCaRL slightly outperforming ECoral in the initial steps. Nonetheless,

ECoral remains highly competitive, demonstrating a robust ability to learn and adapt right from the start.

As additional tasks are introduced (from T1 to T9), the common challenge of catastrophic forgetting becomes more evident, with all methods experiencing a gradual decline in performance. ECoral, however, distinguishes itself by maintaining a more stable and balanced performance compared to baselines like BiC, FedCIL, and Replay, which show more pronounced declines as tasks are added. ECoral's ability to sustain balanced performance across tasks allows it to mitigate forgetting more effectively, maintaining competitive results even as the complexity of the continual learning setting increases.

In the later stages (T8 and T9), while ECoral is occasionally outperformed by Target and iCaRL in last-task accuracy, this is primarily due to the incremental learning emphasis of those methods, which prioritize performance on recent tasks. However, ECoral's superior average accuracy across all tasks underscores its strength in balancing performance over the entire task sequence. This approach ensures that ECoral not only excels in the earlier tasks but also performs well across a wide range of tasks, making it more resilient to the long-term challenges of catastrophic forgetting.

Overall, the results demonstrate ECoral's effectiveness in preserving knowledge across multiple tasks, delivering superior stability and resilience compared to other baseline methods under the CIFAR-100 dataset with $\sigma = 0.5$.

Table 5.1 Results on CIFAR100 with 10 tasks with non-IID levels σ of 0.2, 0.5, and 0.8 across a total of 10 tasks (each task containing 10 classes), evaluated across three metrics: $\mathcal{A}$, $\mathcal{A}^{incre}$, and $\mathcal{A}^{a}$ for both the last task and overall performance. Δ represents the absolute difference compared to the results of our ECoral method.
The results row of our ECoral is highlighted in . Improvements are marked in green, and declines are marked in red.

	$\sigma = 0.2$				$\sigma = 0.5$				$\sigma = 0.8$			
Methods	$\mathcal{A}_{avg}$	Δ	$\mathcal{A}_{last}$	Δ	$\mathcal{A}_{avg}$	Δ	$\mathcal{A}_{last}$	Δ	$\mathcal{A}_{avg}$	Δ	$\mathcal{A}_{last}$	Δ
Replay	32.82	10.70	16.63	12.09	36.72	12.45	18.72	9.25	37.49	11.85	18.05	13.34
iCaRL	31.97	11.55	20.46	8.26	43.85	5.32	21.76	6.21	44.97	4.37	23.58	7.81
EWC	31.73	11.79	14.03	14.69	36.55	12.62	16.56	11.41	38.59	10.75	18.42	12.97
BiC	28.27	15.25	13.08	15.64	33.45	15.72	17.49	10.48	35.62	13.72	16.31	15.08
LwF	36.64	6.88	16.93	11.79	43.13	6.04	21.38	6.59	44.69	4.65	22.90	8.49
TARGET	30.60	12.92	9.42	19.30	41.51	7.66	16.71	11.26	46.05	3.29	20.83	10.56
FedCIL	30.14	13.38	13.62	15.10	34.88	14.29	15.56	12.41	34.98	14.36	16.05	15.34
ECoral	43.52	–	28.72	–	49.17	–	27.97	–	49.34	–	31.39	–
Methods	$\mathcal{A}^{incre}_{avg}$	Δ	$\mathcal{A}^{incre}_{last}$	Δ	$\mathcal{A}^{incre}_{avg}$	Δ	$\mathcal{A}^{incre}_{last}$	Δ	$\mathcal{A}^{incre}_{avg}$	Δ	$\mathcal{A}^{incre}_{last}$	Δ
Replay	45.35	7.61	36.72	12.45	51.01	9.48	32.82	10.70	52.88	7.01	37.49	11.85
iCaRL	46.45	6.51	43.85	5.32	57.37	3.12	36.20	7.32	57.43	2.46	44.97	4.37
EWC	46.32	6.64	36.55	12.62	52.54	7.95	31.73	11.79	55.35	4.54	38.59	10.75
BiC	41.57	11.39	33.45	15.72	48.99	11.50	28.27	15.25	51.33	8.56	35.62	13.72
LwF	49.25	3.71	43.13	6.04	57.18	3.31	36.64	6.88	59.89	–	44.69	4.65
TARGET	45.50	7.46	41.51	7.66	57.47	3.02	30.60	12.92	61.42	1.53	46.05	3.29
FedCIL	44.25	8.71	34.88	14.29	49.84	10.65	30.14	13.38	50.92	8.97	34.98	14.36
ECoral	52.96	–	49.17	–	60.49	–	43.52	–	59.89	–	49.34	–
Methods	$\mathcal{A}^{a}_{avg}$	Δ	$\mathcal{A}^{a}_{last}$	Δ	$\mathcal{A}^{a}_{avg}$	Δ	$\mathcal{A}^{a}_{last}$	Δ	$\mathcal{A}^{a}_{avg}$	Δ	$\mathcal{A}^{a}_{last}$	Δ
Replay	40.39	9.19	28.23	13.74	45.22	11.16	25.26	12.20	46.66	9.47	28.31	14.25
iCaRL	42.68	6.90	35.33	6.64	52.37	4.01	29.71	7.75	52.97	3.16	36.97	5.59
EWC	40.27	9.31	27.02	14.95	46.09	10.29	23.19	14.27	48.53	7.60	28.65	13.91
BiC	36.02	13.56	24.46	17.51	42.44	13.94	20.52	16.94	44.87	11.26	26.36	16.20
LwF	44.25	5.33	34.40	7.57	51.85	4.53	29.03	8.43	54.06	2.07	35.32	7.24
TARGET	39.65	9.93	31.13	10.84	51.88	4.50	21.54	15.92	55.98	0.15	36.12	6.44
FedCIL	38.43	11.15	25.84	16.13	43.91	12.47	21.84	15.62	44.35	11.78	25.60	16.96
ECoral	49.58	–	41.97	–	56.38	–	37.46	–	56.13	–	42.56	–

Table 5.2 Results on Tiny-ImageNet with 10 tasks (10 classes per task) and on Caltech-256 with 16 tasks (16 classes per task), both under a Non-IID setting with $\sigma = 0.5$. The results row of our ECoral is highlighted in . The best result is highlighted with , and the second-best result is highlighted with .

	Tiny-Imagenet (10 Tasks)						Caltech-256 (16 Tasks)					
Methods	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$	$\mathcal{A}^{incre}_{avg}$	$\mathcal{A}^{incre}_{last}$	$\mathcal{A}^{a}_{avg}$	$\mathcal{A}^{a}_{last}$	$\mathcal{A}_{avg}$	$\mathcal{A}_{last}$	$\mathcal{A}^{incre}_{avg}$	$\mathcal{A}^{incre}_{last}$	$\mathcal{A}^{a}_{avg}$	$\mathcal{A}^{a}_{last}$
Replay	35.73	18.73	46.04	33.51	41.72	26.17	24.24	20.92	31.46	24.24	28.20	20.79
iCaRL	36.46	22.80	44.92	35.40	41.56	29.69	26.72	23.52	33.75	26.72	30.54	23.11
EWC	31.10	13.10	45.23	30.14	39.19	21.49	20.68	11.08	34.17	20.68	28.12	13.67
BiC	35.88	20.46	46.31	34.51	42.07	27.57	29.26	22.70	37.78	29.26	33.84	24.51
LwF	35.76	16.78	47.51	33.64	42.53	25.54	23.20	12.88	35.51	23.20	30.06	16.67
TARGET	27.00	10.49	40.83	25.46	34.63	16.90	20.46	10.31	35.28	20.46	27.83	12.15
FedCIL	30.18	12.59	44.67	29.11	38.43	20.13	18.53	10.53	31.16	18.53	25.59	11.92
ECoral	38.78	20.88	48.46	37.80	44.94	31.24	31.06	21.66	40.64	31.06	36.23	25.11

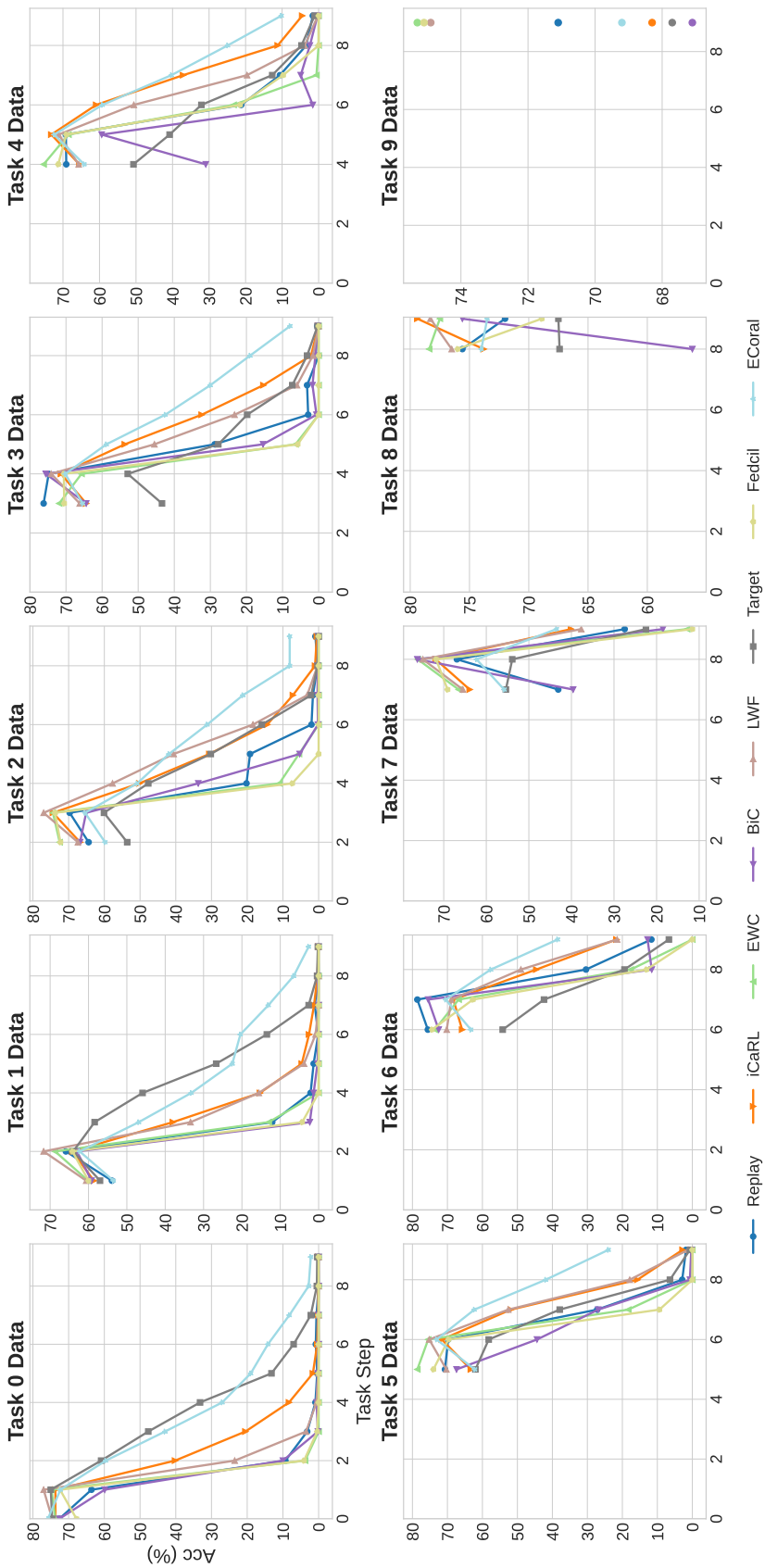

Fig. 5.4 Performance evaluation on CIFAR100 under a Non-IID setting with $\sigma = 0.5$, across 10 tasks. The final accuracy $\mathcal{A}$ (%) for each learned task is reported after the completion of each task.

Ecoral is keep effective at different levels of non-iid. The experimental results, shown in Table 5.1, assessed using three key metrics: Accuracy ($\mathcal{A}$), Averaged Incremental Accuracy ($\mathcal{A}^{incre}$), and Accuracy A ($\mathcal{A}^{a}$), highlight the strong performance of ECoral compared to baseline methods across various non-IID data distributions. ECoral consistently shows higher final accuracy ($\mathcal{A}_{last}$) and average accuracy ($\mathcal{A}_{avg}$) across all tasks, particularly excelling in more challenging non-IID settings such as $\sigma = 0.2$, which represents the most highly skewed data scenario. In this difficult setting, ECoral demonstrates robustness by effectively retaining knowledge from previous tasks and adapting to new ones, significantly reducing catastrophic forgetting. For instance, at $\sigma = 0.2$, ECoral shows marked improvements in both $\mathcal{A}_{last}$ and $\mathcal{A}_{avg}$ compared to other methods. Furthermore, ECoral performs exceptionally well in Averaged Incremental Accuracy ($\mathcal{A}^{incre}$), maintaining higher accuracy throughout the incremental learning process. Even as the non-IID severity decreases (e.g., $\sigma = 0.5$ and $\sigma = 0.8$), ECoral continues to outperform competing approaches, demonstrating adaptability across different levels of data skew. Additionally, ECoral achieves high scores in Accuracy A ($\mathcal{A}^{a}$), which balances performance across tasks irrespective of sample size. Notably, even in the highly skewed $\sigma = 0.2$ scenario, ECoral achieves an $\mathcal{A}^{a}_{avg}$ of 49.58%, outperforming the next best method by a significant margin, further underscoring its robustness and fairness across tasks. Overall, these results confirm ECoral's effectiveness in mitigating catastrophic forgetting and improving task performance in federated continual learning, especially in environments with highly non-IID data distributions.

ECoral achieves superior performance across multiple evaluation metrics. Beyond its strong accuracy, as illustrated in Figure 5.5, ECoral also excels across other important metrics such as Backward Transfer (BwT), Forward Transfer (FwT), Forgetting, and Remembering. ECoral shows lower negative BwT compared to several baseline methods, managing to limit the detrimental effects on previously learned tasks as new tasks are introduced. While iCaRL and Target demonstrate slightly better backward transfer in earlier stages, ECoral avoids the significant negative transfer experienced by methods like FedCIL and BiC, which exhibit steep performance declines when more tasks are added. This demonstrates that ECoral effectively preserves prior knowledge, a key requirement for successful continual learning.

Moreover, ECoral displays strong forward transfer (FwT), suggesting that learning earlier tasks contributes positively to the performance on future tasks. This is especially important in non-IID settings where task distributions may vary significantly. Compared to FedCIL and BiC, which struggle with forward transfer, ECoral leverages knowledge from earlier tasks to improve initial performance on new tasks.

Additionally, ECoral demonstrates significantly lower forgetting, particularly in the later stages of task learning, indicating that it retains information long-term and resists the catastrophic forgetting seen in methods like FedCIL and BiC.

In terms of Remembering, ECoral shows competitive performance, ensuring that it retains knowledge of previously learned tasks effectively. While iCaRL slightly outperforms ECoral in specific cases, ECoral's overall balance between retention, forward transfer, and reduced forgetting ensures it can handle the trade-offs inherent in continual learning, providing robust long-term performance across multiple tasks.

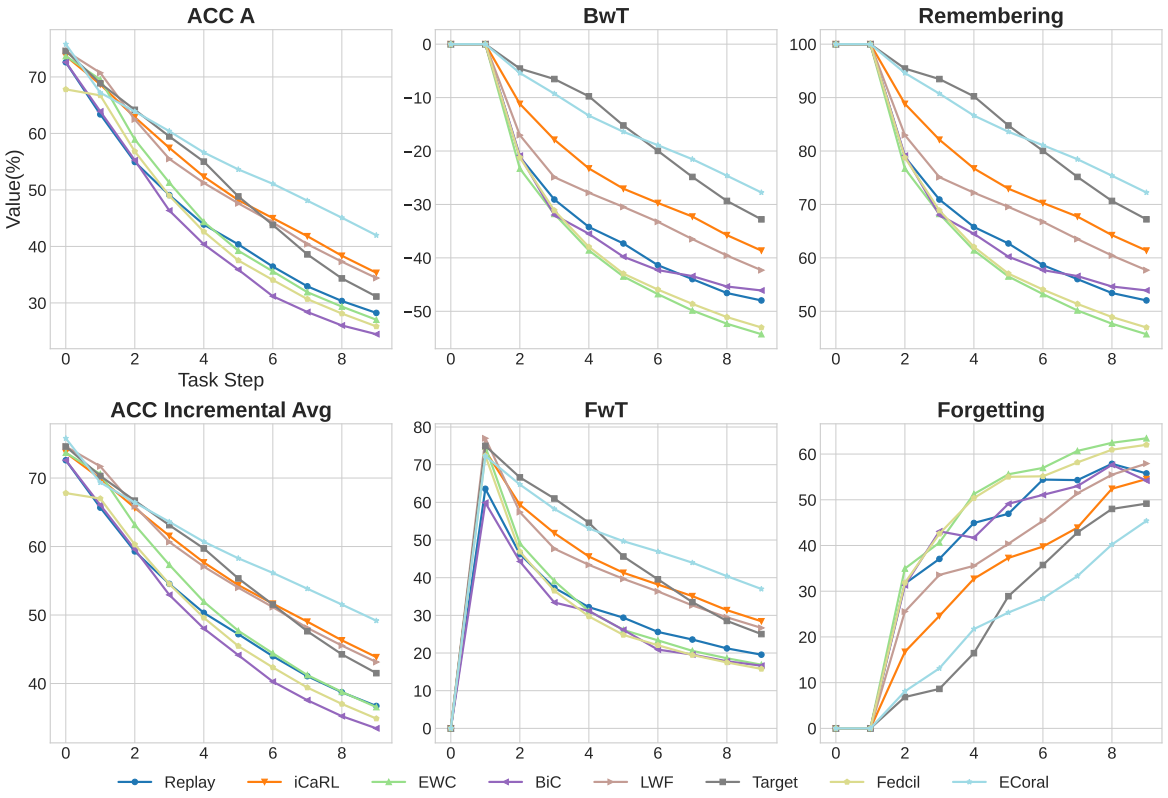

Fig. 5.5 Evaluation of multiple metrics (%) on CIFAR100 under a Non-IID setting with $\sigma = 0.5$, across a total of 10 tasks.

ECoral can perform consistently in a long-term training task. The results shown in Figure 5.6 demonstrate that ECoral significantly outperforms baseline methods in both 20-task and 50-task continual learning setups, particularly in terms of average and final accuracy. In the 20-task setup, ECoral consistently maintains higher average accuracy compared to competing methods, achieving 63.60% for the average accuracy across tasks, while methods such as BiC (51.4%) and Replay (55.27%) fall behind. By the final task, ECoral still retains a strong accuracy of 59.00%, whereas BiC (39.6%) and FedCIL (38.25%) exhibit substantial declines. In the more challenging 50-task

setup, ECoral continues to lead, with an average accuracy of 91.00%, outperforming BiC (90.50%) and iCaRL (90.00%). As the number of tasks increases, ECoral maintains a notable performance edge, with a final accuracy of 64.40% by the last task, significantly higher than BiC (36.90%) and Replay (38.40%). These results highlight ECoral's superior ability to retain knowledge and perform consistently across both short- and long-term continual learning setups, emphasizing its robustness and scalability in FL environments.

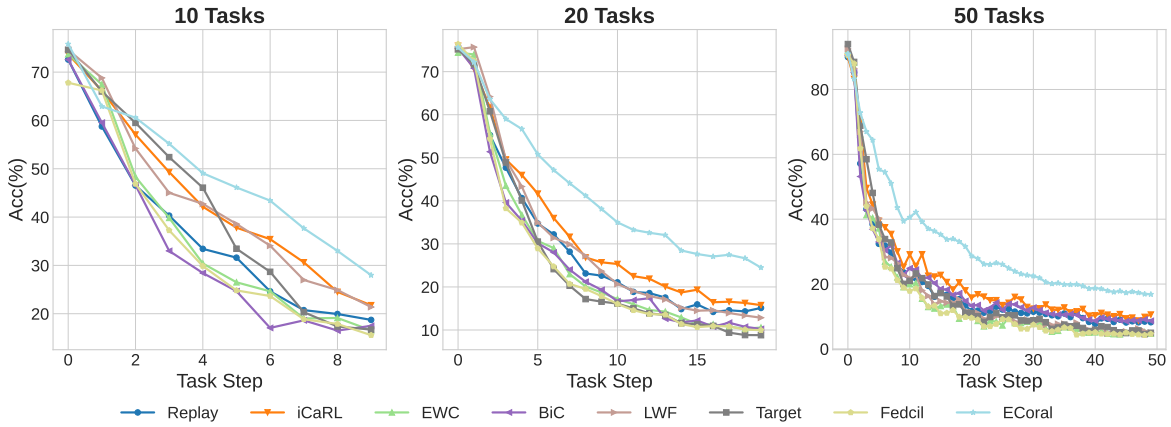

Fig. 5.6 Performance evaluation on CIFAR100 under a Non-IID setting with $\sigma = 0.5$. The final accuracy $\mathcal{A}$ (%) is reported after learning each task. The left plot shows results with 10 steps (10 classes per task), the middle with 20 steps (5 classes per task), and the right with 50 steps (2 classes per task).

ECoral is user privacy friendly. This work addresses user privacy concerns in two key areas. First, we ensure that the Shared-VAE model cannot regenerate raw data from other clients, and that the generated data is not semantically interpretable by humans, preserving data privacy at the FL level. Second, we design the condensed data to be recognizable only as belonging to the cclient'slocal classes, yet indecipherable by humans, to prevent privacy breaches during memory replay.

To illustrate this, Figure 5.7 displays six randomly selected samples of disentangled features generated by Shared-VAE, alongside six condensed exemplars. As seen in the left panel, the disentangled features are highly abstract, containing only basic information such as colors and vague outlines, rendering them indistinguishable to humans. In the right panel, even though the first row of condensed exemplars can be roughly associated with a certain category, the details remain unclear, ensuring that no specific class details from client data are exposed. Importantly, these condensed exemplars are derived solely from the user's local training data, without incorporating

any sensitive information from other clients. The second row of exemplars is entirely unrecognizable, further enhancing in-memory data-level privacy protection.

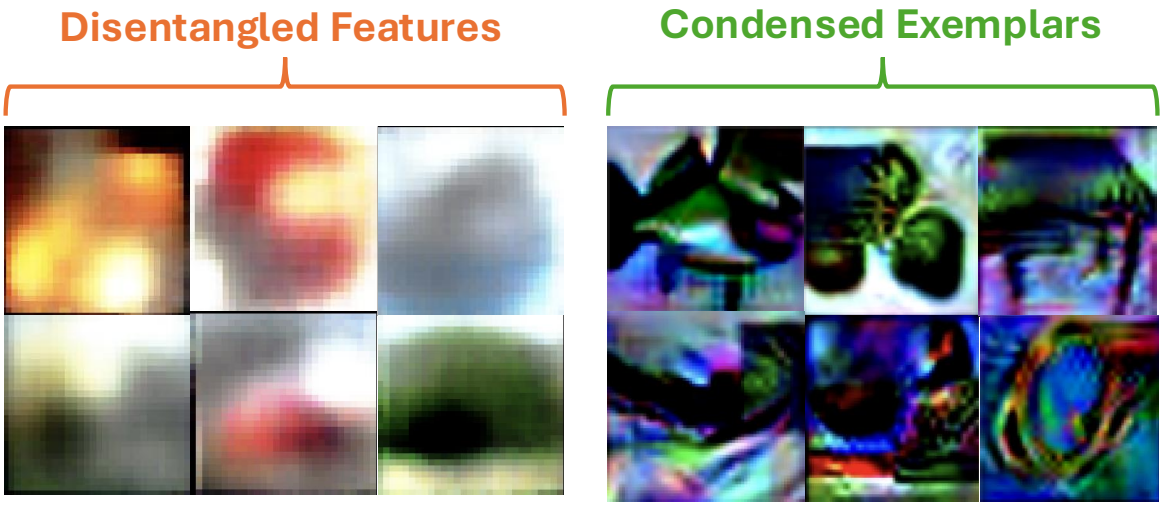

Fig. 5.7 Examples of disentangled features from Shared-VAE and final condensed exemplars.

ECoral saving and computation cost. The ECoral approach employs data condensation via gradient matching and meta-contrastive learning, which introduces approximately a 15% overhead during local training. In return, compressing the data into compact exemplars reduces rehearsal time by about 40%. For instance, if a standard forward and backward pass on the full dataset requires 10 GFLOPs, the additional computation for condensation may increase the cost to around 11.5 GFLOPs, while the overall rehearsal process becomes significantly faster. This method is particularly beneficial for memory-constrained devices because it drastically reduces storage requirements while only modestly increasing computational load.

Ablation Study.

The ablation study in Table 5.3 shows the impact of each component in ECoral. Adding Adjustable Memory gives a modest improvement, emphasizing the importance of efficient memory allocation. Gradient Matching (+3.53%) helps align exemplars with new data, improving task transfer. Feature Matching (+4.20%) ensures consistency between real images and exemplars, while Compensation Matching (+5.01%) addresses meta-knowledge heterogeneity, crucial for non-IID data. The full ECoral method yields the best performance (+12.45% average, +9.25% on the last task), demonstrating that each component contributes to mitigating catastrophic forgetting and improving performance across tasks.

Method	**A**	**G**	**F**	**C**	**K**	**Avg**	**Δ**	**Last**	**Δ**
Replay						36.72	-	18.72	-
	✓					38.12	1.40	20.12	1.40
	✓	✓				40.25	3.53	21.56	2.84
ECoral	✓	✓	✓			40.92	4.20	22.33	3.61
	✓	✓	✓	✓		41.73	5.01	24.14	5.42
	✓	✓	✓	✓	✓	49.17	12.45	27.97	9.25

Table 5.3 Ablation study of ECoral on CIFAR100 with 10 tasks and non-IID level $\sigma = 0.5$. Improvement compared to the Replay baseline is marked as Δ. Components: A (Adjustable Memory), G (Gradient Matching), F (Feature Matching), C (Compensate Matching).

5.2 Summary

In this chapter, we propose the ECoral framework, which successfully addresses critical challenges in Federated Class-Incremental Learning (FCIL) by enhancing memory efficiency and improving resilience against catastrophic forgetting. The combination of exemplar condensation and meta-knowledge contrastive learning allows the model to store more informative and privacy-preserving condensed exemplars, while client-wise feature disentanglement mitigates the negative effects of data heterogeneity. This approach ensures consistent performance across highly non-IID environments, making it well-suited for real-world FL applications where data privacy and resource constraints are key concerns. However, we observe that ECoral's advantage decreases when applied to more complex datasets. Future work will focus on strengthening ECoral's performance in these complex scenarios, ensuring robustness and scalability across diverse real-world data challenges.

Chapter 6

Federated Learning from Domain-incremental Streaming Data

In continuation of our exploration of incremental learning challenges, this chapter addresses the complexities of domain-incremental learning within federated systems. Building on the frameworks introduced in Chapter 5, we extend our focus to scenarios where data from different domains is introduced sequentially. Furthering the information retention strategies from Chapter 4, this chapter emphasizes a model-centric approach, leveraging prompt-based model structure to enhance robustness and generalization across diverse domains. The aim of this research is to ensure that FL systems can maintain high performance even as they encounter new and evolving data distributions, thereby solidifying the comprehensive approach to continuous learning discussed throughout this thesis.

In this chapter, we introduce a rehearsal-free federated domain incremental learning framework, RefFiL, based on a global prompt-sharing paradigm to alleviate catastrophic forgetting challenges in federated domain-incremental learning, where unseen domains are continually learned. Typical methods for mitigating forgetting, such as the use of additional datasets and the retention of private data from earlier tasks, are not viable in federated learning (FL) due to devices' limited resources. Our method, RefFiL, addresses this by learning domain-invariant knowledge and incorporating various domain-specific prompts from the domains represented by different FL participants. A key feature of RefFiL is the generation of local fine-grained prompts by our domain adaptive prompt generator, which effectively learns from local domain knowledge while maintaining distinctive boundaries on a global scale. We also introduce a domain-specific prompt contrastive learning loss that differentiates between locally generated prompts and those from other domains, enhancing RefFiL's precision and effectiveness.

Compared to existing methods, RefFiL significantly alleviates catastrophic forgetting without requiring extra memory space, making it ideal for privacy-sensitive and resource-constrained devices.

6.1 Rehearsal-free Federated Domain-incremental Learning

FL is a remarkable approach that enables various participants, also referred to as clients, to collaboratively train a global machine learning model while keeping their data decentralized, maintaining privacy, and leveraging diverse data sources [100]. A significant limitation of current FL research lies in its primary focus on scenarios with static data distributions [83, 43, 70, 50, 141], which remain unchanged throughout the training process [28, 27]. This limitation is particularly pronounced as it does not reflect the dynamic nature of real-world data. Recent studies have therefore shifted towards Federated Continual Learning (FCL), especially in class-incremental data scenarios [28, 27, 177, 98, 6], which struggles significantly with the critical problem of *catastrophic forgetting* [123], where model rapidly forgets previously learned knowledge upon encountering new data. In this chapter, we explore more challenging domain-incremental data scenarios.

Various methods have been proposed to address catastrophic forgetting. Rehearsal-based approaches, such as those described in [123], mitigate forgetting by replaying a selection of old task data during new task training. Alternatively, regularization-based methods, like Knowledge Distillation (KD) [84, 123, 14, 28], guide the model's optimization using historical data or models. Network expansion techniques [74, 38, 115] adapt the model structure to incorporate new features, but their practicality in memory-constrained FL environments is limited. Recent research like L2P [158] and DualPrompt [157] explores prompt-based, rehearsal-free domain-incremental learning. However, these methods still do not fully address the challenge of using global domain information to enhance local model robustness in Federated Domain-Incremental Learning (FDIL).

Strategies for mitigating catastrophic forgetting in domain-incremental tasks are distinct from those employed in class-incremental scenarios, where the substantial differences between classes make it easier for models to distinguish decision boundaries. In domain-incremental contexts, however, the smaller feature differences between domains make it harder to distinguish decision boundaries. Thus, the priority shifts towards acquiring domain-invariant knowledge, an effort pivotal to enhancing model robustness.

This focus becomes particularly vital when models are required to make predictions across a spectrum of diverse domains. To date, research in this area is notably sparse, especially concerning the effective global utilization of heterogeneous domain information from all clients at various stages of their transition. Such utilization is key to mitigating catastrophic forgetting by strengthening the local models' proficiency in learning domain-invariant knowledge in FDIL and across the broader expanse of FCL.

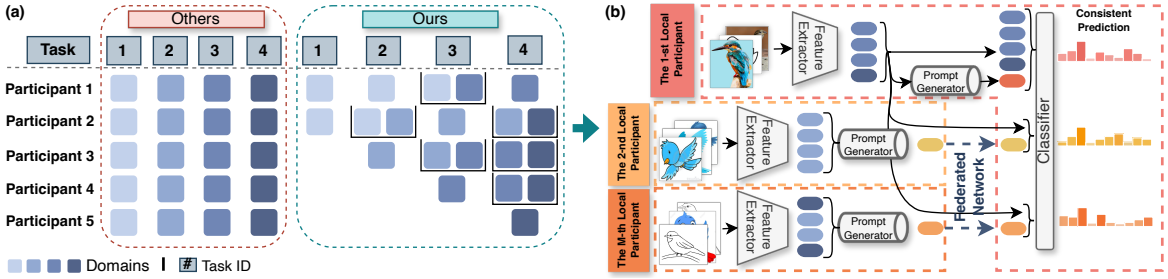

Fig. 6.1 Key steps of RefFil framework. Left panel (a) shows the common setting in existing FCL works, characterized by a cliff-style data transition. Right panel (a) shows our approach, where a subset of participants gradually transitions to new tasks. Panel (b) provides an overview example of a key step in our methodology: the 1st participant processes new domain data using global prompts from the 2nd to m-th participants and local prompts, enhancing robustness by aligning the model's predictions across diverse domain prompts as inputs.

Our proposed framework, **RefFil**, *rehearsal-free federated domain-incremental learning*, harnesses a wide spectrum of global domain information to augment cross-domain prediction robustness by specifically enhancing the acquisition of domain-invariant knowledge. We first recognize the inherent challenges posed by the high autonomy of participants in FL, which introduces significant uncertainty in data distribution and, consequently, a substantial domain gap. Thus, derivating robust prompts becomes critical in learning domain-invariant knowledge.

Hence, firstly, we define a practical FDIL problem. To address the challenge of significant domain gaps in datasets from various resource-sensitive participants, we propose a *client-wise domain adaptive prompt generator*, which creates personalized, instance-level prompts for each participant's data, incorporating domain adaptation knowledge for global sharing. Then, to effectively break down the silos of domain-specific information in FL, we introduce a *global prompt learning paradigm*. This paradigm involves globally distributing and sharing clustered local prompts combined with local domain-invariant knowledge learning through prompting. By doing so, we aim to empower the local model to effectively differentiate between prompt features from various domains, thereby enhancing its ability to learn domain-invariant knowledge

from local data, which is crucial for boosting the model's robustness in multi-domain predictions and significantly reducing its susceptibility to catastrophic forgetting. Finally, we introduce a *novel domain-specific prompt contrastive learning* method with temperature decay aimed at improving the local model's ability to distinguish between semantically related and unrelated prompts. By seamlessly integrating these three key components, the RefFil framework emerges as a highly competitive method in the realm of FDIL.

6.1.1 Related Work

Most recently, several works have begun to concentrate on FCL, especially Federated Class-incremental Learning (FCIL), which involves sequentially learning a series of distinct, non-overlapping classes. The primary challenge in FCIL is catastrophic forgetting, where models tend to lose previously learned knowledge upon task transition. Approaches such as GFLC [28] and its extension LGA [27] use knowledge distillation, employing an old model to regularize the current model's output and a loss function to mitigate global forgetting. Method TARGET [177] uses a global model to generate synthetic data with a global distribution for the current training task to reduce forgetting. CFed [98] addresses data unavailability by generating pseudo-labels on a surrogate dataset for knowledge distillation and proposes a server distillation mechanism to mitigate intra-task forgetting. These methods typically require storing old tasks or extra data, which is impractical in FL with resource-limited devices. MFCL [6] utilized GAN to generate data rather than save raw to achieve rehearsal-free learning from class-incremental data. Despite progress in the class-incremental scenario of FCL, there remains a notable gap in the domain-incremental setting, which focuses on learning domain-invariant knowledge to reduce forgetting, a different approach from FCIL.

To address the devices' resource-constrained limitation in FDIL, the concept of prompt techniques, initially introduced in natural language processing (NLP) for task adaptation [91], offers a promising solution. These techniques append instructions to large pre-trained models, helping them use existing knowledge instead of learning from scratch. Applied to continual learning, approaches like L2P [158] and DualPrompt [157] use prompts to avoid the need for rehearsal. L2P uses a prompt pool for task-specific knowledge, replacing the rehearsal buffer. DualPrompt, designed for resource-limited devices, uses two prompt types: General-Prompt for general instructions and Expert-Prompt for task-specific guidance to replace maintaining a prompt pool. However, in FDIL, the challenge goes beyond learning local domain-invariant information; it involves using data from other participants to enhance model robustness against diverse

domains. This chapter aims to address this specific challenge that is unexplored in current research.

6.1.2 Preliminaries

Federated Domain-incremental Learning. In standard domain-incremental learning [104], a series of sequential tasks are denoted as $\mathcal{T} = \{\mathcal{T}^t\}_{t=1}^{T}$, with T indicating the total number of tasks. Each task $\mathcal{T}^t$, representing the t-th task in the sequence, consists of N^t data-label pairs, represented as $\mathcal{T}^t = \{x_i^t, y_i^t\}_{i=1}^{N^t}$. Although each task is associated with a distinct domain $\mathcal{D}^t$, all tasks share the same label space $\mathcal{Y}$. The entire domain space across all tasks is represented as $\mathcal{D} = \{\mathcal{D}^t\}_{t=1}^{T}$, with the data distribution being completely inconsistent between different task domains, indicated by $\mathcal{D}^t \neq \mathcal{D}^{t-1}$.

We adapt traditional domain-incremental learning in the context of FDIL. In this setup, there are M local participants, denoted as $\mathcal{M} = \{\mathcal{M}_m\}_{m=1}^{M}$, and a central global server $\mathcal{M}_G$. During each global communication round r (where $r = 1, \ldots, R$), a subset of these local participants is randomly chosen for gradient aggregation. When a local participant $\mathcal{M}_l$ is selected for a given global round in the t-th incremental task, it receives the latest global model $\theta^{r,t}$.

The participant then trains this latest model on its local dataset $\mathbf{D}^t \sim \mathcal{P}_l^{|\mathbf{D}^t|}$ specific to the t-th task, where $\mathcal{T}_m^t = \{x_{mi}^t, y_{mi}^t\}_{i=1}^{N_l^t} \subset \mathcal{T}^t$ consists of data from new domains, and $\mathbf{P}_m$ is the data quantity distribution of the m-th participant. These local datasets $\{\mathbf{P}_m\}_{m=1}^{M}$ are not independent and identically distributed (non-iid), showcasing a type of quantity shift in our setting. Each incremental task contains K^t classes, maintaining consistency in class numbers across tasks ($K^1 = K^2 = \cdots = K^t$) and labels ($\mathbf{y}^1 = \mathbf{y}^2 = \cdots = \mathbf{y}^t$), but with varying domain and data distributions ($\mathbf{x}^1 \neq \mathbf{x}^2 \neq \ldots \neq \mathbf{x}^t$) for each task $t \in \mathcal{T}$. Once the local participant $\mathcal{M}_m$ loads $\theta^{r,t}$ and completes training on the t-th task, it obtains an updated local model, denoted as $\theta_l^{r,t}$. All selected participants then transmit their local updated model to the central global server $\mathcal{M}_G$. Here, these models, through aggregation, form the updated global model $\theta^{(r+1),t}$ for the next round. Subsequently, the global server $\mathcal{M}_G$ broadcasts the parameters of $\theta^{r,t}$ back to all participants, marking the beginning of the next global communication round.

Client increment strategy. We divide local participants into three dynamic groups across each incremental task, comprising Old $\mathcal{U}_o$, In-between $\mathcal{U}_b$, and New $\mathcal{U}_n$. Group $\mathcal{U}_o$, with M_o members, is restricted to working solely on data from previous domains, receiving no new domain data for the current task. Group $\mathcal{U}_b$, consisting of M_b

participants, handles both new domain data from the current task and data from previous tasks. The group $\mathcal{U}_n$, comprising M_n new participants, focuses exclusively on the new domain data of the current task. These groups adapt dynamically with the arrival of incremental tasks. Specifically, the composition of $\mathcal{U}_o, \mathcal{U}_b, \mathcal{U}_n$ is randomly redefined at each global round, with $\mathcal{U}_n$ being irregularly supplemented at any round in the FDIL process. This approach gradually increases the total number of participants, $M = M_o + M_b + M_n$, as streaming tasks progress.

Learning with Prompts. Prompt techniques, initially developed for task adaptation in natural language processing [91], have predominantly been applied to transformer-based large models. However, these techniques can also be adapted for models using convolutional neural network (CNN) based feature extractors. This adaptation is particularly relevant in FL environments, where participant devices are more likely to be resource-constrained.

In scenarios where a CNN-based model serves as the backbone for prompt-based learning, the model architecture typically includes a CNN-based feature extractor h, patch embedding layer, attention block b, and the classifier G. Specifically, feature map F is extracted by a feature extractor from an input image x as $F = h(x)$. Let F be split into n d-dimensional patch tokens PT through the patch embedding layer. Then, a d-dimensional trainable class token [CLS] is attached at the beginning of patch tokens along the sequence length dimension, thereby creating the initial set of input tokens $\mathbf{I} \in \mathbb{R}^{(n+1)\times d}$ as

$$\mathbf{I} = [\mathrm{CLS}; \mathrm{PT}_1, \ldots, \mathrm{PT}_n] \in \mathbb{R}^{(n+1)\times d}. \tag{6.1}$$

Following this, $\mathbf{I}$ is forwarded to the b-th attention block, which generates the input $\mathbf{I}_{b+1}$ for the subsequent attention block $(b+1)$ as:

$$\mathbf{I}_{b+1} = \mathrm{LN}\left(\mathbf{I}'_b + \mathbf{I}''_b\right), \tag{6.2}$$

where LN is layer normalization, $\mathbf{I}''_b = \mathrm{MLP}\left(\mathbf{I}'_b\right)$ is multilayer perception (MLP), and $\mathbf{I}'_b = \mathrm{LN}\left(\mathrm{MHSA}\left(\mathbf{I}_b, \mathbf{I}_b, \mathbf{I}_b\right)\right)$ is multi-head self-attention (MHSA) mechanism. This indicates that each attention block involves an MHSA mechanism, succeeded by an MLP with the skip connection [53], and an LN [5] (see Fig. 6.2). The classifier, which is a single feed-forward layer, determines the class labels. It does this by mapping the class token [CLS] from the output of the final attention block as:

$$\hat{y} = G([CLS]_B), \tag{6.3}$$

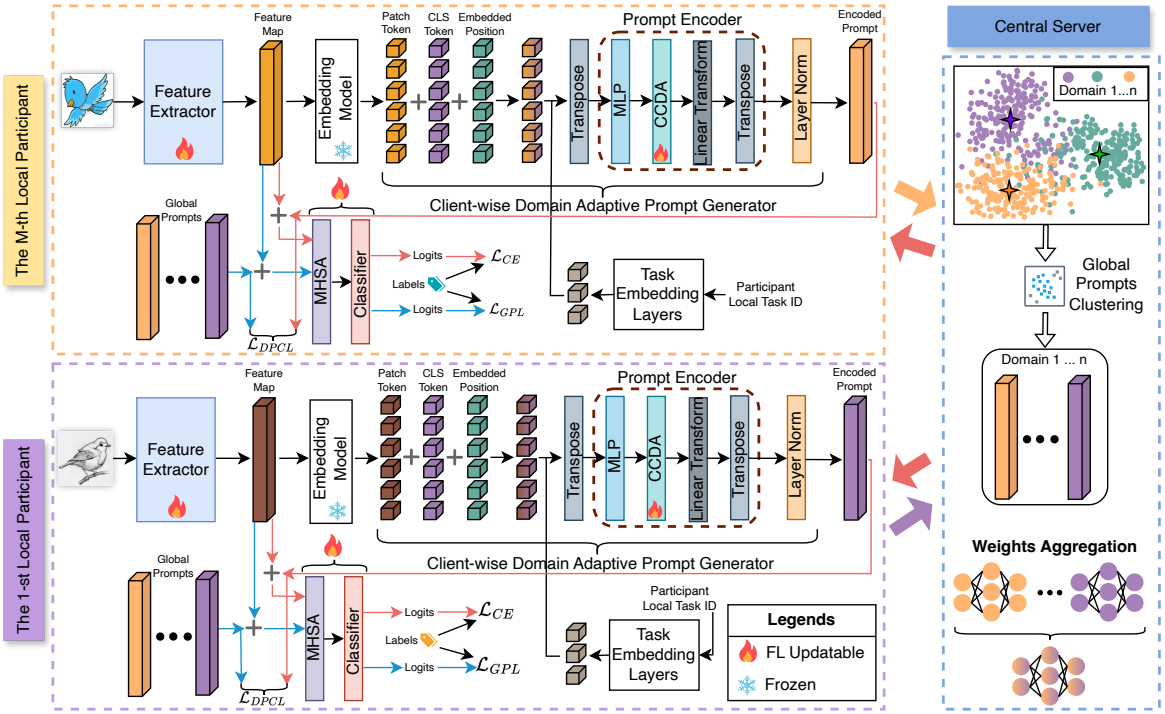

Fig. 6.2 Overview of the RefFil Framework. Each participant first encodes local prompts using the tokenized feature map and task ID embedding. These local prompts are then concatenated with the feature map to compute the loss $\mathcal{L}_{CE}$. Simultaneously, the feature map is combined with global prompts to calculate the loss $\mathcal{L}_{GPL}$, and the loss $\mathcal{L}_{DPCL}$ is determined between global and local prompts. Subsequently, all local prompts, along with the updated local models, are transmitted to the central server. The server then clusters the prompts domain-wise and aggregates the local models for distribution in the next training round.

where $\hat{y}$ is the predicted distribution of class probabilities.

6.1.3 Methodology

RefFil enhances local domain-invariant knowledge learning through the sharing of diverse domain information via global prompts in FDIL settings. As shown in Fig. 6.2, RefFil operates in task stages where participants engage in different domains. The process starts with a feature extractor generating a feature map from the input image, followed by our Client-wise Domain Adaptive Prompt (CDAP) generator. This generator encodes prompts using the feature map with the task ID. These encoded prompts, combined with the feature map, undergo classification involving computation of the cross-entropy loss and the Domain-specific Prompt Contrastive Learning (DPCL) loss with global prompts. Additionally, the feature map is merged with global prompts

to compute the Global Prompt Learning (GPL) loss. All local prompts are sent to the central server to facilitate domain information sharing.

Client-wise Domain Adaptive Prompt Generator. Most of the stages in FDIL training typically involve datasets from various domains, creating a substantial domain gap between participants' local data. To navigate this challenge, generating refined prompts is essential for enabling participants to effectively distinguish and share prompts globally. Common approaches include maintaining a prompt pool, as seen in methods like L2P [158] and DualPrompt [157]. These strategies are similar to traditional continual learning methods, which rely on storing raw data or summarised representative information. However, a significant obstacle in FL is the limited memory resources of participant devices and privacy sensitivity. Additionally, as noted in the [68], a fixed-size prompt pool may not effectively encode domain-specific knowledge, and a scalable pool faces feasibility issues due to uncertainties in task requirements and computational constraints. This makes maintaining an extensive and adaptable prompt pool challenging for continuous learning in a federated context, where the unpredictability of tasks and resource limitations of devices must be considered.

Thus, inspired by the prompt encoder network structure proposed in [68], we propose a client-wise domain adaptive prompt generator. This generator is designed to create personalized prompts for each participant's local data, embedding instance-level, fine-grained instructions with domain adaptation knowledge for global sharing. The CDAP generator $\mathcal{G}$ comprises several components: Layer Normalization (LN), a Multi-Layer Perceptron (MLP), a Cross-Client Domain Adaptation (CCDA) layer, and a Linear Transformation Layer (LT). This configuration enables the generation of adaptive local prompts p_m for each participant m, which could be formulated as:

$$\begin{aligned} P_m &= \mathrm{LT}\left(\mathrm{CCDA}\left(\mathrm{MLP}\left(\mathrm{LN}(\mathbf{I})^\top\right)\right); \phi(v)\right)^\top \\ &= \left(\alpha_v \, \mathrm{CCDA}\left(\mathrm{MLP}\left(\mathrm{LN}(\mathbf{I})^\top\right) + \lambda_v\right)^\top\right) \in \mathbb{R}^{p \times d}, \end{aligned} \tag{6.4}$$

where $\mathbf{I}$ is a set of input tokens, ϕ represents a linear layer that predicts two types of affine transformation parameters, α and λ, from a conditional input embedding v. Initially, v is normalized, transposed, from $(n+1) \times d$ to $d \times (n+1)$ and then passed through an MLP to generate p-dimensional outputs which create current task domain well-fitted and instance-level prompts tailored with domain-specific knowledge. To handle the data that vary significantly in distribution from various domains, we introduce the CCDA layer, a globally transferable linear layer enhancing the CDAP generator's generalization and robustness via FL aggregation. Additionally, we incorporate a

Feature-Wise Linear Modulation (LT) [114] framework to augment the generated prompts with additional instructional guidance for prediction. LT achieves this by applying an affine transformation to the instance-level prompts, using scaling α_v and shifting parameters λ_v, both conditioned on the embedding v as $[\alpha_v, \lambda_v] = \phi(v)$. For the conditional input embedding v, we designed a task-specific key embedding layer that is also employed to encode local task IDs, linking tasks with domain-specific data. However, the task ID contributes only to the training process by helping to distinguish data from different tasks and is not utilized during the inference stage.

Global Prompt Sharing. To overcome domain information silos in FL and address the challenge of data quantity disparity among participants, we propose globally distributing local prompts in a balanced manner. This method prevents network overload in scenarios involving thousands of devices and mitigates biases due to uneven data distribution. Our solution involves averaging across all clients, ensuring equitable influence from each participant on the global model, regardless of their data volume. This strategy is crucial in situations where differences in data quantity between resource-rich and resource-poor participants could significantly skew the direction of the global model optimization. The formulation for the Local Prompts Group (LPG) is as follows:

$$LPG_m = \left[\frac{1}{N} \sum_{j=1}^{N} P_{m,j}^{1}, \ldots, \frac{1}{N} \sum_{j=1}^{N} P_{m,j}^{k} \right] \in \mathbb{R}^d, \tag{6.5}$$

then global prompts set P^g could be summarised as:

$$P^g = [LPG_1, \ldots, LPG_m]\,. \tag{6.6}$$

However, generally, a significant portion of participants, 80% in our scenario, transition to training tasks in new domains while a minority continues with old tasks. This leads to a domain-specific prompt imbalance, with most prompts coming from new domains. This imbalance may bias local model updates towards these majority domains.

Global Prompts Clustering. To tackle the prompt imbalance issue, directly averaging all prompts may lead to a loss of important domain-characterized features, as averaging might not fully capture the unique characteristics essential for effective learning in diverse domains. Therefore, we propose the use of global prompts clustering that filters and selects representative prompts for each class based on their semantic domain characteristics. We adopt FINCH [129], a parameter-free clustering method and particularly well-suited for large-scale prompts clustering with minimal computing resources, which is FL dynamic environment friendly. Additionally, its efficiency in

clustering feature prototypes in FL has been previously demonstrated in [63]. FINCH operates on the principle that the nearest neighbor prompt of each sample can effectively support grouping. This is highly effective for distinguishing between prompts from different domains, as prompts from separate domains are unlikely to be nearest neighbors and are thus grouped separately. In contrast, similar prompts, likely from the same domain, are grouped together. Prompt similarity is evaluated using cosine similarity, where prompts are grouped by their smallest distance. This allows us to sort prompts into domain-specific sets efficiently. For each class k of prompts, we then construct an adjacency matrix as:

$$A^k(m,j) = \begin{cases} 1, & \text{if } j = c_m^k \text{ or } m = c_j^k \text{ or } c_m^k = c_j^k \\ 0, & \text{otherwise} \end{cases}, \tag{6.7}$$

where c_m^k denotes the nearest neighbor (with the highest cosine similarity) of the class k prompt from the m-th participant. Based on this clustering (Eq.6.7), we select representative prompts for each class from the embedding space. The final global clustered prompts are denoted as $\hat{P}^g$, are obtained as:

$$\begin{aligned} \hat{P}^{g,k} = \{LPG_m\}_{m=1}^{C} \xrightarrow{\Psi} \{LPG_m\}_{m=1}^{N} \in \mathbb{R}^{N\times d}, \\ \hat{P}^g = \left[\hat{P}^{g,1}, \ldots, \hat{P}^{g,k}\right]. \end{aligned} \tag{6.8}$$

Finally, C prompts are clustered by cluster model Ψ into N representatives of class k.

Personalised Prompt Generator Optimization. We expect to improve the local model's ability to differentiate between semantically related and unrelated prompts, ensuring a clear class decision boundary by enhancing sensitivity to domain-specific nuances in prompts. Specifically, we encourage the generation of prompts that are closely aligned with a semantically similar prompts set (P_i^+) while being distinctly different from a semantically unrelated prompts set (P_i^-). The prompts sampling strategy is defined as follows: For each class k, global prompts for class k are selected, and their cosine similarity with the locally generated prompt of k is computed. For clients $\mathcal{U}_o$ and $\mathcal{U}_n$ (single domain), the closest prompt is the positive sample P^+. For clients $\mathcal{U}_b$ (two domains), the two closest prompts are positive samples, and all remaining prompts are negative samples P_i^-. Drawing inspiration from the FPL [63], where contrastive learning was used to augment the model's proficiency in learning domain-invariant knowledge, we introduce domain-specific prompt contrastive learning with temperature decay specifically designed to enhance the adaptability of prompt

generation in the CDAP. It is natural to derive the following optimization objective term:

$$\mathcal{L}_{DPCL} = -\log \frac{\exp(\text{sim}(u_i, P_i^+)/\tau')}{\exp(\text{sim}(u_i, P_i^+)/\tau') + \exp(\text{sim}(u_i, P_i^-)/\tau')}. \tag{6.9}$$

Additionally, we address increasing domain diversity globally by proposing a method to calculate the contrastive temperature, τ, which adaptively adjusts the loss. Initially, it allows flexibility in distinguishing between positive and negative prompts, but it becomes more stringent as learning progresses. The calculation of the updated temperature is as follows:

$$\tau' = \max\left(\tau_{\min}, \tau \cdot (1 - (\gamma + (t-1) \cdot \beta))\right), \tag{6.10}$$

where $\tau_{\min}$ is the minimum temperature, γ represents the base decay rate of the temperature, and β is the rate of increment. Both γ and β are confined within the interval [0,1], indicating that their values range from 0 to 1, inclusive.

Local Domain-invariant Knowledge Learning with Prompting. To address the complexity of representing each class in our global clustering model due to varying cluster numbers, we average across all classes. This strategy allows creation of a generalized prompt $\bar{P}^G$ that effectively represents global domain information, ensuring a balanced and simplified approach to encapsulating the diversity of each domain, where

$$\bar{P}^g = \left[AVG(\hat{P}^{g,1}), \ldots, AVG(\hat{P}^{g,k})\right]. \tag{6.11}$$

We enhance our model's prediction robustness and mitigate forgetting by using global heterogeneous domain-specific prompts as diverse stimuli. This method helps the model learn domain-invariant knowledge. To support this, we introduce the GPL loss, formulated by developing a CrossEntory [20] (CE) mechanism, which is defined by the following equation:

$$\mathcal{L}_{GPL} = -\sum_{i=1}^{N} y_i \log\left(\text{softmax}\left(\xi_g\right)\right), \tag{6.12}$$

where $\mathcal{L}_{GPL}$ a loss function, sums over N instances, using y_i as the true label and applying softmax to the output of logits $\xi_g = G\left([\bar{P}^g, h(x)]\right)$, classifier G which combines the global prompt $\bar{P}^g$ with feature $h(x)$. Concurrently, locally CDAP generated prompts P^l are integrated into the CE loss function, $\mathcal{L}_{CE}$, to capture local domain-specific features, where $\xi_l = G\left([P^l, h(x)]\right)$, thus reinforcing the model's discriminative

capability within local domains. The formulation of this loss function is as follows:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{N} y_i \log\left(\text{softmax}\left(\xi_l\right)\right). \tag{6.13}$$

The final optimization objective is succinctly captured as:

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{GPL} + \mathcal{L}_{DPCL}. \tag{6.14}$$

The detailed of the FDIL process is outlined in Algorithm 4.

6.1.4 Experiments

Experimental Setup

Datasets. We evaluate our framework on four widely used datasets for image classification tasks, Digits-five [130, 140, 19], OfficeCaltech10 [150], PACS [81], and a subset of DomainNet [113], we named FedDomainNet.

- **Digits-five** [130, 140, 19]: This dataset is a comprehensive collection of the five most popular digit datasets for digit recognition tasks, encompassing 10 categories (digits 0-9). It aggregates five distinct domains: MNIST (55000 samples), MNIST-M (55000 samples), Synthetic Digits (25000 samples), SVHN (73257 samples), and USPS (7438 samples). In total, it comprises 215,695 images, each of 32x32 pixels.

- **OfficeCaltech10** [150]: This dataset spans four domains: Amazon (958 samples), Caltech(1123 samples), Webcam(295 samples), and DSLR(157 samples). It features 10 classes across each domain, culminating in a total of 2533 images, resized to 32x32 pixels for uniformity.

- **PACS** [81]: Encompassing four domains — Photo (1,670 images), Art Painting (2,048 images), Cartoon (2,344 images) and Sketch (3,929 images) — this dataset is distinguished by its varied artistic representations. Each domain includes seven categories, amounting to a total of 9,991 images. Originally of size 227x227, these images are cropped to 224x224 pixels to standardize the dataset.

- **FedDomainNet**: This dataset, a curated subset derived from DomainNet [113], features six domains: Clipart (9749 samples), Infograph (11010 samples), Painting (15228 samples), Quickdraw (24000 samples), Real (26476 samples), and Sketch

(13898 samples). Each domain offers 48 classes, resulting in a comprehensive collection of 100,361 images. For consistency in analysis, these images are also cropped to a size of 224x224 pixels.

FedDomainNet Dataset Details. In terms of dataset processing, we address the challenge posed by the vast scale of the DomainNet database [113], which comprises an extensive collection of 24 divisions with 345 categories, totaling 586,575 images. Such a large dataset presents considerable computational demands in the context of FL. Specifically, for federated domain-incremental learning, the key objective is to ensure each client sustains a diverse range of categories while possessing sufficient domain variety to mimic a continual data stream. To meet this requirement, we strategically extracted a representative subset from the cleaned version of the original DomainNet dataset for our experiments. As outlined in Table 6.1, from each of the 24 divisions, we carefully selected two classes. These classes were chosen based on their balanced representation across six domains and their clear categorical distinctions, resulting in a total of 48 classes. The subset from the original version comprised 102,986 images; however, utilizing the cleaned version, our final dataset consisted of 100,361 images. This selection was aimed at maintaining a uniform global distribution Figure 6.3 of types and data volumes to underscore the influence of domain shifts on the model.

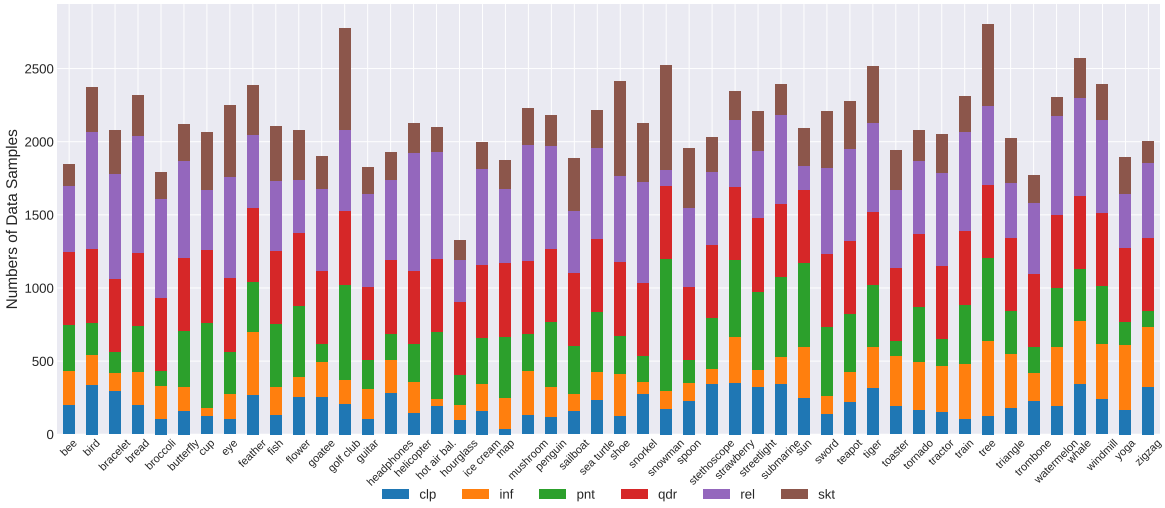

Fig. 6.3 FedDomainNet data distribution statistical information

Algorithm 4: Rehearsal-free federated domain-incremental learning

Input : Total tasks T, Global communication round R, client local epoch E, number of participants M, participant local data $\mathbf{D}_m(x, y)$, participant local model θ_m, learning rate η

Output : Trained global model θ^R

Central Server

  Initialise global model θ^r

  for $t = 1$ **to** T **do**

    for $r = 1$ **to** R **do**

      Participants Random Selection

      Broadcast $\theta^{r,t}, \hat{P}^g \rightarrow$ **Selected Participants**

      $\theta_m^r,\ LPG_m^k \leftarrow$ **Selected Participants**

      $\theta^{r+1} \leftarrow \sum_{m=1}^{N} \frac{|\mathbf{D}_m^t|}{|\mathbf{D}^t|} \theta_m^r$

      $\hat{P}^{g,k} = \{LPG_m\}_{m=1}^{C} \xrightarrow{\Psi} \{LPG_m\}_{m=1}^{N}$ Eq.6.8 ▷ Global prompt clustering

      $\hat{P}^g = \left[\hat{P}^{g,1}, \ldots, \hat{P}^{g,k}\right]$

    $\mathcal{U}_n, \mathcal{U}_b, \mathcal{U}_o \leftarrow M$Sampling ▷ Participant task transfer sampling and increment

    for m *in* $\mathcal{U}_b$ **do**

      $\mathbf{D}_m^t = \text{concat}(\mathbf{D}_m^{t-1}, \mathbf{D}_m^t)$

Participants

  $P_m = \{\}$ ▷ Initialise local prompts collection

  $\theta^{r,t}, \hat{P}^g \leftarrow$ **Central Server**

  $\bar{P}^g \xleftarrow{\text{Eq.6.11}} \hat{P}^g$

  for $e = 0, ..., E$ **do**

    for $x_i, y_i \in \mathbf{D}_m^t$ **do**

      $u_i, p_l, \xi_l, \xi_g = \theta^{r,t}(x_i, \bar{P}^g)$

      $\mathcal{L}_{CE} \leftarrow$ Eq.6.13 (u_i, ξ_l)

      $\mathcal{L}_{GPL} \leftarrow$ Eq.6.12 (u_i, ξ_g)

      $\mathcal{L}_{DPCL} \leftarrow$ Eq.6.9 $(p_l, \hat{P}^g)$

      $\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{GPL} + \mathcal{L}_{DPCL}$

      $\theta_m^r \leftarrow \theta_m^r - \eta \nabla \mathcal{L}(\theta_m^r; b)$

      if $e = (E - 1)$ **then**

        $P_m[k]$ append p_i

  for k *in* P_m *keys* **do**

    ▷ Average local prompts into a representative prompt

    $LPG_m = \left[\frac{1}{N}\sum_{j=1}^{N} P_{m,j}^1, \ldots, \frac{1}{N}\sum_{j=1}^{N} P_{m,j}^k\right]$

  Send $\theta_m^r,\ LPG_m \rightarrow$ **Central Server**

Baselines. RefFil is benchmarked against well-known rehearsal-free methods. Specifically, we compare it with the finetune approach, which involves straightforward model updates but is significantly impacted by catastrophic forgetting. Additionally, we consider four prominent rehearsal-free approaches: LwF [84], EWC [74], L2P [158] and DualPrompt [157]. These methods are fundamentally based on regularization methods.

However, they were originally proposed for centralized continual learning contexts and compared to FDIL, which is an under-explored field. We adopted these methodologies for our FDIL setting, resulting in variants we refer to as FedLwF, FedEWC, FedL2P, and FedDualPrompt, respectively. Particularly, to ensure a fair comparison, the prompt pool feature in FedL2P and FedDualPrompt was initially deactivated. Moreover, to demonstrate the superior efficacy of the RefFiL approach, even when competing methods employ prompt pool as a rehearsal technique, versions of FedL2p and FedDualPrompt with the prompt pool feature reactivated were also evaluated. These special versions are indicated with a † superscript in the Table 6.2 6.4 6.3 6.5.

Evaluation Metrics. The evaluation primarily includes top-1 accuracy, reflecting the model's most probable correct prediction. We also report *Average (Avg %)* accuracy, as defined in iCaRL [123], which is the mean accuracy across all learning steps, indicating the model's stability and consistency. Additionally, we present the *Final (Last %)* accuracy, measured after the last learning step, to assess the model's knowledge retention and consolidation capabilities.

Division	**Class**	**clp**	**inf**	**pnt**	**qdr**	**rel**	**skt**	**Total**
Furniture	teapot	222	209	391	500	631	327	2280
	streetlight	326	113	537	500	463	268	2207
Mammal	tiger	315	285	422	500	607	386	2515
	whale	343	432	357	500	671	272	2575
Tool	stethoscope	343	107	346	500	496	237	2029
	sword	139	124	470	500	591	384	2208
Cloth	shoe	127	291	260	500	587	645	2410
	bracelet	293	123	150	500	715	300	2081
Electricity	headphones	285	224	181	500	551	188	1929
	toaster	196	337	107	500	536	267	1943
Building	golf club	207	169	650	500	552	695	2773
	windmill	245	372	397	500	635	245	2394
Office	cup	128	52	582	500	406	396	2064
	map	42	206	423	500	507	193	1871
Human Body	goatee	255	236	129	500	562	219	1901
	eye	108	168	292	500	695	489	2252
Road Transportation	train	109	373	406	500	681	240	2309
	tractor	154	316	183	500	636	263	2052
Food	bread	197	232	315	500	794	276	2314
	ice cream	160	187	311	500	657	184	1999
Nature	sun	248	352	572	500	161	258	2091
	tornado	169	329	373	500	497	211	2079
Cold Blooded	sea turtle	236	190	410	500	621	254	2211
	fish	130	195	429	500	479	373	2106
Music	guitar	103	204	203	500	632	183	1825
	trombone	227	195	175	500	484	191	1772
Fruit	strawberry	357	308	530	500	454	198	2347
	watermelon	193	401	410	500	671	128	2303
Sport	snorkel	278	81	179	500	689	397	2124
	yoga	165	447	161	500	371	251	1895
Tree	tree	126	511	571	500	536	555	2799
	flower	253	140	485	500	360	336	2074
Bird	bird	336	208	222	500	803	306	2375
	penguin	121	201	447	500	700	209	2178
Vegetable	mushroom	136	298	254	500	788	252	2228
	broccoli	105	229	100	500	679	181	1794
Shape	zigzag	323	412	110	500	515	144	2004
	triangle	183	364	298	500	376	303	2024
Kitchen	spoon	228	127	158	500	534	406	1953
	hourglass	100	100	206	500	289	134	1329
Water Transportation	sailboat	162	119	322	500	422	361	1886
	submarine	344	183	550	500	607	207	2391
Sky Transportation	helicopter	145	216	257	500	804	200	2122
	hot air balloon.	198	48	453	500	732	170	2101
Insect	bee	202	233	313	500	452	144	1844
	butterfly	160	162	387	500	658	249	2116
Others	feather	268	432	344	500	505	336	2385
	snowman	174	123	901	500	114	712	2524
Total	-	9864	11364	16729	24000	26906	14123	102986
Total (Cleaned)	-	9749	11010	15228	24000	26476	13898	100361

Table 6.1 Detailed statistics of the FedDomainNet dataset

Methods	Digits-five				OfficeCaltech10				PACS				FedDomainNet			
	Avg	Δ	Last	Δ	Avg	Δ	Last	Δ	Avg	Δ	Last	Δ	Avg	Δ	Last	Δ
Finetune	77.39	9.55↑	49.80	12.31↑	44.56	9.00↑	19.29	14.37↑	40.18	15.14↑	30.82	13.45↑	28.46	0.47↑	18.07	0.82↑
FedLwF	77.58	9.36↑	56.86	5.25↑	46.78	6.78↑	28.74	4.92↑	40.12	15.20↑	26.61	17.66↑	27.95	0.98↑	17.96	1.02↑
FedEWC	78.20	8.74↑	45.89	16.22↑	44.38	9.18↑	15.55	18.11↑	40.27	15.05↑	27.36	16.91↑	26.10	2.83↑	18.37	0.58↑
FedL2P	83.45	3.49↑	57.65	4.46↑	46.51	7.05↑	26.57	7.09↑	49.68	5.64↑	35.32	8.95↑	25.26	3.67↑	18.42	0.56↑
FedL2P†	84.86	2.08↑	60.17	1.94↑	45.41	8.15↑	25.20	8.46↑	50.00	5.32↑	34.52	9.75↑	22.18	6.75↑	15.59	3.39↑
FedDualPrompt	85.15	1.79↑	59.30	2.81↑	45.15	8.41↑	23.82	9.84↑	54.05	1.27↑	41.07	3.20↑	28.25	0.68↑	18.05	0.93↑
FedDualPrompt†	84.39	2.55↑	58.34	3.77↑	47.86	5.70↑	27.76	5.90↑	52.79	2.53↑	37.62	6.65↑	28.53	0.40↑	17.76	1.22↑
RefFiL	**86.94**	-	**62.11**	-	**53.56**	-	**33.66**	-	**55.32**	-	**44.27**	-	**28.93**	-	**18.98**	-

Table 6.2 Summarised results on four datasets: Digits-five with five incremental tasks, FedDomainNet with six, and both OfficeCaltech10 and PACS with four tasks. Displaying average accuracy (Avg%) and final task accuracy (Last%). The amount of change in our method compared to other methods is marked as Δ, The results of our RefFiL are highlighted in color
† Means this baseline enabled prompt pool

Methods	Digits-five			OfficeCaltech10			PACS			FedDomainNet		
	Avg	Last	Δ	Avg	Last	Δ	Avg	Last	Δ	Avg	Last	Δ
Finetune	59.84	58.20	2.64↑	37.60	25.20	13.19↑	46.99	38.97	7.75↑	31.85	11.58	4.16↑
FedLwF	65.22	59.36	1.48↑	38.76	25.20	13.19↑	43.43	30.17	16.55↑	31.33	11.01	4.73↑
FedEWC	64.00	59.54	1.30↑	38.26	27.95	10.44↑	43.60	30.22	16.50↑	30.38	12.03	3.71↑
FedL2P	66.00	59.84	1.00↑	41.58	34.45	3.94↑	45.99	31.02	15.70↑	25.19	9.51	6.23↑
FedL2P†	64.45	59.74	1.10↑	41.24	31.50	6.89↑	45.39	35.42	11.30↑	22.95	7.32	8.42↑
FedDualPrompt	65.31	**60.94**	0.10↓	40.47	31.50	6.89↑	48.41	42.32	4.40↑	33.09	14.54	1.20↑
FedDualPrompt†	66.61	**60.94**	0.10↓	39.73	30.91	7.48↑	47.64	42.82	3.90↑	30.11	14.54	1.20↑
RefFiL	**69.36**	60.84	-	**44.33**	**38.39**	-	**51.08**	**46.72**	-	**33.34**	**15.74**	-

Table 6.3 Summarised results in new domain order of Table 6.2.

Implementation details. In this study, all methods are implemented using PyTorch [109] and run on a single NVIDIA RTX 4090 GPU, and all baselines are using ResNet10 [53] with attention mechanism as the feature extractor classification model backbone. A simple embedding model as the feature map tokenizer, similar to ViT [31], was designed with initialized-only and frozen parameters for feature embedding for Refil. The FedAvg [100] algorithm was employed for global model aggregation. Each task involves training the model over $R = 30$ rounds. The number of epochs for local updates on each client is set at 20. We use learning 0.04 for FedDomainNet, 0.06 for OfficeCaltech10, and 0.03 for the rest of the datasets to get the proper performance. Additionally, we use SGD as an optimizer for all experiments. Unless specified otherwise, the constraint factor λ in the EWC method is fixed at 300, and the temperature for distillation is set to the default value of 2. Our client increment setting aligns with what we outlined in Section 6.1.2. Specifically, for experiments on the Digit-Five, PACS, and FedDomainNet datasets, we start with 20 clients, select 10, and increment by two clients with each new task. Conversely, for experiments on the OfficeCaltech10 dataset, considering the data sample limitation, we start with 10 clients, select 5, and increment by one client per new task. Since this work focuses on domain-incremental data, all clients had the same number of classes but represented different domains after the first task. In all experiments, 80% of the $\mathcal{U}o$ old clients transition to a new task for each respective task. For the hyperparameters in Eq. 6.10, we fix $\tau_{\min}$ at 0.3, γ at 0.1, β at 0.05, and τ at 0.9.

	Task 1 → 5 on Digit-Five							Task 1 → 4 on OfficeCaltech10				
Methods	MNIST	MNIST-M	USPS	SVHN	SYN	–	Avg	Amazon	Caltech	Webcam	DSLR	Avg
Finetune	99.68	97.75	63.87	75.84	49.80	–	77.39	76.56	57.79	24.58	19.29	44.56
FedLwF	99.68	92.80	69.16	69.39	56.86	–	77.58	76.56	53.24	28.57	28.74	46.78
FedEWC	99.68	97.48	74.63	73.32	45.89	–	78.20	76.56	56.59	29.83	15.55	44.38
FedL2P	99.66	98.06	80.01	81.89	57.65	–	83.45	76.56	51.80	31.09	26.57	46.51
FedL2P†	99.64	97.65	85.18	81.65	60.17	–	84.86	71.35	55.88	29.20	25.20	45.41
FedDualPrompt	99.67	97.96	86.88	81.95	59.30	–	85.15	74.48	50.36	31.93	23.82	45.15
FedDualPrompt†	99.65	97.90	84.68	81.40	58.34	–	84.39	75.90	53.96	33.82	27.76	47.86
RefFiL	**99.68**	**98.25**	**90.96**	**83.70**	**62.11**	–	**86.94**	**78.65**	**61.15**	**40.76**	**33.66**	**53.56**

	Task 1 → 6 on FedDomainNet							Task 1 → 4 on PACS				
Methods	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	Avg	Photo	Cartoon	Sketch	Art Painting	Avg
Finetune	**51.48**	15.89	28.05	**27.84**	29.45	18.07	28.46	61.68	47.45	36.12	30.82	40.18
FedLwF	**51.48**	18.10	26.71	25.98	27.47	17.96	27.95	61.68	47.07	25.11	26.61	40.12
FedEWC	50.76	15.46	22.66	21.87	27.45	18.37	26.10	63.17	47.70	23.66	27.36	40.27
FedL2P	40.55	13.19	21.09	28.15	30.13	18.42	25.26	64.97	48.32	**50.09**	35.32	49.68
FedL2P†	37.63	9.29	16.79	27.09	26.68	15.59	22.18	65.57	54.67	45.25	34.52	50.00
FedDualPrompt	51.17	19.48	28.74	22.68	29.40	18.05	28.25	73.65	56.54	44.93	41.07	54.05
FedDualPrompt†	51.14	20.20	28.91	23.09	30.07	17.76	28.53	**75.75**	54.55	43.23	37.62	52.79
RefFiL	51.27	**20.91**	**29.23**	22.57	**30.62**	**18.98**	**28.93**	73.95	**59.90**	43.17	**44.27**	**55.32**

Table 6.4 Comparison of RefFiL's performance with five baseline methods on four widely used datasets showcasing average accuracy (Avg %) and accuracy for each domain task (%). At each task step, all clients have an equal number of classes but *quantity shift* data.

	Task 1 → 5 on Digit-Five							Task 1 → 4 on OfficeCaltech10				
Methods	SVHN	MNIST	SYN	USPS	MNIST-M	–	Avg	Caltech	Amazon	DSLR	Webcam	Avg
Finetune	94.97	58.35	49.04	38.66	58.20	–	59.84	49.78	58.27	17.15	25.20	37.60
FedLwF	94.97	73.21	54.73	43.82	59.36	–	65.22	49.78	57.79	22.27	25.20	38.76
FedEWC	95.03	64.32	50.22	50.88	59.54	–	64.00	48.00	56.83	20.27	27.95	38.26
FedL2P	94.85	73.54	53.19	48.56	59.84	–	66.00	49.78	58.03	24.05	34.45	41.58
FedL2P†	94.80	73.45	51.07	43.21	59.74	–	64.45	50.67	58.27	**24.50**	31.50	41.24
FedDualPrompt	94.78	70.71	54.06	46.04	**60.94**	–	65.31	48.00	58.75	23.61	31.50	40.47
FedDualPrompt†	94.65	77.02	54.43	46.01	**60.94**	–	66.61	50.22	57.07	20.71	30.91	39.73
RefFiL	**95.35**	**76.03**	**59.90**	**54.68**	60.84	–	**69.36**	**52.00**	**63.31**	23.61	**38.39**	**44.33**

	Task 1 → 6 on FedDomainNet							Task 1 → 4 on PACS				
Methods	Infograph	Sketch	Quickdraw	Real	Painting	Clipart	Avg	Cartoon	Photo	Sketch	Art Painting	Avg
Finetune	68.84	33.94	**28.94**	26.12	21.73	11.58	31.85	68.23	40.97	39.77	38.97	46.99
FedLwF	68.84	34.87	28.82	23.88	20.53	11.01	31.33	68.23	36.11	39.21	30.17	43.43
FedEWC	68.11	34.66	24.63	24.10	18.75	12.03	30.38	69.94	38.23	36.00	30.22	43.60
FedL2P	53.39	26.76	27.57	17.92	15.98	9.51	25.19	68.23	42.34	**42.73**	31.02	45.99
FedL2P†	51.89	24.86	26.37	14.64	12.62	7.32	22.95	66.95	44.71	34.49	35.42	45.39
FedDualPrompt	**70.67**	41.50	25.82	25.30	20.73	14.54	33.09	69.94	41.34	40.03	42.32	48.41
FedDualPrompt†	70.33	40.63	25.69	25.48	21.40	14.54	30.11	66.74	41.72	39.27	42.82	47.64
RefFiL	69.02	**42.48**	24.70	**26.19**	**21.93**	**15.74**	**33.34**	**73.13**	**45.33**	39.14	**46.72**	**51.08**

Table 6.5 Comparison of RefFiL's performance in new domain order of table 6.4.

6.1.5 Results

In this study, we evaluated the performance of seven baseline methods alongside RefFiL across four widely used datasets: Digit-Five, OfficeCaltech10, FedDomainNet, and PACS. The results, detailed in Table 6.2, Table 6.4, Table 6.3, and Table 6.5, indicate that RefFiL consistently outperforms the baseline methods in both average and last task accuracy, even under the impact of catastrophic forgetting. Notably, RefFiL maintains its advanced performance regardless of changes in domain order. These experiments illustrate the accuracy distribution across various domains as tasks change and analyze the decision-making boundaries of all baseline methods.

RefFiL efficiently mitigates forgetting. Detailed results on the performance are available in Table 6.4. The † means the baseline enabled prompt pool to save generated prompts, from which one is sampled each time to better mitigate forgetting. The summarized results in Table 6.2 demonstrate that RefFiL consistently achieves state-of-the-art (SOTA) results, showcasing robust performance across all four datasets in terms of average accuracy (Avg) and performance on the final task of each dataset (Last). Specifically, RefFiL shows a remarkable outperforming over the baseline methods in different scenarios. In smaller and less complex datasets like Digit-Five, RefFiL surpasses other baselines by 1.79% to 9.55% in average accuracy and by 1.94% to 16.22% in the final task accuracy. In the OfficeCaltech10 dataset, RefFiL outperforms baselines by 5.70% to 9.18% in average accuracy and 5.90% to 18.11% in the last task accuracy. Moreover, in larger and more complex image datasets like PACS, RefFiL continues to lead, achieving 1.27% to 15.20% higher average accuracy and 3.20% to 17.66% higher accuracy in the final task compared to the baselines, similarly, despite the challenges posed by the FedDomainNet dataset, such as sparse data distribution across 48 classes and high-resolution images. RefFiL still demonstrated improvement. RefFiL outperforms others with an improvement of 0.40% to 6.75% in average accuracy and 0.56% to 3.39% in the final task accuracy. These results collectively underscore the effectiveness and adaptability of RefFiL in handling diverse and challenging datasets, maintaining consistently high performance both overall and in the crucial final tasks.

RefFiL produces domain order-independent consistency results. To demonstrate the consistency of our results across different domain introduction orders, we generated new results, which are presented in Table 6.5 and summarized in Table 6.3. In these experiments, the order of domain introductions was randomly altered. Specifically, in Table 6.5, the domain orders for Digit-Five, OfficeCaltech10, and FedDomainNet were significantly disrupted, while in PACS, only the order of the first two domains was changed.

Table 6.6 Ablation study of RefFil on OfficeCaltech10 dataset, and the amount of improvement in RefFiL with different components compared to baseline (Finetune) is marked as Δ.

Methods	CDAP	GPL	DPCL	Avg	Δ	Last	Δ
RefFiL				44.56	-	19.29	-
	✓			49.78	5.22↑	27.56	8.27↑
		✓		47.94	3.38↑	26.38	7.09↑
	✓	✓		50.32	5.76↑	25.39	6.10↑
		✓	✓	49.45	4.89↑	30.12	10.83↑
	✓	✓	✓	**53.56**	**9.00**↑	**33.66**	**14.37**↑

This setup highlights the robustness of our method, showing that RefFiL maintains high performance despite variations in the sequence of domain introductions. Compared to the results in Table 6.2, although FedDualPrompt and FedDualPrompt† achieved better performance in the Last metric, RefFiL consistently outperformed these methods in the more critical Avg metric, indicating its superior overall performance. For both the OfficeCaltech10 and PACS datasets, RefFiL's advantages in Avg and Last metrics were magnified compared to most of the baselines. In the most challenging dataset, FedDomainNet, while RefFiL's advantage in Avg reduced from 0.40% to 0.20%, its Last metric slightly improved from 0.93% to 1.20%. These results significantly boosted RefFiL's advantages over others.

These results illustrate that RefFiL efficiently and stably mitigates catastrophic forgetting by enhancing the model's ability to learn from domain-invariant knowledge. The consistent high performance across various domain orders underscores RefFiL's robustness and adaptability in FDIL scenarios.

RefFiL enables stable accuracy distribution across different domains. We also plot the accuracy distribution for each domain of all tasks, presented in Figure 6.4, illustrating the distribution of accuracy across various baseline methods for different tasks and domains. The subplots show a clear comparison between the baseline methods, with each box plot providing insights into the median accuracy, interquartile range, and variability for each method.

The subplot corresponding to our ReFiL stands out due to its narrow interquartile range and high median accuracy across all tasks, indicating a consistent and reliable performance. For example, in the MNIST task, ReFiL achieves a median accuracy of 99.64%, with minimal variability among the accuracy values. This consistency is also

observed in more complex tasks like MNIST-M and USPS, where ReFiL maintains a high median accuracy compared to other baselines.

Additionally, ReFiL's box plot shows fewer outliers than other baselines, indicating that its performance is less susceptible to variability or domain shift. This resilience is especially notable in challenging tasks like SVHN, where ReFiL still achieves a median accuracy of 95.18%, outperforming several other baselines. Overall, ReFiL's superior performance in terms of both high accuracy and low variability is consistently demonstrated across all tasks, highlighting its robustness and effectiveness as a baseline method.

RefFiL provides clearer decision-making boundaries. To demonstrate the model's capability to learn a generalizable decision boundary across various class domains, we present a t-SNE visualization in Figure 6.5, comparing five baselines with our RefFiL on the Digits-Five Dataset. This visualization is particularly effective in showing how well the model differentiates between class domains at each task stage. The incremental task structure is as follows: Task 1 evaluates the model using only the MNIST dataset. Task 2 tests the model on a combined dataset of MNIST and MNIST-M. Task 3 expands the assessment to include MNIST, MNIST-M, and USPS. In Task 4, the model faces a dataset that integrates MNIST, MNIST-M, USPS, and SYN. Finally, Task 5 challenges the model with all previously learned datasets, adding SVHN into the mix. Compared to other baseline methods, RefFiL demonstrates a superior performance starting from Task 2, as evidenced by the greater clarity and distinctness of each cluster's boundaries in the t-SNE visualization. Notably, the key areas marked by red boxes, which indicate regions where different classes overlap or are challenging to separate, are consistently smaller with RefFiL. This indicates that RefFiL is more effective at distinguishing between classes, leading to less confusion and more precise clustering than the baselines.

More specifically, in Figure 6.5, we present a detailed t-SNE visualization of Task 5, illustrating the global model's performance after the completion of this task. This figure evaluates the task 5 output model's performance in handling all learned domain datasets ranging from MNIST to SVHN. It is evident from the visualization that RefFiL achieves a more distinct decision boundary, particularly for the MNIST, USPS, and SYN datasets. Moreover, for the MNIST-M and SVHN datasets, RefFiL demonstrates significant diffusion of cluster boundaries. This diffusion is indicative of RefFiL's proficiency in achieving effective class separation across a diverse range of domain datasets.

RefFiL saving and computation cost. RefFiL eliminates storage costs entirely by avoiding the need to save raw or condensed data. Instead, it learns domain-invariant features through prompt tuning and contrastive learning. This additional processing increases local computation by about 20%. For example, if a baseline training epoch requires 10 GFLOPs, incorporating RefFiL's modules may raise the cost to around 12 GFLOPs per epoch. The trade-off between these approaches depends on device capabilities: while ECoral is ideal for environments with limited memory resources, RefFiL is more advantageous in settings where computational power is abundant.

Ablation Study

To understand how different components can mitigate catastrophic forgetting problems, we conduct the experiments as shown in Table 6.6, wherein the impact and the synergistic effects of each module within RefFil are dissected. Each experimental configuration involving a separate component is indicated by a checkmark (✓).

Our findings reveal that using either the CDAP or GPL module individually enhances both average and final task accuracy, but their combination results in only a slight improvement in average accuracy compared to when they are used separately. Considering that the DPCL module cannot function in isolation, we explored its effectiveness in conjunction with the GPL module. This combination markedly improved the model's performance, especially in the accuracy of the final task. Most importantly, when all three core components, CDAP, GPL, and DPCL, are integrated, we observe a substantial improvement in both average and last-task accuracy.

An important observation is that while combining CDAP with GPL can improve the model's average performance, it does not outperform the individual use of CDAP or GPL. This may be due to the increased domain data heterogeneity in the final task. The personalized prompts generated by CDAP and shared by GPL might cause difficulty for the model in distinguishing between heterogeneous domain information.

This comprehensive ablation study confirms the essentiality and complementarity of each module. Their collective operation significantly enhances the global model's performance in domain-incremental scenarios. The results validate the necessity of each component, underscoring their collective contribution to achieving optimal outcomes in FDIL.

Hyperparameter sensitivity analysis

To evaluate the sensitivity of the hyperparameters in Eq.6.10, we conducted a grid search. Based on our preliminary experiments, we identified a suitable range for each

Exp	τ	$\tau_{\min}$	γ	β	τ' (3rd)	**Avg**	**Last**
1	0.5	0.2	0.15	0.1	0.325	42.17	36.22
2	0.5	0.4	0.05	0.05	0.425	42.32	37.01
3	0.7	0.3	0.1	0.05	0.560	43.14	37.80
4	0.9	0.2	0.05	0.1	0.675	42.89	36.81
5	0.9	0.4	0.05	0.01	0.837	42.30	36.02
w/o τ'	-	-	-	-	-	41.97	34.39
Our	0.9	0.3	0.1	0.05	0.720	44.33	38.39

Table 6.7 Experimental configurations for sensitivity analysis of hyperparameters in OfficeCaltech10 dataset, with domain order shown in Table 6.5.

hyperparameter as follows: τ in $0.5, 0.7, 0.9$, $\tau_{\min}$ in $0.2, 0.3, 0.4$, γ in $0.05, 0.1, 0.15$, and β in $0.01, 0.05, 0.1$. These parameters can form a total of 81 combinations. Given that the final value of τ' influences the loss function $\mathcal{L}_{DPCL}$, we selected five representative groups of hyperparameter combinations to explore how these variations affect the model's performance. Additionally, we conducted a comparative experiment without applying the overall decay parameter τ', marked as w/o τ'. As illustrated in Table 6.7, different combinations of these parameters resulted in only minor variations in the model's performance metrics. Notably, in the last task, where the overall domain diversity increased, τ' effectively mitigated the impact of heterogeneous domain data on model performance.

6.2 Summary

In this chapter, we introduced RefFil, a rehearsal-free federated domain-incremental learning framework designed to augment cross-domain prediction robustness by enhancing the acquisition of domain-invariant knowledge. Our approach includes a client-wise domain adaptive prompt generator for creating personalized, instance-level prompts, a global prompt learning paradigm for breaking down domain-specific information silos, and a novel domain-specific prompt contrastive learning method with temperature decay for discriminating between relevant and irrelevant prompts. Through extensive experiments, we show that the RefFil framework excels in capturing domain invariant knowledge, mitigating catastrophic forgetting in federated domain-incremental learning. It is also important to acknowledge that while RefFil is proposed for learning domain-

invariant knowledge through prompting, it may also contribute to the development of finetuning large pre-trained models in domain-specific areas.

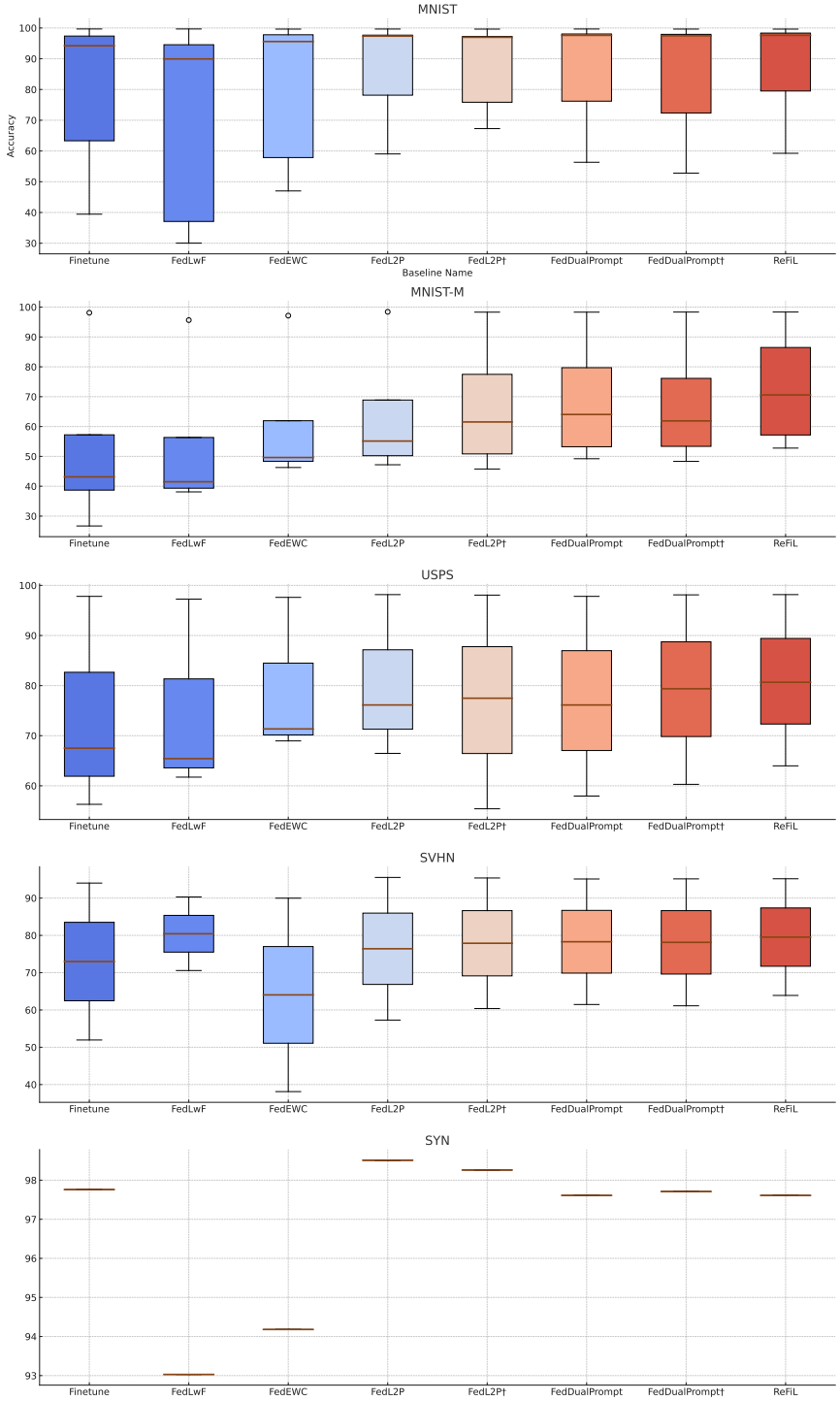

Fig. 6.4 Box plots of the accuracy distribution for each domain in every task.

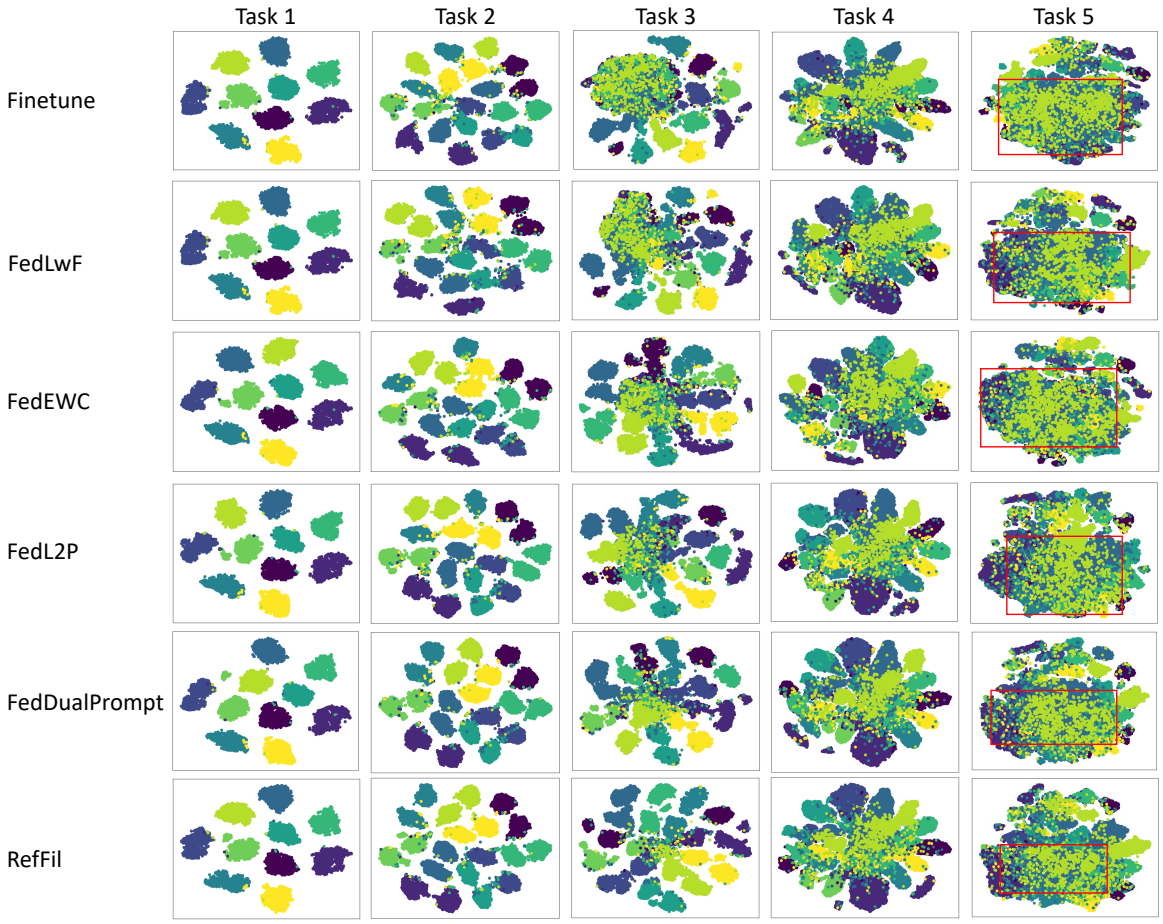

Fig. 6.5 t-SNE Visualization of Five Baselines and Our RefFiL Model Across Task Steps in the Digits-Five Dataset: This illustration captures the global model's ability to learn a generalizable decision boundary at each task step, evaluated across all previously learned domains. The greater the clarity and diffusion of each cluster's boundaries, the better. The core areas where different classes are challenging to distinguish are highlighted with red boxes. Smaller box sizes indicate fewer data points that are difficult to cluster.

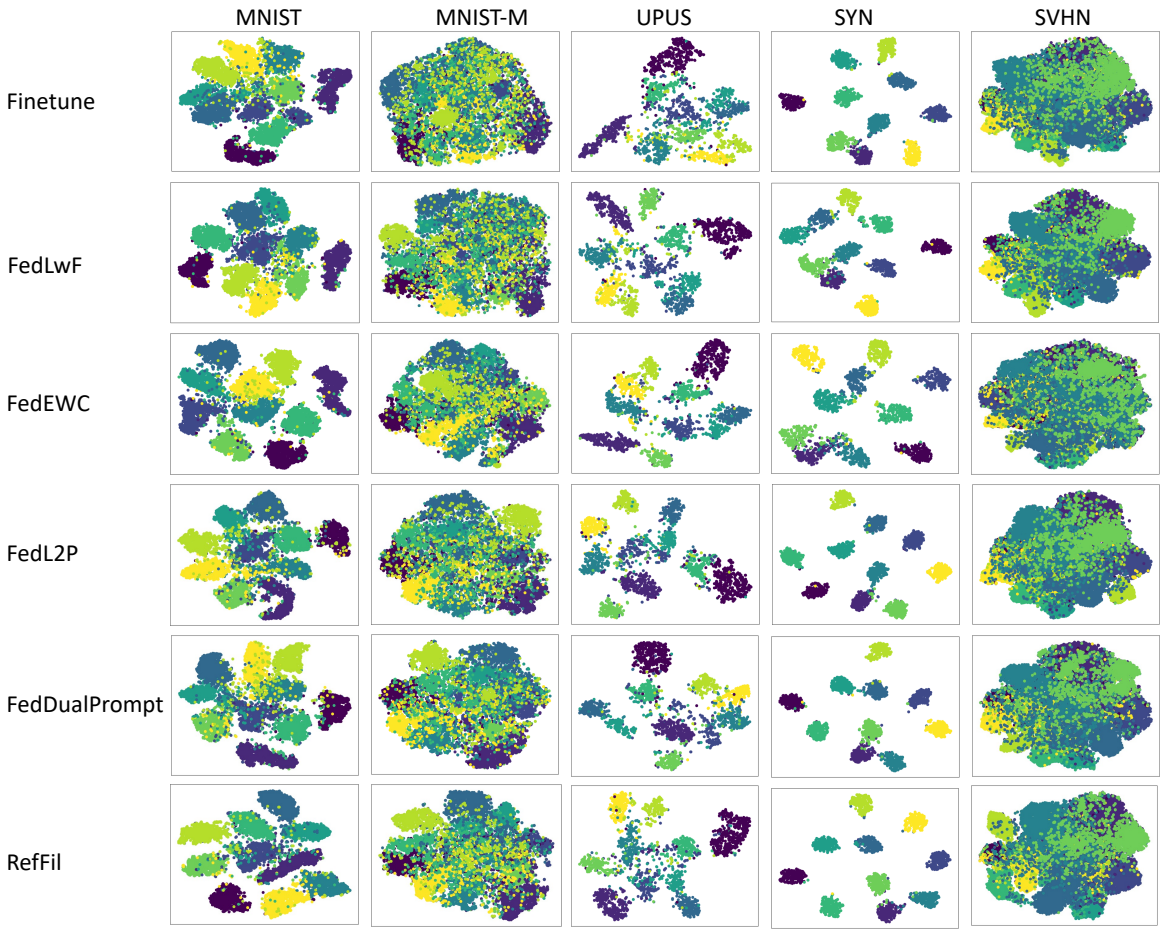

Fig. 6.6 Detailed t-SNE Visualization of Task 5 in Figure 6.5 for Five Baselines and Our RefFiL Model. This figure showcases the global model, updated post Task 5, evaluated separately on datasets from MNIST to SVHN, reflecting the learning progression from Task 1 to 5. Enhanced clarity and diffusion of each cluster's boundaries signify the global model's proficiency in developing generalizable decision boundaries.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has addressed key challenges in federated learning (FL) systems, focusing on improving adaptability and resilience in dynamic, distributed environments. The research spans the intersections of FL, continual learning, and neural architecture search, offering comprehensive solutions to enhance performance, scalability, and efficiency in the application characterized by privacy constraints, device heterogeneity, and evolving data distributions.

In **Chapter 3**, the thesis first tackled the challenge of designing an adaptive model selection system for FL, which balances training efficiency and model performance across hardware-heterogeneous environments (**Research Question 1**). To address this, we introduced FedMSA, a system that utilizes Hardware-aware Neural Architecture Search (HW-NAS) to dynamically select model architectures tailored to the computational capabilities of participating devices. This work enables federated systems to efficiently adapt to diverse hardware, improving overall deployment feasibility.

Building on this foundation, **FedDAS** was developed to further enhance the robustness and operational stability of FL systems. By integrating diagnostics and anomaly detection, FedDAS extends the adaptive capabilities of FedMSA, ensuring that FL processes can remain efficient and reliable even as network conditions and device availability fluctuate.

In **Chapter 4**, the focus shifted to improving the model's ability to retain knowledge during local model updates in FL, particularly under class-incremental learning scenarios (**Research Question 2**). We introduced DVKA, a Dual Variational Knowledge Attention mechanism designed to optimize the balance between retaining existing knowledge and integrating new information in Transformer models. This approach

mitigates catastrophic forgetting, enabling continuous learning at the client level while ensuring effective global model aggregation.

Chapter 5 addressed the challenge of optimizing knowledge retention techniques for FL in resource-constrained environments, particularly focusing on the efficient use of limited exemplar memory (**Research Question 3**). The chapter introduced ECoral, an exemplar-condensed framework that enhances the information density of stored exemplars. ECoral improves data efficiency by ensuring that federated systems can handle class-incremental learning without compromising data privacy or suffering from significant performance degradation due to memory constraints.

Finally, **Chapter 6** tackled the broader issue of handling domain-incremental data in FL, with an emphasis on improving cross-domain prediction robustness and minimizing catastrophic forgetting in resource-constrained settings (**Research Question 4**). The chapter introduced RefFil, a framework that leverages global prompt-sharing and domain-specific contrastive learning to enhance the robustness of local models. RefFil ensures that FL systems can generalize effectively across diverse and evolving data distributions without relying on rehearsal techniques, thus maintaining high performance even as they encounter new domains.

To ensure both comprehensive evaluation and real-world applicability, an extensive experimental setup was designed across three critical dimensions. The orchestration systems (FedMSA and FedDAS) were deployed on real devices, showcasing practical feasibility and cross-platform compatibility—with real-world diagnostic data used for anomaly detection. In parallel, class/domain-incremental learning was rigorously evaluated on a suite of public datasets (e.g., CIFAR-100, TinyImageNet, ImageNet variants, Caltech-256, Digits-Five, OfficeCaltech10, PACS, and FedDomainNet) across varied task settings (ranging from 10 to 50 tasks) to reflect diverse levels of complexity and real-world class distributions. Finally, federated learning experiments simulated dynamic client participation (starting with 10 or 20 clients and incorporating new participants as tasks evolved) and accounted for data heterogeneity by configuring multiple levels of non-IID distributions. These experimental settings underline the robustness and generalizability of the proposed methods, affirming their potential for deployment in real-world FL scenarios.

In synthesizing these contributions, this thesis has introduced novel frameworks and methodologies that significantly advance the state-of-the-art in FL. By addressing critical challenges related to system heterogeneity, continuous learning, data efficiency, and domain shifts, these innovations push the boundaries of what FL systems can achieve. The research provides a strong foundation for future work aimed at further

enhancing the scalability, adaptability, and efficiency of machine learning in complex environments.

7.2 Future research directions

FL, as an open-world scenario and application, still has many critical real-world challenges that need to be solved, both from the machine learning and system sides. Here, we provide a number of areas of future research that can be inspired by the work done in this thesis.

7.2.1 Federated fine-grained model architecture search for evolutionary task

In this thesis, we proposed FedMSA, which integrates neural architecture search (NAS) to find a globally optimal model while accounting for different hardware configurations. However, FedMSA faces limitations in dynamic systems where clients' hardware configurations vary, such as mobile apps that need to detect new classes while maintaining the ability to recognize previous ones. In these scenarios, extending the model architecture to handle new information may result in a structure that is suboptimal for certain clients.

Future work could focus on fine-grained model architecture search at a more granular level, allowing for incremental adjustments to the model's architecture. Instead of searching for a complete model, the system could adapt specific components or layers, ensuring that the model can grow to handle new tasks while remaining efficient for clients with varying hardware. Additionally, techniques such as model partitioning could be explored, where clients with limited resources run a smaller, optimized portion of the model, while more powerful devices run the entire architecture. This approach would enable more flexible, scalable FL in dynamic environments, improving the adaptability and performance of models across heterogeneous client hardware configurations.

7.2.2 Federated novel class discovery in a collaborative manner

In this thesis, we presented ECoral and RefFil, which focus on FL for streaming new class and domain data. While our solution effectively mitigates the catastrophic forgetting problem, it assumes that the FL system is aware of when new classes or domains arrive and how many classes exist. However, in real-world FL applications, especially on edge devices like anomaly detection cameras, the system must continually

detect new behaviors. Although some methods, such as GLFC [28], use entropy to detect the arrival of new classes, this becomes challenging when the amount of new data is limited compared with existing data and its classification remains unclear.

Future work should explore how FL can leverage collaboration among clients to enhance the detection of new data in such resource-constrained environments. By utilizing information from other clients, the system could more accurately identify novel data patterns and adapt the global model to support the recognition of new classes. This collaborative approach could improve the overall system's ability to dynamically adjust to new inputs, ensuring a more robust and adaptive learning process in federated settings.

7.2.3 Efficient federated learning with large model

In this thesis, we mainly focused on conventional neural network-based models like ResNet and MobileNet for FL, except DVKA, as they are more resource-efficient and suitable for edge devices with limited computational power and bandwidth. However, larger transformer-based models, such as BERT [23] and GPT [12], have demonstrated superior performance in recent AI advancements. To enhance the capabilities of edge AI, it is crucial to explore how to efficiently train transformer-based models in federated environments. While these models are more powerful, they are also computationally intensive, presenting challenges in resource-constrained settings.

One promising approach is Federated Split Learning [110], where the model is divided between clients and a central server. Clients only update the first and last layers, while the bulk of the model is trained on the server, offloading significant computation. However, this approach still requires transferring large intermediate outputs, which can strain network bandwidth and pose privacy risks, as these middle states contain detailed features of the data.

Future work should focus on optimizing the training of transformer-based models in FL, specifically by addressing the network overhead and privacy concerns of split learning. This might involve compressing intermediate data or leveraging more efficient communication strategies. Moreover, developing methods to adaptively balance computational load between the client and server while maintaining model performance would enhance the feasibility of deploying large models in federated environments. This could significantly improve the performance of edge AI systems, enabling them to handle complex tasks while minimizing resource constraints.

7.2.4 Asynchronous Federated Continual learning

In the ECoral and RefFil works, we aimed to make the experimental setup more realistic by allowing clients to gradually transition to new tasks, with some retaining both new and old task data. Like most continual learning approaches, we assumed that new data arrives in a sequential manner; for example, all of the clients might learn classes 1 to 10 in order or skip some of the tasks but still follow the same general sequence. However, in real-world FL systems, clients exhibit unique behaviors, and new tasks can start at any time with entirely different data. For instance, client one might classify cats and dogs as new classes, while client two might classify planes, trains, and bikes, with each client moving to new tasks at different, asynchronous times.

This asynchronous and diverse task distribution among clients presents a significant challenge for model aggregation. Simply aggregating all client updates without considering their task misalignment could degrade global model performance, as updates from clients working on different tasks could conflict with one another.

Future work should focus on developing more sophisticated aggregation methods that can account for asynchronous client updates and differing tasks. This may involve adaptive aggregation strategies that weigh model updates based on task similarity or dynamically adjust the global model to accommodate varying client progress. Such approaches will be crucial for improving the robustness and flexibility of FL in real-world, asynchronous settings.

7.2.5 Machine Unlearning for long-term federated learning

Although the works presented in this thesis aim to address long-term, continuous FL challenges, they do not consider scenarios where clients may quit the training process. In real-world FL, it is unrealistic to assume that all clients will remain motivated to contribute throughout the entire task. This raises a significant challenge: *once a client leaves, how can we remove or clean up the model knowledge that was contributed by this client's data?*

This challenge aligns with the emerging field of machine unlearning, which focuses on selectively removing the influence of specific data from trained models. In FL, this becomes more complex due to the decentralized nature and diverse data distributions across clients, where the model knowledge is aggregated without direct access to the underlying data.

Future work should explore federated unlearning techniques that efficiently remove the influence of data contributed by departing clients without degrading the model's

overall performance. One approach could involve developing mechanisms that track client-specific updates or contributions, allowing targeted removal of those influences. Addressing this challenge will be essential for ensuring data privacy and maintaining model integrity in dynamic FL environments.

7.2.6 Real-World deployment conditions and system requirements.

A key future research direction is to investigate the realistic conditions required for deploying federated learning models in practice. In real-world scenarios, federated systems must operate over a diverse range of hardware architectures, from ARM-based CPUs to x86 systems and GPUs. For instance, incorporating adaptive components like FedMSA's platform detector could automatically adjust the model architecture based on the specific computing resources available on each edge device. Additionally, intermittent connectivity and client dropouts present significant challenges in realistic network environments. Techniques similar to FedDAS, which provide real-time diagnosis and adaptation to fluctuating network conditions, are essential to ensure robust model updates despite these connectivity issues.

7.2.7 Balancing privacy, adaptability, and resource constraints.

Another critical aspect for real-world deployment is ensuring strict privacy compliance and model adaptability under dynamic data conditions. Federated learning models, such as ECoral and RefFil, address privacy concerns by either using condensed exemplars or eliminating storage of raw data, thereby aligning with regulations like GDPR. However, beyond privacy, models must continuously adapt to evolving data streams without suffering from catastrophic forgetting. Future work should explore methods that balance these competing demands by, for example, leveraging global prompt sharing and contrastive learning for domain-invariant feature extraction while ensuring efficient computation. Addressing these realistic conditions—hardware diversity, intermittent connectivity, privacy compliance, and model adaptability—will help bridge the gap between current research and practical deployment in diverse, real-world environments.

References

[1] Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2016). Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

[2] Aljundi, R., Chakravarty, P., and Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375.

[3] Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019). Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32.

[4] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

[5] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[6] Babakniya, S., Fabian, Z., He, C., Soltanolkotabi, M., and Avestimehr, S. (2024). A data-free approach to mitigate catastrophic forgetting in federated class incremental learning for vision tasks. *Advances in Neural Information Processing Systems*, 36.

[7] Belouadah, E. and Popescu, A. (2019). Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 583–592.

[8] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

[9] Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., and Lane, N. D. (2020). Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*.

[10] Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

[11] Bonicelli, L., Boschini, M., Porrello, A., Spampinato, C., and Calderara, S. (2022). On the effectiveness of lipschitz-driven rehearsal in continual learning. *arXiv preprint arXiv:2210.06443*.

[12] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[13] Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. (2018). Understanding disentangling in -vae. *arXiv preprint arXiv:1804.03599*.

[14] Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930.

[15] Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision*, pages 233–248.

[16] Cazenavette, G., Wang, T., Torralba, A., Efros, A. A., and Zhu, J.-Y. (2022). Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4750–4759.

[17] Cha, H., Lee, J., and Shin, J. (2021). Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9516–9525.

[18] Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. (2018). Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547.

[19] Cho, W., Park, J., and Kim, T. (2023). Complementary domain adaptation and generalization for unsupervised continual domain shift learning. *arXiv preprint arXiv:2303.15833*.

[20] De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of operations research*, 134:19–67.

[21] De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

[22] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[23] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[24] Dhar, S., Guo, J., Liu, J., Tripathi, S., Kurup, U., and Shah, M. (2021). A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Transactions on Internet of Things*, 2(3):1–49.

[25] Diao, E., Ding, J., and Tarokh, V. (2020). Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*.

[26] Díaz-Rodríguez, N., Lomonaco, V., Filliat, D., and Maltoni, D. (2018). Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*.

[27] Dong, J., Cong, Y., Sun, G., Zhang, Y., Schiele, B., and Dai, D. (2023a). No one left behind: Real-world federated class-incremental learning. *arXiv preprint arXiv:2302.00903*.

[28] Dong, J., Wang, L., Fang, Z., Sun, G., Xu, S., Wang, X., and Zhu, Q. (2022). Federated class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10164–10173.

[29] Dong, J., Zhang, D., Cong, Y., Cong, W., Ding, H., and Dai, D. (2023b). Federated incremental semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3934–3943.

[30] Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*.

[31] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[32] Douillard, A., Cord, M., Ollion, C., Robert, T., and Valle, E. (2020). Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 86–102. Springer.

[33] Douillard, A., Ramé, A., Couairon, G., and Cord, M. (2022). Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9285–9295.

[34] Duan, H., Long, Y., Wang, S., Zhang, H., Willcocks, C. G., and Shao, L. (2023). Dynamic unary convolution in transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[35] Duan, H., Wang, S., and Guan, Y. (2020). Sofa-net: Second-order and first-order attention network for crowd counting. *arXiv preprint arXiv:2008.03723*.

[36] Duan, H., Wang, S., Ojha, V., Wang, S., Huang, Y., Long, Y., Ranjan, R., and Zheng, Y. (2024). Wearable-based behaviour interpolation for semi-supervised human activity recognition. *Information Sciences*, 665:120393.

[37] Ermis, B., Zappella, G., Wistuba, M., Rawal, A., and Archambeau, C. (2022a). Continual learning with transformers for image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3774–3781.

[38] Ermis, B., Zappella, G., Wistuba, M., Rawal, A., and Archambeau, C. (2022b). Memory efficient continual learning with transformers. *Advances in Neural Information Processing Systems*, 35:10629–10642.

[39] European Parliament and Council of the European Union (2016). Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). Official Journal of the European Union, L 119, 4 May 2016, pp. 1-88. https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[40] Fan, Z., Hu, G., Sun, X., Wang, G., Dong, J., and Su, C. (2022). Self-attention neural architecture search for semantic image segmentation. *Knowledge-Based Systems*, 239:107968.

[41] Fedorov, I., Adams, R. P., Mattina, M., and Whatmough, P. (2019). Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers. *Advances in Neural Information Processing Systems*, 32.

[42] Ghiasi, G., Lin, T.-Y., and Le, Q. V. (2019). Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7036–7045.

[43] Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. (2020). An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597.

[44] Goetz, J. and Tewari, A. (2020). Federated learning via synthetic data. *arXiv preprint arXiv:2008.04489*.

[45] Griffin, G., Holub, A., Perona, P., et al. (2007). Caltech-256 object category dataset. Technical report, Technical Report 7694, California Institute of Technology Pasadena.

[46] Gu, J., Wang, K., Jiang, W., and You, Y. (2024). Summarizing stream data for memory-constrained online continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12217–12225.

[47] Guan, C., Wang, X., Zhang, Q., Chen, R., He, D., and Xie, X. (2019). Towards a deep and unified understanding of deep neural models in nlp. In *International Conference on Machine Learning*, pages 2454–2463. PMLR.

[48] Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24(12):1028–1040.

[49] Han, J., Kamber, M., and Pei, J. (2006). Data mining: concepts and techniques morgan kaufmann. *San Francisco*.

[50] Hanzely, F., Hanzely, S., Horváth, S., and Richtárik, P. (2020). Lower bounds and optimal algorithms for personalized federated learning. *Advances in Neural Information Processing Systems*, 33:2304–2315.

[51] He, C., Li, S., So, J., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M., and Avestimehr, S. (2020). Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*.

[52] He, C., Mushtaq, E., Ding, J., and Avestimehr, S. (2021a). Fednas: Federated deep learning via neural architecture search.

[53] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[54] He, X., Zhao, K., and Chu, X. (2021b). Automl: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622.

[55] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

[56] Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3.

[57] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[58] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[59] Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 831–839.

[60] Hsu, T.-M. H., Qi, H., and Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.

[61] Hu, S., Goetz, J., Malik, K., Zhan, H., Liu, Z., and Liu, Y. (2022). Fedsynth: Gradient compression via synthetic data in federated learning. *arXiv preprint arXiv:2204.01273*.

[62] Huang, W., Ye, M., and Du, B. (2022). Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10143–10153.

[63] Huang, W., Ye, M., Shi, Z., Li, H., and Du, B. (2023). Rethinking federated learning with domain shift: A prototype view. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16312–16322. IEEE.

[64] Insider, B. (2020.05). The internet of things 2020: Here's what over 400 iot decision-makers say about the future of enterprise connectivity and how iot companies can use it to grow revenue. [Online]. https://www.businessinsider.com/internet-of-things-report?r=US&IR=T.

[65] Jiang, Y., Ma, B., Wang, X., Yu, G., Yu, P., Wang, Z., Ni, W., and Liu, R. P. (2024). Blockchained federated learning for internet of things: A comprehensive survey. *ACM Computing Surveys*, 56(10):1–37.

[66] Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K., and Tassiulas, L. (2022). Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10374–10386.

[67] Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

[68] Jung, D., Han, D., Bang, J., and Song, H. (2023). Generating instance-level prompts for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11847–11857.

[69] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210.

[70] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR.

[71] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

[72] Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. (2022). Transformers in vision: A survey. *ACM Computing Surveys*, 54(10s):1–41.

[73] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[74] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

[75] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[76] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[77] Lai, Q., Li, Y., Zeng, A., Liu, M., Sun, H., and Xu, Q. (2021). Information bottleneck approach to spatial attention learning. *arXiv preprint arXiv:2108.03418*.

[78] Laptev, N., Amizadeh, S., and Flint, I. (2015). Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1939–1947.

[79] Le, Y. and Yang, X. (2015). Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3.

[80] Li, C., Yu, Z., Fu, Y., Zhang, Y., Zhao, Y., You, H., Yu, Q., Wang, Y., and Lin, Y. (2021a). Hw-nas-bench: Hardware-aware neural architecture search benchmark. *arXiv preprint arXiv:2103.10584*.

[81] Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. (2017). Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550.

[82] Li, M., Andersen, D. G., Smola, A. J., and Yu, K. (2014). Communication efficient distributed machine learning with the parameter server. *Advances in Neural Information Processing Systems*, 27.

[83] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450.

[84] Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

[85] Li, Z., Zhong, C., Liu, S., Wang, R., and Zheng, W.-S. (2021b). Preserving earlier knowledge in continual learning with the help of all previous feature extractors. *arXiv preprint arXiv:2104.13614*.

[86] Lin, J., Chen, W., Cai, H., Gan, C., and Han, S. (2021). Mcunetv2: Memory-efficient patch-based inference for tiny deep learning. arxiv. *arXiv preprint arXiv:2110.15352*.

[87] Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., and Fei-Fei, L. (2019). Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 82–92.

[88] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2018a). Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34.

[89] Liu, H., Simonyan, K., and Yang, Y. (2018b). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

[90] Liu, P., Yu, X., and Zhou, J. T. (2022). Meta knowledge condensation for federated learning. *arXiv preprint arXiv:2209.14851*.

[91] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

[92] Liu, Y., Fan, T., Chen, T., Xu, Q., and Yang, Q. (2021). Fate: An industrial grade platform for collaborative learning with data protection. *Journal of Machine Learning Research*, 22(226):1–6.

[93] Liu, Y., Su, Y., Liu, A.-A., Schiele, B., and Sun, Q. (2020). Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 12245–12254.

[94] Long, G., Tan, Y., Jiang, J., and Zhang, C. (2020). Federated learning for open banking. In *Federated learning: privacy and incentive*, pages 240–254. Springer.

[95] Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

[96] Lu, N., Wang, Z., Li, X., Niu, G., Dou, Q., and Sugiyama, M. (2022). Federated learning from only unlabeled data with class-conditional-sharing clients. *arXiv preprint arXiv:2204.03304*.

[97] Luo, Z., Liu, Y., Schiele, B., and Sun, Q. (2023). Class-incremental exemplar compression for class-incremental learning. *arXiv preprint arXiv:2303.14042*.

[98] Ma, Y., Xie, Z., Wang, J., Chen, K., and Shou, L. (2022). Continual federated learning based on knowledge distillation. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, volume 3.

[99] Masarczyk, W. and Tautkute, I. (2020). Reducing catastrophic forgetting with learning on synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 252–253.

[100] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017a). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

[101] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017b). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

[102] Meiseles, A. and Rokach, L. (2020). Source model selection for deep learning in the time series domain. *IEEE Access*, 8:6190–6200.

[103] Mellor, J., Turner, J., Storkey, A., and Crowley, E. J. (2021). Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR.

[104] Mirza, M. J., Masana, M., Possegger, H., and Bischof, H. (2022). An efficient domain-incremental learning approach to drive in all weather conditions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3001–3011.

[105] Nekrasov, V., Chen, H., Shen, C., and Reid, I. (2019). Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9126–9135.

[106] Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., and Poor, H. V. (2021). Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658.

[107] Nguyen, D. C., Pham, Q.-V., Pathirana, P. N., Ding, M., Seneviratne, A., Lin, Z., Dobre, O., and Hwang, W.-J. (2022). Federated learning for smart healthcare: A survey. *ACM Computing Surveys (Csur)*, 55(3):1–37.

[108] Nishio, T. and Yonetani, R. (2019). Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE.

[109] 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.

[110] Pathak, R. and Wainwright, M. J. (2020). Fedsplit: An algorithmic framework for fast federated optimization. *Advances in neural information processing systems*, 33:7057–7066.

[111] Pelosin, F., Jha, S., Torsello, A., Raducanu, B., and van de Weijer, J. (2022). Towards exemplar-free continual learning in vision transformers: an account of attention, functional and weight regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3820–3829.

[112] Peng, J., Sun, M., ZHANG, Z.-X., Tan, T., and Yan, J. (2019a). Efficient neural architecture transformation search in channel-level for object detection. *Advances in Neural Information Processing Systems*, 32.

[113] Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. (2019b). Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415.

[114] Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[115] Pham, Q., Liu, C., and Hoi, S. (2021). Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34:16131–16144.

[116] Pokhrel, S. R. and Choi, J. (2020a). A decentralized federated learning approach for connected autonomous vehicles. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6. IEEE.

[117] Pokhrel, S. R. and Choi, J. (2020b). A decentralized federated learning approach for connected autonomous vehicles. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6. IEEE.

[118] Pokhrel, S. R. and Choi, J. (2020c). Federated learning with blockchain for autonomous vehicles: Analysis and design challenges. *IEEE Transactions on Communications*, 68(8):4734–4746.

[119] Qi, D., Zhao, H., and Li, S. (2023). Better generative replay for continual federated learning. *arXiv preprint arXiv:2302.13001*.

[120] Rajasegaran, J., Hayat, M., Khan, S., Khan, F. S., Shao, L., and Yang, M.-H. (2019). An adaptive random path selection approach for incremental learning. *arXiv preprint arXiv:1906.01120*.

[121] Ramachandram, D. and Taylor, G. W. (2017). Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34(6):96–108.

[122] Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789.

[123] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

[124] Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., and Wang, X. (2021). A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34.

[125] Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. (2019). Experience replay for continual learning. *Advances in neural information processing systems*, 32.

[126] Rosasco, A., Carta, A., Cossu, A., Lomonaco, V., and Bacciu, D. (2021). Distilled replay: Overcoming forgetting through synthetic samples. In *International Workshop on Continual Semi-Supervised Learning*, pages 104–117. Springer.

[127] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[128] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

[129] Sarfraz, S., Sharma, V., and Stiefelhagen, R. (2019). Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8934–8943.

[130] Schrod, S., Lippl, J., Schäfer, A., and Altenbuchinger, M. (2023). Fact: Federated adversarial cross training. *arXiv preprint arXiv:2306.00607*.

[131] Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR.

[132] Shanahan, M., Kaplanis, C., and Mitrović, J. (2021). Ensembles and encoders for task-free continual learning.

[133] Shwartz-Ziv, R. and Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

[134] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

[135] Song, F., Li, L., You, I., Yu, S., and Zhang, H. (2022). Optimizing high-speed mobile networks with smart collaborative theory. *IEEE Wireless Communications*, 29(3):48–54.

[136] Song, Y.-Y. and Ying, L. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130.

[137] Sun, R., Duan, H., Dong, J., Ojha, V., Shah, T., and Ranjan, R. (2024a). Rehearsal-free federated domain-incremental learning. *arXiv preprint arXiv:2405.13900*.

[138] Sun, R., Li, Y., Shah, T., Sham, R. W., Szydlo, T., Qian, B., Thakker, D., and Ranjan, R. (2022). Fedmsa: a model selection and adaptation system for federated learning. *Sensors*, 22(19):7244.

[139] Sun, R., Zhang, Y., Shah, T., Sun, J., Zhang, S., Li, W., Duan, H., Wei, B., and Ranjan, R. (2024b). From sora what we can see: A survey of text-to-video generation. *arXiv preprint arXiv:2405.10674*.

[140] Sun, Y., Chong, N., and Ochiai, H. (2023). Feature distribution matching for federated domain generalization. In *Asian Conference on Machine Learning*, pages 942–957. PMLR.

[141] T Dinh, C., Tran, N., and Nguyen, J. (2020). Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405.

[142] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828.

[143] Taylor, B., Marco, V. S., Wolff, W., Elkhatib, Y., and Wang, Z. (2018). Adaptive deep learning model selection on embedded systems. *ACM SIGPLAN Notices*, 53(6):31–43.

[144] Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

[145] Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., and Jégou, H. (2021). Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42.

[146] Van de Ven, G. M., Tuytelaars, T., and Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197.

[147] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[148] Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeyer, J. S. (2020). A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2):1–33.

[149] Wan, F., Miao, X., Duan, H., Deng, J., Gao, R., and Long, Y. (2024). Sentinel-guided zero-shot learning: A collaborative paradigm without real data exposure. *arXiv preprint arXiv:2403.09363*.

[150] Wang, J., Feng, W., Chen, Y., Yu, H., Huang, M., and Yu, P. S. (2018a). Visual domain adaptation with manifold embedded distribution alignment. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 402–410.

[151] Wang, K., Zhao, B., Peng, X., Zhu, Z., Yang, S., Wang, S., Huang, G., Bilen, H., Wang, X., and You, Y. (2022a). Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12196–12205.

[152] Wang, L., Zhang, X., Su, H., and Zhu, J. (2023). A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*.

[153] Wang, L., Zhang, X., Yang, K., Yu, L., Li, C., Hong, L., Zhang, S., Li, Z., Zhong, Y., and Zhu, J. (2022b). Memory replay with data compression for continual learning. *arXiv preprint arXiv:2202.06592*.

[154] Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. (2018b). Dataset distillation. *arXiv preprint arXiv:1811.10959*.

[155] Wang, Z., Liu, L., Duan, Y., Kong, Y., and Tao, D. (2022c). Continual learning with lifelong vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 171–181.

[156] Wang, Z., Liu, L., Kong, Y., Guo, J., and Tao, D. (2022d). Online continual learning with contrastive vision transformer. In *Proceedings of the European Conference on Computer Vision*, pages 631–650. Springer.

[157] Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.-Y., Ren, X., Su, G., Perot, V., Dy, J., et al. (2022e). Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer.

[158] Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. (2022f). Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149.

[159] Wei, J., Chu, X., Sun, X.-Y., Xu, K., Deng, H.-X., Chen, J., Wei, Z., and Lei, M. (2019). Machine learning in materials science. *InfoMat*, 1(3):338–358.

[160] Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q., and Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469.

[161] Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. (2019a). Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742.

[162] Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. (2019b). Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382.

[163] Xie, C., Koyejo, S., and Gupta, I. (2019). Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*.

[164] Xiong, Y., Wang, R., Cheng, M., Yu, F., and Hsieh, C.-J. (2023). Feddm: Iterative distribution matching for communication-efficient federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16323–16332.

[165] Xu, J., Glicksberg, B. S., Su, C., Walker, P., Bian, J., and Wang, F. (2021a). Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19.

[166] Xu, J., Glicksberg, B. S., Su, C., Walker, P., Bian, J., and Wang, F. (2021b). Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19.

[167] Yan, S., Xie, J., and He, X. (2021). Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023.

[168] Yang, E., Shen, L., Wang, Z., Liu, T., and Guo, G. (2023). An efficient dataset condensation plugin and its application to continual learning. *Advances in Neural Information Processing Systems*, 36.

[169] Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019a). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.

[170] Yang, W., Zhang, Y., Ye, K., Li, L., and Xu, C.-Z. (2019b). Ffd: A federated learning based method for credit card fraud detection. In *International conference on big data*, pages 18–32. Springer.

[171] Yang, W., Zhang, Y., Ye, K., Li, L., and Xu, C.-Z. (2019c). Ffd: A federated learning based method for credit card fraud detection. In *International conference on big data*, pages 18–32. Springer.

[172] Yegnanarayana, B. (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.

[173] Yoon, J., Jeong, W., Lee, G., Yang, E., and Hwang, S. J. (2021). Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pages 12073–12086. PMLR.

[174] Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2017). Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

[175] Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. (2018). Slimmable neural networks. *arXiv preprint arXiv:1812.08928*.

[176] Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., and Liu, Y. (2020a). {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 493–506.

[177] Zhang, J., Chen, C., Zhuang, W., and Lyu, L. (2023). Target: Federated class-continual learning via exemplar-free distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4782–4793.

[178] Zhang, L. L., Yang, Y., Jiang, Y., Zhu, W., and Liu, Y. (2020b). Fast hardware-aware neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 692–693.

[179] Zhao, B. and Bilen, H. (2023). Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523.

[180] Zhao, B., Mopuri, K. R., and Bilen, H. (2020a). Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*.

[181] Zhao, B., Xiao, X., Gan, G., Zhang, B., and Xia, S.-T. (2020b). Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13208–13217.

[182] Zhou, D.-W., Wang, Q.-W., Qi, Z.-H., Ye, H.-J., Zhan, D.-C., and Liu, Z. (2023). Deep class-incremental learning: A survey. *arXiv preprint arXiv:2302.03648*.

[183] Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.