SCALABLE CONTINUAL LEARNING FOR NEURAL NETWORK-BASED MODELS

YUQING ZHAO

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University
Department of Computing

Scalable Continual Learning for Neural Network-Based Models

Yuqing ZHAO

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
May 2025

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Yuqing ZHAO

Abstract

Neural networks serve as the powerful foundation of modern AI, achieving remarkable success in areas like vision, language, and robotics due to their ability to learn complex patterns. However, they have a significant drawback: they are static. When confronted with new data in a changing environment, they tend to forget previous knowledge catastrophically. To learn a new task, they must be retrained from the beginning, which is often inefficient and impractical. This is where continual learning comes into play as a promising direction. Continual learning aims to enable models to learn continuously from a stream of data without the need for retraining. But for continual learning to transition from a lab concept to a real-world technology, it must be scalable, which means that a model's performance should not degrade catastrophically as the number of datasets or the duration of learning increases. Scalability can be broken down into three essential requirements: seamless retention of accuracy (being task agnostic), robustness to heterogeneous data, and the ability to adapt sustainably over the long term. Existing continual learning methods have limited scalability as they primarily focus on accuracy preservation, often sacrificing other requirements. Rehearsal and regularization methods help reduce forgetting by storing past data or constraining weight updates. While they are task-agnostic, they struggle with long-term accuracy and data heterogeneity. In contrast, structural methods add components for new tasks and provide high accuracy but require manual task annotations and are resource-intensive, making them unsuitable for long-term use. Overall, existing methods struggle to meet all three requirements, which is the gap

this thesis aims to address.

This gap arises from three fundamental challenges presented by real-world data streams. Current research primarily focuses on the first challenge: the issue of non-IID (Independent and Identically Distributed) data, which leads to catastrophic forgetting. This problem has not yet been sufficiently resolved to ensure seamless retention of accuracy. The second challenge is that neural networks must manage the heterogeneity of datasets, which may vary in size, similarity, and complexity. Finally, to support sustainable long-term adaptation, neural networks must be able to handle incremental datasets. The goal of my PhD is to comprehensively tackle these challenges and facilitate scalable continual learning for neural network-based models, as summarized below.

Addressing the first forgetting challenge of non-IID datasets, I drew inspiration from the neural reuse principle in the brain and introduced a parameter reuse algorithm featuring task-agnostic parameter isolation to overcome catastrophic forgetting. This algorithm involves isolating key neural network parameters by freezing their gradient updates during training. It allows the remaining parameters to adapt to new data while retaining previously acquired knowledge. I also extended this problem to the more complex Continual Federated Learning setting, where data is non-IID not only over time but also across clients. This scenario introduces both temporal forgetting and spatial bias. To address these issues, we developed FedDistill, which employs group distillation to reduce bias for underrepresented classes and incorporates a hybrid local-global architecture to combat forgetting.

Furthermore, to tackle heterogeneity in sequential datasets, this thesis introduces a novel adaptive continual learning method. Specifically, it employs fine-grained data-driven pruning to adapt to variations in data complexity and dataset size. It also combines previously proposed task-agnostic parameter isolation to mitigate the impact of varying degrees of catastrophic forgetting caused by differences in data similarity. This study builds several challenging scenarios in which the data domains

and classes are incremental and conducts experiments to test the performance of the proposed algorithms.

Finally, to enable sustainable long-term adaptability over incremental new data, this thesis analyzes model growth, in which a pre-trained neural network model grows in parameters to cope with limited network capacity. However, improper model growth can lead to severe degradation of previously learned knowledge, an issue this thesis identifies and names as growth-induced forgetting (GIFt). I propose a novel sparse model growth approach that employs data-driven sparse layer expansion and on-data initialization to overcome the issue of GIFt while enhancing adaptability over new data.

Publications Arising from the Thesis

1. Yuqing Zhao, Divya Saxena, and Jiannong Cao (2022). "Memory-Efficient Domain Incremental Learning for Internet of Things", in *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems (SenSys) workshop, pp. 1175–1181* .

2. Yuqing Zhao, Divya Saxena, and Jiannong Cao (2023). "AdaptCL: Adaptive Continual Learning for Tackling Heterogeneity in Sequential Datasets", in *IEEE Transactions on Neural Networks and Learning Systems (TNNLS).*

3. Yuqing Zhao, Jiannong Cao, Divya Saxena, Xiaoyun Liu, Changlin Song, Bo Yuan,and Julie McCann. "SparseGrow: Addressing Growth-Induced Forgetting in Task-Agnostic Continual Learning", manuscript submitted to *Knowledge Based Systems (KBS).*

4. Changlin Song, Divya Saxena, Jiannong Cao and Yuqing Zhao*, "FedDistill: Global Model Distillation for Local Model De-Biasing in Non-IID Federated Learning", manuscript submitted to *IEEE Transactions on Neural Networks and Learning Systems (TNNLS).*

5. Xiaoyun Liu, Divya Saxena, Jiannong Cao, Yuqing Zhao, and Penghui Ruan, "MGAS: Multi-Granularity Architecture Search for Effective and Efficient Neu-

ral Networks", manuscript submitted to *IEEE Transactions on Neural Networks and Learning Systems (TNNLS).*

6. Divya Saxena, Jiannong Cao, Tarun Kulshrestha, and Yuqing Zhao "MGAS: Dynamic Generative Adaptation for Data-Efficient GAN", in *2025 International Joint Conference on Neural Networks (IJCNN 2025).*

7. Xiaoyun Liu, Divya Saxena, Jiannong Cao, Yuqing Zhao, and Penghui Ruan, "MGAS: Fine-Grained Knowledge-Aware Compression for Large Language Models", manuscript submitted to *2025 International Conference on Computer Vision (ICCV2025).*

8. Wenhao Li, Wei Li, Yuqing Zhao, Yinghui Wang, and Xiaowu Zhang. "Parameter-Efficient Continual Learning Network via Kernel Generation", manuscript submitted to *the Fortieth AAAI Conference on Artificial Intelligence (AAAI-26).*

Acknowledgments

Throughout my Ph.D. journey, I have navigated a path filled with enriching experiences, encountering both highs and lows. I have weathered moments of despair and the temptation of giving up, only to later reignite my determination. This doctoral phase has been instrumental in reshaping my values: transitioning from reliance on others for information to proactively seeking knowledge independently, shifting my focus from methods to goal-oriented approaches, and fostering a deeper sense of self-assurance.

I have cherished the time spent pursuing my Ph.D. at PolyU and the vibrant atmosphere of our IMCL lab. I am deeply grateful to Professor Jiannong Cao for providing me with this life-changing opportunity and entrusting me with the freedom to explore my interests and research directions. I am also thankful that he recognized my potential and offered me the chance to pursue a Ph.D. despite my initial lack of foundation when I crossed paths with him during a workshop in Chengdu. I extend my heartfelt gratitude to Professor Cao, my dedicated supervisor, whose guidance has been invaluable throughout my research.

I want to express my heartfelt gratitude to my family for their unwavering encouragement. I fondly remember my beloved uncle, whose caring nature brought us happiness and created cherished memories. Their constant support has been a pillar of strength for me, and I am truly indebted to them for their love and companionship throughout this journey.

I am deeply thankful to Dr. Divya Saxena, whose guidance has helped my academic and professional growth. With her meticulous suggestions, I have evolved from a newbie in academia and computer science to an expert in my field. Dr. Saxena's intelligence, logical approach, and attention to detail have made our collaborative years not only productive but also enjoyable.

To my fellow lab members, I extend my appreciation for their friendship and collaborative spirit. They are a group of exceptional individuals—kind, supportive, and always ready to lend a helping hand and provide valuable insights. It has been an honor to work alongside such outstanding colleagues.

I am also grateful for the insights and mentorship provided by Professor Julie McCann, who has generously shared her guidance and kindly encouraged me on my paper rebuttal during my research visit at the AESE lab of Imperial College.

Lastly, I wish to express my gratitude to all those who have offered their assistance in the past. Your contributions have played a significant role in shaping my journey and reaching the milestones that stand before me today. Your support has been invaluable, and I am truly grateful for the impact you have had on my academic pursuits. This acknowledgement is a reflection of the profound impact each of you has had on my academic and personal growth. Thank you for being part of this transformative journey.

Table of Contents

Abstract **i**

Publications Arising from the Thesis **iv**

Acknowledgments **vi**

List of Figures **xiv**

List of Tables **xxii**

1 Introduction **1**

1.1 Neural Networks: Advantages and Limitations 1

1.2 Scalable Continual Learning 2

1.3 Research Framework 6

1.4 Research Scope 7

1.5 Thesis Organization 9

2 Literature Review **13**

2.1 Tackling Catastrophic Forgetting 13

2.1.1 Activation Strategies 13
2.1.2 Penalization Strategies 14
2.2 Tackling bias 16
2.2.1 Non-IID Federated Learning (FL) 16
2.2.2 Forgetting in Non-iid Federated Learning. 17
2.2.3 Federated Learning with local model de-biasing. 18
2.3 Tackling Heterogeneity 19
2.3.1 Rehearsal-Based 19
2.3.2 Regularization-Based 20
2.3.3 Structure-Based 21
2.4 Enlarging Model Capacity 23
2.4.1 Lateral Connection 23
2.4.2 Layer Expansion 24
2.4.3 In-Depth Growth 25
2.4.4 Growth-Induced Forgetting 26

3 Research Contribution 1a: Overcoming Catastrophic Forgetting in Non-IID Task-Agnostic Continual Learning 28

3.1 Introduction 28
3.2 Related Works 30
3.3 Methodology 32
3.3.1 Problem setting and objective 32
3.3.2 Task-agnostic parameter isolation 33

3.3.3 Data-driven pruning 34
3.4 Experiments 36
3.4.1 Datasets 36
3.4.2 Evaluation metrics 38
3.4.3 State-of-the-art baselines 39
3.4.4 Implementation details 39
3.5 Results and discussions 40
3.6 Chapter Summary 45

4 Research Contribution 1b: Overcoming Catastrophic Forgetting in Non-IID Task-Agnostic Continual Federated Learning 46

4.1 Introduction 46
4.2 Related Works 52
4.3 Method 55
4.3.1 Preliminary 55
4.3.2 Understanding Forgetting in Federated Learning 55
4.3.3 FedDistill Framework 58
4.4 Experiments and Results 64
4.4.1 Experimental Setup 64
4.4.2 Performance Analysis 68
4.4.3 Communication Efficiency 71
4.4.4 Effect of group distillation 72
4.4.5 Ablation Study 76

4.5 Chapter Summary 81

5 Research Contribution 2: Adaptive Continual Learning for Tackling Heterogeneity in Sequential Datasets 82

5.1 Introduction 82

5.2 Related Works 85

5.2.1 Task-Agnostic Continual Learning 86

5.2.2 Task-Specific Continual Learning 88

5.3 Problem setting and objective 89

5.4 Adaptive Continual Learning 90

5.4.1 Fine-Grained Data-Driven Pruning 91

5.4.2 Task-Agnostic Parameter Isolation 93

5.4.3 Adaptive Continual Learning Training Flow 95

5.5 Experiments 95

5.5.1 Datasets 96

5.5.2 Networks Used 100

5.5.3 Evaluation Metrics 100

5.5.4 Baselines 101

5.5.5 Implementation Details 102

5.6 Results 103

5.6.1 Performance on Datasets with Varied Size (Q1,Q2,Q3) 103

5.6.2 Performance on Datasets with Varied Complexity (Q1,Q2,Q3) 107

5.6.3 Performance on Datasets with Varied Similarity (Q1,Q4) . . . 110

5.6.4 Performance on Different Networks (Q1) 115
5.6.5 Ablation Study (Q3,Q4) 117
5.7 Chapter Summary 119

6 Research Contribution 3: Addressing Growth-Induced Forgetting with SparseGrow in Continual Learning 123

6.1 Introduction 123
6.2 Related Works 127
6.2.1 Growing Approaches 127
6.2.2 Overcoming growth-induced forgetting 130
6.3 Problem Formulation 132
6.4 Methodology 133
6.4.1 Increased Gradient Sparsity 133
6.4.2 Increased Parameter Sparsity 136
6.5 Experimental Setup 140
6.5.1 Datasets 140
6.5.2 Evaluation Metrics 140
6.5.3 Experiment Settings 141
6.5.4 Baselines 141
6.6 Results 143
6.6.1 Comparison of Model Growth Methods 143
6.6.2 Method Scalability and Adaptability 144
6.6.3 Evaluation of Overcoming growth-induced forgetting 147

6.6.4 Class-Incremental Learning Setting Results 148
6.7 Chapter Summary 149

7 Conclusions and Future Directions **155**

7.1 Conclusions 155
7.2 Future Directions 157

References **159**

List of Figures

1.1 Catastrophic forgetting where a model learns task B and forgets task A 2

1.2 (c) Continual learning compared with (a) multi-task learning and (b) finetuning. 3

1.3 Radar chart for the three requirements of scalable continual learning. 4

1.4 Existing continual learning methods. 5

1.5 The three-layer framework of the research on scalable continual learning. 6

3.1 Overview of proposed domain incremental learning in an IoT scenario. It incrementally adapts to domain shifts in a fixed neural network while retaining the accuracy of all learned domains. After deployment, it can make inferences without giving task labels of the domains. 29

3.2 E-DomainIL training process. E-DomainIL can overcome catastrophic forgetting through task-agnostic parameter isolation that freezes learned parameters and allows reusing them later in training. During inference, it uses all parameters, therefore does not require task labels or storing masks. Through data-driven pruning, E-DomainIL can adjust the ratio of the parameters according to the data and maintain a balance between accuracy and parameter efficiency. (Best viewed in color) . . 32

3.3 Examples of MNIST Variants and DomainNet. 37

3.4 Visualization of test accuracy on three MNIST Variant datasets during training using SGD, EWC, PackNet*, and E-DomainIL. (Best viewed in color) 42

3.5 Visualization of test accuracy on three DomainNet datasets during training using SGD, EWC, PackNet*, and E-DomainIL. (Best viewed in color) 43

3.6 Visualization of test accuracy on Food Safety datasets during training using SGD, EWC, PackNet*, and E-DomainIL. (Best viewed in color) 44

4.1 (a) Impact of imbalanced class distribution on a model's gradient updates. Gradient changes in a neural network across four scenarios (s1, s2, s3, s4) for ten classes (Class ID 0-9). The bars indicate gradient magnitudes, with colors representing different scenarios. Significant gradient increases for classes 3 and 9 in s2 and s4 point to shifts in learning focus due to class imbalance. (b) Training on a balanced dataset, the model shows no bias towards any classes. After training on an imbalanced dataset (c), the model exhibits a clear bias with significantly varied probabilities for each class. The elevated probability for certain classes suggests that the model has become more confident in these classes likely due to their overrepresentation in the training data. Conversely, the reduced probability for other classes indicates a loss of confidence, which can be interpreted as the model 'forgetting' or failing to recognize these underrepresented classes. 47

4.2 Illustration of softmax output distribution disparities between global and local models under imbalanced class distribution on client 0, highlighting the local model's inclination towards fitting its specific dataset and the global model's balanced approach. 56

4.3 Overview of the FedDistill framework, detailing the interplay between global and local models' feature extractors and classifiers for an input x_i. Specifically, x_i is the input for the global feature extractor E_g and local feature extractor E_l, respectively. After that, the features will be input into the classifiers FC_g and FC_l of both the global and the local model. $\hat{y}_{gg}$, $\hat{y}_{gl}$ denotes the output of global and local classifier with the global feature, respectively, and $\hat{y}_{lg}$, $\hat{y}_{ll}$ denotes the output of global and local classifier with the local feature, respectively. y_i is the real output. 59

4.4 Illustration of the traditional KD and our G-KD. We reformulate the traditional KD into three parts: (1) The true class KL loss (TC-KD), which has been discussed in DKD[110]. (2) The rich sample KL loss (RC-KD), denotes the KL loss for the rich sample classes that were highly represented by the local modal. (3) The few sample KL loss (FC-KD), denotes the KL loss for the few sample classes that were underrepresented by the local model. By separating the classes, we intended to accommodate the imbalance in the local dataset distribution by adjusting the weight $(\alpha_t, \alpha_r, \alpha_f)$ correspondingly. 60

4.5 Data distribution with different non-iid levels in the first ten clients for all datasets. (a) - (c) demonstrate how non-iid level affects the data sparsity. 65

4.6 Top-1 Accuracy 72

4.7 Neuron Class Preference: The figure illustrates the neuron class preference of the last linear layer of global models trained on CIFAR-10 across various methods. Colors represent the class preference of individual neurons. "DG" refers to the distributed global model from the previous round, "Locals" indicates the local models, and "AG" represents the aggregated global model of the current round. 74

4.8 Weight Divergence: The x-axis represents the data distribution divergence, while the y-axis indicates the weight divergence. A smaller data distribution divergence implies that global models are more general and capable of fitting diverse local data distributions. In contrast, a larger weight divergence signifies that local models deviate further from the global model. 74

4.9 The t-SNE visualization of the features extracted by the feature-extractor of different models from MNIST with non-iid level as 0.1. The black circles denote mixed class boundaries and the blue circles denote clearer class boundaries. Compared to baseline methods, our model yield a clearer class boundary, demonstrating that our model learn a more discriminative feature extractor. 77

4.10 Effects of hyperparameters on the performance of FedDistill. 79

5.1 (a) Traditional parameter isolation methods divide the network into non-interfering modules during inference. However, these methods are limited to task-specific continual learning (aka task incremental learning). They require manual selection of output layers and parameters, resulting in limited generalization and higher parameter usage. (b) AdaptCL achieves task-agnostic parameter isolation by fine-grained data-driven parameter partitioning, enabling high accuracy on heterogeneous datasets without module selection, while also optimizing parameter reuse and saving resources. 83

5.2 The Adaptive Continual Learning (AdaptCL) training flow. It facilitates adaptive learning via fine-grained data-driven pruning to respond effectively to variations in data complexity and dataset size. Additionally, it enables task-agnostic parameter isolation to ensure optimal model performance on datasets ranging in similarity without requiring manual selection of modules. 90

5.3 Examples of input images and size of datasets used in the experiments. (a) Large-Scale, Diverse Binary-Class Food Quality Dataset. (b) Few-Shot, Multi-Class Food Quality Dataset (c) DomainNet comprises datasets with heterogeneous complexity and size. (d) MNIST Variants with heterogeneous similarity. 98

5.4 Test accuracy comparison of continual learning methods on the Large-Scale, Diverse Binary-Class Food Quality Dataset. Our proposed method, AdaptCL, achieves higher average accuracy while consistently preventing catastrophic forgetting in real-world applications with heterogeneous data, outperforming other methods. (Best viewed in color) . . 104

5.5 Results of continual learning methods on the DomainNet that comprises datasets with heterogeneous complexity and size. AdaptCL achieves the best average accuracy and is the most robust to datasets with varied complexity and size. (Best viewed in color) 109

5.6 Test accuracy comparison of continual learning methods on three dissimilar MNIST Variant datasets. Compared with using separated models for learning (SML), AdaptCL achieved comparable AAC results while using only 31.2% of SML's parameters. AdaptCL's ability to achieve minimal forgetting while learning dissimilar datasets, coupled with its parameter efficiency, establishes the effectiveness of our approach. (Best viewed in color) 112

5.7 Visualization of each method's test accuracy training on more similar MNIST Variant datasets. Still, AdaptCL retains the best performance compared with other CL methods. Compared with using separated models for learning (SML), AdaptCL can increase the accuracy on similar unseen datasets via forward knowledge transfer. (Best viewed in color) 113

5.8 Comparison of model keep ratio (1- pruning ratio) between (a) PackNet* model and (b) our model. The diagrams present the change of model remaining ratios during the whole training process (60 epochs). PackNet* manually assigns the same ratio to each dataset. Our model adaptively assigns the corresponding ratio to learning incremental datasets in a data-driven way. It assigns a very high proportion of parameters to learning the first dataset from scratch, and only a small proportion of parameters to generalize existing parameters to the subsequent dataset. 118

5.9 (a) Model on ResNet-18 remaining ratio and sparse accuracy compared with dense accuracy, using $\alpha = 10^{-4}$. (b) Change of model remaining ratio during training on MNIST Variants. (Best viewed in color) . . . 118

5.10 Illustration of the parameter execution patterns for continually trained models on MNIST Variants. Heatmap (a) showcases the firing probability of the xth parameters within the 1st Conv2d layer demonstrated in the x-axis; it illustrates that subsequent datasets maintain and reuse the previous parameters and generalize by adding new connections to them. The heatmap (b) shows the utilization ratio of different layers. 120

5.11 (a) Pruning effect in ResNet-18 for different value of hyper-parameter α. (b) Change of model remaining ratio with ResNet-18 and LeNet-5 for different α. (Best viewed in color) 121

6.1 The three types of model growth methods applied to ResNet and their performance compared to not applying model growth (No Growth) in handling sequential domains. These methods include: 1) Layer Expansion, which widens each layer of the model; 2) Lateral Connection, introducing new lateral layers connected to adjacent layers; and 3) In-Depth Growth, adding hidden layers to increase model depth. (a) Depicts the change in average accuracy following the application of different model growth methods. (b) Shows the average accuracy and backward knowledge transfer of these methods. In this context, Layer Expansion demonstrates the highest average accuracy and the least forgetting in terms of backward transfer, even surpassing the performance of No Growth. (Best viewed in color) 124

6.2 The training flow of SparseGrow involves sparse neural expansion at the structural level and on-data initialization at the weight level at the conclusion of the training process. Initially, model expansion and random initialization are employed to increase model capacity for improved adaptation to new data. Subsequently, growth-induced forgetting is addressed through frozen sparse training and on-dataset fine-tuning. Notably, while 0-initialization of grown parameters prevents growth-induced forgetting, it also stops the update of these parameters. In contrast, the case differs for pruned parameters initialized to 0 value. 133

6.3 Average accuracy comparison of continual learning methods (EWC, LwF, PackNet, PRE-DFKD, and AdaptCL) alongside different model growth techniques (IDGrow, LatConn, LayerExp) and several CL strategies incorporating LayerExp across an increasing number of observed domains. Notably, rehearsal methods like LwF and PRE-DFKD exhibit an initial decline when combined with LayerExp, suggesting potential unsuitability for direct combination with expansion, possibly leading to more MF. Yet, the later positive effects of model capacity enhancement from expansion seem to surpass the negative impact of MF as domains increase. SparseGrow has demonstrated superior performance in knowledge retention, with its effectiveness improving as domain number rises. 145

6.4 Epoch-wise average accuracy of observed domains using different continual learning methods with layer expansion on FreshStale datasets with six sequential domains. 146

6.5 Epoch-wise average accuracy of observed domains using different continual learning methods with layer expansion on DomainNet datasets with four sequential domains. 147

List of Tables

3.1 The number of training dataset and testing dataset sample splits used in the datasets. 36

3.2 Average accuracy ACC (%), backward knowledge transfer BWT (%), forward knowledge transfer FWT (%) and used parameters of E-DomainIL and other baseline methods evaluated on MNIST Variants. 41

3.3 Average accuracy ACC (%), backward knowledge transfer BWT (%), forward knowledge transfer FWT (%) and used parameters of E-DomainIL and other baseline methods evaluated on DomainNet. 43

3.4 Average accuracy ACC (%), backward knowledge transfer BWT (%), forward knowledge transfer FWT (%) and used parameters of E-DomainIL and other baseline methods evaluated on Food Safety datasets. 44

4.1 Configurations of datasets 64

4.2 Top-1 Accuracy (%) of baselines and our method on datasets. The number inside the bracket is the Forgetting measure $\mathcal{F}$. 68

4.3 Communication Efficiency. It demonstrates the number of communication rounds that each approach achieves the final accuracy of FedAvg on the same dataset (The lower the better). 69

4.4 Demonstration of group distillation loss impact on CIFAR10 accuracy, showcasing the method's effectiveness in improving learning outcomes for few-sample classes. 73

4.5 Accuracy (%) of our method with/without $\mathcal{L}_E$ and $\mathcal{L}_{\text{FC}}$ on CIFAR10 with non-iid level $\alpha = 0.5$ 78

5.1 Performance evaluation of continual learning methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and number of used parameters on the Large-Scale, Diverse Binary-Class Food Quality Dataset. Our proposed method, AdaptCL, achieves the best AAC and BWT with fewer parameters compared to other methods. 105

5.2 Comparison of average accuracy (AAC↑), backward knowledge transfer (BWT↑), forward knowledge transfer (FWT↑), Test Accuracy↑, and the number of used parameters (Param$\times 10^7\downarrow$) of various continual learning methods on the Few-Shot, Multi-Class Food Quality Dataset with different training orders. AdaptCL obtains better accuracy than using separated models for learning (SML) on small-size datasets. . . 106

5.3 Results of different continual learning methods on the DomainNet dataset, including their AAC, BWT, FWT, and the number of used parameters. Our proposed method, AdaptCL, demonstrates the best AAC and BWT, indicating its ability to handle datasets with heterogeneous dataset size and complexity. 107

5.4 Comparison of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and the number of used parameters of different continual learning methods on the dissimilar MNIST Variants dataset. Our proposed method, AdaptCL, achieves significantly higher AAC, BWT and FWT with fewer parameters compared to other methods. AdaptCL achieved comparable AAC results while using 31.2% of SML's parameters. 110

5.5 Comparison of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and the number of used parameters of different continual learning methods on more similar MNIST Variant datasets. AdaptCL retains the best performance compared with other CL methods that use one model for learning. . 114

5.6 Results of different continual learning methods applied on LeNet-5 with limited network capacity, including their AAC, BWT, FWT, and the number of used parameters. AdaptCL is also quite effective on LeNet-5, with the best BWT and comparable AAC with PRE-DFKD. It comes in second of AAC after PRE-DFKD, which may be due to less parameter usage. 115

5.7 Results of different continual learning methods applied on VGG-16, including their AAC, BWT, FWT, and the number of used parameters. AdaptCL is still very effective on VGG-18, having the best AAC and BWT with less parameter usage compared to other CL methods. . . 116

6.1 Performance evaluation of different model growth methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and number of used parameters (Param $\times 10^7$) after observing four domains of Permuted MNIST. 143

6.2 Comprehensive performance evaluation of different continual learning methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), and forward knowledge transfer (FWT) on twelve domains of Permuted MNIST. Here we have three groups of comparison: 1) Simple baselines, and 2) SGD with model growth methods IDGrow, LatConn and LayerExp, 3) Baselines with LayerExp,. 152

6.3 Performance evaluation of continual learning methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), and forward knowledge transfer (FWT) on the domain-incremental datasets of FreshStale and DomainNet. 153

6.4 Comparison of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and each dataset's test accuracy using different continual learning methods in the most challenging class-incremental setting. 153

6.5 verage accuracy comparison of continual learning methods (EWC, LwF, PackNet, PRE-DFKD, and AdaptCL) alongside different model growth techniques (IDGrow, LatConn, LayerExp) and several CL strategies incorporating LayerExp across an increasing number of observed domains. 154

6.6 Performance evaluation of continual learning methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and number of used parameters on the Large-Scale, Diverse Binary-Class Food Quality Dataset. 154

Chapter 1

Introduction

1.1 Neural Networks: Advantages and Limitations

Neural networks (NNs), foundations of artificial intelligence, are strong computing systems that learn by training offline on a balanced dataset. Neural network-based models have had great success in recognizing images, helping with diagnoses, autonomous driving, etc. Neural networks can be beyond human capacity in structured scenarios. However, these systems have a critical flaw: they are inherently static. When confronted with new data in a dynamic environment, they often experience catastrophic forgetting 1.1, where they lose the knowledge and skills they previously acquired. This phenomenon occurs because the models do not adapt or integrate new information effectively; instead, they overwrite the old data when trained on new tasks. As a result, to learn a new task, they must be retrained from scratch, a process that is not only time-consuming but also resource-intensive. This inefficiency limits the ability of these systems to operate in real-world scenarios where continual learning and adaptation are essential for success.

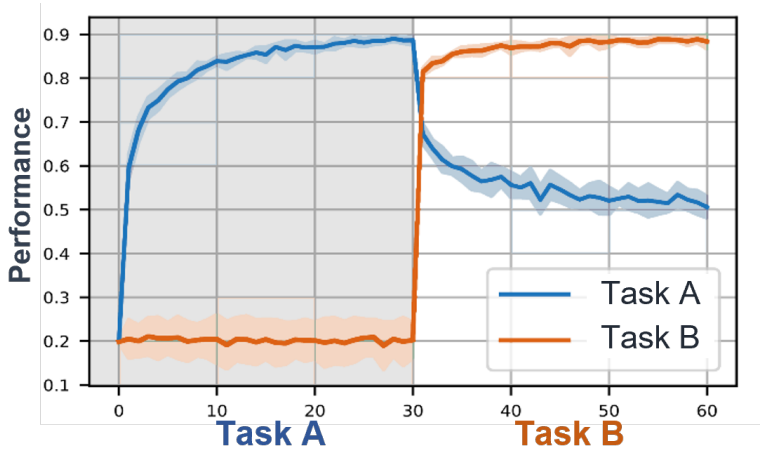


Figure 1.1: Catastrophic forgetting where a model learns task B and forgets task A

1.2 Scalable Continual Learning

To address this, we study Continual Learning (CL), aka lifelong learning and incremental learning, to enable model to learn continually from a stream of datasets without retraining [111, 114, 115, 87].

In comparison to multi-task learning and fine-tuning, continual learning is more efficient and practical. In dynamic environments, we cannot constantly gather and shuffle data to retrain the model, as this approach is both inefficient and impractical. Continual learning also allows for better generalization than relying on separate models for learning or fine-tuning. It is essential to retain both new and old knowledge in a changing context. Additionally, an effective system should leverage related data to facilitate inductive knowledge transfer. Continual learning can be applied to various scenarios with changing data distributions. For example, in video surveillance, it enables the detection of incremental classes of targets, such as cars, animals, and humans without helmets. In medical diagnosis, continual learning allows a model to generalize continuously to different instruments, standard images, symptoms, and cases from various hospitals, without the need to gather all the data for retraining.

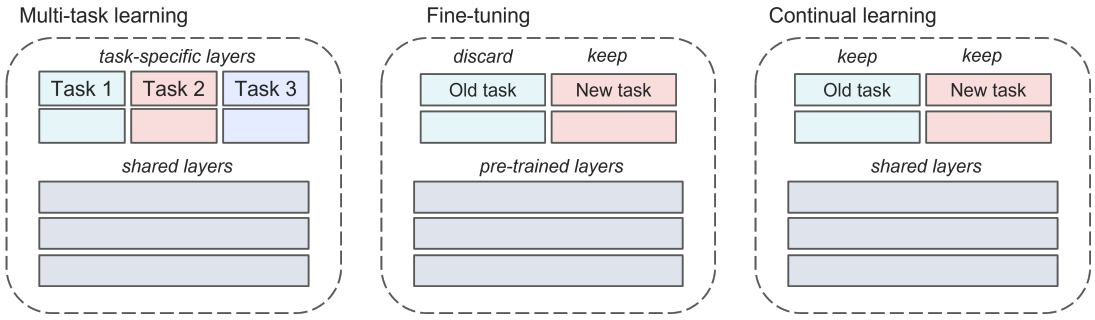


Figure 1.2: (c) Continual learning compared with (a) multi-task learning and (b) finetuning.

For autonomous vehicles, this approach helps the model adapt to different environments, such as daylight and darkness, as well as different cities. In the inspection of pipe robots, continual learning aids in generalizing to various pipe conditions.

In a continual learning scenario, we are given a sequence of usually non-I.I.D. datasets $D_1, D_2, \ldots D_n$. Each dataset consists of a group of labeled data $(X, Y) \in D$, where X and Y are input variables and the corresponding output variables, respectively. It is worth noting that, when accessing the current dataset D at time t, it is impractical or impossible to access any previous or future dataset.

However, for continual learning to transition from a lab concept to real-world technology, it must be scalable, which means that the performance of a model should not degrade catastrophically as the number of data sets or the duration of learning increases. Scalability can be broken down into three essential requirements: (1) achieve seamless accuracy retention, (2) be robust to dataset heterogeneity that may contain variations in dataset size, similarity, and complexity, and (3) enable sustainable long-term adaptation. We can visualize these three requirements in greater detail using a five-dimensional radar chart 1.3. Here, 'Task-Agnostic' represents the seamlessness, which means that the model learns and makes inferences without the need for any

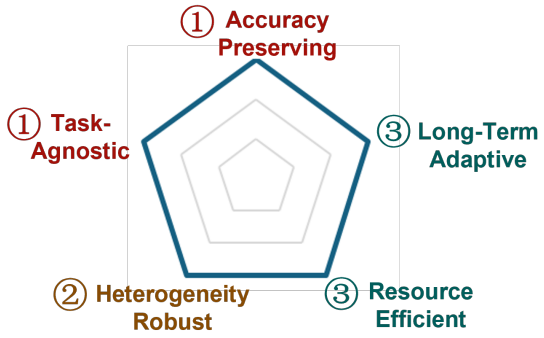


Figure 1.3: Radar chart for the three requirements of scalable continual learning.

manual task annotations or boundaries. The concept of 'Heterogeneity Robustness' differs from Non-IID (not independently and identically distributed). While non-IIDness refers to distribution shifts that lead to forgetting, heterogeneity is a broader and more challenging issue. It encompasses variations in dataset size, similarity, and complexity, which can result in uneven forgetting. For 'Resource Efficient.' This requirement is essential for sustainability, signifying that the model should be efficient in terms of memory and computational resources. Overall, achieving high performance across all five of these axes is what defines truly scalable continual learning.

Existing approaches in this field 1.4 often represent a compromise among various requirements. Most of them prioritize accuracy preservation but tend to address only part of the overall requirements at the expense of others. On one hand, rehearsal and regularization methods help mitigate forgetting by either storing past data or constraining updates to model weights. While these methods are task-agnostic, they struggle with maintaining long-term accuracy and are not robust enough to handle data heterogeneity. On the other hand, structural methods involve modularizing or adding components to models for new tasks. These methods are generally accurate; however, they require manual task annotations and can be resource-intensive, making them impractical for long-term adaptation. In summary, existing methods face challenges in simultaneously achieving all three key objectives, and this is the gap my

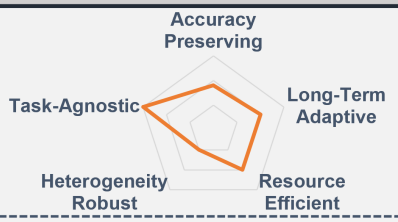
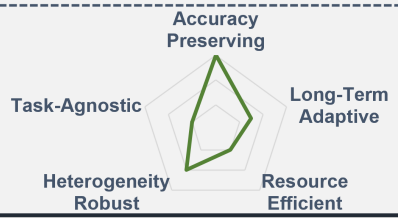
Existing methods	What do they do	Examples	Strengths and limitations
Rehearsal & regularization-based	Mitigate forgetting by storing past data or constraining weight updates.	EWC, PRE-DFKD, LwF	
Structure-based	Modularize or add new model components for new datasets.	PackNet, Progressive Learning, P&C	

Figure 1.4: Existing continual learning methods.

research aims to address.

To achieve scalable continual learning, several challenging issues need to be solved.

Sequential non-IID datasets: Due to distribution shifts among non-IID datasets, learned knowledge will be forgotten in neural networks. This phenomenon is known as catastrophic forgetting. To preserve learned knowledge seamlessly in a neural network-based model without manual task annotations, we need to tackle this challenge and identify neural network parameters to preserve and reuse.

Datasets heterogeneity: Data with varying levels of heterogeneity can have different impacts on the learning process of neural networks. This heterogeneity can manifest in several ways: the size of datasets can range from few-shot examples to large-scale samples; the complexity of the data can differ based on the variety and intricacy of features; and the similarity among data points can vary, making it challenging to distinguish task boundaries. To address these challenges, we need a more adaptive approach that can respond to data heterogeneity in a data-driven manner.

Incremental data: Finally, facing incremental datasets, it is difficult to determine the corresponding model size. The mismatch between model capacity and data

amount will affect the model's performance and adaptability. To ensure sustainable long-term adaptation, it is essential for us to select a suitable strategy for growing the model structure. This approach should enhance the network's long-term adaptability while preserving the knowledge that has already been learned.

1.3 Research Framework

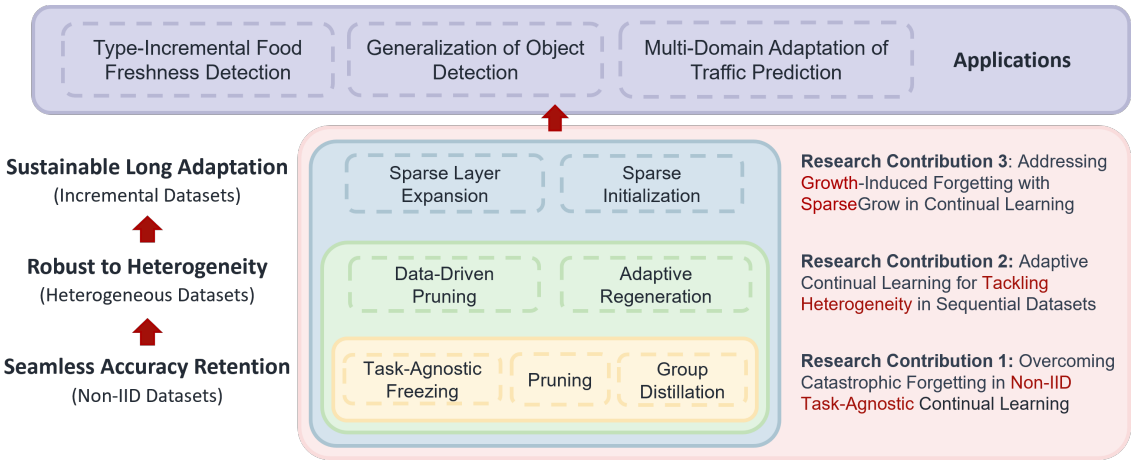


Figure 1.5: The three-layer framework of the research on scalable continual learning.

In this section, we first introduce the research framework for enabling scalable continual learning in neural networks. Then, the key components and challenging issues are discussed. Finally, we present our solutions to solve these problems. As shown in figure 1.5., the research framework consists of technical layers and an application layer. The bottom layers are the three technical layers to tackle the challenges and enable scalable continual learning. The upper part of the research framework consists of continual learning applications that have been used as use cases to apply the underlying techniques.

It includes three layers of techniques from bottom to top. The first bottom layer of techniques is used to enable task-agnostic accuracy retention in neural networks

through task-agnostic freezing and group distillation. This layer contributes to overcoming catastrophic forgetting and lays a foundation for scalable continual learning. Then, based on the first layer, we further utilize techniques such as adaptive regeneration and data-driven pruning, which adaptively control parameter usage among heterogeneous datasets in a data-driven way. Finally, we further propose to enable long-term adaptability over incremental new data through sparse neural expansion with on-data frozen initialization based on previous layers. The methods have been applied to applications like type-incremental food freshness detection, generalization of object detection, and multi-domain adaptation of traffic prediction to verify their performance.

1.4 Research Scope

Given the above requirements and corresponding challenges to achieve scalable continual learning, the first and second work of this thesis mainly aims to solve the first challenge, namely the catastrophic forgetting problem and bias problem of neural network learning caused by sequential non-IID data. When encountering domain shifts, neural network-based models suffer from performance degradation and need to retrain from scratch to adapt to domain shifts incrementally. Therefore, incremental learning is needed to adapt a model to domain shifts without retraining. Existing methods using the parameter isolation technique perform well in incremental learning of new domains without performance degradation. However, they cannot be directly adopted as they store masks and require users to label the task to indicate task-specific parameters during inference, which is memory inefficient and cumbersome. We propose a memory-efficient method to incrementally adapt to domain shifts in a fixed neural network while retaining the accuracy of learned knowledge, named E-DomainIL. Our method freezes learned parameters and allows reusing them later in training to avoid interference between different domains. E-DomainIL does not require task labels or

storing masks as it uses all parameters during inference. We use data-driven pruning to adjust the parameter ratio according to the dataset, thus maintaining the balance between accuracy and parameter efficiency. In continual federated learning, to preserve privacy and improve efficiency, we introduce FedDistill. This novel knowledge distillation framework mitigates the forgetting of global knowledge in local models. To address the issue of local imbalanced distributions, we propose group distillation, a method that segments classes into groups based on their frequency in local datasets, enabling targeted distillation that focuses on underrepresented classes. Additionally, to prevent bias, we debias the local feature extractor by distilling local features using the global classifier.

For the second contribution, we solve the challenge of dataset heterogeneity, which may contain variation in dataset sizes, similarity, and complexity. Managing heterogeneous datasets that vary in complexity, size, and similarity in continual learning presents a significant challenge. Task-agnostic continual learning is necessary to address this challenge, as datasets with varying similarity pose difficulties in distinguishing task boundaries. Conventional task-agnostic continual learning practices typically rely on rehearsal or regularization techniques. However, rehearsal methods may struggle with varying dataset sizes and regulating the importance of old and new data due to rigid buffer sizes. Meanwhile, regularization methods apply generic constraints to promote generalization but can hinder performance when dealing with dissimilar datasets lacking shared features, necessitating a more adaptive approach. We propose AdaptCL, a novel adaptive continual learning method to tackle heterogeneity in sequential datasets. AdaptCL employs fine-grained data-driven pruning to adapt to variations in data complexity and dataset size. It also utilizes task-agnostic parameter isolation to mitigate the impact of varying degrees of catastrophic forgetting caused by differences in data similarity.

My third contribution tackles the challenging issue of incremental data in a continual learning scenario to enable long-term adaptability while being parameter-efficient as

the number of datasets increases. In continual learning (CL), model growth enhances adaptability over new data, improving knowledge retention for more tasks. However, improper model growth can lead to severe degradation of previously learned knowledge, an issue we name as growth-induced forgetting (GIFt), especially in task-agnostic CL using the entire grown model for inference. Existing works, despite adopting model growth and random initialization for better adaptability, often fail to recognize the presence of GIFt caused by improper model growth. This oversight limits its comprehensive control of forgetting and hinders full utilization of model growth. We are the first in CL to identify this issue and conduct an in-depth study on the root cause of GIFt, where layer expansion stands out among model growth strategies, widening layers without affecting model functionality. Nonetheless, direct adoption of layer expansion presents challenges. It lacks data-driven control and initialization of expanded parameters to balance adaptability and knowledge retention. We propose a novel sparse model growth (SparseGrow) approach to overcome the issue of GIFt while enhancing adaptability over new data. SparseGrow employs data-driven sparse layer expansion to control efficient parameter usage during model growth, reducing GIFt from excessive growth and functionality changes. It also combines the sparse growth with on-data initialization at training late-stage to create partially zero-valued expansions that fit learned distribution, enhancing retention and adaptability. To further minimize forgetting, freezing is applied by calculating the sparse mask, allowing data-driven preservation of important parameters.

1.5 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter 2 reviews existing work in scalable continual learning. This chapter presents a systematic classification of approaches to addressing diverse challenges in the pursuit of scalable continual learning. It includes a comprehensive

taxonomy of methods for mitigating catastrophic forgetting, a detailed classification of task-agnostic continual learning methods for addressing dataset heterogeneity, and an in-depth examination of existing model growth strategies. Additionally, it discusses methods for implicitly addressing growth-induced forgetting.

- Chapter 3 introduces my first research contribution 1a that aims to overcome catastrophic forgetting caused by non-iid data. We propose efficient domain-incremental learning (E-DomainIL), a memory and parameter-efficient continual learning method for incremental domains, achieving the best performance with fewer parameters. Our method freezes learned parameters and allows reusing them later in training to avoid interference between different domains. E-DomainIL does not require task labels or storing masks as it uses all parameters during inference. This work considers neural network models in application areas like IoT devices and proposes a memory-efficient method for IoT to incrementally adapt to domain shifts in a fixed neural network.

- Chapter 4 introduces the research contribution 1b in a continual federated learning scenario, where there is spatial bias caused by non-iid data among local clients, and temporal bias caused by non-iid data through time. Continual federated learning faces significant challenges due to the imbalanced data distribution across different local devices, often termed as non-IID challenges. Recent studies have shown that while the global model tends to learn more generalized representations, this knowledge is often forgotten during local training due to this challenge. This forgetting leads to biases in local models, resulting in a degraded global model with lower accuracy and communication efficiency. We proposed a novel knowledge distillation framework that overcomes the spatial and temporal bias caused by imbalanced data distribution. Our method uses group distillation that segments classes into groups based on their frequency in local datasets, enabling targeted distillation that focuses on underrepresented

classes. Moreover, it applies local feature distillation that de-bias the local feature extractor by distilling local features using the global classifier.

- In Chapter 5, we introduce our second research contribution, AdaptCL, which is a novel adaptive continual learning method designed to address the heterogeneity present in sequential datasets. AdaptCL employs fine-grained data-driven pruning to adapt to variations in data complexity and dataset size. It also utilizes task-agnostic parameter isolation to mitigate the impact of varying degrees of catastrophic forgetting caused by differences in data similarity. Managing heterogeneous datasets that vary in complexity, size, and similarity in continual learning presents a significant challenge. Task-agnostic continual learning is necessary to address this challenge, as datasets with varying similarity pose difficulties in distinguishing task boundaries. Conventional task-agnostic continual learning practices typically rely on rehearsal or regularization techniques. However, rehearsal methods may struggle with varying dataset sizes and regulating the importance of old and new data due to rigid buffer sizes. Meanwhile, regularization methods apply generic constraints to promote generalization but can hinder performance when dealing with dissimilar datasets lacking shared features, necessitating a more adaptive approach.

- In Chapter 6, we study how to enhance the long-term adaptability of neural network models with model growth, over incremental new data. In continual learning (CL), model growth enhances adaptability over new data, improving knowledge retention for more tasks. However, improper model growth can lead to severe degradation of previously learned knowledge, an issue we name as growth-induced forgetting (GIFt), especially in task-agnostic CL using entire grown model for inference. Existing works, despite adopting model growth and random initialization for better adaptability, often fail to recognize the presence of GIFt caused by improper model growth. This oversight limits comprehensive control of forgetting and hinders full utilization of model growth. We are the

first in CL to identify this issue, and conduct an in-depth study on root cause of GIFt, where layer expansion stands out among model growth strategies, widening layers without affecting model functionality. Nonetheless, direct adoption of layer expansion presents challenges. It lacks data-driven control and initialization of expanded parameters to balance adaptability and knowledge retention. In this chapter, we propose a novel sparse model growth (SparseGrow) approach to overcome the issue of GIFt while enhancing adaptability over new data.

- In Chapter 7, I present a summary of the research problems and contributions addressed in this thesis. Additionally, I identify potential directions for future research on the role of scalable continual learning in advancing artificial general intelligence and enabling scalable self-learning in artificial intelligence.

Chapter 2

Literature Review

2.1 Tackling Catastrophic Forgetting

Existing strategies to alleviate catastrophic forgetting can be classified into activation and penalization strategies. Activation strategies preserve the activation of old data, while the penalization strategies penalize change of important parameters. Parameter isolation is a class of hard penalization strategies that can be immune to catastrophic forgetting by completely freezing learned parameters.

2.1.1 Activation Strategies

Methods in this category include rehearsal strategies as well as some regularization methods that retain activation of old data. Activation strategies mitigate forgetting by activating the network with previous experience, requiring growing or fixed memory power. For example, in [55, 75, 76, 77] samples are saved into the memory for activation, or use GAN [93, 85] and autoencoder [39] to synthesizes learned data to perform activation. More efficiently, [12, 13] leverages small episodic memory for rehearsal, allowing the same or even better performance as GEM while being com-

putationally and memory efficient. LwF [51] preserves responses of the network on older tasks by using a distillation loss, which does not require the storage of older training data.

Activation strategies can allow both backward and forward knowledge transfer and are more scalable for growing datasets as they do not rely on network capacity growth. However, activation strategies generally do not significantly mitigate catastrophic forgetting [61], especially for datasets with strong distribution shifts.

2.1.2 Penalization Strategies

Penalization strategies give hard or soft penalization to the change of important learned knowledges. For some soft penalization methods, change in important parameters or descent of gradient is controlled and reduced with penalizations if it causes a performance downgrade on prior tasks to suppress the forgetting of learned knowledge. For instance, inspired by Bayesian Learning, in elastic weight consolidation (EWC) [41] the change of important parameters is preserved in terms of the Fisher information matrix. P&C [81] compress learned knowledge and distill it into knowledge base, and preserve its changes with EWC, while using active column to progress new data. Using a Bayesian NN, CBLN [46] preserves distinctive parameters for different datasets for retaining performance. Similarly, [68] introduced developmental memory (DM) into a CNN, continually growing sub-memory networks to preserve important features of learned tasks while allowing faster learning. Each sub-memory can store task-specific knowledge by using a memory loss function and preserve it during continual adaptations. HAT [83] learns an attention mask over important parameters. By aligning local representations, P-TNCN [66] replaces the back-propagation method that descent steepest, punishing parameter updates to a more generalized result, therefore mitigating catastrophic forgetting.

These methods can make trade-offs model capacity and catastrophic forgetting, but

they are often inadequate facing strong distribution shifts.

Parameter-isolation based approaches Parameter isolation is a class of hard penalization strategies that can be immune to catastrophic forgetting by completely freezing learned parameters. Parameter isolation methods either separately model each task with modularization or rigidly factorization the architecture into shared and task-specific parts.

For instance, Progressive Neural Nets (PNNs) [80] statically grow the architecture with equal size modules while allowing forward knowledge transfer through connections between them. PathNet [19] propose a giant modularized neural network that has many possible paths from input to output. The network can choose which path to use with given task or dataset label. Similarly, RPS-Net [72] modularize the network into different paths, and progressively chooses optimal paths for the new tasks. It combines a distillation loss for regularization and retrospection replay to further reduce forgetting. CAT [38] mask used parameters and block gradients flow through used units for dissimilar tasks. Task mask is stored according to task ID or label, and need to be kept during test. Deep Adaptation Modules (DAM) [79] assign learning of each domain to a fraction or the network, each fraction has the same percentage, typically 13To make trade-offs between knowledge transfer and catastrophic forgetting, CLAW [2] adaptively identify which parts of the network to share in a data driven way, and use trainable mask to flexible decide the shared part to adapt or task-specific part to preserve. CLNP [22] and PackNet [61] use pruning to compromise between model sparsity and performance. PackNet prune the parameters according to certain percentage of the number of parameters, while the CLNP prune the parameters according to certain percentage of parameters' highest value. After pruning, the learned parameters are frozen and isolated while using new fractions for learning new task. During inference, the parameters are selected according to given task label. Reinforced Continual Learning RCL [96] adaptively expands each layer using reinforcement learning, allow sharing parameter. Due to separation, task label

is needed as input to indicate the parameters to use during test. PiggyBack [60] learn binary masks that "piggyback" on an existing fixed network. It adapts the network to new task by learning trainable masks on network weights. There is no transfer of knowledge, and the mask is kept during test given which task to perform.

By freezing and separating the learned parameters, parameter isolation methods are proved [17, 80] to be more forgetting-free and can achieve the highest overall performance compared with other famous CL methods like iCaRL [75], GEM [55], HAT [83], MAS [4], replay methods with partial memory like GEM [55], EWC [41], and LwF [51], etc.

Nevertheless, it has room for improvement. First, parameter isolation methods are considered [1, 80] can only be applied in the task-incremental settings where the task ID is given during inference but fail in domain incremental settings where what dataset the model performs on is unknown during inference, which is due to improper isolation strategies. Modularization and factorization create different paths in the parameters for different datasets or tasks, and therefore during inference, the neural network cannot determine which module and path should be selected by itself. Second, previous paths and modules are preserved without further modification, preventing backward transfer of knowledge.

2.2 Tackling bias

2.2.1 Non-IID Federated Learning (FL)

Federated Learning, a paradigm for training models across decentralized datasets without compromising privacy, faces significant challenges in non-iid scenarios. Despite FedAvg's foundational role in FL [63], its performance degrades under non-iid data distributions [112]. Subsequent efforts to mitigate these challenges have bifur-

cated into enhancing global aggregation [14, 52, 92, 108] and refining local training methodologies [64, 106, 37, 48]. While global-side improvements are constrained by privacy regulations, leading to minimal data-driven modifications, local-side enhancements have flourished. Innovations like FedAlign[64] emphasize optimizing local model generalization, while approaches like FedUFO[106] and SCAFFOLD[37] introduce mechanisms to foster client-agnostic feature learning and update calibration, respectively. Notably, MOON[48] leverages contrastive learning to align local model features with the global model, underscoring the evolving landscape of strategies aimed at reconciling local and global learning objectives.

2.2.2 Forgetting in Non-iid Federated Learning.

The phenomenon of forgetting, borrowed from lifelong learning, has been recognized in FL, particularly in the context of local training. [86] is among the first proposing that forgetting might be an issue in non-iid Federated Learning. [44, 95, 3, 99] provides different empirical results showing local models forgetting the generalization ability from the global model during training. However, they didn't analyze the reason why this forgetting happens. In this paper, we argue that forgetting is incurred due to insufficient positive update for the classes with few or no samples, surprising the local model from retaining the generalization ability of the global model.

With different observations, approaches are proposed to alleviate the forgetting in non-iid federated learning. [86, 25] proposed a regularization-based method that prevents the parameter changes from the global model to the local model, which limits the learning ability of the local model. FedNTD[44] proposed to use the global model as the teacher to guide the local model with not-true distillation; FedRAD[90] proposed to use the entropy of the global model as an indicator to adaptively adjust the weight of knowledge distillation and cross-entropy. But both of them didn't consider the imbalanced nature of the local data distribution.

FedCSD[99] proposed to use logit as guidance, where an additional logit prototype for each class will be maintained through the training, an extra teacher model is also needed to calculate the logit from the logit prototype to guide the training of the local model. Both the logit prototype and teacher models need to be uploaded/downloaded for each local training, bringing in extra communication overhead. Flashback[3] also noticed the imbalanced data distribution across clients and used the label count as the indicator of knowledge distillation, however, they rely on a global label count estimation which requires the label count of clients from the last round, which brings potential privacy concerns. In our paper, we don't need any other information except the global model on the local side and don't upload any extra information except the trained local model, which poses a more challenging question compared to other methods.

FedReg[95] proposed to generate pseudo data on the local side that indicates the biggest changes of prediction from the global model to the local model. This behavior brings potential privacy concerns and burdens the local training process.

2.2.3 Federated Learning with local model de-biasing.

Apart from forgetting to describe the model inconsistency from the global model and across client models, some researchers believe the local models are skewed by the local data. [57] observe that the local model has parameter drift from the global model the drift level of which decreases from the bottom layers to the top. With this observation, they decompose the model into a feature extractor and classifier and calibrate the biased classifier by single-regularizing the parameter drift of the classifier. FedBR[24] takes local feature extractor de-biasing into consideration. FedBR designs an adversarial learning framework on the local side, where a discriminator is trained to discriminate the feature extracted from local and global models on pseudo data while minimizing the distance between local and global features on local data. In our

paper, we consider both feature extractor and classifier de-biasing without pseudo data, by further leveraging the global model.

Given that most of the existing methods either ignore the importance of imbalanced local data in local model forgetting or rely on extra information sharing, we proposed FedDistill. Our contribution to this evolving field includes a novel decomposition of knowledge distillation into three parts: true class, few-sample class, and rich-sample class, directly addressing the challenge of imbalanced data distribution. Furthermore, by decomposing both the global and local models, we apply KD in a manner that effectively prevents drift from the global model to local models, ensuring a more balanced and representative learning process. Notably, our method emphasizes improvements on the local side, without any additional communication overhead and privacy concerns typically associated with global-side enhancements.

2.3 Tackling Heterogeneity

Continual learning methods are crucial tools in the field of machine learning, aiding in the effective handling of tasks that evolve over time. The existing methods primarily fall into three categories: rehearsal-based, regularization-based, and structure-based. This section provides a detailed overview of these methods, highlighting their strengths and limitations, particularly when applied to task-agnostic continual learning and the management of heterogeneous datasets.

2.3.1 Rehearsal-Based

These techniques seek to overcome catastrophic forgetting, a significant challenge in continual learning, by replaying previous training data periodically. Early methods like GEM and A-GEM [55, 12] relied on storing a portion of past training data and reusing it in future training phases. This approach has been further refined

with the incorporation of generative models to create synthetic data distributions for pseudo-rehearsals [53]. LwF [51] introduces knowledge distillation that utilizes a teacher network to distil knowledge and soft targets to a student network while training on new tasks, enabling retention of knowledge from previous tasks. Some combine replay with knowledge distillation like Andrea et al. [78] keep a very small buffer for highly informative samples and combine with distillation playback and Jingyuan et al. [89] distils knowledge and replays experience from previous tasks when fitting on a new task. ICaRL [75] adopts a combination of rehearsal and regularization through learning a compact and discriminative feature representation to enable class-incremental learning. Similarly, [28] adopts a combination of rehearsal and regularization that uses the nearest class mean (NCM) classifier on food image classification dataset Food1k-100; the class mean of all data seen so far is estimated by the online mean update standard during the training phase. PRE-DFKD[9] further refines these strategies and proposes to rehearse the model using the data-free knowledge distillation through the distribution of the previously observed synthetic samples from a Variational Autoencoder (VAE).

Despite these advancements, rehearsing techniques face limitations when managing datasets of varying sizes and maintaining the balance between old and new data. However, with AdaptCL, the model allocates parameters based on the accuracy in a data-driven way, allowing it to retain knowledge as parameter-level representations, independent of the data volume.

2.3.2 Regularization-Based

These methods incorporate regularization techniques, such as weight decay or dropout, to prevent catastrophic forgetting in neural networks when learning multiple tasks sequentially. Inspired by Bayesian Learning, Elastic Weight Consolidation (EWC) [41, 33] mitigates catastrophic forgetting by tracking changes using the Fisher In-

formation Matrix. [27] adopts knowledge distillation on augmented exemplars in a class-incremental setting on food image classification. Selvarajah et al. [91] propose an indicator loss that is associated with a distillation mechanism that preserves the existing knowledge. Guanglei et al. [100] introduce an attentive feature distillation approach to mitigate forgetting. P&C [81] compress learned knowledge and distil it into the knowledge base, and preserve knowledge with EWC while using the active column to progress new data. Using a Bayesian neural network, CBLN [46] preserves distinctive parameters for different datasets for retaining performance. Similarly, [68] introduced developmental memory (DM) into a CNN, continually growing sub-memory networks to preserve important features of learned tasks while allowing faster learning. Each sub-memory can store task-specific knowledge by using a memory loss function and preserve it during continual adaptations. HAT [83] learns an attention mask over important parameters. SCML [88] proposes to learn a meta-learner for updating a unified model than updating the weights inappropriately through the optimizer. By aligning local representations, P-TNCN [66] replaces the back-propagation method that descent steepest, punishing parameter updates to a more generalized result, therefore mitigating catastrophic forgetting.

Despite the potential of regularization-based methods, they can face challenges when handling heterogeneous datasets, especially those that are dissimilar and have few shared features. While through parameter isolation in a data-driven manner, AdaptCL can effectively adapt to datasets with varying levels of similarity, including dissimilar ones.

2.3.3 Structure-Based

Structure-based methods are primarily employed in task-specific scenarios, and these methods use parameter isolation to handle both similar and dissimilar domains effectively. They divide the network into separate modules to mitigate interference during

inference. While these techniques excel in managing catastrophic forgetting, they present difficulties when directly applied to task-agnostic scenarios.

One approach, exemplified by Progressive Neural Nets (PNNs) [80], involves a static growth of the architecture with equal-sized modules, allowing for forward knowledge transfer between them. However, this method lacks a data-driven approach and requires task-specific settings for subsequent tasks, limiting its flexibility. Another approach, represented by SILF [58], addresses parameter isolation by pruning unimportant parameters, isolating the important ones to mitigate forgetting. However, SILF relies on manual pruning ratio setting instead of leveraging a data-driven approach. Reinforced Continual Learning (RCL) [96] expands each layer using reinforcement learning and enables parameter sharing. Nevertheless, this method necessitates task labels as additional inputs during inference to determine the parameters to use. To strike a balance between knowledge transfer and catastrophic forgetting, CLAW [2] identifies which parts of the network should be shared or preserved for specific tasks. PathNet [19] and RPS-Net [72] adopt a modularized network with multiple possible paths from input to output. They choose specific paths based on tasks or dataset labels. Additionally, RPS-Net includes a distillation loss and retrospection replay to further minimize forgetting. CAT [38] masks used parameters and blocks gradient flow through unused units for dissimilar tasks. Task masks are stored according to task ID or label and need to be retained during testing. Other methods, such as DAM [79], CLNP [22], and PackNet [61], leverage pruning to strike a balance between model sparsity and performance. DAM assigns learning of each domain to a fraction of the network, typically with the same percentage (e.g., 13%). CLNP and PackNet prune parameters based on specific percentages.

Notably, the power of structure-based parameter isolation methods like PackNet has been demonstrated through recent advancements [17] that have shown superior performance compared to other continual learning methods [75, 55, 83, 4, 41, 51].

However, challenges persist, particularly when dealing with heterogeneous datasets

in task-agnostic settings, which calls for more adaptive approaches. Our previous work [113] priorly applied the structure-based parameter isolation method to the task-agnostic scenario. However, its coarse-grained pruning resulted in limited adaptability to the heterogeneity of dataset size and similarity, leading to sub-optimal accuracy. Additionally, more adequate validation is needed on heterogeneous datasets.

2.4 Enlarging Model Capacity

At present, there are three kinds of model growth methods. One is the lateral connection in width, the second is layer expansion in width, the third is growing in depth.

2.4.1 Lateral Connection

Lateral connection grows new modules or layers in parallel with existing layers and allows connections with previous layers.

For instance, Progressive Neural Nets (PNNs) [80] statically grow the architecture with randomly initialized modules while retaining lateral connections to previously frozen modules. However, this method is limited to specific simple networks. Schwarz et al. [81] use randomly initialized active lateral columns (1x1 conv) to learn new tasks by connecting them to lateral columns that store previous knowledge. They also distill knowledge from the active column to knowledge base layers. This method is applicable to Conv2d layers but is limited to specifically designed simple networks and lacks generality. Zhang et al. [105] adopt AutoML-based model growing with both lateral connections and in-depth growth. They also use knowledge distillation to compress the model after learning a task. This method is applied to complex networks like VGG. However, using AutoML or similar methods often requires defining the layers for growth during the initial model definition, limiting its applicability. In-

depth and lateral growth can also lead to more forgetting. Ardywibowo et al. [6] propose VariGrow, which detects if a new task is arriving through an energy-based novelty score. If the novelty score is high and the sample is "detected" as a new task, VariGrow grows a new expert module to handle it. Otherwise, the sample is assigned to one of the existing experts that is most familiar with it. Unlike other methods, VariGrow is task-agnostic continual learning and does not require task identification during inference, making it suitable for class and domain incremental continual learning. However, this method can prevent the update of old modules when encountering similar data. In Hu et al. [31], given a new task t, a new branch called the task t expert is added while freezing existing experts. This is achieved by introducing dense connections between the intermediate layers of the task expert networks. Li et al. [50] propose a hybrid solution for lateral connection and layer expansion, involving three operations: 'new', 'adaptation', and 'reuse'. The 'new' operation introduces a randomly initialized 3x3 layer trained from scratch. In the 'reuse' strategy, existing frozen weights are reused. The 'adaptation' strategy adds a 1x1 convolution layer in parallel to a 3x3 convolution layer, keeping the original 3x3 kernel fixed while learning the parameters of the 1x1 adapter. However, this hybrid solution is limited to specially designed Conv2d networks and lacks generality.

2.4.2 Layer Expansion

DEN [102] uses layer expansion in a top-down manner, growing every layer if the loss does not meet a threshold. Similar to DEN, Hung et al. [32] expand the number of filters (weights) for new tasks. Moreover, they adopt gradual pruning to compact the model. However, different from our sparse growing method, their pruning is not data-driven, and their method requires manual help of task IDs during inference, making it unsuitable for domain incremental or class incremental learning. Ostapenko et al. [67] introduce Dynamic Generative Memory (DGM) and expand the same number of neurons used in a layer in the generator of a GAN for the scalability of rehearsal. The

scope of rehearsal is expanded. Geng et al. [20] expand the hidden size H_k for the k-th task from H_{k-1} by the pruning ratio of task $k-1$. After random initialization, on-data finetuning is performed for the newly added parameters. Yang et al. [101] grow a randomly initialized expanded filter and concatenate it to the network. Xu et al. [96] adaptively expand each layer of the network when the t-th task arrives. This method is also applicable for simple convolutional networks and fully-connected networks. Yan et al. [98] expand the model with new parameters by creating a separate feature extractor F_t for incoming data and taking a uniform distribution as the prior distribution. This method is also applicable for class incremental learning.

2.4.3 In-Depth Growth

Some non-continual learning methods, such as Wen et al. [94] and Yuan et al. [104], increase the depth of the neural network in the hidden layer to achieve faster convergence and efficiency. Yan et al. [98] also increase the depth of the neural network in the hidden layer to achieve faster convergence and efficiency. They use random initialization, which can help escape from a bad starting point. Yuan et al. [104] propose a budget-aware growing process that starts from a small, simple seed architecture and dynamically grows and prunes both layers and filters to make the network wider and deeper. They also adopt the initialization of ResNet or VGG.

In the field of continual learning, a few papers also utilize in-depth growth to increase model capacity: Kozal et al. [42] add new layers on top of existing ones. Zhang et al. [105] adopt AutoML-based model growing in both width and depth by adding lateral layers and in-depth layers. They also use knowledge distillation to compress the model after learning a task. Nevertheless, adding new layers in depth within the neural network can have a profound impact on the model's structure compared to growing in width, leading to more severe growth-induced forgetting and altering the learned features of previously acquired tasks. This effect undermines the network's

ability to retain prior acquired knowledge and, therefore, is not suitable for continual learning.

2.4.4 Growth-Induced Forgetting

In this section, we explore the related works in overcoming growth-induced forgetting.

The issue of growth-induced forgetting might have been indirectly addressed in the context of task-agnostic continual learning, where certain studies have focused on preserving knowledge post model growth. Several studies in continual learning have developed methods to handle model growth and maintain accuracy on previous tasks, albeit without explicitly recognizing or referring to the phenomenon of growth-induced forgetting.

Here, we focus on task-agnostic continual learning, excluding task-specific continual learning, which relies on manual labelling of task IDs to assist machines in selecting the model's inference region, thereby bypassing growth-induced forgetting.

Existing task-agnostic continual learning approaches typically employ regularization and rehearsal methods to mitigate the forgetting problem encountered by models. For example, Madaan et al. [59] build on top of the distillation family of techniques and modify it for a new setting where a weaker model takes the role of a teacher. They propose Quick Deep Inversion (QDI) to recover prior task visual features to enhance distillation.

Zhang et al. [105] use RWC to alleviate forgetting while employing knowledge distillation for compression. Schwarz et al. [81] use Elastic Weight Consolidation (EWC) to overcome forgetting caused by adding an active column. Ostapenko et al. [67] expand the same number of neurons used in a layer in the generator of GAN for the scalability of rehearsal. Ardywibowo et al. [6] propose a different method called VariGrow. It detects if a new task is arriving through an energy-based novelty score.

If the novelty score is high and the sample is "detected" as a new task, VariGrow will grow a new expert module to be responsible for it. Otherwise, the sample will be assigned to one of the existing experts who is the most "familiar" with it. However, this method has a drawback in that when encountering similar data, the old modules cannot be updated with new data, which hinders knowledge reuse and propagation between modules and is not parameter-efficient.

Similar to our approach's task-agnostic freezing method, Yan et al. [98] freeze the previously learned representation and augment it with additional feature dimensions from a new learnable feature extractor.

Chapter 3

Research Contribution 1a: Overcoming Catastrophic Forgetting in Non-IID Task-Agnostic Continual Learning

3.1 Introduction

Consider a household robot using a Deep Neural Network (DNN) model to identify multiple objects. Changes in user requirements and environment such as from daylight to nightlight may compromise the robot's recognition ability due to domain shift problem caused by changing data patterns. To overcome the performance degradation, existing DNN models in IoT need to retrain from scratch to adapt to domain shifts incrementally. To fill this gap, incremental learning is needed to enable DNN models in embedded systems and IoT units to perform incrementally with domain shifts, rather than being trained from scratch each time new data is collected. However, when a model learns a new domain, it often forgets previously learned domain

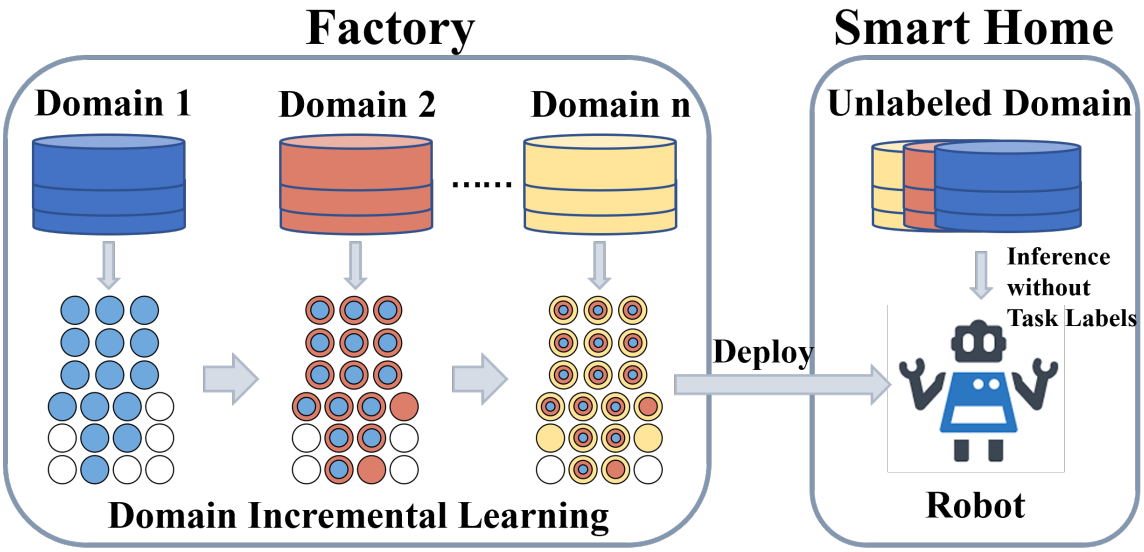


Figure 3.1: Overview of proposed domain incremental learning in an IoT scenario. It incrementally adapts to domain shifts in a fixed neural network while retaining the accuracy of all learned domains. After deployment, it can make inferences without giving task labels of the domains.

knowledge and suffers from catastrophic forgetting problem.

Existing incremental learning methods using the parameter isolation technique are promising as they are immune to catastrophic forgetting [17, 81] but cannot perform well in a domain incremental learning setting. These methods prevent interference between domains in the network by freezing learned parameters from gradient updates during training and masking task-specific parameters for inference. However, by isolating parameters into modules and storing task-specific masks during inference, these methods consume high memory, and can only be applied in task-incremental settings that require users to label the task during inference, which is cumbersome and not applicable for many IoT applications. Moreover, many parameter isolation methods such as PackNet [19], and PNN [80] give the same isolation ratio for each dataset, which will affect the model accuracy, and is not very parameter efficient.

We propose E-DomainIL, a memory-efficient method for IoT that incrementally adapts to changing domains without task labels or storing masks during inference, as shown in Figure 3.1. E-DomainIL uses task-agnostic parameter isolation that freezes learned parameters and allows reusing them later in training to avoid interference between different domains and uses all parameters without separation during inference. In this way, the model does not require task labels to specify task-specific parameters for inference. At the same time, inductive knowledge can transfer backward to help increase the accuracy of similar previous domains. Furthermore, instead of manually assigning the same ratio of parameters to each domain, we propose a data-driven pruning strategy that adjusts the parameter ratio according to the dataset, ensuring accuracy and parameter efficiency. We evaluate the E-DomainIL on several datasets in a challenging domain-incremental setting, where the lack of task labels prevents task-aware inference in the model. This setting requires merging output units into a single-headed classifier with competing data of different domains, often leading to more severe forgetting [1]. Our method outperforms the competing alternatives in average accuracy, overcoming forgetting and parameter efficiency.

3.2 Related Works

Incremental Learning for IoT. In order to adapt to domain shifts incrementally without forgetting the learned domains, existing studies [45, 11, 74] store data of previous domains explicitly or implicitly to shuffle all data to train from scratch or rehearse past domain data during the learning of the new domain. Such methods require more training time to consolidate previous domains during training of new domains, and require more memory for storage.

Parameter Isolation. In the field of incremental learning, parameter isolation is a tempting method for models to adapt to new domains while avoiding catastrophic forgetting. By freezing the learned parameters during training, and and separating

them during inference, parameter-isolation-based methods prevent the interference of different domain in the network, and are are proved [17, 80] to be forgetting-free and can achieve the highest overall performance compared with other famous CL methods like iCaRL [75], GEM [55], HAT [83], MAS [4], replay methods with partial memory like GEM [55], EWC [41], and LwF [51], etc.

Parameter isolation methods are considered [1, 80] to be only applicable in task-incremental settings where the task labels are given during inference but fail in domain-incremental settings where which domain the model performs on is unknown during inference. For instance, Progressive Neural Nets (PNNs) [80] statically grow the architecture with equal size modules while allowing forward knowledge transfer through connections between them. Reinforced Continual Learning (RCL)[96] adaptively expands each layer using reinforcement learning and allows parameter sharing. Due to separation, task labels are needed as input to indicate the parameters to use during inference. Deep Adaptation Modules (DAM) [79] assign learning of each domain to a fraction of the network; each fraction has the same percentage, typically 13%. CLNP [22] and PackNet [61] use pruning to compromise between model sparsity and performance. PackNet prunes the parameters according to a certain percentage of the number of parameters, while the CLNP prunes the parameters according to a certain percentage of parameters' highest value. After pruning, the learned parameters are frozen and isolated while using new fractions for learning a new task. During inference, task masks are kept for selecting parameters according to given task labels. CAT[38] mask used parameters and block gradients flow through used units for dis-similar tasks. Task masks are stored according to task labels and need to be kept during the test. PathNet [19] proposes a massive modularized neural network with many possible paths from input to output. The network can choose which path to use with a given task label. Similarly, RPS-Net [72] modularizes the network into different paths and progressively chooses optimal paths for the new tasks.

3.3 Methodology

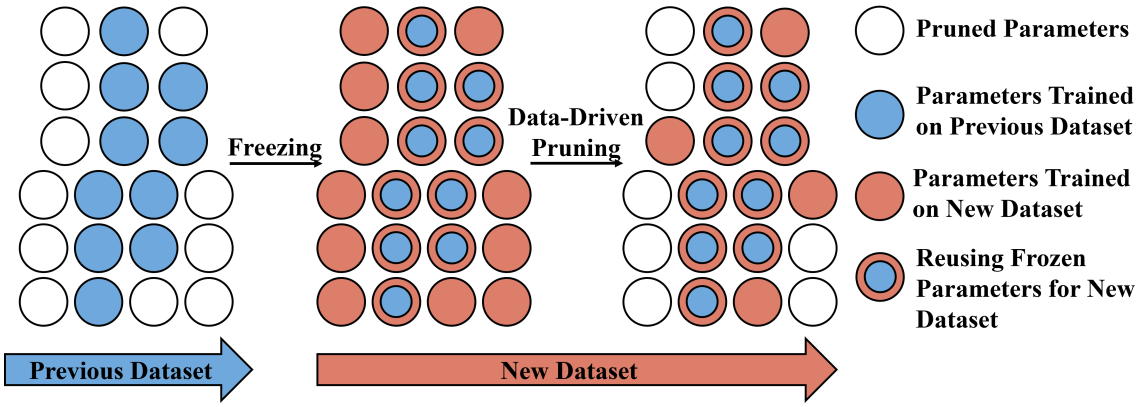


Figure 3.2: E-DomainIL training process. E-DomainIL can overcome catastrophic forgetting through task-agnostic parameter isolation that freezes learned parameters and allows reusing them later in training. During inference, it uses all parameters, therefore does not require task labels or storing masks. Through data-driven pruning, E-DomainIL can adjust the ratio of the parameters according to the data and maintain a balance between accuracy and parameter efficiency. (Best viewed in color)

3.3.1 Problem setting and objective

We are given a sequence of datasets $D_1, D_2, ...D_n$ from different domains for a fixed classification problem, each dataset consists of a group of labeled data $(X, Y) \in D$, where X and Y are input variables and the corresponding output variables, respectively. A domain-incremental setting requires to optimize:

$$\max_{\theta} E_{t\sim D}[E_{(X,Y)\sim D_t}[\log p_\theta(Y|X)]] \tag{3.1}$$

where θ identifies the parametrization of the network. Such a maximization problem is subject to the incremental learning constraints: When accessing the current dataset D at time t, it is impractical or impossible to access any previous or future dataset. We aim to train a neural network with fixed memory and compute to learn continuously from a sequence of datasets while retaining inference accuracy on learned datasets.

3.3.2 Task-agnostic parameter isolation

Existing parameter isolation methods completely isolate parameters during training and inference by freezing learned parameters from gradient update and masking task-specific parameters for inference. We propose to freezes learned parameters and allows reusing them later in training, but use all parameters during inference.

Given the training dataset D, a neural network with parameter W can be trained with a back-propagation algorithm taking the cost J as the following equations 3.2,3.3.

$$J(W) = \frac{1}{n} \sum_{i=1}^{n} L\left(D; W\right) \tag{3.2}$$

$$W_{new} = W - \alpha \frac{\delta J}{\delta W} \tag{3.3}$$

The $L\left(D; W\right)$ for loss function, such as the classification of cross-entropy loss, and a small value α called the learning rate.

For a neural network that consists of a set of parameters $\{W_i : 1 \leq i \leq C\}$, W_i denotes the parameter matrix at layers i, and C denotes the number of layers in this network. For each fully connected layer, the corresponding parameter is defined as $W_i \in R^{c_o \times c_i}$, in which c_o is the output dimension and c_i is the input dimension of each layer. During the training of the new dataset, we set the gradient descent $\frac{1}{n} \sum_{i=1}^{n} L\left(D; W_{ij}\right)$ of previous parameters to 0 to suppress the change of learned knowledge.

To calculate the current parameter gradient to be frozen, a freeze mask M_{ij}^{f} needs to be updated after training each dataset. We utilize a unit step function $S\left(x\right)$ to obtain the masks according to the magnitude of parameters, as equation 3.4 and 3.5.

$$S\left(x\right) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{3.4}$$

$$M^f{}_{ij} = S\left(|W_{ij}|\ \right), \quad 1 \leq i \leq c_o, 1 \leq j \leq c_i \tag{3.5}$$

While training new data, the gradient of parameter update will be frozen with mask M^f_{ij} as in equation 3.6, where $\circ$ is an element-wise product operator. During test, all parameters are used for inference.

$$J\left(W_{ij}\right) = \frac{1}{n}\sum_{i=1}^{n} L\left(D; W\right)\ \circ\ M^f_{ij} \tag{3.6}$$

3.3.3 Data-driven pruning

We use data-driven pruning to assign different ratios of parameters according to the datasets of different domains. For one dataset, data-driven pruning can find the most suitable sparse network structure with pruning without harming accuracy. It creates free parameters for learning new datasets without adding additional network capacity, ensuring high parameter efficiency. For incremental datasets, data-driven pruning enables similar datasets to reuse more of each other's parameters.

Inspired by [54], we set a trainable threshold value t for each fully connected, recurrent and convolutional neural network layer to separate pruned and retained parameters. To achieve this, we add a sparse regularization term L_s to the training loss that penalizes the low threshold value. For each trainable masked layer with threshold t, the corresponding regularization term is $R = \sum_{i=1}^{c_o} \exp\left(-t_i\right)$. Thus, the sparse regularization term L_s for a neural network with C trainable masked layers is:

$$L_s = \sum_{i=1}^{C} R_i \tag{3.7}$$

$exp(-x)$ is used as the regularization function since it is asymptotical to zero as x increases. Given the training dataset D, a sparse neural network can be trained directly with a back-propagation algorithm by adding the sparse regularization term

L_s to the loss function as follows:

$$J(W) = \frac{1}{n}\sum_{i=1}^{n}\left[L(D; W, t) + \beta L_s\right] \tag{3.8}$$

where β is the scaling coefficient for the sparse regularization term, which can control the percentage of parameters remaining. Therefore, data-driven pruning can find the sparse structure in a trainable way that appropriately balances the model sparsity and performance. Accordingly, the formula of masked gradient descent is also updated as:

$$J(W_{ij}) = \frac{1}{n}\sum_{i=1}^{n}\left[L(D; W, t) + \beta L_s\right] \circ M_{ij}^{f} \tag{3.9}$$

In each training step, we utilize the unit step function $S(x)$ to obtain a binary pruning mask M^p according to the magnitude of parameters and corresponding threshold values, as equation 3.10.

$$M^p{}_{ij} = S\left(|W_{ij} - t_i|\right), \quad 1 \le i \le c_o, 1 \le j \le c_i \tag{3.10}$$

The corresponding element in the pruning masks $M^p{}_{ij}$ will be set to 0 if W_ij needs to be pruned.

According to [54], a derivative estimation is needed to make the binary step function $S(x)$ in threshold vector t trainable via back-propagation. A long-tailed higher-order estimator $H(x)$ proposed by [97] is adopted for a balance of tight approximation and smooth back-propagation.

$$\frac{d}{dx}S(x) \approx H(x) = \begin{cases} 2 - 4|x| & -0.4, \le x \le 0.4 \\ 0.4, & 0.4 < |x| \le 1 \\ 0, & otherwise \end{cases} \tag{3.11}$$

The training flow of this method is shown in Algorithm 1. The threshold parameter is initialized at the beginning of each new dataset training. Each gradient descent step is frozen by the current freezing mask (if any) during training. At the same time,

Algorithm 1 E-DomainIL Training Flow

Require: weight of parameter W, threshold vector t is initialized with zero tensors.
for dataset $d = 0, 1, 2, \ldots$ **do**
  for layer in the model **do**
    Reset threshold $t \leftarrow 0$
  end for
  for epoch **do**
    for step **do**
      update pruning mask ${M^p}_{ij} \leftarrow S\left(|W_{ij} - t_i|\right)$
      update pruned weight $W \leftarrow W \circ M^p$
      for layer in the model **do**
        update the loss $L(\cdot) \leftarrow L(D; W, t) + \beta L_s$
      end for
      if $d == 0$ **then**
        gradient descent $J(W) \leftarrow \frac{1}{n}\sum_{i=1}^{n} L(\cdot)$
      else
        gradient descent with freeze masks $J\left(W_{ij}\right) \leftarrow \frac{1}{n}\sum_{i=1}^{n} L(\cdot) \circ M^f_{ij}$
      end if
    end for
  end for
  update freeze mask ${M^f}_{ij} \leftarrow S\left(|W_{ij}|\right)$
end for

Table 3.1: The number of training dataset and testing dataset sample splits used in the datasets.

	MNIST Variant			**DomainNet**			**Food Safety**	
	Dataset A	Dataset B	Dataset C	Dataset A	Dataset B	Dataset C	Dataset A	Dataset B
	MNIST	Permute MNIST	Invert MNIST	Sketch	Quickdraw	Clipart	Apple	Bread
Train Set	60000	60000	60000	4607	2100	1163	55	91
Test Set	10000	10000	10000	400	200	100	57	93

the threshold parameter is calculated and updated at each step of backpropagation to update the pruning mask and realize data-driven pruning step by step.

3.4 Experiments

3.4.1 Datasets

Following the domain-incremental setups, our method is evaluated on two sequences or benchmark datasets, MNIST and DomainNet, and a use case dataset, Food Safety datasets.

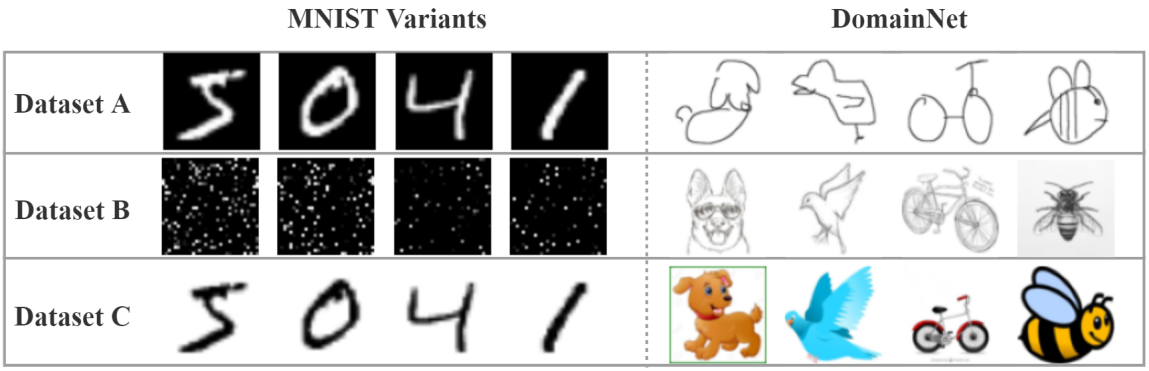


Figure 3.3: Examples of MNIST Variants and DomainNet.

MNIST Variants: To preliminarily verify our model, we select MNIST, Permuted MNIST, and Inverted MNIST datasets to form an MNIST Variants sequence in a domain-incremental setting: {MNIST, Permuted MNIST, Inverted MNIST}. The three datasets of different domains, with no data overlap among them. The datasets each consist of 70,000 images of handwritten digits from 0 to 9 of size 32×32. In each dataset, 60,000 images are used for training and 10,000 for the test, as listed in Table 3.1.

- MNIST [43] consists of 70,000 binary images of handwritten digits from 0 to 9.
- Permuted MNIST [23] is a variant of MNIST that performs a fixed random permutation of the pixels of the MNIST digits. It also includes the same number of handwritten numbers.
- Inverted MNIST is another variant of the MNIST dataset that changes the color of the MNIST image from black to white and from white to black.

DomainNet: To further validate our method, we selected DomainNet [70] datasets that are more complex and close to real-life data, with varying amounts of data and stronger domain shifts. The DomainNet dataset consists of image data from six distinct domains, each with a different amount of data, including real photos, paintings,

clipart, infograph, QuickDraw, and sketches. There are 48K - 172K images categorized into 345 classes per domain. Following the domain-incremental setting, ten categories are selected from the Sketch, Quickdraw, and Clipart domains as datasets to form a sequence with the number of data shown in Table 3.1.

Food Safety: The Food Safety datasets was used to verify our solution as a real-life IoT application case. The Food Safety datasets consists of Apple and Bread images with their freshness score. The freshness grading is scaled from 0 to 4, 0 indicating total corruption and 4 for total freshness. The apple dataset contains 57 total apple images, while the bread dataset contains 93 total bread images. Due to the small amount of data, we compromised some overlaps when dividing the training and test datasets, as shown in Table 3.1.

3.4.2 Evaluation metrics

For a principled evaluation, we adopt the evaluation metrics from GEM[55]. We consider access to a testing dataset for each of the D domains. After the model finishes learning about the domain t_i, we evaluate its test performance on all T domains. By doing so, we construct the matrix $R \in R^{t \times t}$, where $R_{i,j}$ is the test classification accuracy of the model on the domain t_j after observing the last sample from domain t_i. Letting $\bar{b}$ be the vector of test accuracies for each task at random initialization, then we have:

- Average Accuracy: $ACC = \frac{1}{T} \sum_{i=1}^{T} R_{T,i}$
- Backward Transfer: $BWT = \frac{1}{T-1} \sum_{i=1}^{T-1} (R_{T,i} - R_{i,i})$
- Forward Transfer: $FWT = \frac{1}{T-1} \sum_{i=2}^{T-1} R_{i-1,i} - \bar{b_i}$

3.4.3 State-of-the-art baselines

To validate the effectiveness of our method in alleviating forgetting and enabling knowledge transfer, we compared our model with the following described state-of-the-art algorithms.

- **Separated model learning (SML)**: Separate models are trained for every task, achieving the highest possible accuracy by dedicating all the network resources to that single dataset. In this case, there is no inductive knowledge transfer or catastrophic forgetting.
- **SGD**[10]: A traditional model used in IoT trained with stochastic gradient descent.
- **EWC**[41]: A representative model for the regularization method, utilizing the Fisher information matrix as a quadratic penalty.
- **PackNet**[61]: This model also uses the pruning technique; it freezes learned parameters and uses free parameters for learning new data.

3.4.4 Implementation details

We apply the methodology to ResNet-18[29] architecture for the experiments. ResNet-18 contains approximately 11.17 million parameters with parameter size 42.62 MB. We used Pytorch[71] and Torchvision[62] libraries to implement neural networks. All of the training images were scaled and normalized before training as preprocessing. Identical processes were applied to the test images. The optimizer was stochastic gradient descent (SGD), with a 0.001 learning rate, 0.9 as the momentum value, and Nesterov Accelerated Gradient for regularization.

To guarantee completely reproducible results, we set seed value as 5 for the random function of Numpy, python Random, Pytorch, Pytorch Cuda, and set Pytorch back-

ends Cudnn benchmark as False, with Deterministic as True, configuring PyTorch to avoid using nondeterministic algorithms for some operations, so that multiple calls to those operations, given the same inputs, will produce the same result.

Algorithm 2 shows the learning procedure of E-DomainIL. We keep all the settings the same for our method and the baselines.

For Separated model learning, we use one network for training on every single dataset and do not finetune it on other datasets.

For naïve settings with SGD, we simply finetune the network on each new dataset without any network change.

To implement EWC, the Fisher information matrix was tracked using training data from previous tasks. Before starting the training, the network copied its weight values and compared them with current weights, considering the Fisher matrix. We used EWC hyper-parameter λ as 0.3 and 1.

For PackNet, we implement it in a domain-incremental setting that merge output units into a single-headed classifier, marked as **PackNet*** in this chapter. We train the same epochs as other methods rather than using a pre-trained model. We select ten epochs of sparse training after pruning, as the setting mentioned in its paper. We prune 1/3 of the network for training on each dataset.

3.5 Results and discussions

The performance of our method and other baselines is summarized in tables. We report the average accuracy (ACC), backward knowledge transfer (BWT), and forward knowledge transfer (FWT) for MNIST Variants, DomainNet, and Food Safety dataset sequences. E-DomainIL outperforms incremental learning baselines on ACC and BWT of all dataset sequences.

Table 3.2: Average accuracy ACC (%), backward knowledge transfer BWT (%), forward knowledge transfer FWT (%) and used parameters of E-DomainIL and other baseline methods evaluated on MNIST Variants.

	MNIST Variants			
Methods	ACC	BWT	FWT	Used Params ($\times 10^7$)
SML	98.89	-	-	3.3518
SGD	75.48	-35.09	0.55	1.1174
EWC (λ=0.3)	71.07	-41.71	1.87	1.1174
EWC (λ=1)	75.32	-35.38	0.95	1.1174
PackNet*	84.09	-22.18	0.93	1.1174
E-DomainIL	**96.37**	**-2.744**	**2.81**	**0.5130**

Specifically, as shown in Table 3.2, for the MNIST Variants, E-DomainIL improves the network average accuracy ACC by 27.68% and the ability to alleviate forgetting BWT by 92.18%. It outperforms PackNet* and EWC in ACC and BWT. Compared with separated model learning (SML), where separate models are trained for every task, achieving the highest possible accuracies by dedicating all the network resources for that single dataset with no catastrophic forgetting, E-DomainIL almost matches its ACC using only about a quarter of SML's parameters. As can be seen from Figure 3.4, the method can effectively overcome catastrophic forgetting through task-agnostic parameter isolation. In the training process of dataset B (20-40 epochs), the test accuracy of dataset A is improved with data-driven pruning, indicating that pruning may promote model generalization to some extent.

For DomainNet sequence in Table 3.3, E-DomainIL improves the network performance by 18.14% in average Accuracy and 47.92% in backward transfer. With fewer parameters, it beats the PackNet* and EWC in ACC and BWT. It still has some gaps in ACC compared with SML, as DomainNet data is more complex and has more sig-

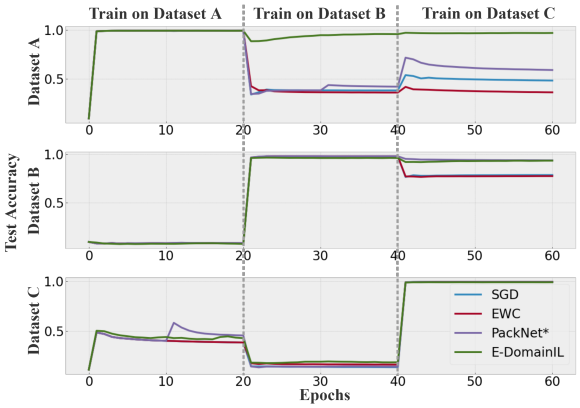


Figure 3.4: Visualization of test accuracy on three MNIST Variant datasets during training using SGD, EWC, PackNet*, and E-DomainIL. (Best viewed in color)

nificant domain shifts than MNIST. From the Figure 3.5, our method can significantly inhibit catastrophic forgetting, but due to the limit of the number of parameters, the learning speed of new datasets will be affected. This is due to the improper selection of the hyperparameter β, which leads to a large proportion of network sparsity in the loss function. On the other hand, the network's learning ability is also limited by its capacity. Since our algorithm is stepwise, the selection of hyperparameters is influenced by the training iteration, which is related to the number of images in the dataset. In the future, we will change the training hyperparameters according to the dataset to better adapt to different types of datasets.

As shown in Table 3.4, when evaluated on the Food Safety dataset, E-DomainIL has a positive backward knowledge transfer of 1.075%. By referring to Figure 3.6, we can see that with E-DomainIL, the learning of dataset B can slightly improve dataset A's accuracy. Due to the limited number of parameters, our method does not achieve the same accuracy as training SML with separate models, but it can achieve approximate accuracy using only about $\frac{1}{6}$ of the total parameters of SML.

The results show that through task-agnostic parameter isolation, neural networks can overcome forgetting to a great extent without the need for task labels. Furthermore, with data-driven pruning, the accuracy of the model learning dataset is improved,

Table 3.3: Average accuracy ACC (%), backward knowledge transfer BWT (%), forward knowledge transfer FWT (%) and used parameters of E-DomainIL and other baseline methods evaluated on DomainNet.

	DomainNet			
Methods	ACC	BWT	FWT	Used Params ($\times 10^7$)
SML	57.75	-	-	3.3518
SGD	36.75	-36.00	9.25	1.1174
EWC (λ=0.3)	37.17	-36.62	7.75	1.1174
EWC (λ=1)	37.75	-37.25	7.50	1.1174
PackNet*	39.25	-31.50	4.75	1.1174
E-DomainIL	**43.42**	**-18.75**	1.25	**0.4323**

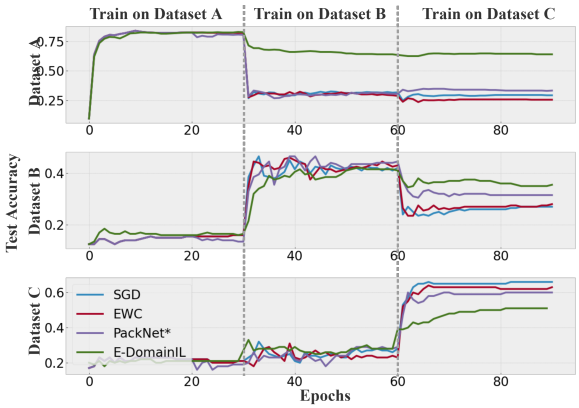


Figure 3.5: Visualization of test accuracy on three DomainNet datasets during training using SGD, EWC, PackNet*, and E-DomainIL. (Best viewed in color)

Table 3.4: Average accuracy ACC (%), backward knowledge transfer BWT (%), forward knowledge transfer FWT (%) and used parameters of E-DomainIL and other baseline methods evaluated on Food Safety datasets.

	Food Safety			
Methods	ACC	BWT	FWT	Used Params ($\times 10^7$)
SML	98.92	-	-	3.3518
SGD	77.42	-43.01	36.84	1.1174
EWC (λ=0.3)	75.47	-45.16	36.84	1.1174
EWC (λ=1)	76.34	-45.16	36.84	1.1174
PackNet*	89.45	-17.20	28.07	1.1174
E-DomainIL	**95.95**	**1.075**	**40.35**	**0.5600**

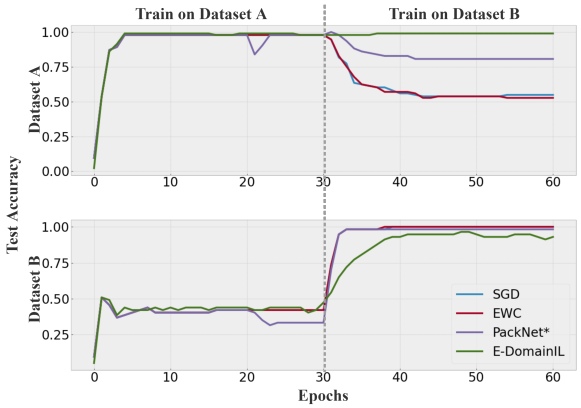


Figure 3.6: Visualization of test accuracy on Food Safety datasets during training using SGD, EWC, PackNet*, and E-DomainIL. (Best viewed in color)

and more parameters are saved.

3.6 Chapter Summary

We propose E-DomainIL, a memory-efficient method for IoT systems that enables incremental learning of domain shifts in a fixed neural network. Our method uses task-agnostic parameter isolation to learn domains incrementally while retraining the accuracy of learned domains without requiring additional task labels. It uses data-driven pruning to adjust the parameter ratio according to the dataset, thus more parameter efficient than existing methods. E-DomainIL has shown outstanding performance in experiments, supporting maximum accuracy using half the parameters. Our method is a general design, and we can easily adopt it for several IoT scenarios such as smart homes, autonomous vehicles, smart manufacturing, wearable devices, etc.

Chapter 4

Research Contribution 1b: Overcoming Catastrophic Forgetting in Non-IID Task-Agnostic Continual Federated Learning

4.1 Introduction

In the era of data-driven decision-making, Internet of Things (IoT) has become an important data source due to the wild deployment of mobile devices and IoT equipment. However, traditional centralized machine-learning training is limited due to privacy constraints [56]. To this end,Federated Learning (FL) has emerged as a pivotal technology, enabling collaborative on-device model training across multiple devices while preserving data privacy. This approach not only enhances user privacy but also facilitates the utilization of decentralized data sources, making it a cornerstone for the

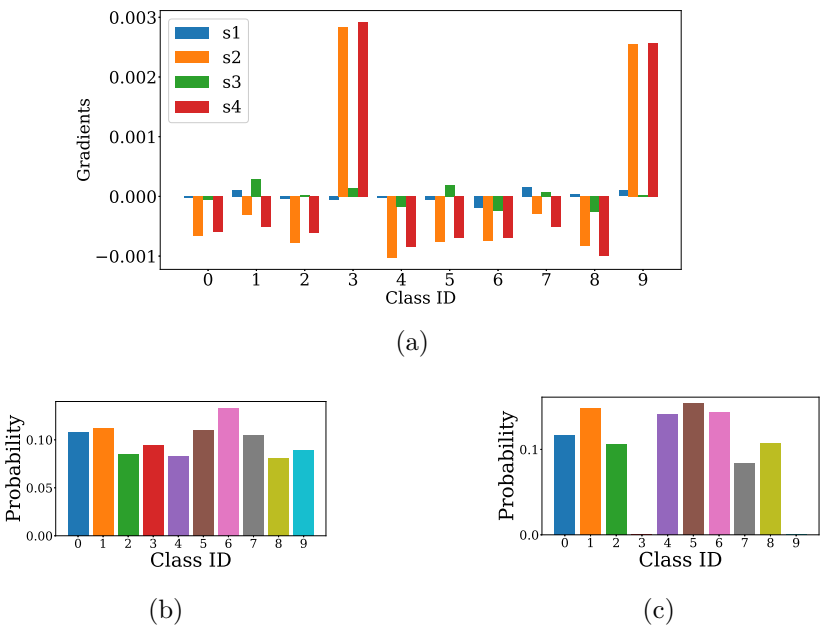


Figure 4.1: (a) Impact of imbalanced class distribution on a model's gradient updates. Gradient changes in a neural network across four scenarios (s1, s2, s3, s4) for ten classes (Class ID 0-9). The bars indicate gradient magnitudes, with colors representing different scenarios. Significant gradient increases for classes 3 and 9 in s2 and s4 point to shifts in learning focus due to class imbalance. (b) Training on a balanced dataset, the model shows no bias towards any classes. After training on an imbalanced dataset (c), the model exhibits a clear bias with significantly varied probabilities for each class. The elevated probability for certain classes suggests that the model has become more confident in these classes likely due to their overrepresentation in the training data. Conversely, the reduced probability for other classes indicates a loss of confidence, which can be interpreted as the model 'forgetting' or failing to recognize these underrepresented classes.

next generation of machine learning applications, such as medical image processing [73], cybersecurity [21], and edge computing [107].

Despite its potential, FL faces critical challenges in achieving uniform model per-

formance and generalization, particularly when dealing with non-independent and identically distributed (non-iid) data across different devices. This heterogeneity in data distribution leads to significant disparities in local model performance, as the local objective functions diverge from the global objective, pushing the aggregated global model away from the optimal solution. This often induces lower accuracy and communication efficiency of the model, more communications are required to achieve target accuracy. The non-iid problem is not only theoretical but exists in many FL applications [**34**], potentially compromising the efficacy of FL systems.

We first train a basic model with resnet18 on dataset 's1' with five epochs, and train another five epochs on 's1', 's2', 's3', 's4' respectively. The blue, orange, green and red bar denotes the four datasets respectively. The dataset 's1', 's3' are subset of CIFAR10 with all classes. The dataset 's2', 's4' are subset of CIFAR10 with classes 3 and 9 removed. The bar demonstrate the gradient of the corresponding class to the classifier, which is ∇w_i. From the graph, we can see that when a class is removed, it will show a completely different update direction. But if it's trained in a balanced distribution, there are no certain trends.

Several strategies have been proposed to mitigate these issues. The majority of attempts focused on designing alternative of raw data for sharing, e.g., parameter-sharing [57], feature sharing[106], pseudo data sharing [24], and distribution statistics sharing [3]. While these approaches aim to enrich local models with broader insights, they inherently increase communication costs and raise privacy concerns, as they necessitate additional data or metadata exchange across the network. Another stream applying generalization theories [64], proposing that enhancing the generalization capability of local models could indirectly address the non-iid challenge. Yet, without direct access to knowledge of the global data distribution, these generalization efforts often fall short, providing limited improvements in overall model performance.

In response to these limitations, the global model gained attention, since in the default FL setting, the global model will be distributed to each client, and has not been

utilized other than an initialization of local models. Although global models are often more generalized than locals [48], this ability is often biased [57] or forgotten [44, 95]. They termed this phenomenon as local model forgetting, often considered a degradation in the ability to accurately predict underrepresented classes over time. By mitigating the local model forgetting, they observe an increment of top-1 accuracy and communication efficiency, making it an important challenge in non-iid FL [44, 95, 86, 3, 99].

In response to this challenge, the concept of utilizing the global model as a teaching mechanism for local models has emerged [82]. This approach leverages the more generalized nature of the global model, aiming to guide local training without the direct exchange of data. However, the effectiveness of such knowledge transfer is markedly reduced in the presence of imbalanced local data distributions. Recent studies have shown the importance of considering the imbalanced data distribution during the distillation [3, 90], but additional information from other clients is required. Given the challenges posed by non-IID data distributions in FL, particularly the phenomenon of model forgetting in local models, we ask:

How can we effectively leverage the global model to enhance the learning and generalization ability of local models in a federated learning system without extra information sharing?

This study delves into the phenomenon of local model forgetting in the context of non-iid data distribution. Our investigation reveals that the core of the forgetting issue is primarily rooted in the skewed distribution of local data, where certain classes are represented by fewer samples than others. Such an imbalance leads to insufficient "positive update" during the training process, meaning that the model does not receive enough reinforcement to accurately learn and retain information about these sparsely represented classes.

Figure 4.1a illustrates the effects of imbalanced class distribution on gradient updates within FL context, using the ResNet-18 architecture trained on CIFAR10 dataset variations. We observe the model's performance across four different training scenarios: "s1" and "s3" datasets, which include all classes, and "s2" and "s4" datasets, from which class 3 and 9 are removed intentionally to simulate imbalanced class distribution. The analysis of gradients—specifically, how they change in response to the absence or presence of certain classes—reveals that removing classes leads to significant shifts in the model's learning focus. This is depicted through divergent gradient updates for class 3 and 9, indicating the model's "forgetting" or loss of ability to generalize to those classes that became underrepresented or absent in the training data.

Following the initial observations, our investigation delves deeper into the mechanisms behind these shifts in learning focus. By analyzing the probability outputs for each class before and after training on both balanced ("s1", "s3") and imbalanced ("s2", "s4") datasets as shown in Figure 4.1, we gain insights into how the model's predictive confidence is affected by the presence or absence of certain classes.

These two experiments demonstrate that the absence of samples for specific categories will bias local models in non-iid FL through gradients, leading to the local model forgetting. This analysis not only confirms the impact of imbalanced datasets on the learning process but also underscores the necessity of addressing this challenge to prevent the degradation of model generalization in FL environments.

We propose a new learning framework, FedDistill, designed to enhance knowledge transfer from the global model to local models in FL, effectively addressing the issue of imbalanced class distribution.

This approach integrates group distillation with a novel de-biasing approach for the local classifier and feature extractor, targeting the root causes of the 'forgetting' phenomenon in local models. We introduce group distillation, a strategy that seg-

ments classes into categories based on their sample abundance in local datasets. This categorization enables a targeted distillation process, where knowledge transfer is customized to bolster the learning of few sample classes, ensuring they receive adequate attention and reinforcement. Furthermore, the global model is decomposed into two components: a feature extractor and a classifier so that local models can learn more effectively from the global knowledge. The feature extractor part of the global model helps local models learn generalized representations of the input data, while the classifier part assists in accurately predicting the class labels. We design a new way of de-bias these two parts, which is improving the generalization ability of local feature extractor and classifier with two additional loss. By leveraging the global model as a more effective teacher through group distillation and strategic decomposition, we enable local models to retain their generalization ability over time. This approach mitigates the adverse effects of data imbalance, ensuring that local models do not forget underrepresented classes but instead become more adept at recognizing and classifying them accurately.

Our contributions are as follows:

- To the best of our knowledge, this is the first in-depth study to identify and analyze the main reason behind the forgetting in local models within the FL framework. We demonstrate that the core issue is the insufficient positive updates for classes with few or no samples during local training, leading to an imbalanced classifier that significantly limits the model's generalization capabilities.

- We introduce a new learning framework, FedDistill, that incorporates group distillation, tailored to local data distribution, and a novel approach to de-bias the decomposed local model. This solution is designed to optimize the guidance provided by the global model to local ones, specifically addressing the imbalance in class representation without introducing extra communication overhead or

privacy concerns.

- Through extensive experiments, we comprehensively analyze the efficacy of our proposed components. Our empirical studies on commonly used benchmark datasets reveal that our method achieves state-of-the-art performance, significantly mitigating the forgetting issue and enhancing model generalization across diverse classes. In addition, our method shows a higher communication efficiency against all baselines.

4.2 Related Works

Non-IID Federated Learning (FL). Federated Learning, a paradigm for training models across decentralized datasets without compromising privacy, faces significant challenges in non-iid scenarios. Despite FedAvg's foundational role in FL [63], its performance degrades under non-iid data distributions [112]. Subsequent efforts to mitigate these challenges have bifurcated into enhancing global aggregation [14, 52, 92, 108] and refining local training methodologies [64, 106, 37, 48]. While global-side improvements are constrained by privacy regulations, leading to minimal data-driven modifications, local-side enhancements have flourished. Innovations like FedAlign[64] emphasize optimizing local model generalization, while approaches like FedUFO[106] and SCAFFOLD[37] introduce mechanisms to foster client-agnostic feature learning and update calibration, respectively. Notably, MOON[48] leverages contrastive learning to align local model features with the global model, underscoring the evolving landscape of strategies aimed at reconciling local and global learning objectives.

Forgetting in Non-iid Federated Learning. The phenomenon of forgetting, borrowed from lifelong learning, has been recognized in FL, particularly in the context of local training. [86] is among the first proposing that forgetting might be an issue in

non-iid Federated Learning. [44, 95, 3, 99] provides different empirical results showing local models forgetting the generalization ability from the global model during training. However, they didn't analyze the reason why this forgetting happens. In this paper, we argue that forgetting is incurred due to insufficient positive update for the classes with few or no samples, surprising the local model from retaining the generalization ability of the global model.

With different observations, approaches are proposed to alleviate the forgetting in non-iid federated learning. [86, 25] proposed a regularization-based method that prevents the parameter changes from the global model to the local model, which limits the learning ability of the local model. FedNTD[44] proposed to use the global model as the teacher to guide the local model with not-true distillation; FedRAD[90] proposed to use the entropy of the global model as an indicator to adaptively adjust the weight of knowledge distillation and cross-entropy. But both of them didn't consider the imbalanced nature of the local data distribution.

FedCSD[99] proposed to use logit as guidance, where an additional logit prototype for each class will be maintained through the training, an extra teacher model is also needed to calculate the logit from the logit prototype to guide the training of the local model. Both the logit prototype and teacher models need to be uploaded/downloaded for each local training, bringing in extra communication overhead. Flashback[3] also noticed the imbalanced data distribution across clients and used the label count as the indicator of knowledge distillation, however, they rely on a global label count estimation which requires the label count of clients from the last round, which brings potential privacy concerns. In our paper, we don't need any other information except the global model on the local side and don't upload any extra information except the trained local model, which poses a more challenging question compared to other methods.

FedReg[95] proposed to generate pseudo data on the local side that indicates the biggest changes of prediction from the global model to the local model. This behavior

brings potential privacy concerns and burdens the local training process.

Federated Learning with local model de-biasing. Apart from forgetting to describe the model inconsistency from the global model and across client models, some researchers believe the local models are skewed by the local data. [57] observe that the local model has parameter drift from the global model the drift level of which decreases from the bottom layers to the top. With this observation, they decompose the model into a feature extractor and classifier and calibrate the biased classifier by single-regularizing the parameter drift of the classifier. FedBR[24] takes local feature extractor de-biasing into consideration. FedBR designs an adversarial learning framework on the local side, where a discriminator is trained to discriminate the feature extracted from local and global models on pseudo data while minimizing the distance between local and global features on local data. In our paper, we consider both feature extractor and classifier de-biasing without pseudo data, by further leveraging the global model.

Given that most of the existing methods either ignore the importance of imbalanced local data in local model forgetting or rely on extra information sharing, we proposed FedDistill. Our contribution to this evolving field includes a novel decomposition of knowledge distillation into three parts: true class, few-sample class, and rich-sample class, directly addressing the challenge of imbalanced data distribution. The true class represents the actual, ground truth category or label of a data sample. Furthermore, by decomposing both the global and local models, we apply KD in a manner that effectively prevents drift from the global model to local models, ensuring a more balanced and representative learning process. Notably, our method emphasizes improvements on the local side, without any additional communication overhead and privacy concerns typically associated with global-side enhancements.

4.3 Method

4.3.1 Preliminary

In the FL setting, following the FedAvg algorithm, we consider a set of clients S, where each client $s \in S$ possesses a private dataset $\mathcal{D}_s$. Each client has an associated image classification model represented by parameters θ_s. The global model, shared among all clients, is denoted by θ_g, and the collective dataset from all clients is represented as $\mathcal{D} = \bigcup_{s \in S} \mathcal{D}_s$. The FL process is structured into T communication rounds. In each round t, a subset of clients $S^{(t)}$ is selected, and the global model from the previous round $\theta_g^{(t-1)}$ is distributed to each selected client $s \in S^{(t)}$. This model initializes their local model for the current round as $\theta_s^{(t)} = \theta_g^{(t-1)}$, upon which local training is conducted.

The aggregation of trained local models to update the global model is performed by averaging their weights, given by:

$$\theta_g^{(t)} = \frac{\sum_{s \in S^{(t)}} |\mathcal{D}_s| \theta_s^{(t)}}{\sum_{s' \in S^{(t)}} |\mathcal{D}_{s'}|} \tag{4.1}$$

The overall objective is to minimize the empirical loss of the global model across the entire dataset, formalized as:

$$\min_{\theta_g} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[\mathcal{L} \left(f(\boldsymbol{x}; \theta_g), y \right) \right] \tag{4.2}$$

where $\mathcal{L}$ denotes the loss function, $(\boldsymbol{x}, y)$ are the input image and corresponding ground-truth label pairs, and $f(\boldsymbol{x}; \theta_g)$ represents the mapping from the input space $\mathcal{X}$ to the label space $\mathcal{Y}$, parameterized by the global model θ_g.

4.3.2 Understanding Forgetting in Federated Learning

The phenomenon of forgetting within local models during Federated Learning (FL) training sessions significantly hampers their ability to generalize, particularly for

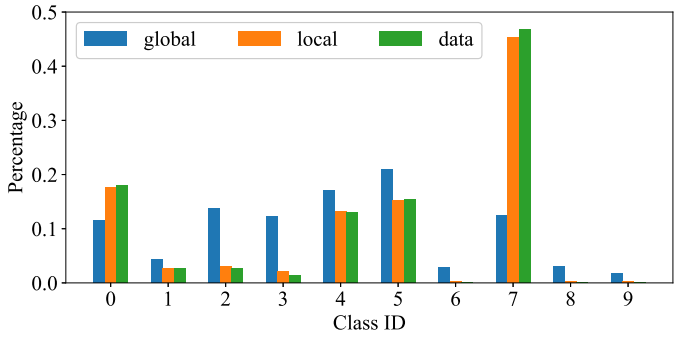


Figure 4.2: Illustration of softmax output distribution disparities between global and local models under imbalanced class distribution on client 0, highlighting the local model's inclination towards fitting its specific dataset and the global model's balanced approach.

classes that are underrepresented in their datasets. This subsection delves into the mechanisms contributing to this issue and proposes a novel perspective on mitigating its impact through the strategic use of the global model.

Previous investigations have identified client drift, backward transfer (BwT), and unbalanced loss as indicators of forgetting during local training sessions [57, 44, 95]. However, these studies fall short of pinpointing the precise mechanism of forgetting induced by non-iid data distributions. Our analysis reveals that the core issue lies within the cross-entropy loss function, a common choice for classification tasks in FL.

The role of cross-entropy loss. For a given classification model θ, which comprises a feature extractor E ($E(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}^d$, mapping inputs to a feature space) and a classifier FC ($FC(E(\boldsymbol{x})) = W^T E(\boldsymbol{x})$, mapping features to class probabilities), we can express the output probability for a class $c \in \mathcal{C}$ as:

$$q_c(\boldsymbol{x}) = \frac{\exp\left(\boldsymbol{w}_c^T E(\boldsymbol{x})\right)}{\sum_k^{|\mathcal{C}|} \exp\left(\boldsymbol{w}_k^T E(\boldsymbol{x})\right)},$$

with the cross-entropy loss calculated as:

$$\sum_{c \in \mathcal{C}} p_c \log \left(q_c(\boldsymbol{x}) \right),$$

where p_c represents the ground truth. The derivative of the loss with respect to the weight of the ground-truth class c is positive, promoting learning for this class:

$$\frac{\partial \log \left(q_c \right)}{\partial \boldsymbol{w}_c} = \left(1 - q_c \right) E(\boldsymbol{x}),$$

while for non-true classes $\bar{c}$, it's negative, potentially decreasing their weight:

$$\frac{\partial \log \left(q_c \right)}{\partial \boldsymbol{w}_{\bar{c}}} = -q_{\bar{c}} E(\boldsymbol{x}).$$

This differential treatment can lead to the forgetting of underrepresented classes, as their weights diminish over time due to insufficient positive update.

Fitting local data and losing generalization. Fig. 4.2 showcases a FedAvg scenario where the local model's softmax output distribution significantly deviates towards classes overrepresented in its dataset, compromising its ability to generalize. In contrast, the global model maintains a more balanced output distribution despite the local data imbalance.

Proposed mitigation through global model guidance. In response to these challenges, our approach advocates for a re-envisioned application of the global model in guiding local training. Unlike prior efforts that solely focus on optimizing local models' generalization abilities or sharing parameters/data characteristics, we propose a method that incorporates class imbalanced data distribution directly into the training process. This involves using the global model to provide targeted guidance to local models, enhancing their ability to retain knowledge of underrepresented classes without introducing additional communication burdens or privacy concerns.

By situating the global model as a more generalized teacher and decomposing it

into distinct feature extractor and classifier components, we offer a nuanced strategy that not only addresses the forgetting issue but also respects the federated learning paradigm's foundational principles of efficiency and privacy.

4.3.3 FedDistill Framework

The FedDistill framework introduces an advanced approach to Federated Learning (FL) by addressing the challenges posed by non-iid data distributions across clients. At the heart of FedDistill is the innovative use of group distillation (GD) Loss, which modifies the traditional knowledge distillation process to better suit the federated context. This section outlines the FedDistill framework, emphasizing the strategic components designed to enhance local model performance through global model insights.

Fig. 4.3 illustrates the framework's structure, where both global and local models contribute to a cross-learning environment. This setup not only leverages the generalized capabilities of the global model but also specifically targets the improvement of local models' generalization and robustness against class imbalance.

Addressing local model forgetting. In federated learning (FL), where data is inherently non-IID across clients, traditional knowledge distillation (KD) methods relying on KL divergence often fall short. These methods aim to minimize the discrepancy between the teacher (global) and student (local) models, but this approach inadvertently restricts local models from learning new knowledge, particularly when there is a significant divergence between global and local data distributions.

To address this limitation, we propose preserving knowledge from the global model for underrepresented classes (few-sample classes) in local clients while allowing local models the flexibility to learn new knowledge from well-represented classes (rich-sample classes). By separating classes into two groups during distillation, our approach, termed Group Distillation (GD), ensures a balanced trade-off between preserving

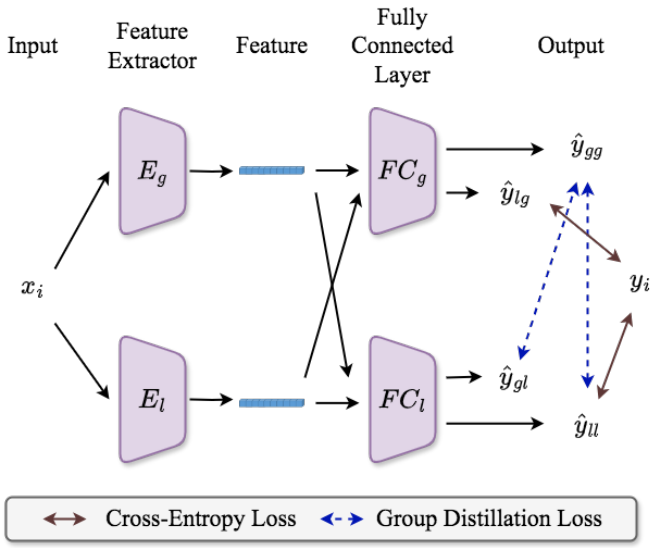
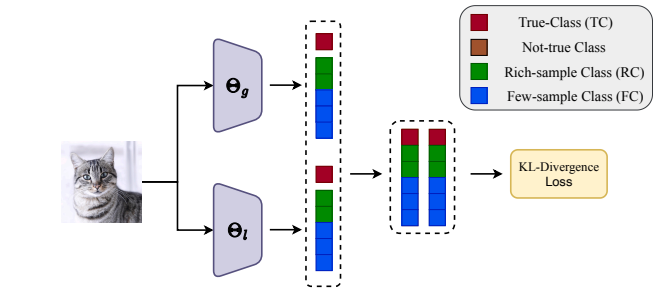


Figure 4.3: Overview of the FedDistill framework, detailing the interplay between global and local models' feature extractors and classifiers for an input x_i. Specifically, x_i is the input for the global feature extractor E_g and local feature extractor E_l, respectively. After that, the features will be input into the classifiers FC_g and FC_l of both the global and the local model. $\hat{y}_{gg}$, $\hat{y}_{gl}$ denotes the output of global and local classifier with the global feature, respectively, and $\hat{y}_{lg}$, $\hat{y}_{ll}$ denotes the output of global and local classifier with the local feature, respectively. y_i is the real output.

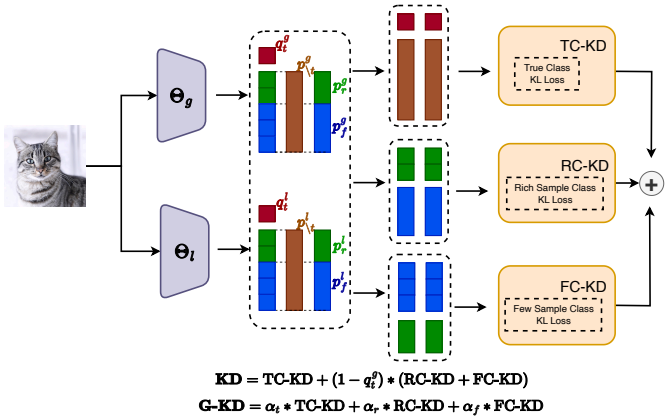
global knowledge and enabling local adaptation.

As demonstrated in section 4.4.4, GD achieves higher global model consistency and improved top-1 accuracy during training, even with slightly increased local model divergence. This increased divergence is anticipated, as it reflects the local models' ability to assimilate new knowledge effectively.

Our novel GD Loss approach modifies the traditional use of KL divergence to better accommodate the unique distributional characteristics of FL. As stated in Sec.4.3.2, forgetting was caused by the insufficient positive update, which associates with the number of local samples. Empirically, we take classes with samples more than $\frac{1}{|C|}$ as rich-sample classes, and those have fewer as few-sample classes. We term this thresh-



(a) Traditional Knowledge Distillation



(b) Group Distillation

Figure 4.4: Illustration of the traditional KD and our G-KD. We reformulate the traditional KD into three parts: (1) The true class KL loss (TC-KD), which has been discussed in DKD[110]. (2) The rich sample KL loss (RC-KD), denotes the KL loss for the rich sample classes that were highly represented by the local modal. (3) The few sample KL loss (FC-KD), denotes the KL loss for the few sample classes that were underrepresented by the local model. By separating the classes, we intended to accommodate the imbalance in the local dataset distribution by adjusting the weight $(\alpha_t, \alpha_r, \alpha_f)$ correspondingly.

old as 'few-sample threshold γ'. Each client's data is then classified into rich-sample classes if the sample proportion exceeds γ; otherwise, it is categorized into few-sample classes. Apart from the rich-sample and few-sample classes, we also separate the true classes as discusses in DKD[110]. We apply a differentiated distillation technique that recalibrates the focus of local models towards achieving a more balanced learn-

ing outcome across all classes. A visual demonstration of GD Loss is shown in Figure 4.4.

To detail, we reformulate the KL divergence to encapsulate the differentiated treatment of each class group, thus ensuring a more equitable learning process across classes with varying levels of representation:

$$\begin{aligned} \mathrm{GD}\left(\boldsymbol{q}^g \| \boldsymbol{q}^l\right) = \alpha_t \, &\mathrm{TC\text{-}KD}(\boldsymbol{q}^g, \boldsymbol{q}^l) \\ + \alpha_r \, &\mathrm{RC\text{-}KD}(\boldsymbol{q}^g, \boldsymbol{q}^l) \\ + \alpha_f \, &\mathrm{FC\text{-}KD}(\boldsymbol{q}^g, \boldsymbol{q}^l), \end{aligned} \tag{4.3}$$

where TC-KD, RC-KD, and FC-KD represent the components of our GD Loss corresponding to true-class, rich-sample class, and few-sample class, respectively. Each component is scaled by its respective weighting factor (α_t, α_r, and α_f) to ensure that the distillation process is finely tuned to the specific learning needs and challenges posed by the class distribution within each local dataset. More specifically, their formulations are as follows:

$$\mathrm{TC\text{-}KD}\left(\boldsymbol{q}^g \| \boldsymbol{q}^l\right) = q_t^g \log\left(\frac{q_t^g}{q_t^l}\right) + p_{\setminus t}^g \log\left(\frac{p_{\setminus t}^g}{p_{\setminus t}^l}\right) \tag{4.4}$$

where q_t denotes the logit of the true class, and $p_{\setminus t}$ denotes the logit sum of the not-true classes.

$$\mathrm{RC\text{-}KD}\left(\boldsymbol{q}^g \| \boldsymbol{q}^l\right) = \left(\sum_{i \in \mathcal{C}_r \setminus \{t\}} \tilde{q}_i^g \log\left(\frac{\tilde{q}_i^g}{\tilde{q}_i^l}\right) + \tilde{p}_f^g \log\left(\frac{\tilde{p}_f^g}{\tilde{p}_f^l}\right)\right) \tag{4.5}$$

where $\tilde{q}_i = \frac{q_i}{p_{\setminus t}}$, $\tilde{p}_f = \frac{\sum_{i \in \mathcal{C}_f \setminus \{t\}} q_i}{p_{\setminus t}}$, and $\mathcal{C}_r$, $\mathcal{C}_f$ denotes the rich-sample and few-sample classes respectively.

$$\text{FC-KD}\left(\boldsymbol{q}^g \| \boldsymbol{q}^l\right) = \left(\sum_{i \in \mathcal{C}_f \setminus \{t\}} \tilde{q}_i^g \log\left(\frac{\tilde{q}_i^g}{\tilde{q}_i^l}\right) + \tilde{p}_r^g \log\left(\frac{\tilde{p}_r^g}{\tilde{p}_r^l}\right) \right) \tag{4.6}$$

where $\tilde{p}_r = \frac{\sum_{i \in \mathcal{C}_r \setminus \{t\}} q_i}{p_{\setminus t}}$. When α_t, α_r, and α_f are set to 1, $p_{\setminus t}$, $p_{\setminus t}$ respectively, GD Loss is equivalent to KL divergence Loss.

Our reconceptualization of KL divergence, pivotal to the GD Loss, addresses the prevalent class imbalance issue in FL, ensuring effective knowledge transfer from the global to local models. By ensuring that the distillation process respects and responds to the unique class distribution of each local dataset, we facilitate a more balanced and effective knowledge transfer from global to local models. This strategy not only counteracts the adverse effects of non-iid data on model performance but also significantly enhances the robustness and generalization capabilities of FL systems.

Local model de-biasing. Inspired by [57, 24], we decomposed models into feature extractor and classifier and designed a novel distillation framework to prevent the biasing of local imbalanced data but without leveraging extra information. Since the global model can be viewed as a more generalized model compared to local ones [48], we assume that the feature extractor and the classifier of the global model are also generalized. To preserve the generalization ability, we intend to let global teach the local model, by utilizing the group distillation.

$$\mathcal{L}_{\text{L}} = \text{GD}(\hat{y}_{gg} \| \hat{y}_{ll}) \tag{4.7}$$

where $\hat{y}_{gg}, \hat{y}_{ll}$ are the output of the global and local model with the same input $\boldsymbol{x}$. The GD process aims to align the local model's predictions more closely with those of the global model, thereby leveraging the global model's generalization capabilities to mitigate biases and imbalances within the local model's learning process.

Acknowledging the influence of imbalanced local data on the classifier's performance, an innovative step is taken to integrate the global classifier in evaluating the local

feature extractor. By processing local features through the global classifier (FC_g), we obtain a less biased prediction ($\hat{y}_{lg}$), which serves as a more reliable indicator of the local feature extractor's (E_l) generalization performance. This process aims to ensure that the local feature extractor remains discriminative and effective across both biased and unbiased classifiers, with the goal articulated through the following loss function.

$$\mathcal{L}_{\mathrm{E}} = \mathrm{CE}(\hat{y}_{lg}, y) \tag{4.8}$$

focusing on minimizing the cross-entropy (CE) loss between $\hat{y}_{lg}$ and the ground-truth label y.

Further, to address biases within the local classifier, we explore the concept of preserving the information conveyed by the global classifier. This is achieved by processing features extracted by the global feature extractor (E_g) through the local classifier (FC_l), generating a prediction ($\hat{y}_{gl}$) that reflects the performance of an unbiased feature extractor when coupled with a biased classifier. This strategy, aimed at preserving ranking information, is quantified through the minimization of Group Distillation between $\hat{y}_{gl}$ and the global model's predictions ($\hat{y}_{gg}$).

$$\mathcal{L}_{\mathrm{FC}} = \mathrm{GD}(\hat{y}_{gg} \| \hat{y}_{gl}) \tag{4.9}$$

In conjunction with these targeted interventions, the original CE loss between the local model's predictions and the ground-truth labels is maintained, ensuring the local model's foundational predictive capabilities are not compromised. The composite loss function, incorporating CE loss and the components designed to enhance feature extraction and classification, is presented as follows:

$$\mathcal{L} = \mathrm{CE}(\hat{y}_{ll}, y) + \beta_{\mathrm{L}}\mathcal{L}_{\mathrm{L}} + \beta_{\mathrm{E}}\mathcal{L}_{\mathrm{E}} + \beta_{\mathrm{FC}}\mathcal{L}_{\mathrm{FC}} \tag{4.10}$$

Table 4.1: Configurations of datasets

Datasets	CIFAR10	CIFAR100	MNIST	Tiny-ImageNet
Dataset Classes	10	100	10	200
Training Instance	50,000	50,000	60,000	100,000
Test Instance	10,000	10,000	10,000	10,000
Model	SimpleCNN	Resnet18	SimpleCNN	Resnet18
Local epochs	10	10	10	10
Batch Size	64	64	64	64
Number of Clients	100	100	100	100
Sample ratio	10%	10%	10%	10%
Non-iid Level (α)	$\{0.1, 0.3, 0.5\}$	0.1	0.1	0.1

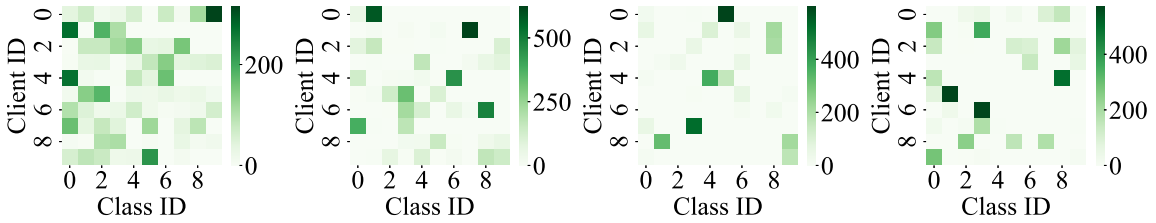
with β_{L}, β_{E}, and β_{FC} serving as weighting factors for the respective loss components. This structured approach ensures a comprehensive and balanced enhancement of local models' capabilities, addressing the nuances of non-IID data distribution and class imbalance within the federated learning paradigm.

4.4 Experiments and Results

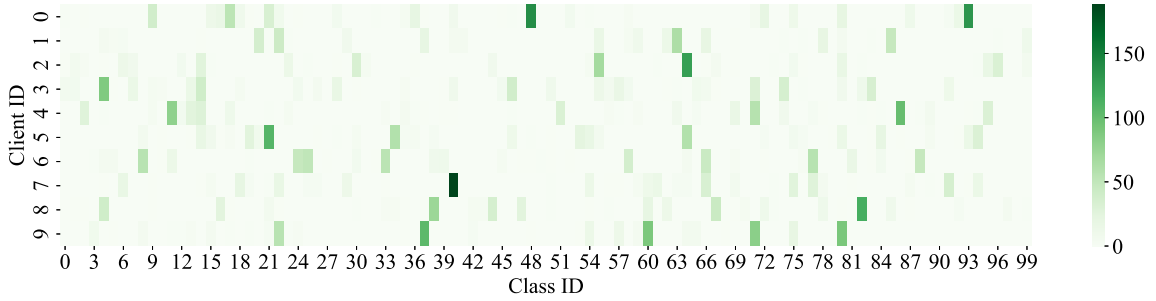
4.4.1 Experimental Setup

Datasets. We conduct experiments on commonly used datasets in FL, which are MNIST, CIFAR10, CIFAR100, and Tiny-ImageNet. We also adopt a commonly used heterogeneous dataset partition method [47, 108, 48] using Dirichlet distribution $\mathrm{Dir}(\alpha)$ to assign the label distribution among to simulate the non-iid data distribution across clients. The α is 0.1 for MNIST and CIFAR100, and $\{0.1, 0.3, 0.5\}$ for CIFAR10 to show the performance of our method under different non-iid levels. As shown in Fig. 4.5, we visualize the data distribution with responses to classes in the first ten clients. From Fig. 4.5a to Fig. 4.5c, A clear increase of data sparsity is observed. For all datasets, we set 100 clients and sampled 10% of them for each communication round. There are a total of 100 communication rounds for each dataset. More details of those datasets are shown in Table. 4.1.

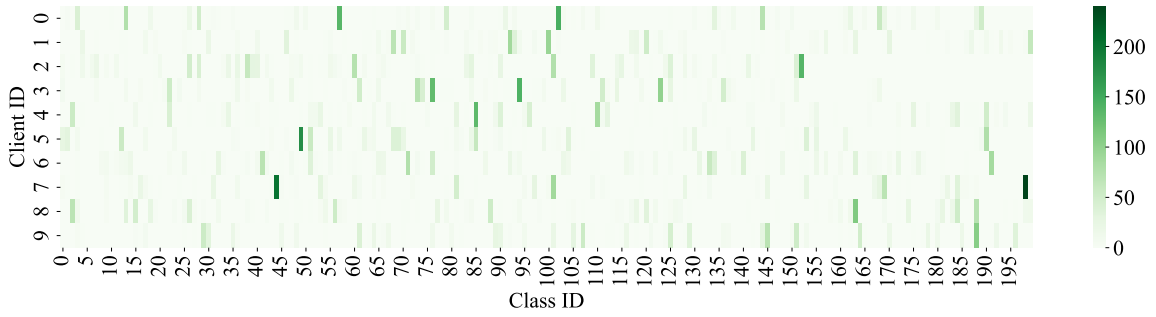
Training and Hyperparameters Configurations. Following existing works [48,



(a) CIFAR10: $\alpha = 0.5$ (b) CIFAR10: $\alpha = 0.3$ (c) CIFAR10: $\alpha = 0.1$ (d) MNIST: $\alpha = 0.1$



(e) CIFAR100: $\alpha = 0.1$



(f) Tiny-ImageNet: $\alpha = 0.1$

Figure 4.5: Data distribution with different non-iid levels in the first ten clients for all datasets. (a) - (c) demonstrate how non-iid level affects the data sparsity.

44, 108], we adopt SimpleCNN and resnet18 as the classification model. We apply resnet18 in CIFAR100 to demonstrate our method is applicable in modern CNN-based architecture. For all datasets, the number of local epochs is 10 and the batch size is 64. We take 0.01 as the learning rate for SimpleCNN and we adopt SGD as optimizer with momentum as 0.9 and weight-decay as $1e^{-5}$. We first search an optimal α_t, α_r and α_f from $\{0.0, 0.1, 0.2, \ldots, 1.0\}$. We will freeze the α_t, α_r and α_f and search the β_{L}

from $\{0.5, 0.8, 1.0, 1.1, 1.2, 1.4\}$, and β_{E}, β_{FC} from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, respectively. For the few-sample threshold γ, we empirically set it to $\frac{1}{|\mathcal{C}|}$ for each dataset.

In the main experiment, we will only search the best hyper-parameter configuration on CIFAR10 with non-iid level $\alpha = 0.1$ and maintain this setting for the rest of scenarios. It shows that without further hyper-parameter tuning, our method can still have a good performance.

Additionally, Section-4.4.5 discuss the performance influence of different hyper-parameters. The result is that our model are sensitive to α_f and α_r and need careful tuning on them, while it perform robustly with the rest four hyper-parameters.

Baselines. In our work, the adoption of widely recognized baselines such as FedAvg, FedProx, SCAFFOLD, MOON, and the more recent FedNTD, underscores a comprehensive approach to evaluating the effectiveness of our proposed solution in the context of FL under non-IID data distributions. Each of these baselines represents a significant contribution to the FL landscape, addressing various aspects of the challenges posed by non-IID data:

- **FedAvg** [63] is the foundational algorithm in FL that aggregates model updates from a diverse set of clients. It serves as the standard against which improvements in handling non-IID data are measured.
- **FedProx** [49] introduces modifications to FedAvg to better handle statistical heterogeneity and system challenges, making it a relevant comparison for assessing enhancements in model robustness and stability.
- **SCAFFOLD** [37] tackles the variance in updates due to non-IID data by correcting the client drift, providing a direct comparison for evaluating improvements in model convergence and accuracy.
- **MOON** [48] leverages contrastive learning to mitigate the effects of data heterogeneity, offering a novel perspective on enhancing model generalization across

clients with diverse data distributions.

- **FedNTD** [44] specifically addresses the issue of forgetting in FL by incorporating concepts from lifelong learning, making it an apt benchmark for comparing the efficacy of our method in maintaining model performance over time.

- **FedBR** [24] adopt adversarial learning to calibrate the feature extractor and classifier of local models in FL with auxiliary data. By comparison with FedBR, the performance difference in FL can be demonstrated with and without auxiliary data.

- **FedACG** [40] leverage meta-learning to improve the consistency among Federated learning. It records the weight update of the global model each communication round as global momentum and adds this momentum before distributing the global model to each client. The underlying motivation is to let the client consider not only their local data, but also the updates made to the global model in previous rounds.

For the implementation of baseline methods, we follow the original implementation. More specifically, we set $\mu = 0.1$ for FedProx, $\mu = 1.0$, $\tau = 1.0$ for MOON and $\mu = 1.0$, $\tau = 1.0$ for FedNTD where μ denotes the coefficient of the losses proposed by these methods, and τ denotes the temperature of the distillation. For FedBR, we use Mixture to construct the auxiliary data, and only transfer those data once for each global round. There is no hyperparameter to tune for FedAvg and SCAFFOLD. For all the baseline methods, we adopted the same network, and the same training setting(learning rate, optimizer settings, batch size, and local epochs) as our method.

By comparing our approach against these baselines, we can effectively demonstrate its advantages in terms of de-biasing local models, enhancing generalization, and reducing the impact of class imbalance in FL environments.

Evaluation. For all the results, we take the top-1 accuracy as the primary metric.

Table 4.2: Top-1 Accuracy (%) of baselines and our method on datasets. The number inside the bracket is the Forgetting measure $\mathcal{F}$.

Methods	MNIST	CIFAR10			CIFAR100	TinyImageNet
		$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$		
FedAvg	84.21 (0.157)	35.37 (0.636)	49.68 (0.432)	55.6 (0.331)	28.09 (0.374)	29.01 (0.293)
FedProx	73.63 (0.26)	34.29 (0.63)	41.77 (0.474)	44.4 (0.414)	25.87 (0.21)	29.48 (0.254)
SCAFFOLD	83.41 (0.168)	32.26 (0.332)	50.17 (0.426)	55.33 (0.332)	28.4 (0.341)	28.74 (0.265)
MOON	83.47 (0.167)	34.83 (0.642)	49.48 (0.428)	55.72 (0.327)	28.47 (0.339)	28.81 (0.315)
FedNTD	**90.68** (0.087)	<u>47.11</u> (0.462)	<u>54.85</u> (0.303)	<u>57.04</u> (0.246)	31.82 (0.151)	<u>32.12</u> (0.151)
FedBR	78.40 (0.219)	29.95 (0.684)	40.18 (0.522)	41.85 (45.31)	20.27 (0.476)	25.63 (0.327)
FedACG	81.49 (0.180)	15.46 (0.814)	48.13 (0.419)	55.97 (0.324)	**32.9** (0.25)	29.11 (0.196)
FedDistill (**Ours**)	<u>90.22</u> (0.092)	**47.37** (0.461)	**56.17** (0.309)	**61.14** (0.222)	<u>32.59</u> (0.132)	**33.04** (0.145)

We take the average of three separate runs with random seeds in 2022, 2023, and 2024 for the convenience of reproduction. Apart from the top-1 accuracy, we also introduced Backward Transfer $\mathcal{F}$ proposed in [7] as a measurement of forgetting of models. The formulation of $\mathcal{F}$ is as follows:

$$\mathcal{F} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \max_{t \in \{1,\ldots,T-1\}} \mathcal{A}_c^t - \mathcal{A}_c^T$$

where $\mathcal{A}_c^t$ denotes the accuracy of the global model on class c in the t-th communication round, and $\mathcal{A}_c^T$ is the same of the last round. All methods adopt the same model architecture and training setup specified in Table. 4.1.

4.4.2 Performance Analysis

We evaluate our model with four aspects — top-1 accuracy, forgetting measure $\mathcal{F}$, communication efficiency, and generic of extracted features, which showcase the performance, efficiency, and generic of our models against baselines. The α represents the intensity of distribution shift set during data partitioning.

Our comprehensive evaluation, as detailed in Table 4.2, showcases the top-1 accuracy and forgetting measure ($\mathcal{F}$) across various datasets for our FedDistill approach against established baselines such as FedAvg, FedProx, SCAFFOLD, MOON, FedNTD and

Table 4.3: Communication Efficiency. It demonstrates the number of communication rounds that each approach achieves the final accuracy of FedAvg on the same dataset (The lower the better).

Methods	MNIST	CIFAR10			CIFAR100	Tiny-ImageNet
		$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$		
FedAvg	100	100	100	100	100	100
FedProx	N/A	91	N/A	N/A	N/A	378
SCAFFOLD	67	94	59	62	90	498
MOON	67	N/A	59	73	80	353
FedNTD	<u>29</u>	<u>34</u>	<u>33</u>	<u>48</u>	47	<u>125</u>
FedBR	67	N/A	N/A	N/A	N/A	N/A
FedACG	81	N/A	78	78	<u>42</u>	380
FedDistill (**Ours**)	**24**	**18**	**25**	**43**	**38**	**105**

FedBR. Notably, our method demonstrates superior performance in terms of both accuracy and the mitigation of forgetting, indicative of its robustness and efficacy in non-IID Federated Learning environments.

Comparative Performance. The results show FedDistill's ability to significantly enhance model accuracy across all evaluated datasets, including MNIST, CIFAR10 with varying degrees of data heterogeneity ($\alpha = 0.1, 0.3, 0.5$), and CIFAR100. Specifically, our method achieves the highest top-1 accuracy of 90.35% on MNIST and notable improvements in CIFAR10 and CIFAR100, surpassing the next best method, FedNTD, by a margin that highlights the effectiveness of our distillation strategy.

The forgetting measure ($\mathcal{F}$) further validates our method's capability to retain learned knowledge more effectively than competing approaches. FedDistill exhibits the lowest $\mathcal{F}$ values across all datasets, affirming its superiority in addressing the critical challenge of model forgetting in FL. This is particularly evident in the CIFAR10 dataset, where our approach not only enhances accuracy but also significantly reduces the extent of forgetting compared to other methods.

Our experimental findings also reveal that FedDistill offers higher communication efficiency, where a significant fewer communication round is required for same accu-

racy. This communication efficiency is coupled with stability across different network architectures, showcasing our method's versatility. Specifically, the experiments on CIFAR100 and the comparative analysis between network architectures (resnet18 vs. simpleCNN) demonstrate that our method is adaptable to varying class numbers and network complexities.

Fig. 4.9 showcases the t-SNE visualization of features extracted by the feature-extractor of different models from MNIST with non-iid level as 0.1. The mixed class boundaries were highlighted with black circles on the graph, while the blue circle denotes a clearer class boundary compared to other methods. From the graph, we found that: (1) Almost all baselines are struggling to classify class 2 (green), 3 (red), 5 (brown), and 8 (yellow) (2) There are also mixed class boundaries between class 7 (grey) and 9 (blue), 4 (purple) and 9, and 8 and 9, which is MOON, SCAFFOLD and FedNTD, and FedNTD, respectively. (3) Our method has clear boundaries among those classes. It shows that features extracted by baseline models often failed to have clear class boundaries even with a high top-1 accuracy (89.46), while our method could extract more discriminative features, indicating that our method is capable of learning a more generic feature.

The key observations are as follows. (1) Our method effectively de-biases local models by leveraging global model insights, significantly enhancing local model generalization and reducing the impact of non-IID data distributions. (2) The direct correlation between the forgetting measure ($\mathcal{F}$) and performance underscores the importance of addressing forgetting in improving FL algorithm performance. FedDistill's lower $\mathcal{F}$ values across datasets highlight its effectiveness in mitigating forgetting. (3) Our method is able to extract more discriminative features from the data, which is aligned with our motivation of feature extraction enhancement. (4) The consistent performance of our method across diverse datasets and network architectures attests to its robustness and adaptability, making it a promising solution for a wide range of FL applications.

In summary, the empirical evidence from our analysis firmly establishes the FedDistill method as a significant advancement in FL, particularly in addressing the challenges posed by non-IID data distributions. By effectively de-biasing local models and mitigating forgetting, our approach not only enhances model performance but also contributes to the development of more generic, and efficient distributed learning systems.

4.4.3 Communication Efficiency

The computational efficiency of federated learning methods is crucial for their practical deployment, especially considering the limited communication bandwidth and the computational resources of participating clients. Table 4.3 and Fig. 4.6 presents a comparative analysis of the communication efficiency across different federated learning approaches, including our proposed FedDistill method. This metric is defined by the number of communication rounds required by each method to reach the final accuracy benchmark established by FedAvg on various datasets.

Our observations from the results highlight several key points regarding the efficiency of FedDistill compared to established methods such as FedAvg, FedProx, SCAFFOLD, MOON, and FedNTD. (1) FedDistill significantly outperforms all baseline methods in terms of communication efficiency. It achieves the benchmark top-1 accuracy set by FedAvg with considerably fewer communication rounds across all datasets tested. For instance, on MNIST, FedDistill requires only 16 rounds compared to FedAvg's 100 rounds, showcasing a drastic reduction in communication needs. (2) The efficiency of FedDistill is particularly pronounced under various levels of non-IID data distribution (α values). As the non-IID level increases from 0.5 to 0.1, our method demonstrates an increasing advantage in communication efficiency, indicating its robustness and adaptability to different degrees of data heterogeneity. (3) When compared to more recent approaches like FedNTD, FedDistill not only achieves higher

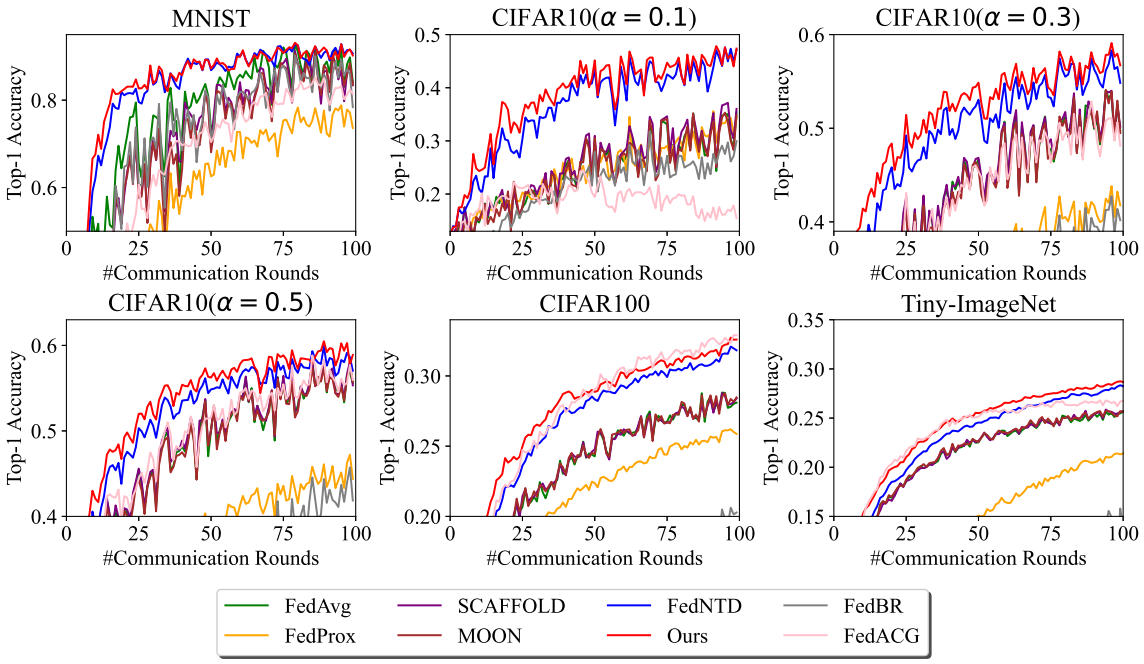


Figure 4.6: Top-1 Accuracy

accuracy but also requires fewer rounds to do so. This efficiency becomes more significant with more pronounced non-IID distributions, highlighting the effectiveness of our method in handling the challenges posed by skewed data distributions. (4) The communication efficiency of FedDistill suggests that it can significantly reduce the communication cost associated with federated learning. This reduction is critical for real-world applications where bandwidth and communication costs are limiting factors, particularly in environments with constrained resources or high data privacy requirements.

4.4.4 Effect of group distillation

This section analyses the impact of group distillation (GD) on performance, weight alignment, and weight divergence during training.

Performance. table 4.4 presents the top-1 accuracy for various GD component

Table 4.4: Demonstration of group distillation loss impact on CIFAR10 accuracy, showcasing the method's effectiveness in improving learning outcomes for few-sample classes.

Dataset	TC-KD	FC-KD	RC-KD	Top-1	Δ
CIFAR10	FedKD			43.3	
	✓			20.48	-22.82
		✓		46.9	+3.6
			✓	45.94	+2.64
	✓	✓		43.31	+0.01
	✓		✓	40.47	-2.83
		✓	✓	47.99	+4.19

combinations. FedKD serves as the baseline, employing traditional knowledge distillation (KD) during local training. The results indicate that KD on few-sample classes plays the most crucial role, aligning with our assumptions. Combining KD on both rich-sample and few-sample classes further improves performance, highlighting the necessity of KD on rich-sample classes to prevent excessive divergence of local models from the global model. However, the weight assigned to KD for rich-sample classes is generally smaller compared to that for few-sample classes, a topic explored further in section 4.4.5.

Weight alignment.

Since model aggregation is performed by averaging the weights of individual neurons, it is crucial to analyse the class preference of individual neurons. As noted in [44, 103], a neuron has a class preference vector defined as: $\mathcal{H} = [h_1, h_2 \dots h_{\mathcal{C}}]$, where $h_c = \sum_{i=1}^{N_c} \mathcal{O}(x_{c,i})$, $\mathcal{O}(x_{c,i})$ denotes the neuron's activation on data x_i of class c, and N_c is the number of samples for class c. A neuron's class preference is determined by the class with the highest activation ($\arg\max(\mathcal{H})$).

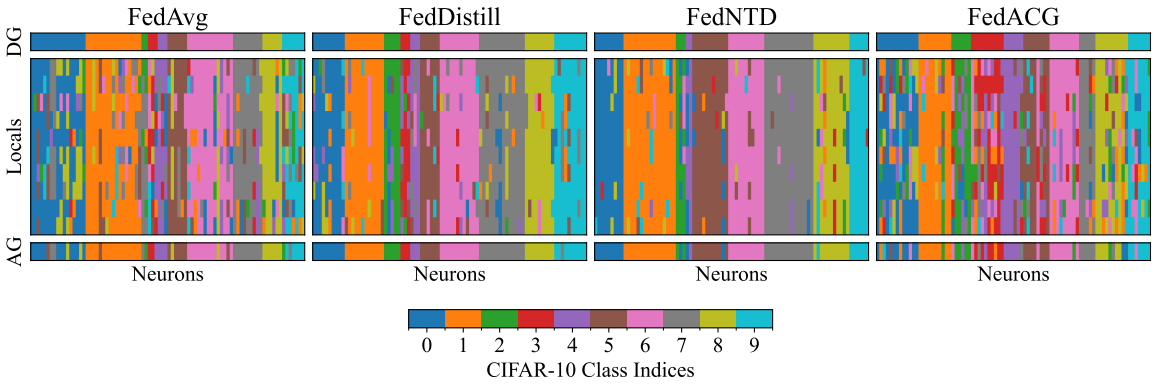


Figure 4.7: Neuron Class Preference: The figure illustrates the neuron class preference of the last linear layer of global models trained on CIFAR-10 across various methods. Colors represent the class preference of individual neurons. "DG" refers to the distributed global model from the previous round, "Locals" indicates the local models, and "AG" represents the aggregated global model of the current round.

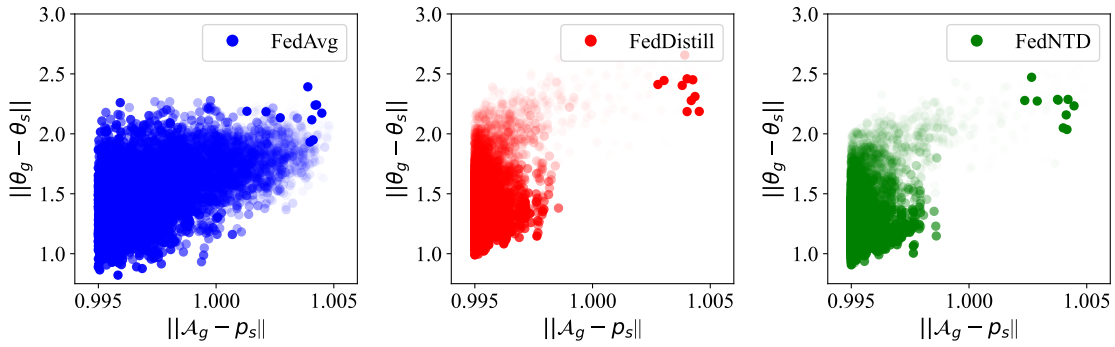


Figure 4.8: Weight Divergence: The x-axis represents the data distribution divergence, while the y-axis indicates the weight divergence. A smaller data distribution divergence implies that global models are more general and capable of fitting diverse local data distributions. In contrast, a larger weight divergence signifies that local models deviate further from the global model.

fig. 4.7 illustrates the class preferences of neurons in the last linear layer (excluding the classifier) of a Simple CNN trained on CIFAR10. The models shown include:

- DG: Distributed global model (same as the global model from the previous round).
- Locals: Local models from individual clients.
- AG: Aggregated global model (global model after this round).

The colors in the figure indicate the class preferences of individual neurons.

Our method demonstrates the highest alignment between DG and AG compared to other methods, signifying consistent training. For local models, our approach shows better alignment with the global model than FedAvg and FedACG, though slightly worse than FedNTD. This outcome is expected, as FedAvg and FedACG do not employ knowledge distillation during local training, while FedNTD does. Notably, fig. 4.7 reveals that despite greater local model divergence, our method achieves higher global model alignment, supporting the argument in section 4.3.3.

An intriguing observation is the absence of neurons preferring class-3 in both DG and AG of FedNTD. Moreover, certain classes, such as class-2 and class-4, show reduced neuron counts in AG compared to DG, indicating an imbalance in class preferences in FedNTD. In contrast, our method maintains balanced class preferences and retains neurons for underrepresented classes.

In summary, our method achieves balanced neuron class preferences and greater global model alignment during communication rounds while allowing for larger local model divergences to facilitate learning new knowledge.

Weight divergence.

Assuming a global data distribution $\mathcal{A}_g$ under which the global model is trained, the weight divergence increases when there is a significant divergence between $\mathcal{A}_g$ and local data distributions p_s. Following [63], the normalized test accuracy of the global model is used to estimate $\mathcal{A}_g$: $\mathcal{A}_g = \frac{1}{A} \cdot [a_1, \ldots, a_{|C|}]$, where A denotes the test accuracy, $a_1, \ldots, a_{|C|}$ denote class-wise test accuracy.

Figure 4.8 illustrates the relationship between weight divergence and distribution divergence during training on Tiny-ImageNet. The opacity increases with later communication rounds (from 1 to 500).

- FedAvg: Throughout training, FedAvg exhibits large data distribution divergence, indicating that its global model is biased and struggles to accommodate diverse local data distributions.

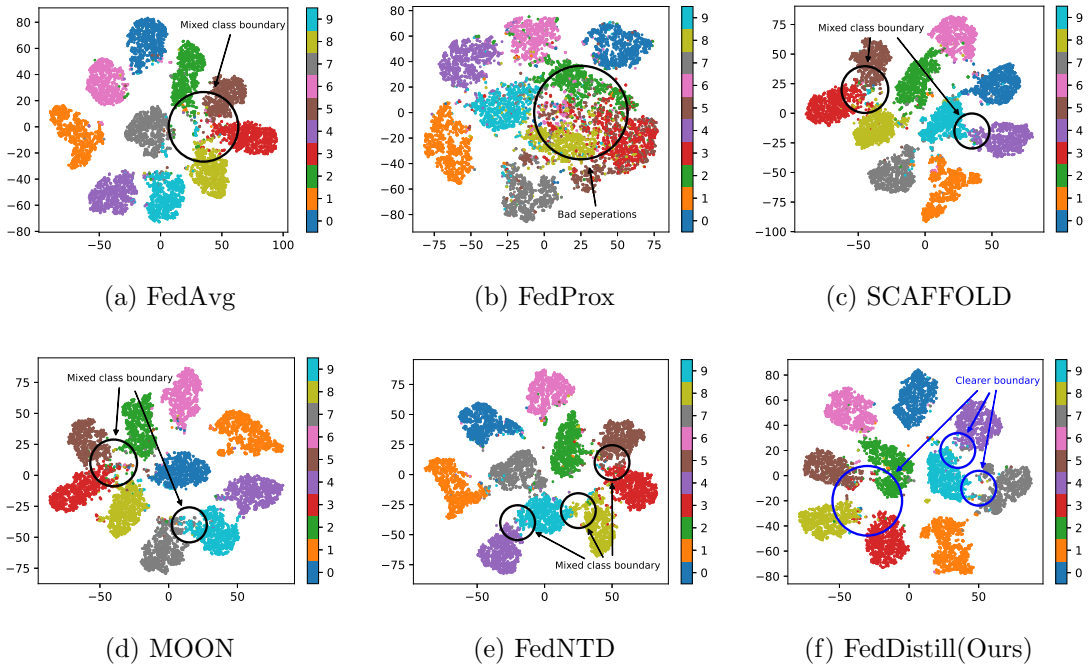
- FedNTD vs. Our Method (FedDistill): Compared to FedNTD, our global model achieves smaller data distribution divergence on average, showcasing its generalizability. Furthermore, our method requires fewer communication rounds to achieve a similar level of divergence, demonstrating faster convergence.

Our global model better aligns with local data distributions while allowing higher divergence among local models. This supports our argument in section 4.3.3, emphasizing the importance of focusing knowledge distillation (KD) on underrepresented classes while enabling local models to learn more from well-represented classes. This approach does not compromise the global model's consistency but instead enhances alignment during training.

In summary, this section validates the argument in section 4.3.3. By preserving knowledge for underrepresented classes and enabling local models to learn more from well-represented ones, our method improves global model consistency and alignment during training.

4.4.5 Ablation Study

The ablation study, as presented in Table 4.5 and Fig. 4.10, systematically investigates the impact of each proposed loss component on the overall performance of our FL method. This analysis crucially dissects the role of each component to under-



(a) FedAvg (b) FedProx (c) SCAFFOLD

(d) MOON (e) FedNTD (f) FedDistill(Ours)

Figure 4.9: The t-SNE visualization of the features extracted by the feature-extractor of different models from MNIST with non-iid level as 0.1. The black circles denote mixed class boundaries and the blue circles denote clearer class boundaries. Compared to baseline methods, our model yield a clearer class boundary, demonstrating that our model learn a more discriminative feature extractor.

stand their contributions towards enhancing model generalization and addressing the challenge of data imbalance across clients.

Impact of $\mathcal{L}$. Both $\mathcal{L}_E$ and $\mathcal{L}_{\text{FC}}$ play an important part in our method. The observations from the results are as follows. (1) Removing $\mathcal{L}_E$ leads to a noticeable drop in top-1 accuracy (from 0.5967 to 0.5854), underscoring its essential role in enabling the local feature extractors to generalize effectively. This loss component, by facilitating the processing of local features through both biased and balanced (global) classifiers, ensures that the feature extractor is not solely tailored to the local data distribution but also retains the capability to learn and adapt based on the more generalized global model insights. (2) The removal of $\mathcal{L}_{\text{FC}}$ also results in

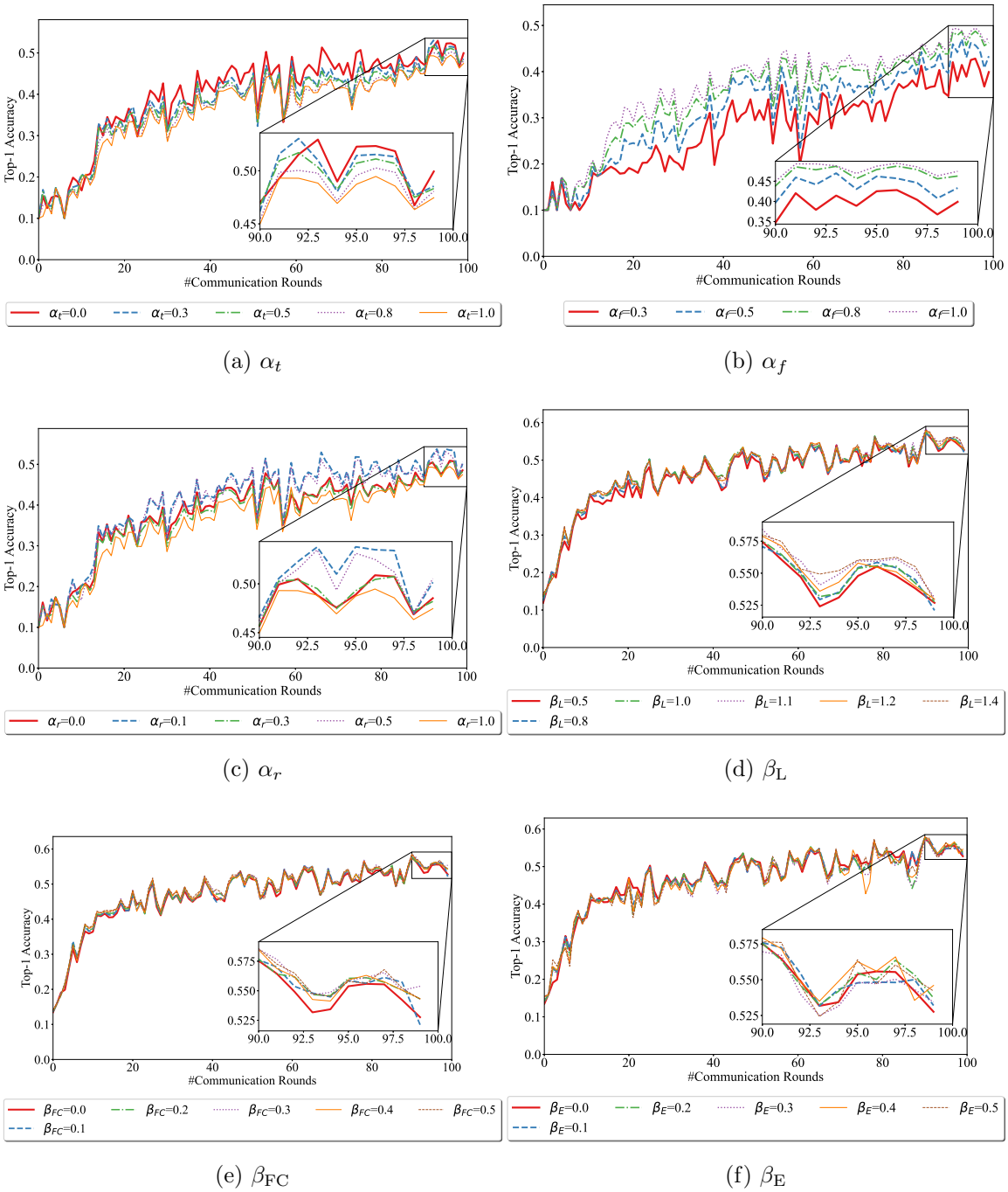
Table 4.5: Accuracy (%) of our method with/without $\mathcal{L}_E$ and $\mathcal{L}_{\text{FC}}$ on CIFAR10 with non-iid level $\alpha = 0.5$

Methods	CIFAR10 $\alpha = 0.5$
Ours	59.67
w/o $\mathcal{L}_{\text{E}}$	58.54
w/o $\mathcal{L}_{\text{FC}}$	58.08

a significant decrease in performance (to 0.5808). This observation suggests that debiasing the local classifier is also critical for enhancing model performance. (3) From Fig. 4.10d, we observe that although top-1 accuracy of last epoch is relatively robust for different β_{L} the average top-1 accuracy of last 10 epochs is increased with higher β_{L}. This observation demonstrates that $\mathcal{L}_{\text{L}}$ also contributes to the training stability. (4) The study clearly demonstrates that both $\mathcal{L}_E$ and $\mathcal{L}_{\text{FC}}$ are integral to achieving the method's high performance, with $\mathcal{L}_E$ identified as a more influential factor. This aligns with the earlier discussion on the imbalanced local data's effect on the model, particularly highlighting the classifier's vulnerability to such imbalances.

This study reveals the indispensable role of both $\mathcal{L}_E$ and $\mathcal{L}_{\text{FC}}$ in ensuring the local model's generalization capability by calibrating local models with the global model with intermediate features. It highlights the necessity of addressing both feature extraction and classification processes to combat the challenges posed by imbalanced local data in Federated Learning.

Impact of group distillation loss. The results from the demonstration of group distillation loss's impact on CIFAR10 accuracy highlight the significant role of each component within the group distillation process and its effectiveness in addressing few-sample class challenges (see Table. 4.4). The baseline performance on CIFAR10 is set at a Top-1 accuracy of 43.3%, serving as a benchmark for comparing the influ-



(a) α_t (b) α_f

(c) α_r (d) β_{L}

(e) β_{FC} (f) β_{E}

Figure 4.10: Effects of hyperparameters on the performance of FedDistill.

ence of incorporating True-Class KD (TC-KD), Few-sample Class KD (FC-KD), and Rich-sample Class KD (RC-KD) components individually and in combination. More

specifically, the baseline model is FedKD, where we directly use knowledge distillation during local training.

The observations from the results are as follows: (1) The combination of FC-KD and RC-KD contribute to the performance, with the most significant improvement (+4.19%), highlighting the nuanced interplay between focusing on few-sample and rich-sample classes. (2) The inclusion of FC-KD or RC-KD along also benefit to the result (+3.6%, +2.64%, respectively). Among FC-KD and RC-KD, FC-KD play a more important role with a higher performance increment, indicating its critical role in enhancing learning outcomes for few-sample classes. (3) The removal of FC-KD along brings a huge performance drop (-2.83%), while the removal of RC-KD doesn't cause much performance change(+0.01%). This suggests that focusing on underrepresented classes through targeted distillation significantly contributes to overall model performance. (4) If we remove both FC-KD and RC-KD, we will meet the most significant performance decrease (-22.82%) showing that the focus of true-class is often harmful. We need to focus on the rest of classes, especially those with few samples.

More specifically, Fig. 4.10a - 4.10c demonstrates the influence of α_t, α_r, and α_f on top-1 accuracy on CIFAR10 with non-iid level as 0.3. The figure shows that (1) FC-KD (α_f) plays the most important part of our Group Distillation loss: from 0.3 to 0.5 and from 0.5 to 0.8 the top-1 accuracy was significantly improved, but from 0.8 to 1.0 the improvement was not as significant as it was. (2) RC-KD also contribute to our Group Distillation loss, but a high(1.0) or low(0.0) α_r both result in a decrease on top-1 accuracy. It demonstrate that although RC-KD contribute to the performance, it should be suppress to yield a better top-1 accuracy. (3) TC-KD shows a negative impact on the top-1 accuracy: from 0.0 to 1.0 the increase of α_t correspond to the decrease of performances.

In addition, we can observe that our model is sensitive to α_f and α_r, while performing robustly with different setting of the rest four hyper-parameters. It shows that, although we are introducing six hyper-parameters in our method, only two of them

need comprehensive tuning, which makes our model more general and efficient.

4.5 Chapter Summary

In this chapter, we introduce FedDistill, a novel distillation framework for non-IID Federated Learning (FL) that addresses the challenge of imbalanced local data distributions without requiring extra communications. Our extensive experiments demonstrate that FedDistill not only surpasses a range of baseline approaches across various datasets but also significantly improves communication efficiency. This is achieved by grouping data classes according to the number of local samples and re-weighting them during the distillation process, enabling more effective and efficient knowledge transfer under non-IID conditions in FL.

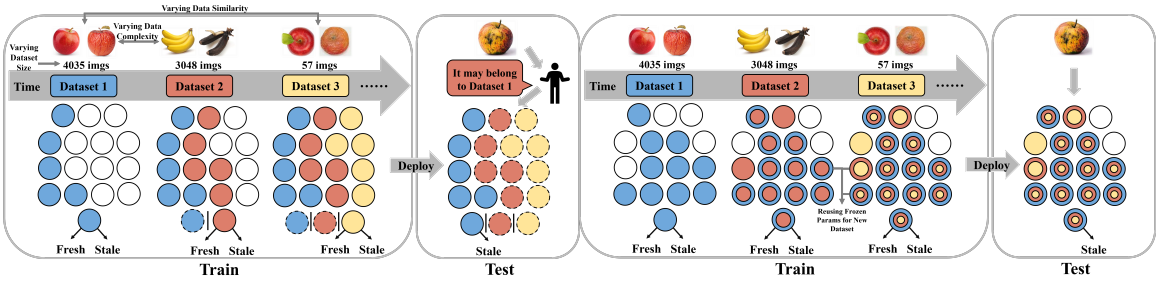
Chapter 5

Research Contribution 2: Adaptive Continual Learning for Tackling Heterogeneity in Sequential Datasets

5.1 Introduction

The last decade has witnessed a surge in data generation, facilitated by sensor-equipped devices and rapid digitization, across diverse domains such as healthcare, the Internet of Things (IoT), transportation, food safety and etc. However, datasets associated with these domains often exhibit heterogeneity, encompassing variations in size, complexity, and similarity. This heterogeneity presents unique challenges, particularly in implementing continual learning algorithms.

As machine learning models, particularly continual learning models, gain prominence in these domains, it becomes evident that they must be robust and flexible enough



(a) Traditional Parameter Isolation based Methods

(b) AdaptCL: Adaptive Learning with Task-Agnostic Parameter Isolation

Figure 5.1: (a) Traditional parameter isolation methods divide the network into non-interfering modules during inference. However, these methods are limited to task-specific continual learning (aka task incremental learning). They require manual selection of output layers and parameters, resulting in limited generalization and higher parameter usage. (b) AdaptCL achieves task-agnostic parameter isolation by fine-grained data-driven parameter partitioning, enabling high accuracy on heterogeneous datasets without module selection, while also optimizing parameter reuse and saving resources.

to accommodate the inherent heterogeneity of datasets. This heterogeneity often manifests in several ways: the size of the dataset can range from few-shot examples to large-scale samples; the complexity of data can differ based on the range and intricacy of features; and the similarity of data can vary, which can create difficulties in distinguishing task boundaries. Conventional continual learning methods for these scenarios [27, 111, 28] are typically task-agnostic and depend on either rehearsal or regularization techniques, and they have limitations when dealing with such datasets. The rehearsals often struggle with size variability due to a rigid buffer size that makes the importance regulation between old and new data challenging, while regularization techniques may hinder performance when dealing with dissimilar datasets that lack shared features. These challenges underscore the need for a more adaptive approach to handling heterogeneous datasets.

On the other hand, structure-based methods like parameter isolation show great promise in handling both similar and dissimilar domains. These methods segment the network into distinct modules that do not interfere with each other during inference 5.1. However, they are primarily suitable for task-specific continual learning, where the manual selection of the parameter module, based on the task category during inference, is feasible. In the case of task-agnostic continual learning, using all parameters for integrated inference can lead to significant interference and a drop in accuracy [1]. Therefore, the direct application of parameter isolation to task-agnostic continual learning is unsuitable without an appropriate adaptation mechanism.

Building on our previous work [113], we propose Adaptive Continual Learning (AdaptCL). AdaptCL enables adaptive learning through fine-grained data-driven pruning, effectively responding to variations in data complexity and dataset size. It also employs task-agnostic parameter isolation to ensure optimal model performance across datasets with varying similarity levels, all without the need for manual module selection. AdaptCL draws inspiration from the human brain's adaptive nervous system, a complex neural network that dynamically prunes redundant synapses [69, 35] and

reuses neural circuits for different tasks without compromising the original functions during development [5].

To the best of our knowledge, AdaptCL is the first task-agnostic parameter isolation continual learning approach designed specifically to tackle heterogeneity in sequential datasets. The key contributions of this research can be summarised as follows:

- We conduct the first comprehensive investigation into adaptive continual learning for managing heterogeneous datasets, irrespective of their complexity, size, and similarity. This innovative approach does not require retraining or different models for varying batches of data, marking a significant leap in continual learning techniques.

- Our method, AdaptCL, uniquely employs a combination of fine-grained data-driven pruning and task-agnostic parameter isolation to address the problems of catastrophic forgetting and variations in data complexity and dataset size. These adaptive mechanisms enable the model to respond effectively to different scenarios, increasing both flexibility and robustness.

- Extensive experiments conducted on several datasets, including MNIST Variants, DomainNet, and large-scale diverse and few-shot, multi-class food quality datasets, demonstrate the general applicability and resilience of AdaptCL. The method consistently outperformed existing solutions, providing higher average accuracy and versatility across different networks and applications.

5.2 Related Works

Continual learning methods are crucial tools in the field of machine learning, aiding in the effective handling of tasks that evolve over time. The existing methods primarily fall into three categories: rehearsal-based, regularization-based, and structure-

based. This section provides a detailed overview of these methods, highlighting their strengths and limitations, particularly when applied to task-agnostic continual learning and the management of heterogeneous datasets.

5.2.1 Task-Agnostic Continual Learning

Rehearsal-Based

These techniques seek to overcome catastrophic forgetting, a significant challenge in continual learning, by replaying previous training data periodically. Early methods like GEM and A-GEM [55, 12] relied on storing a portion of past training data and reusing it in future training phases. This approach has been further refined with the incorporation of generative models to create synthetic data distributions for pseudo-rehearsals [53]. LwF [51] introduces knowledge distillation that utilizes a teacher network to distil knowledge and soft targets to a student network while training on new tasks, enabling retention of knowledge from previous tasks. Some combine replay with knowledge distillation like Andrea et al. [78] keep a very small buffer for highly informative samples and combine with distillation playback and Jingyuan et al. [89] distils knowledge and replays experience from previous tasks when fitting on a new task. ICaRL [75] adopts a combination of rehearsal and regularization through learning a compact and discriminative feature representation to enable class-incremental learning. Similarly, [28] adopts a combination of rehearsal and regularization that uses the nearest class mean (NCM) classifier on food image classification dataset Food1k-100; the class mean of all data seen so far is estimated by the online mean update standard during the training phase. PRE-DFKD[9] further refines these strategies and proposes to rehearse the model using the data-free knowledge distillation through the distribution of the previously observed synthetic samples from a Variational Autoencoder (VAE).

Despite these advancements, rehearsing techniques face limitations when managing

datasets of varying sizes and maintaining the balance between old and new data. However, with AdaptCL, the model allocates parameters based on the accuracy in a data-driven way, allowing it to retain knowledge as parameter-level representations, independent of the data volume.

Regularization-Based

These methods incorporate regularization techniques, such as weight decay or dropout, to prevent catastrophic forgetting in neural networks when learning multiple tasks sequentially. Inspired by Bayesian Learning, Elastic Weight Consolidation (EWC) [41, 33] mitigates catastrophic forgetting by tracking changes using the Fisher Information Matrix. [27] adopts knowledge distillation on augmented exemplars in a class-incremental setting on food image classification. Selvarajah et al. [91] propose an indicator loss that is associated with a distillation mechanism that preserves the existing knowledge. Guanglei et al. [100] introduce an attentive feature distillation approach to mitigate forgetting. P&C [81] compress learned knowledge and distil it into the knowledge base, and preserve knowledge with EWC while using the active column to progress new data. Using a Bayesian neural network, CBLN [46] preserves distinctive parameters for different datasets for retaining performance. Similarly, [68] introduced developmental memory (DM) into a CNN, continually growing sub-memory networks to preserve important features of learned tasks while allowing faster learning. Each sub-memory can store task-specific knowledge by using a memory loss function and preserve it during continual adaptations. HAT [83] learns an attention mask over important parameters. SCML [88] proposes to learn a meta-learner for updating a unified model than updating the weights inappropriately through the optimizer. By aligning local representations, P-TNCN [66] replaces the back-propagation method that descent steepest, punishing parameter updates to a more generalized result, therefore mitigating catastrophic forgetting.

Despite the potential of regularization-based methods, they can face challenges when handling heterogeneous datasets, especially those that are dissimilar and have few shared features. While through parameter isolation in a data-driven manner, AdaptCL can effectively adapt to datasets with varying levels of similarity, including dissimilar ones.

5.2.2 Task-Specific Continual Learning

Structure-Based

Structure-based methods are primarily employed in task-specific scenarios, and these methods use parameter isolation to handle both similar and dissimilar domains effectively. They divide the network into separate modules to mitigate interference during inference. While these techniques excel in managing catastrophic forgetting, they present difficulties when directly applied to task-agnostic scenarios.

One approach, exemplified by Progressive Neural Nets (PNNs) [80], involves a static growth of the architecture with equal-sized modules, allowing for forward knowledge transfer between them. However, this method lacks a data-driven approach and requires task-specific settings for subsequent tasks, limiting its flexibility. Another approach, represented by SILF [58], addresses parameter isolation by pruning unimportant parameters, isolating the important ones to mitigate forgetting. However, SILF relies on manual pruning ratio setting instead of leveraging a data-driven approach. Reinforced Continual Learning (RCL) [96] expands each layer using reinforcement learning and enables parameter sharing. Nevertheless, this method necessitates task labels as additional inputs during inference to determine the parameters to use. To strike a balance between knowledge transfer and catastrophic forgetting, CLAW [2] identifies which parts of the network should be shared or preserved for specific tasks. PathNet [19] and RPS-Net [72] adopt a modularized network with multiple possible paths from input to output. They choose specific paths based on tasks or dataset

labels. Additionally, RPS-Net includes a distillation loss and retrospection replay to further minimize forgetting. CAT [38] masks used parameters and blocks gradient flow through unused units for dissimilar tasks. Task masks are stored according to task ID or label and need to be retained during testing. Other methods, such as DAM [79], CLNP [22], and PackNet [61], leverage pruning to strike a balance between model sparsity and performance. DAM assigns learning of each domain to a fraction of the network, typically with the same percentage (e.g., 13%). CLNP and PackNet prune parameters based on specific percentages.

Notably, the power of structure-based parameter isolation methods like PackNet has been demonstrated through recent advancements [17] that have shown superior performance compared to other continual learning methods [75, 55, 83, 4, 41, 51].

However, challenges persist, particularly when dealing with heterogeneous datasets in task-agnostic settings, which calls for more adaptive approaches. Our previous work [113] priorly applied the structure-based parameter isolation method to the task-agnostic scenario. However, its coarse-grained pruning resulted in limited adaptability to the heterogeneity of dataset size and similarity, leading to sub-optimal accuracy. Additionally, more adequate validation is needed on heterogeneous datasets.

5.3 Problem setting and objective

We are given a sequence of non-IID datasets $D_1, D_2, ...D_n$ for a fixed task. Each dataset consists of a group of labelled data $(X, Y) \in D$, where X and Y are input variables and the corresponding output variables, respectively. A task-agnostic continual learning setting aims to optimize:

$$\max_{\theta} E_{t\sim D}[E_{(X,Y)\sim D_t}[\log p_\theta(Y|X)]] \tag{5.1}$$

where θ identifies the parametrization of the network. Such a maximization problem is subject to continual learning constraints: when accessing the current dataset D at

time t, it is impractical or impossible to access any previous or future dataset. We aim to develop a task-agnostic continual learning method that can effectively handle a sequence of heterogeneous datasets.

Here, the absence of known task or dataset labels prevents task-aware inference in the model. The task-agnostic setting requires merging the output units into a single-headed classifier, with more serious task interference between data from different domains, which leads to more severe forgetting [1].

5.4 Adaptive Continual Learning

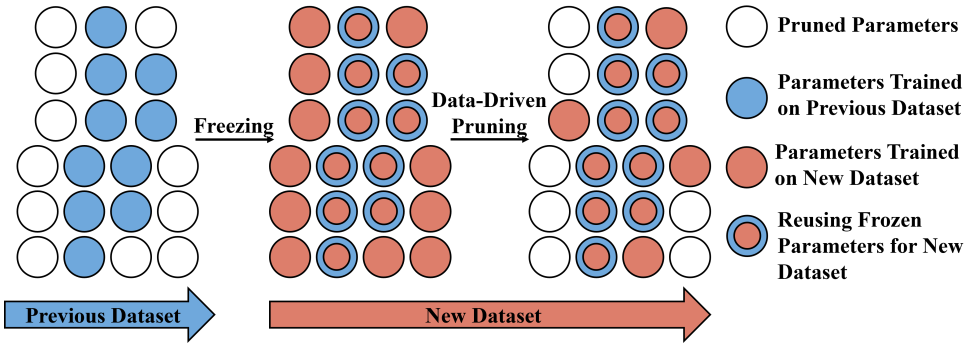


Figure 5.2: The Adaptive Continual Learning (AdaptCL) training flow. It facilitates adaptive learning via fine-grained data-driven pruning to respond effectively to variations in data complexity and dataset size. Additionally, it enables task-agnostic parameter isolation to ensure optimal model performance on datasets ranging in similarity without requiring manual selection of modules.

AdaptCL (Figure 5.2) employs adaptive learning that utilizes fine-grained data-driven pruning to adapt to variations in data complexity and dataset size. It also employs a form of task-agnostic parameter isolation to mitigate the impact of varying degrees of catastrophic forgetting caused by differences in data similarity.

5.4.1 Fine-Grained Data-Driven Pruning

In continuous learning with heterogeneous datasets, effectively managing model complexity becomes crucial within a limited computational budget. Fine-grained pruning goes beyond traditional pruning approaches by compressing the model while maintaining or even increasing accuracy. This data-driven pruning method aims to strike a balance between network accuracy and sparsity, facilitating better parameter reuse among similar datasets and improved fitting accuracy for complex or dissimilar datasets. Let's consider a neural network with a parameter set $\{W_i : 1 \leq i \leq C\}$, where W_i represents the parameter matrix at layer i and C denotes the number of layers. For fully connected layers, the corresponding parameter is $W_i \in R^{c_o \times c_i}$, where c_o is the output dimension and c_i is the input dimension. For convolutional layers, a convolution kernel $K_i \in R^{c_o \times c_i \times w \times h}$ exists, where c_o represents the number of output channels and c_i, w, and h denote the number of input channels, width, and height respectively. Pruning involves applying a binary mask M^p to each parameter W, setting unimportant parameters to 0. To determine the masks, a trainable pruning threshold vector t is introduced. The magnitude of parameters is compared to the corresponding threshold values using a unit step function $S(x)$, as shown in the following equation.

$$S\left(x\right) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{5.2}$$

$$M^p{}_{ij} = S\left(|W_{ij}| - t_i\right), \quad 1 \leq i \leq c_o, 1 \leq j \leq c_i \tag{5.3}$$

The corresponding element in pruning mask $M^p{}_{ij}$ will be set to 0 if W_{ij} needs to be pruned.

Unlike traditional methods that use a fixed threshold value, achieving fine-grained pruning requires a high-dimensional threshold, denoted as t, in order to ensure more

precise pruning. For a fully connected layer or recurrent layer with a parameter size of $W \in R^{c_o \times c_i}$, our threshold tensor size is $t \in R^{c_o}$. Each weight W_{ij} will have a neuron-wise threshold, denoted as t_i, where W_{ij} represents the jth weight associated with the ith output neuron. Similarly, for convolutional layers, the thresholds are filter-wise. Consequently, each neural network layer will be pruned based on high-dimensional thresholds, where each row of the tensor has its unique threshold. This approach ensures a more fine-grained pruning, avoiding the removal of potentially important parameters. For fully connected and recurrent layers, instead of using the dense parameter W, the sparse product $W \circ M^p$ is used in the batched matrix multiplication, where $\circ$ represents the Hadamard product operator. As for convolutional layers, each convolution kernel is flattened to obtain W, following a process similar to that of fully connected layers.

Inspired by the dynamic sparse training [54], we separate important and unimportant parameters by learning a threshold for each fully connected and convolutional neural network layer during training on one dataset. This threshold is a trainable parameter that is updated along with the backpropagation of the neural network to achieve a stepwise update. In order to make the binary step function $S(x)$ in threshold vector t trainable via back-propagation, a derivative estimation is needed. A long-tailed higher-order estimator $H(x)$ proposed by [97] is adopted for a balance of tight approximation and smooth back-propagation.

$$\frac{d}{dx} S\left(x\right) \approx H\left(x\right) = \begin{cases} 2 - 4|x| & -0.4, \leq x \leq 0.4 \\ 0.4, & 0.4 < |x| \leq 1 \\ 0, & otherwise \end{cases} \tag{5.4}$$

To get the pruning masks M^p with high sparsity, higher pruning thresholds are needed. To achieve this, a sparse regularization term L_s is added to the training loss that penalizes the low threshold value. For each trainable masked layer with threshold t, the corresponding regularization term is $R = \sum_{i=1}^{c_o} exp(-t_i)$. Thus, the sparse

regularization term L_s for a neural network with C trainable masked layers is:

$$L_s = \sum_{i=1}^{C} R_i \tag{5.5}$$

$exp(-x)$ is used as the regularization function since it is asymptotical to zero as x increases. Consequently, it penalizes low thresholds without encouraging them to become extremely large. Given the training dataset D, a sparse neural network can be trained directly with backpropagation algorithm by adding the sparse regularization term L_s to the loss function as follows:

$$W^*, t^* = argmin[L\,(D; W) + \alpha\ L_s] \tag{5.6}$$

where $L\,(\cdot)$ is the loss function, e.g., cross-entropy loss for classification, and α is the scaling coefficient for the sparse regularization term, which can control the percentage of parameters remaining. It is calculated according to the total number of iterations of one dataset. For a new dataset, the pruning threshold t is re-initialized, and a new round of fine-grained data-driven pruning is restarted and be applied to neural network parameters not occupied by previous datasets.

5.4.2 Task-Agnostic Parameter Isolation

To address the issue of catastrophic forgetting with varying similarity datasets, we enhance the technique of parameter isolation. Traditional methods freeze learned parameters during both training and inference, preventing them from being updated and masking task-specific parameters. In contrast, our data-driven approach progressively learns from frozen parameters while utilizing all parameters during inference. This allows effective handling of heterogeneity in data similarity during continuous learning. Parameter freezing in neural networks involves preventing specific parameters from being updated during training. In our approach, we introduce a binary freeze mask, denoted as M, of the same shape as the parameters. This mask has a

value of 1 for parameters that are allowed to be updated and 0 for frozen parameters. We obtain the frozen parameters, θ_f, by element-wise multiplying the original parameters by this mask:

$$\theta_f = \theta \odot M$$

During training, the gradients computed with respect to the loss function are applied only to the non-frozen parameters, updating them according to the optimization algorithm. The frozen parameters remain unchanged throughout the training process. At the end of training on each dataset, we calculate a freeze mask M^f, which is the result of the union between the existing pruning mask and the freeze mask from the previous round. This mask is used to freeze the learned parameters during the next dataset training. The freeze mask M^f is calculated as follows:

$$M^f{}_{ij} = S\left(\left|M^f{}_{ij} + M^p{}_{ij}\right|\right), \quad 1 \leq i \leq c_o, 1 \leq j \leq c_i \tag{5.7}$$

where S is the sign function, c_o is the number of output channels, c_i is the number of input channels, and $M^p{}_{ij}$ is the pruning mask obtained after pruning.

In order to ensure that the corresponding gradient of the parameters in the freeze mask $M^f{}_{ij}$ is set to 0 when W_{ij} needs to be frozen, we use the following equation:

$$W^*, t^* = argmin\left[L\left(D; W, t\right) + \alpha L_s\right] \circ \left(1 - M^f\right) \tag{5.8}$$

Here, $L(D; W, t)$ denotes the loss function on the current dataset, L_s is the penalty for changes in learned parameters, α is the learning rate, W^* and t^* denotes the optimal value of the weight and threshold respectively, and $\circ$ denotes element-wise multiplication between the matrices. During inference, all the parameters, including the frozen ones, are used to make predictions, as the model has already learned useful representations from them. By applying parameter freezing, a neural network can retain knowledge from previous tasks while allowing for further learning without catastrophic forgetting. For a new dataset, adaptive continual learning initiates a

fresh iteration while preserving important frozen parameters. Pruning is only applied to other free neural network parameters.

5.4.3 Adaptive Continual Learning Training Flow

Referring to the algorithm flow of our proposed method, depicted in Algorithm 2, at the start of each new round of dataset training, threshold parameters are initialized. During training, these threshold parameters are calculated and updated at each step of backpropagation, leading to the refinement of the pruning mask.

$$M^p{}_{ij} = S\left(|W_{ij}| - t_i\right)$$

The refinement process, being fine-grained, data-driven, and step-wise, allows AdaptCL to adapt to variations in data complexity and dataset size. Throughout the training process, AdaptCL freezes the gradient descent at each step based on the freeze mask $(1 - M^f)$. This protects the current parameters from further modification, preserving the knowledge learned in previous training rounds and mitigating the impact of catastrophic forgetting caused by variations in data similarity. At the end of each round of training, AdaptCL generates an updated freeze mask to protect the current set of parameters for future training. This allows AdaptCL to continue learning from new data while retaining the knowledge gained from previous training rounds. Overall, the adaptive learning with fine-grained data-driven pruning approach, coupled with task-agnostic parameter isolation, enables AdaptCL to effectively adapt to variations in data complexity and dataset size while mitigating the impact of variation of data similarity during the training process.

5.5 Experiments

Our method is evaluated on a range of benchmark datasets with heterogeneous characteristics, encompassing various domains and tasks. To assess the performance of

our method, we apply the method to the widely recognized ResNet-18, LeNet-5 and VGG-16 architectures. To establish a solid benchmark for comparison, we implement several other baseline algorithms in the domain incremental setting. These algorithms include SGD as the naive setting, as well as EWC, LwF, PRE-DFKD, PackNet*, and Separated Models for Learning (SML). Particularly, PackNet* represents an extension of PackNet specifically designed for our task-agnostic evaluation.

By conducting experiments on these benchmark datasets and comparing our method against these baseline algorithms, we aim to gain insights into the performance of our proposed approach and to assess its effectiveness in addressing the challenges of tackling data heterogeneity in continual learning. In particular, we aim to answer the following research questions:

- **Q1**: How does AdaptCL compare to other baseline continual learning methods in terms of average accuracy and parameter efficiency?

- **Q2**: What is the effectiveness of AdaptCL in managing heterogeneity in sequential datasets from different application domains, such as Food Quality and DomainNet?

- **Q3**: What is the impact of AdaptCL's fine-grained data-driven pruning technique on adapting to differences in data complexity and dataset size?

- **Q4**: How does AdaptCL's task-agnostic parameter isolation approach mitigate catastrophic forgetting in the presence of varying degrees of data similarity?

5.5.1 Datasets

To evaluate our method, we choose the following four datasets:

Large-Scale, Diverse Binary-Class Food Quality Dataset

The dataset comprises a total of 14,683 images of six different types of fruits and vegetables, as shown in Figure 5.3(a), including apples, bananas, bitter gourds, capsicums, oranges, and tomatoes. Each image in the dataset is classified as either fresh or stale. The datasets vary in size, and the images are obtained from various sources such as online image repositories, self-captured images, or artificially-generated images through data augmentation techniques, resulting in different levels of complexity and similarity among the datasets. The datasets are designed to be heterogeneous and challenging to evaluate the robustness and generalization of machine learning models. All the images in the dataset have been preprocessed to ensure a uniform size and aspect ratio of 64×64 pixels. The total size of the dataset is approximately 2GB.

Few-Shot, Multi-Class Food Quality Dataset

The used as a real-life application case to verify our solution on a small dataset size, which poses a more challenging scenario compared to the previous binary classification dataset. This dataset comprises images of Apples and Bread, each associated with a freshness score label. The freshness scores range from 0 to 4, where 0 represents total corruption and 4 indicates total freshness. The Apple dataset consists of a total of 57 images, while the Bread dataset contains 93 images, as illustrated in Figure 5.3(b). This dataset aims to evaluate the model's performance in adapting to very few samples, and the ability to transfer knowledge to solve under-fitting.

DomainNet with heterogeneous complexity and size

The DomainNet[70] dataset consists of image data from six domains, each with a different amount of data, including real photos, painting, clipart, infograph, quickdraw,

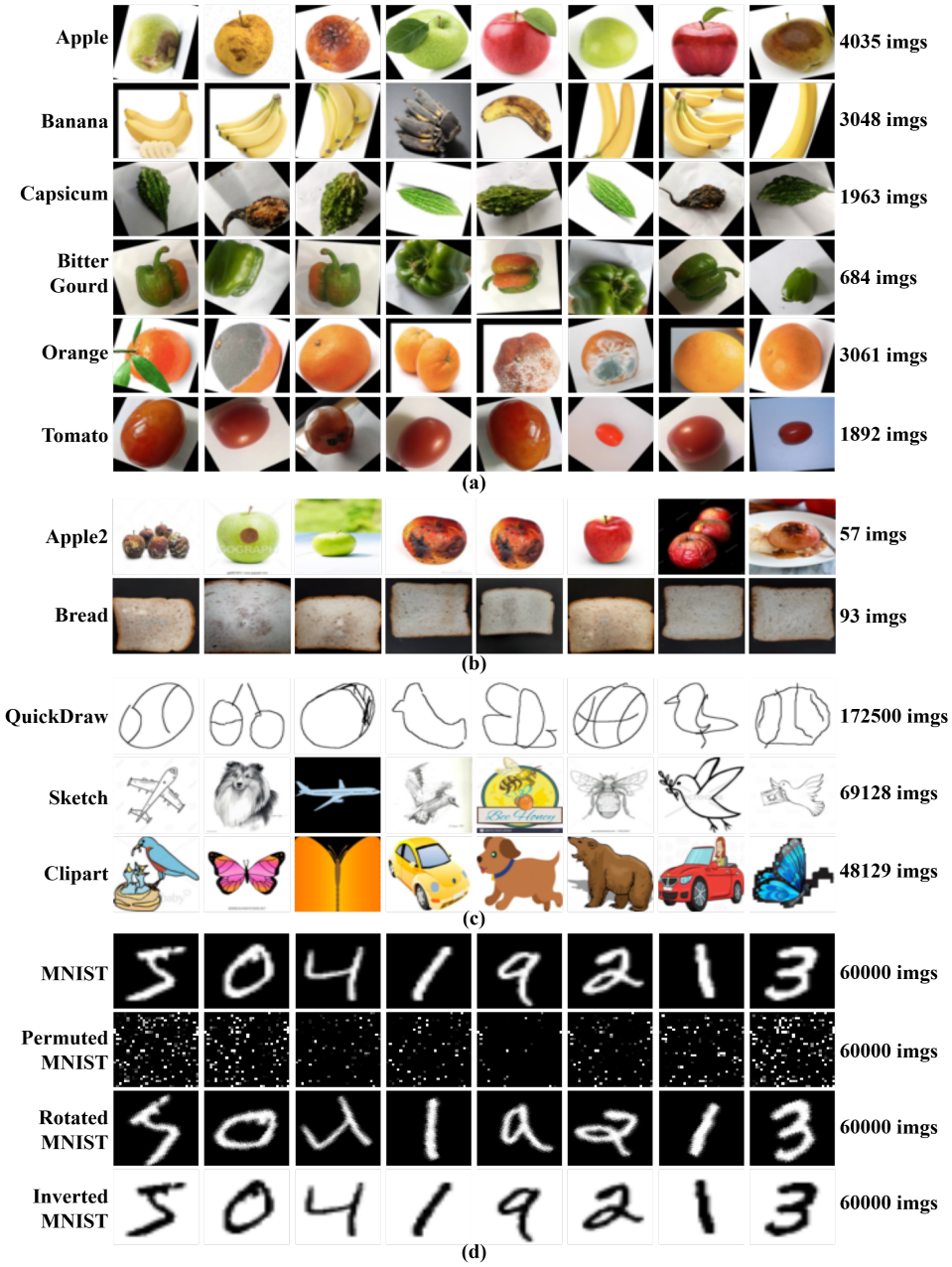


Figure 5.3: Examples of input images and size of datasets used in the experiments. (a) Large-Scale, Diverse Binary-Class Food Quality Dataset. (b) Few-Shot, Multi-Class Food Quality Dataset (c) DomainNet comprises datasets with heterogeneous complexity and size. (d) MNIST Variants with heterogeneous similarity.

and sketch. There are 48K - 172K images (600K in total) categorized into 345 classes per domain. DomainNet comprises datasets with heterogeneous complexity and size. For instance, the Quickdraw dataset holds 172,500 images but requires only 439MB of storage, while the Sketch dataset includes 69,128 images but occupies 2.5GB of storage. The Sketch, Quickdraw, and Clipart domains are selected as datasets as shown in Figure 5.3(c) to evaluate models' performance on datasets with different complexity.

MNIST Variants with heterogeneous similarity

To provide additional validation for our model, we choose to utilize MNIST Variants, which include the MNIST, Permuted MNIST, Inverted MNIST, and Rotated MNIST datasets. These datasets are organized into two sequences, each reflecting a different level of similarity. The first sequence, referred to as the dissimilar sequence, consisted of MNIST, Permuted MNIST, and Inverted MNIST. On the other hand, the second sequence, referred to as the more similar sequence, comprised MNIST, Permuted MNIST, and Rotated MNIST.

The datasets each consist of 70,000 images of handwritten digits from 0 to 9 of size 32×32. In each dataset, 60,000 images are used for training and 10,000 images for the test, as listed in Figure 5.3(d).

- **Permuted MNIST** is an MNIST variant that applies a fixed random permutation of the pixels of the MNIST digits. It also includes the same number of images of handwritten digits. Permuted MNIST bears no resemblance to MNIST at all.

- **Inverted MNIST** is another variant of the MNIST dataset inverting the color of MNIST images from black to white. The Inverted MNIST and MNIST are the exact opposite in the color of the input data and the same in the output.

- **Rotated MNIST** is also a variant of the MNIST dataset. It rotates MNIST data randomly by 0-45 degrees. There is some overlap of data between the MNIST and the Rotated MNIST, making the two datasets similar to each other.

5.5.2 Networks Used

To evaluate the applicability of our proposed technique on networks of different sizes and structures, we conducted our experiments using three popular network architectures: LeNet-5, ResNet-18, and VGG-16.

LeNet-5 is a relatively simple architecture with 61,706 parameters and a compact size of 0.24 MB. It was primarily designed for digit recognition in checks and consists of 7 convolutional layers. However, due to its limited number of convolutional layers, LeNet-5 may face resource constraints when processing sequential datasets.

ResNet-18, on the other hand, is a more complex architecture with 11,172,810 parameters and a larger size of 42.62 MB. This network incorporates a greater number of convolutional layers, making it better equipped to handle complex image recognition tasks. The increased number of parameters allows for a larger network capacity, which is advantageous for continuous learning scenarios involving sequential datasets.

Lastly, we utilized the VGG-16 architecture, which is more parameter-rich, with 14,986,570 parameters and a size of 57.17 MB. This architecture offers a high degree of expressiveness due to its numerous convolutional and fully connected layers.

5.5.3 Evaluation Metrics

For a principled evaluation, we adopt the following evaluation metrics[55]:

- Average Accuracy: $ACC = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}$
- Backward Transfer: $BWT = \frac{1}{T-1}\sum_{i=1}^{T-1} (R_{T,i} - R_{i,i})$

- Forward Transfer: $FWT = \frac{1}{T-1}\sum_{i=2}^{T-1} R_{i-1,i} - \bar{b_i}$

We consider access to a testing dataset for each of the D datasets. After the model finishes learning about the domain t_i, we evaluate its test performance on all T datasets. By doing so, we construct the matrix $R \in R^{t\times t}$, where $R_{i,j}$ is the test classification accuracy of the model on the dataset t_j after observing the last sample from dataset t_i. Letting $\bar{b}$ be the vector of test accuracy for each task at random initialization. For comparison, our primary criterion for evaluating performance is the average accuracy (AAC) metric, where higher values indicate better performance. Additionally, we consider the metrics of backward and forward transfer efficiency (BWT and FWT), with higher values being preferred. Furthermore, we calculate parameters (Params) to assess parameter efficiency. To gain a deeper understanding of model performance across datasets, we also compare test accuracy for each dataset.

5.5.4 Baselines

To validate the effectiveness of our method in continual learning with heterogeneous datasets, we compare our model with baseline algorithms. We implement all of the following described baselines in our code base:

- Separated model learning (SML): Separate models are trained for every task, achieving the highest possible accuracy by dedicating all the network resources to that single dataset. In this case, there is no knowledge transfer or catastrophic forgetting. It requires manual selection of the model during inference.
- SGD[10]: A naïve model trained with direct stochastic gradient descent.
- EWC[41]: A regularization technique in continual learning that uses diagonal elements of Fisher Information Matrix to constrain the weights of the neural network and avoid catastrophic forgetting.

- LwF[51]: A rehearsal-based method that uses knowledge distillation to preserve previously learned knowledge along with training on new tasks.

- PRE-DFKD[9]: A recently proposed rehearsal strategy that rehearses the model using the data-free knowledge distillation through the distribution of the previously observed synthetic samples from a Variational Autoencoder (VAE).

- PackNet[61]: A structure-based parameter isolation method that prunes a specific ratio of the network during training to sequentially "pack" multiple tasks into a single network. It requires knowing the number of datasets ahead to calculate the pruning ratio. Also, it needs to select masks to indicate network modules to perform during inference. We implement it in the task-agnostic setting referred to as PackNet* later in this thesis.

5.5.5 Implementation Details

We use Pytorch and Torchvision libraries to implement neural networks. All of the training images are scaled and normalized before training as preprocessing. Identical processes are applied to the test images. The optimizer is stochastic gradient descent (SGD), with a 0.001 learning rate, 0.9 as the momentum value, and Nesterov Accelerated Gradient for regularization. To guarantee completely reproducible results, we set seed value as 5 for the random function of Numpy, python Random, Pytorch, Pytorch Cuda, and set Pytorch backends Cudnn benchmark as False, with Deterministic as True, configuring PyTorch to avoid using nondeterministic algorithms for some operations, so that multiple calls to those operations, given the same inputs, will produce the same result. Algorithm 2 shows the learning procedure of AdaptCL. We keep all the settings the same for our method and the baselines.

Considering the Fisher matrix of EWC, we use EWC λ as 1. Regarding PackNet, we implement it in a domain-incremental setting, which we refer to as PackNet* in

the thesis. Instead of using a pre-trained model, we train it for the same number of epochs as other methods, selecting ten epochs of sparse training following pruning, as discussed in PackNet's paper. To ensure each dataset received equal attention, we prune the network to assign the same ratio of $1/T$ parameters per dataset, where T is the number of datasets. For PRE-DFKD, we follow the default setting and use Kullback–Leibler Divergence (KLD) loss with a hyperparameter of 10^{-5}. Regarding LwF, we set the hyperparameters Alpha and Temperature to 1 and 3, respectively. For the naïve settings with stochastic gradient descent (SGD), we simply fine-tune the network on each new dataset without making any network modifications. For Separated model learning, we use one network for training on every single dataset and do not fine-tune it on other datasets.

We will make the implementation details and **code publicly available** upon publication, ensuring transparency, reproducibility, and facilitating further research in the field.

5.6 Results

5.6.1 Performance on Datasets with Varied Size (Q1,Q2,Q3)

Large-Scale, Diverse Binary-Class Food Quality Dataset

Our method achieves an average accuracy of 78.2% on the six food datasets, surpassing baseline methods by 4.32%. It outperforms other approaches in terms of final accuracy and has the lowest parameter count (1.091×10^7) while effectively overcoming catastrophic forgetting. Despite having fewer parameters, our model successfully fits up to six datasets, with some minor accuracy gaps compared to SML due to data complexity and the need for increased capacity. To unlock its full potential, we recommend scaling up the model for improved accuracy in continual learning.

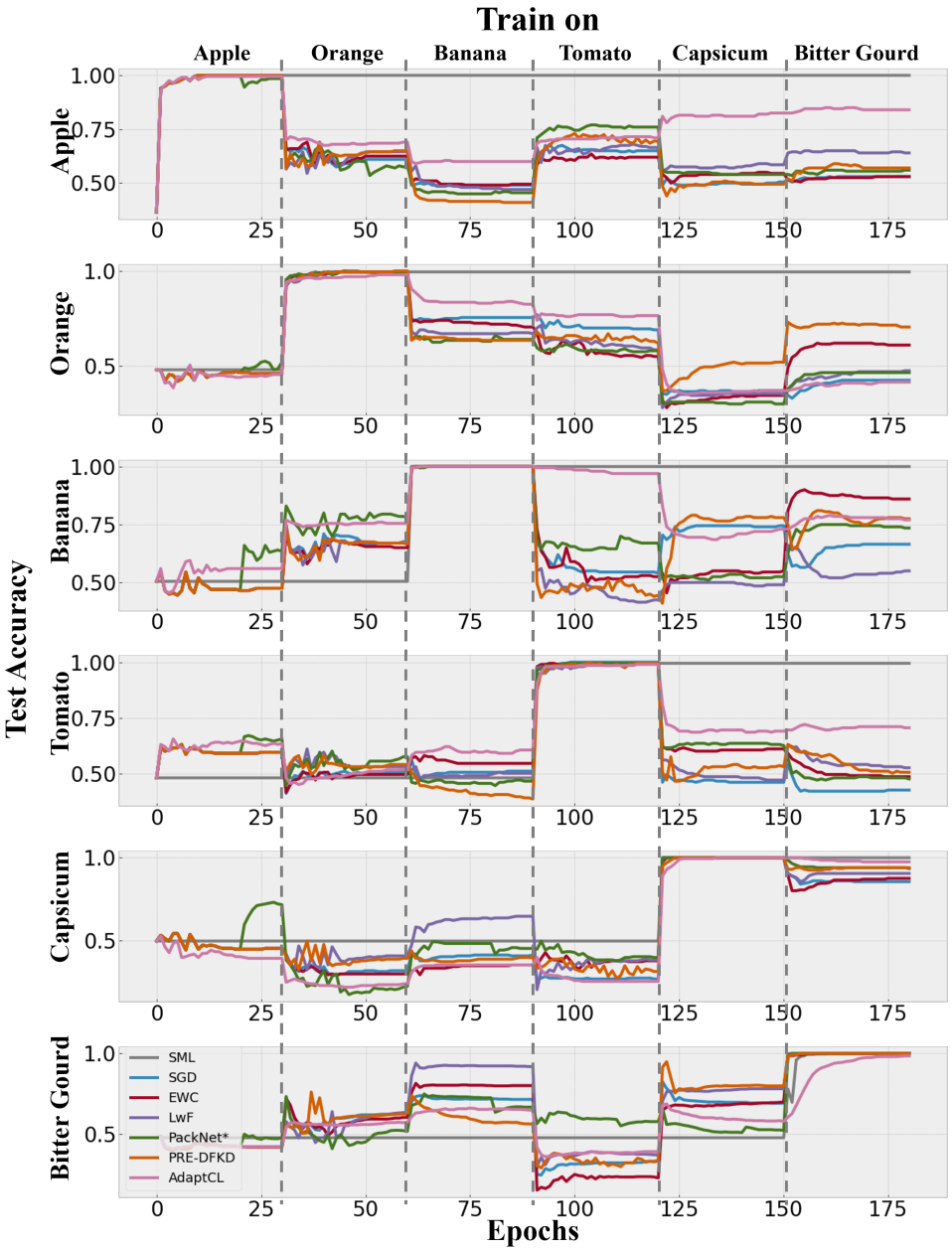


Figure 5.4: Test accuracy comparison of continual learning methods on the Large-Scale, Diverse Binary-Class Food Quality Dataset. Our proposed method, AdaptCL, achieves higher average accuracy while consistently preventing catastrophic forgetting in real-world applications with heterogeneous data, outperforming other methods. (Best viewed in color)

Table 5.1: Performance evaluation of continual learning methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and number of used parameters on the Large-Scale, Diverse Binary-Class Food Quality Dataset. Our proposed method, AdaptCL, achieves the best AAC and BWT with fewer parameters compared to other methods.

	AAC↑	BWT ↑	FWT ↑	Params ($\times 10^7$) ↓	Test Accuracy↑					
					Apple	Orange	Banana	Tomato	Gourd	Capsicum
SML	0.998	-	-	6.704	1.000	0.995	1.000	0.995	1.000	1.000
SGD	0.650	-0.419	0.033	1.117	0.530	0.425	0.665	0.425	0.855	1.000
LwF	0.683	-0.379	0.072	1.117	0.640	0.475	0.550	0.525	0.905	1.000
EWC	0.727	-0.326	0.058	1.117	0.530	0.610	0.860	0.485	0.875	1.000
PackNet*	0.695	-0.361	0.049	1.117	0.56	0.465	0.735	0.475	0.935	1.000
PRE-DFKD	0.749	-0.300	0.085	1.117	0.570	0.705	0.775	0.505	0.940	1.000
AdaptCL	**0.782**	**-0.252**	0.041	**1.091**	**0.840**	0.415	**0.770**	**0.705**	**0.975**	0.984

Analyzing Figure 5.4, we observe the varying impact of learning across datasets due to their heterogeneity. In most cases, our model maintains the highest accuracy on the learned datasets. Comparing it to PackNet*, which also uses pruning methods, we notice a notable accuracy increase on unlearned datasets during pruned epochs, indicating the efficacy of pruning for enhancing generalization in continual learning with relevant datasets. Our model struggles with the Orange dataset following training on Capsicum and Bitter Gourd datasets due to conflicting features, mainly caused by the low data resolution of 64×64 pixels, which led the model to primarily rely on color and shape to differentiate images. This issue, observed in all baseline models, can be resolved by increasing data resolution.

Few-Shot, Multi-Class Food Quality Dataset

We also test our method on a Few-Shot, Multi-Class Food Quality Dataset to evaluate its ability to generalize on small datasets. As the human brain is a few-shot learner,

Table 5.2: Comparison of average accuracy (AAC↑), backward knowledge transfer (BWT↑), forward knowledge transfer (FWT↑), Test Accuracy↑, and the number of used parameters (Param$\times 10^7\downarrow$) of various continual learning methods on the Few-Shot, Multi-Class Food Quality Dataset with different training orders. AdaptCL obtains better accuracy than using separated models for learning (SML) on small-size datasets.

	Bread → Apple2						**Apple2 → Bread**					
	AAC	BWT	FWT	Param	Test Accuracy		AAC	BWT	FWT	Param	Test Accuracy	
					Bread	Apple					Apple	Bread
SML	0.989	-	-	2.235	0.978	1.000	0.989	-	-	2.235	1.000	0.978
SGD	0.774	-0.430	0.368	1.117	0.548	1.000	0.849	-0.281	0.452	1.117	0.719	0.978
LwF	0.782	-0.398	0.368	1.117	0.581	0.982	0.849	-0.281	0.452	1.117	0.719	0.978
EWC	0.763	-0.452	0.368	1.117	0.527	1.000	0.831	-0.316	0.452	1.117	0.684	0.978
PackNet*	0.894	-0.172	0.281	1.117	0.806	0.982	0.893	-0.193	0.398	1.117	0.807	0.978
PRE-DFKD	0.859	-0.086	0.404	1.117	0.892	0.825	0.867	-0.158	0.409	1.117	0.842	0.892
AdaptCL	**0.995**	**0.011**	0.316	**1.014**	**0.989**	**1.000**	**0.980**	**-0.018**	0.441	**1.005**	**0.982**	**0.978**

able to generalize from a few examples, we find that our AdaptCL method, designed based on the neural reuse principle, can improve learning efficiency and performance on small sample datasets similar to how humans learn. As shown in Table 5.2, when evaluating the Few-Shot, Multi-Class Food Quality Dataset, AdaptCL achieves 99.5% accuracy using 10% fewer parameters than baseline methods, even producing a rare positive backward knowledge transfer of **1.08%**, meaning the positive consequence of inductive knowledge transfer is more significant than catastrophic forgetting. Since the first dataset is small, the model is not fully trained, and easy to overfit; the new dataset can make the network more robust to have higher accuracy during inference on the test dataset. Our model outperforms the baselines' AAC by 11.20% and is superior to using a separated model for learning (SML) on the Few-Shot, Multi-Class Food Quality Dataset sequence with only 45% of SML's parameters. These results demonstrate the potential advantages of our model when encountering a continuous stream of smaller datasets.

Effect of Training Orders To investigate the impact of training orders, we conduct

experiments using different dataset sequences. Our method, AdaptCL, consistently achieves the best results in both forward and reverse training orders, as shown in Table 5.2. Unlike baseline methods, the average accuracy (AAC) of AdaptCL remains unaffected by the training sequence, while the final accuracy of methods like SGD, LwF, and EWC is heavily influenced by the order of training. This can be attributed to AdaptCL's fine-grained pruning and task-agnostic parameter isolation, which minimize catastrophic forgetting and promote model generalization, enabling adaptation to new datasets regardless of their presentation order. These findings demonstrate the significant impact of training order on the performance of traditional methods, likely due to the tendency to overfit early datasets during training. The robustness of AdaptCL to training order positions it as a preferred method for domains requiring frequent learning and adaptation to new datasets, as it effectively avoids the limitations associated with traditional methods.

5.6.2 Performance on Datasets with Varied Complexity (Q1,Q2,Q3)

Table 5.3: Results of different continual learning methods on the DomainNet dataset, including their AAC, BWT, FWT, and the number of used parameters. Our proposed method, AdaptCL, demonstrates the best AAC and BWT, indicating its ability to handle datasets with heterogeneous dataset size and complexity.

	AAC ↑	BWT ↑	FWT ↑	Params($\times 10^7$) ↓	Test Accuracy↑		
					Quickdraw	Sketch	Clipart
SML	0.624	-	-	3.404	0.774	0.573	0.524
SGD	0.449	-0.220	0.209	1.135	0.394	0.414	0.539
LwF	0.448	-0.237	0.103	1.135	0.381	0.411	0.552
EWC	0.457	-0.203	0.103	1.135	0.421	0.414	0.536
PackNet*	0.484	-0.180	0.098	1.135	0.483	0.448	0.521
PRE-DFKD	0.461	-0.206	0.109	1.135	0.464	0.416	0.504
AdaptCL	**0.531**	**-0.131**	0.106	**1.051**	**0.570**	**0.510**	0.512

We evaluate the performance of AdaptCL on the DomainNet sequence, which is a heterogeneous classification dataset made up of images from different domains, each of varying size and complexity [70]. On the DomainNet dataset, rehearsal-based methods like LwF and PRE-DFKD perform poorly, especially LwF, even lower than SGD without any Continual Learning method assistance. This is likely due to the large and disparate sizes of the three subsets in DomainNet, making it challenging to adjust simple knowledge distillation methods based on dataset size. Additionally, comparing SML with other methods on the last subset, Clipart, we observe that learning models on sequential datasets can facilitate faster learning and forward knowledge transfer, resulting in higher test accuracy compared to separated model learning (SML). The AdaptCL doesn't achieve higher accuracy than SML on this subset due to pruning, which makes the model more parameter efficient, but simultaneously slows down the learning of new data because of insufficient model capacity. This issue can be solved by network expansion. As shown in Table 5.4, AdaptCL outperforms the baselines, improving network performance by 18.24% in average accuracy and 44.79% in backward transfer compared to SGD, while beating the baselines CL methods by 9.7% in AAC and 30.69% in BWT, by using only 92.65% of their parameters. Despite the impressive results, gaps in accuracy persisted compared to using separate models for learning, primarily due to the complex nature of the DomainNet data that demand increased model capacity to handle more complex information with significant distribution shifts within the dataset. From Figure 5.5, we can see that even with significant differences in data, Clipart, Sketch, and Quickdraw can rely on forward knowledge transfer to achieve faster learning. In the context of continual learning, old datasets can improve the accuracy of new datasets, making CL methods more accurate than using separate models for learning (SML) to learn new data. Among the methods evaluated, rehearsal is the most effective in promoting faster learning, while AdaptCL excelled in accuracy retention.

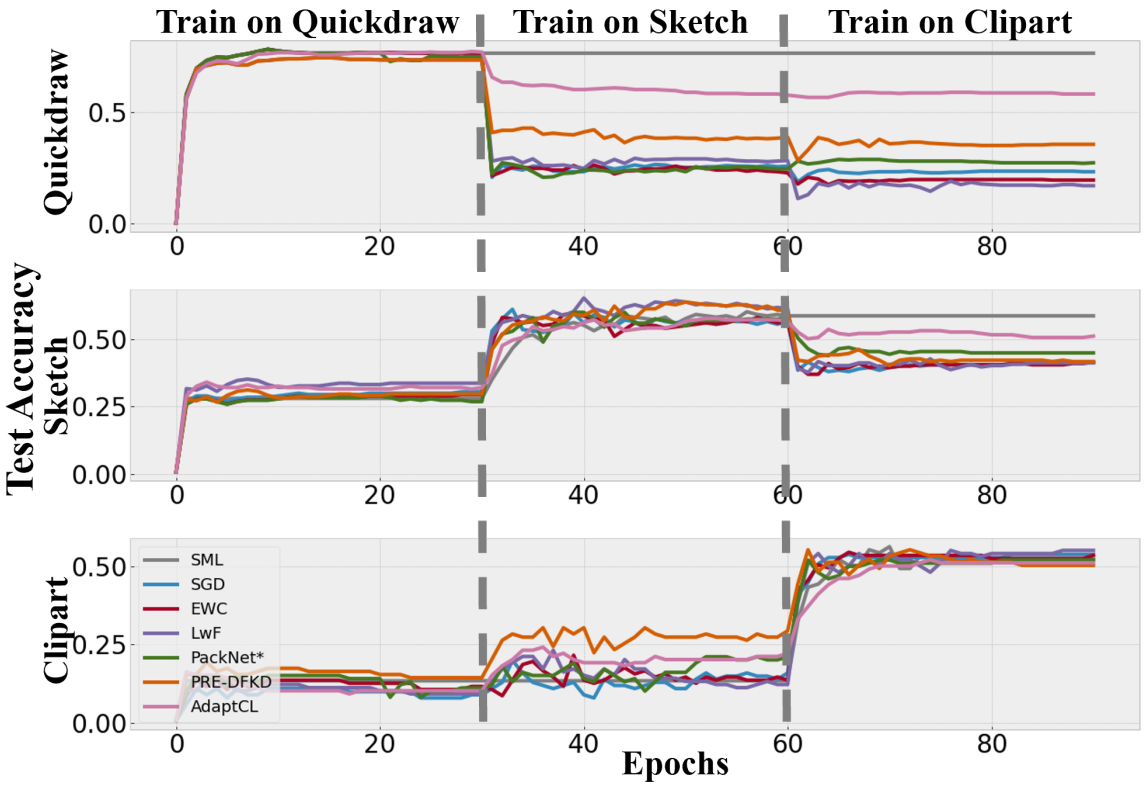


Figure 5.5: Results of continual learning methods on the DomainNet that comprises datasets with heterogeneous complexity and size. AdaptCL achieves the best average accuracy and is the most robust to datasets with varied complexity and size. (Best viewed in color)

5.6.3 Performance on Datasets with Varied Similarity (Q1,Q4)

Dissimilar MNIST Variants

Table 5.4: Comparison of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and the number of used parameters of different continual learning methods on the dissimilar MNIST Variants dataset. Our proposed method, AdaptCL, achieves significantly higher AAC, BWT and FWT with fewer parameters compared to other methods. AdaptCL achieved comparable AAC results while using 31.2% of SML's parameters.

	AAC↑	BWT↑	FWT↑	Params ($\times 10^7$)↓	Test Accuracy↑		
					MNIST	Permuted MNIST	Inverted MNIST
SML	0.989	-	-	3.352	0.993	0.980	0.993
SGD	0.755	-0.351	0.006	1.117	0.483	0.787	0.994
LwF	0.643	-0.521	-0.002	1.117	0.331	0.604	0.994
EWC	0.753	-0.354	0.009	1.117	0.363	0.776	0.993
PackNet*	0.841	-0.222	0.009	1.117	0.591	0.939	0.993
PRE-DFKD	0.784	-0.274	0.011	1.117	0.723	0.686	0.943
AdaptCL	**0.967**	**-0.019**	**0.023**	**1.046**	**0.980**	**0.936**	**0.986**

We screen two sets of MNIST Variants to compose similar and dissimilar sequences, with the dissimilar sequence as MNIST, Permuted MNIST, and Inverted MNIST. Regarding the dissimilar MNIST Variant sequence, AdaptCL significantly improves the network's average accuracy (AAC) by 28.14% and alleviates forgetting (BWT) by 94.50% (Table 5.3). It outperforms baselines by 15.03% in AAC and 91.30% in BWT on this sequence. Compared to separated model learning (SML) where separate models are trained for each task, AdaptCL achieves comparable AAC while utilizing only 31.2% of SML's parameters. Our method's ability to minimize forgetting while learning dissimilar datasets, its parameter efficiency, and generalization contribute to its effectiveness.

Rehearsal and regularization-based methods like EWC, LwF, and PRE-DFKD per-

form poorly on the Dissimilar MNIST Variants dataset due to the vast data amount and dissimilarity between datasets. Parameter isolation-based methods like PackNet* and our method, AdaptCL, demonstrate significant advantages on this dataset. Training with plain SGD leads to catastrophic forgetting and a performance decline of at least 50% (Figure 5.6). EWC slows down the performance decline initially, but it deteriorates over time. While PackNet* shows some improvement through pruning during learning on Dataset A, it is not as effective as AdaptCL in inhibiting catastrophic forgetting. AdaptCL, with a fixed neural network, adapts to new datasets while maintaining high performance on previous datasets without significant forgetting.

To understand why EWC and LwF performed worse than SGD, we test them with different hyper-parameters and epochs and find that they perform better with fewer training epochs. This is because MNIST has a large dataset size, requiring numerous iterations of training. When training on the new dataset, using fewer epochs will lead to less over-fitting on the new dataset. For example, with 10 epochs of training per dataset, SGD has an AAC of 78.88%, -30.02% BWT, and 1.11% FWT, while EWC has an AAC of 79.38%, -29.30% BWT, and 5.71% FWT. However, as the number of epochs increased, these methods struggled to suppress over-fitting in the new dataset.

More similar MNIST Variants

In a similar MNIST Variant sequence consisting of MNIST, Permuted MNIST, and Rotated MNIST, our method maintains the highest accuracy, outperforming baselines by 21.8% while using fewer parameters (Table 5.5). AdaptCL achieves significant forward knowledge transfer on similar datasets with minimal catastrophic forgetting. Although it may not achieve the best accuracy on particularly similar datasets like MNIST and Rotated MNIST, our model consistently performs well on datasets with heterogeneous similarity, avoiding overfitting to similar datasets. As dataset similar-

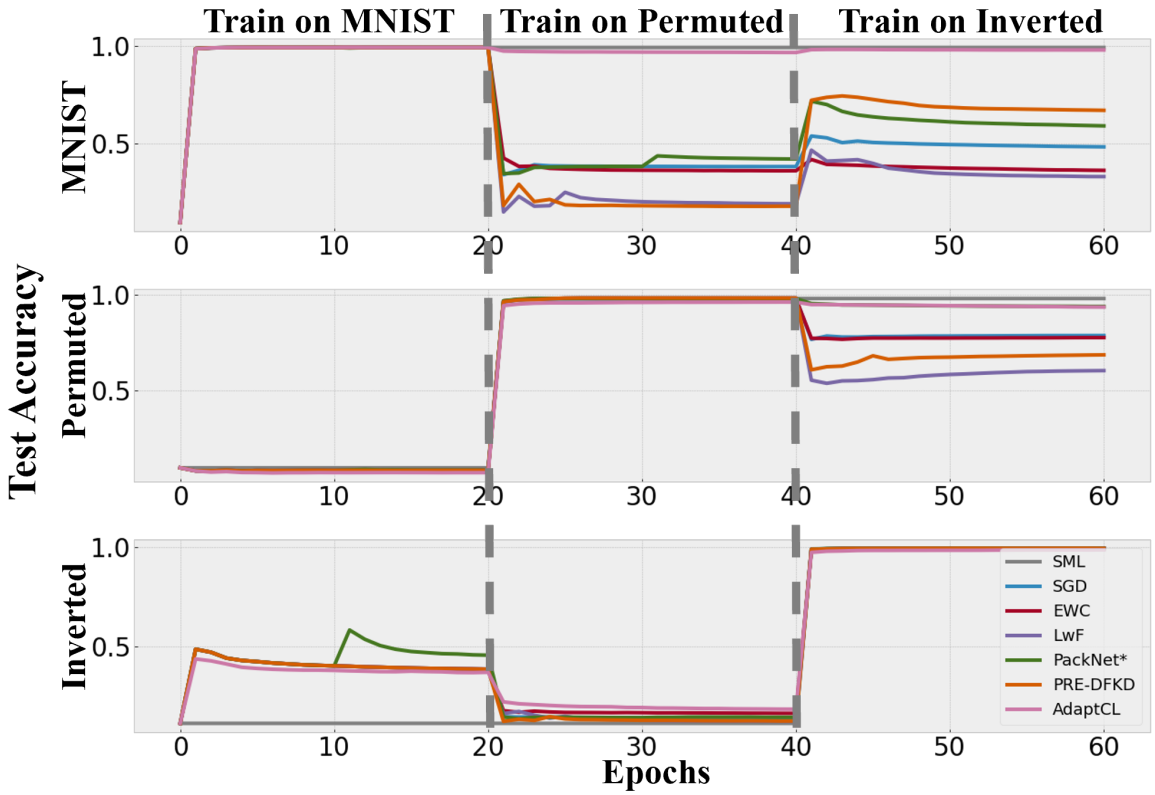


Figure 5.6: Test accuracy comparison of continual learning methods on three dissimilar MNIST Variant datasets. Compared with using separated models for learning (SML), AdaptCL achieved comparable AAC results while using only 31.2% of SML's parameters. AdaptCL's ability to achieve minimal forgetting while learning dissimilar datasets, coupled with its parameter efficiency, establishes the effectiveness of our approach. (Best viewed in color)

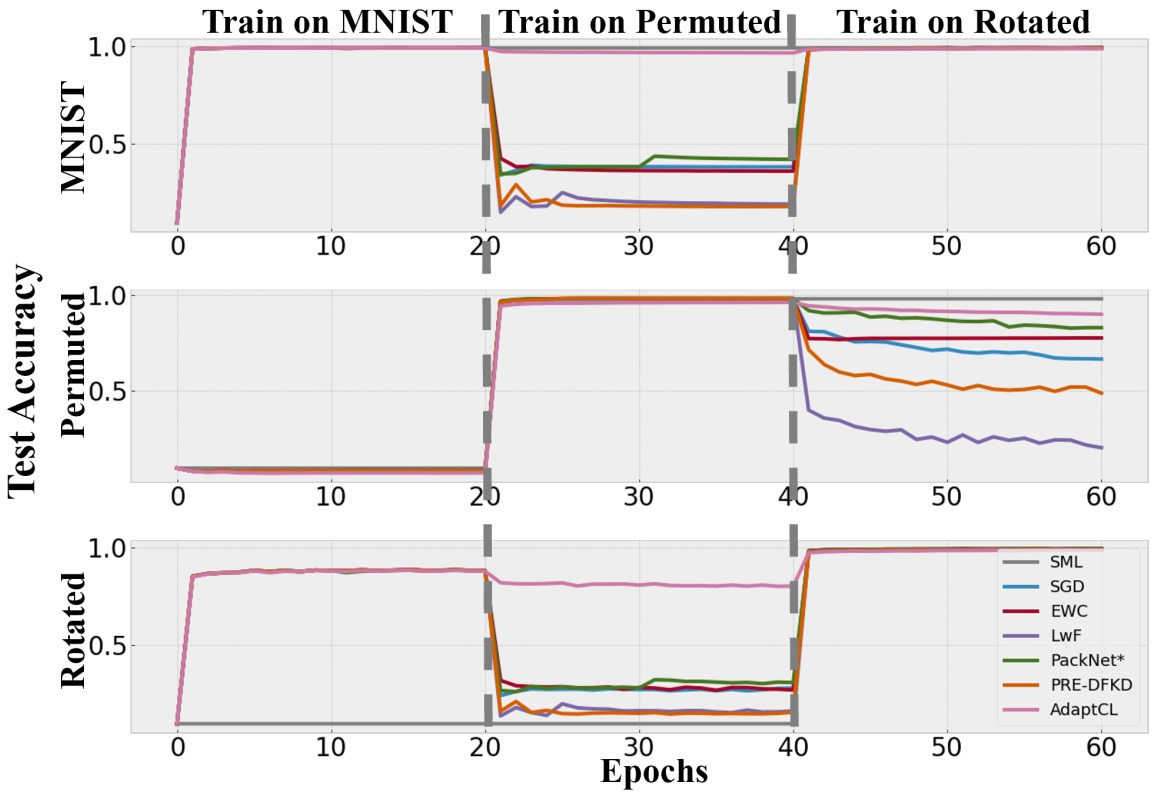


Figure 5.7: Visualization of each method's test accuracy training on more similar MNIST Variant datasets. Still, AdaptCL retains the best performance compared with other CL methods. Compared with using separated models for learning (SML), AdaptCL can increase the accuracy on similar unseen datasets via forward knowledge transfer. (Best viewed in color)

Table 5.5: Comparison of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and the number of used parameters of different continual learning methods on more similar MNIST Variant datasets. AdaptCL retains the best performance compared with other CL methods that use one model for learning.

	AAC↑	BWT↑	FWT↑	Params ($\times 10^7$)↓	Test Accuracy↑		
					MNIST	Permuted MNIST	Rotated MNIST
SML	0.989	-	-	3.352	0.993	0.980	0.992
SGD	0.884	-0.156	0.077	1.117	0.994	0.666	0.993
LwF	0.729	-0.391	0.023	1.117	0.993	0.204	0.991
EWC	0.889	-0.149	0.088	1.117	0.994	0.681	0.992
PackNet*	0.938	-0.076	0.101	1.117	0.994	0.830	0.991
PRE-DFKD	0.822	-0.240	0.179	1.117	0.991	0.488	0.988
AdaptCL	**0.958**	**-0.032**	**0.339**	**1.044**	0.990	**0.900**	0.985

ity increases, EWC effectively suppresses catastrophic forgetting and achieves higher average accuracy than SGD. This differs from previous experiments on dissimilar datasets where EWC had negative effects. LwF and PRE-DFKD struggle to balance the importance of different datasets, leading to significantly lower accuracy for Permuted MNIST, indicating that rehearsal-based methods are not suitable for large datasets with heterogeneous similarities. In contrast, our method shows robust performance in such scenarios. In Figure 5.7(b), the bottom of the chart shows that CL methods' inference accuracies on Rotated MNIST increase in the first 20 epochs due to the dataset's similarity to MNIST. Compared to separated model learning (SML), other CL methods enhance model generalization, leading to better accuracy and faster learning on unseen datasets. However, catastrophic forgetting still occurs when learning Permuted MNIST with a strong distribution shift. AdaptCL minimizes catastrophic forgetting, enabling the model to maintain high accuracy and generalization. We also observe that catastrophic forgetting sharply increases at the beginning of new data training, eventually reaching equilibrium as the accuracy of the new

dataset balances. Notably, rehearsal-based methods like LwF and PRE-DFKD exacerbate catastrophic forgetting on the dissimilar second dataset, Permuted MNIST, when learning the third dataset, Rotated MNIST, due to their high similarity to the initial MNIST dataset. This highlights the challenge for rehearsal-based methods in adapting to varying dataset similarity, resembling the challenge posed by varying data volumes.

5.6.4 Performance on Different Networks (Q1)

Table 5.6: Results of different continual learning methods applied on LeNet-5 with limited network capacity, including their AAC, BWT, FWT, and the number of used parameters. AdaptCL is also quite effective on LeNet-5, with the best BWT and comparable AAC with PRE-DFKD. It comes in second of AAC after PRE-DFKD, which may be due to less parameter usage.

	AAC↑	BWT↑	FWT↑	Params ($\times 10^4$)↓	Test Accuracy↑		
					MNIST	Permuted MNIST	Inverted MNIST
SML	0.968			18.51	0.948	0.978	0.977
SGD	0.479	-0.740	0.049	6.171	0.233	0.220	0.983
LwF	0.542	-0.646	0.045	6.171	0.243	0.398	0.984
EWC	0.098	-0.839	0.003	6.171	0.098	0.098	0.098
PackNet*	0.533	-0.644	-0.018	6.171	0.141	0.482	0.975
PRE-DFKD	0.579	-0.479	0.027	6.171	0.421	0.332	0.985
AdaptCL	0.567	**-0.457**	0.041	**6.162**	0.220	**0.740**	0.742

AdaptCL can be applied to neural networks with fully connected, recurrent, or convolutional layers. We also apply our method to the shallow Lenet-5 network and more complex VGG-16 to test its performance over different networks.

As shown in Table 5.6, we test the performance of different continual learning methods applied on LeNet-5 with network capacity constraints, and catastrophic forgetting increased significantly compared to ResNet-18. AdaptCL is also quite effective on

Table 5.7: Results of different continual learning methods applied on VGG-16, including their AAC, BWT, FWT, and the number of used parameters. AdaptCL is still very effective on VGG-18, having the best AAC and BWT with less parameter usage compared to other CL methods.

	AAC↑	BWT↑	FWT↑	Params ($\times 10^4$)↓	Test Accuracy↑		
					MNIST	Permuted MNIST	Inverted MNIST
SML	0.990	-	-	4.496	0.995	0.980	0.995
SGD	0.730	-0.390	-0.004	1.499	0.441	0.753	0.994
LwF	0.763	-0.342	-0.342	1.499	0.532	0.762	0.995
EWC	0.747	-0.363	0.021	1.499	0.455	0.792	0.994
PackNet*	0.802	-0.281	-0.015	1.499	0.527	0.885	0.885
PRE-DFKD	0.865	-0.182	0.010	1.499	0.841	0.767	0.989
AdaptCL	**0.963**	**-0.027**	-0.003	**1.396**	**0.979**	**0.922**	0.987

LeNet-5, with the best BWT and second-highest AAC. It comes in second after PRE-DFKD, which may be due to less parameter usage compared to PRE-DFKD. And in terms of overcoming catastrophic forgetting, even with limited network capacity, our method outperformed other methods in BWT. The EWC's result is deficient on LeNet-5 because the model collapsed during training on the third dataset, exceeding its capacity. We find that rehearsal-based approaches also perform better in this situation where model parameters and capacities are insufficient, presumably because they do not regulate the model itself so much as the data.

According to the results of different continual learning methods on VGG-16 5.7, still, AdaptCL demonstrate its effectiveness on MNIST Variants datasets. It shows the best average accuracy of AAC and BWT and exceeds the baseline method. It is worth noticing that when VGG-16 was used, most CL methods obtained negative FWT. From the experiment, the test accuracy of Permuted MNIST was not improved but actually decreased after only learning the dataset MNIST. This is probably because Permuted MNIST and MNIST have no similarity in the image structure, and the network structure of VGG prevents it from being able to observe permutation, while

ResNet-18 is slightly better in this respect.

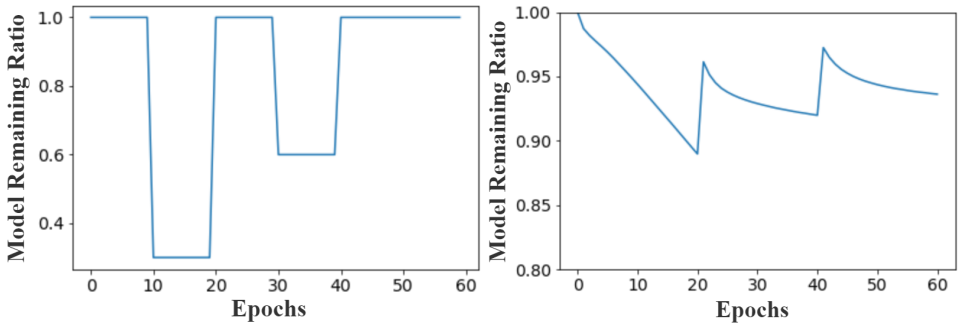
5.6.5 Ablation Study (Q3,Q4)

Parameter Isolation Ratio

To validate our intuition, we analyze and visualize the pruning ratio of the model in different datasets. Figure 5.9(a) displays the epoch-wise accuracy changes of the sparse network compared to fine-grained data-driven pruning and the dense network without pruning, along with the corresponding model remaining ratio during training. Our method dynamically and adaptively learns the model remaining ratios during training on each dataset, rather than manually setting fixed ratios as in other pruning methods. Additionally, fine-grained data-driven pruning enables a highly sparse pruned network to achieve the same accuracy as a dense network. Further, Figure 5.9(b) illustrates the change in the remaining ratios of the ResNet-18 model for each dataset of MNIST Variants at each epoch. Figure 5.9(b) demonstrates that it is possible to fit the new dataset without sacrificing the accuracy of the old dataset by adding only a few parameters, even when there are significant differences in data distribution between the old and new datasets. Consequently, manually assigning the same parameter ratio to all datasets is not reasonable.

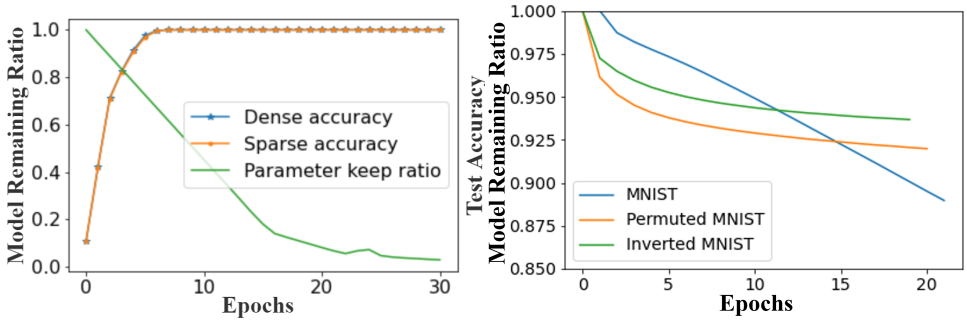
Parameter Execution Pattern

To analyze the parameter reuse in AdaptCL, we visualize the pattern of parameter execution during training and the proportion of each layer in the neural network occupied progressively by different datasets. Figure 5.10(a) displays the pattern of parametric ignition of the first Conv2d layer (flattened) on the MNIST Variants. Figure 5.10(b) presents the proportion of parameters occupied during adaptive learning in each convolution and fully connected layer of ResNet-18. From these figures, we ob-



(a) (b)

Figure 5.8: Comparison of model keep ratio (1- pruning ratio) between (a) PackNet* model and (b) our model. The diagrams present the change of model remaining ratios during the whole training process (60 epochs). PackNet* manually assigns the same ratio to each dataset. Our model adaptively assigns the corresponding ratio to learning incremental datasets in a data-driven way. It assigns a very high proportion of parameters to learning the first dataset from scratch, and only a small proportion of parameters to generalize existing parameters to the subsequent dataset.



(a) (b)

Figure 5.9: (a) Model on ResNet-18 remaining ratio and sparse accuracy compared with dense accuracy, using $\alpha = 10^{-4}$. (b) Change of model remaining ratio during training on MNIST Variants. (Best viewed in color)

serve that the parameters activated during previous task training remain unchanged when learning a new dataset. Additionally, we notice that some previously unused parameters (black) become activated during the new dataset training, contributing to the overall generalization of the network.

Hyperparameters

The hyperparameter α controls the intensity of pruning in the loss function during training, and it plays a crucial role in determining the final balance of model sparsity. We explore the effect of different α values, ranging from 10^{-3} to 10^{-7}, in our experiments. The value of α is mainly determined based on the data amount and training epochs, as pruning is primarily performed in each iteration of the training step. We aim to ensure that the product of the total number of iterations (e.g., image numbers multiplied by epochs) and α is approximately 1, allowing for efficient and effective pruning. To investigate the influence of different α values on the model's pruning intensity, we conduct experiments using different models, such as ResNet-18 and LeNet-5. Additionally, we examine the change in the model's remaining ratio for various α values. The results are shown in Figure 5.11, providing insights into the relationship between α, pruning intensity, and the model's remaining parameters for each architecture.

5.7 Chapter Summary

In this chapter, we aimed to tackle the challenge of managing heterogeneous datasets in continual learning. We observed unstable performance of rehearsal, regularization, and non-adaptive parameter isolation-based methods when dealing with multiple heterogeneous datasets in experiments. Inspired by the neural-reuse principle of human brains, we presented AdaptCL, a novel continual learning algorithm. Our proposed

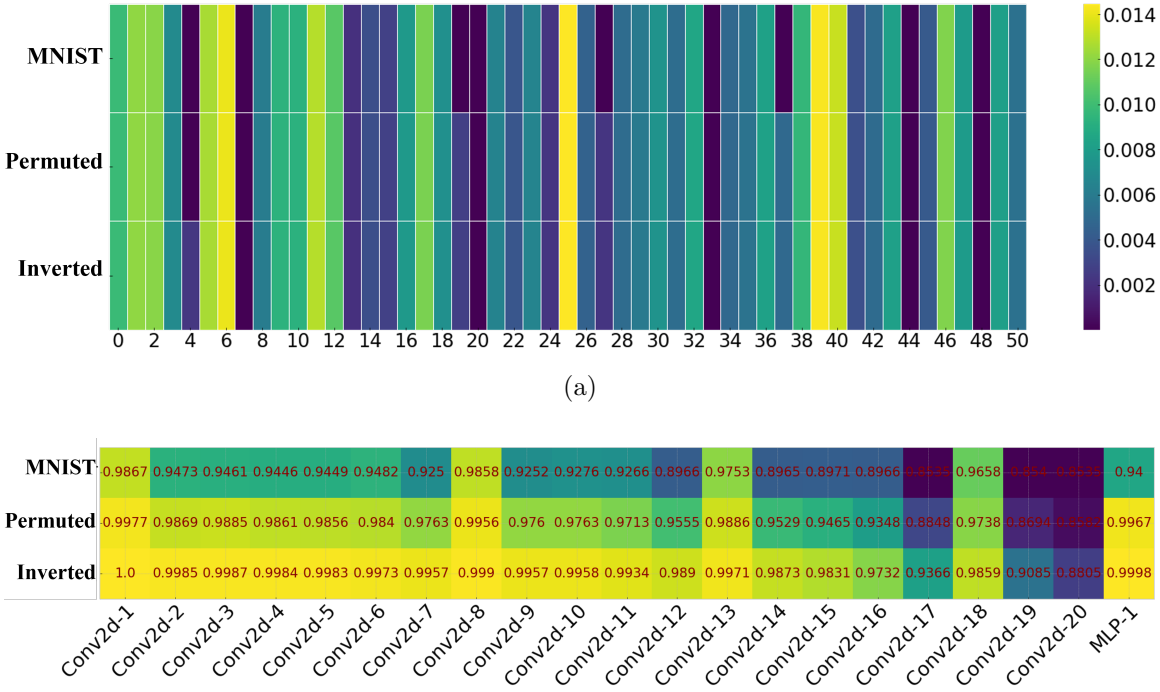
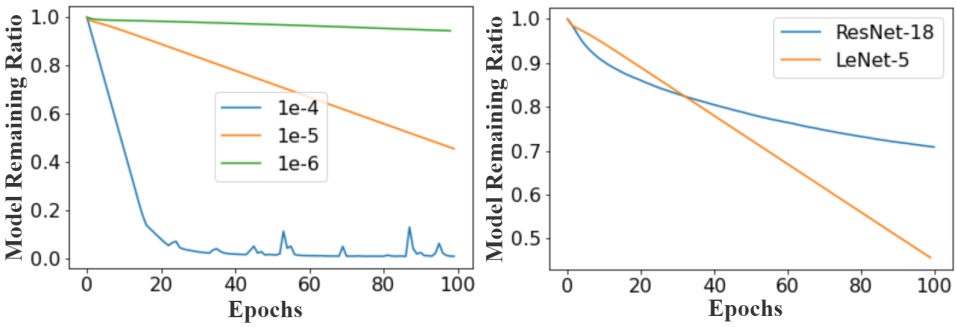


Figure 5.10: Illustration of the parameter execution patterns for continually trained models on MNIST Variants. Heatmap (a) showcases the firing probability of the xth parameters within the 1st Conv2d layer demonstrated in the x-axis; it illustrates that subsequent datasets maintain and reuse the previous parameters and generalize by adding new connections to them. The heatmap (b) shows the utilization ratio of different layers.



(a) (b)

Figure 5.11: (a) Pruning effect in ResNet-18 for different value of hyper-parameter α. (b) Change of model remaining ratio with ResNet-18 and LeNet-5 for different α. (Best viewed in color)

method effectively addresses the challenge of managing heterogeneous datasets in continual learning, outperforming existing approaches in terms of robustness and achieving higher average accuracy. Additionally, AdaptCL proves to be a proficient few-shot learner, exhibiting the capability to make generalizations based on limited examples similar to human cognitive abilities. By introducing fine-grained data-driven pruning and task-agnostic parameter isolation, we address catastrophic forgetting and demonstrate the effectiveness of AdaptCL across heterogeneous datasets in diverse applications. Our work contributes to the field by providing a novel algorithm that improves performance in heterogeneous dataset scenarios. While our approach is computationally efficient, we acknowledge the limitation of reduced learning efficiency with insufficient model capacity. To address this, future work will focus on introducing network expansion techniques to enhance scalability on a growing number of heterogeneous datasets.

Algorithm 2 Training Flow of AdaptCL

Require: weight of parameter W, threshold vector t is initialized with zero tensor.

for dataset $d = 0,\ 1,\ 2, \ldots$ **do**
  for layer in model **do**
    Reset threshold $t \leftarrow 0$
  end for
  for epoch **do**
    for step **do**
      update pruning mask $M^p{}_{ij} = S\left(|W_{ij}| - t_i\right)$
      update pruned weight $W = W \circ M^p$
      for layer in model **do**
        update the loss $L\left(\cdot\right) = L\left(D; W\right) + \alpha L_s$
      end for
      if $d == 0$ **then**
        gradient decent $W^*, t^* = argminL\left(\cdot\right)$
      else
        gradient decent with frozen parameters $W^*, t^* = argminL\left(\cdot\right) \circ \left(1 - M^f\right)$
      end if
    end for
  end for
  update freeze mask $M^f{}_{ij} = S\left(\left|M^f{}_{ij} + M^p{}_{ij}\right|\right)$
end for

Chapter 6

Research Contribution 3: Addressing Growth-Induced Forgetting with SparseGrow in Continual Learning

6.1 Introduction

In the field of continual learning (CL), model growth becomes essential for better adaptation to new data and experiences, thereby improving scalability for managing incremental knowledge [26, 15]. In real-life scenarios with evolving needs—such as self-evolving agents, autonomous vehicles, and language models—model growth is crucial for effectively handling information updates, increasing task complexity, and incorporating new features or modalities. As experiences accumulate and task complexity rises, a fixed-capacity model can quickly become a bottleneck, struggling to integrate new information while retaining the knowledge it has previously acquired. To address the above challenge, the model needs to intelligently expand its parameter

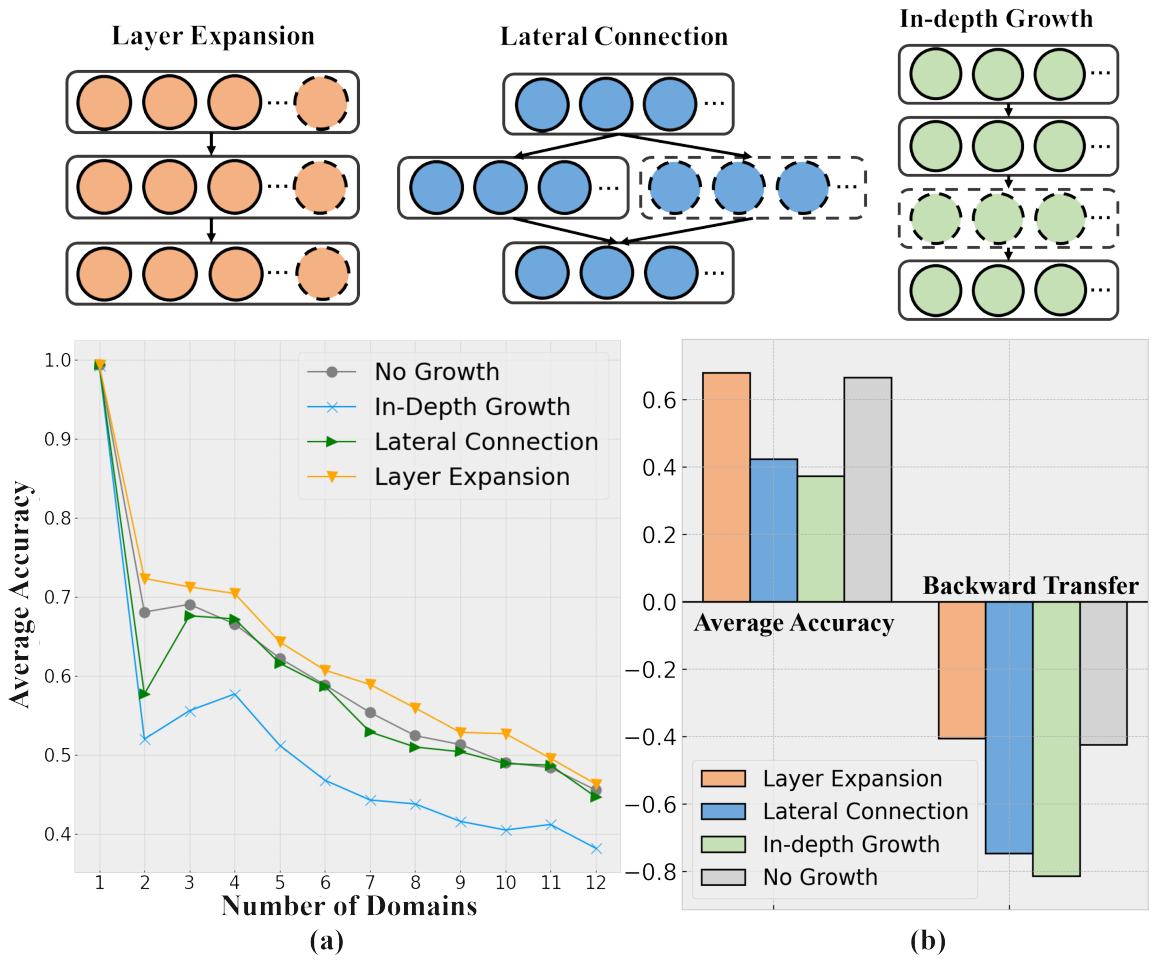


Figure 6.1: The three types of model growth methods applied to ResNet and their performance compared to not applying model growth (No Growth) in handling sequential domains. These methods include: 1) Layer Expansion, which widens each layer of the model; 2) Lateral Connection, introducing new lateral layers connected to adjacent layers; and 3) In-Depth Growth, adding hidden layers to increase model depth. (a) Depicts the change in average accuracy following the application of different model growth methods. (b) Shows the average accuracy and backward knowledge transfer of these methods. In this context, Layer Expansion demonstrates the highest average accuracy and the least forgetting in terms of backward transfer, even surpassing the performance of No Growth. (Best viewed in color)

capacity. This helps ensure high adaptability and sustainable scalability, forming the foundation for truly lifelong learning systems.

However, improper model growth often leads to a degradation of performance on previously learned tasks, an issue we identify as growth-induced forgetting, especially in task-agnostic continual learning settings such as domain and class incremental learning settings [114, 84, 30]. In these scenarios, the task label is unavailable, and the entire model, including expanded parameters, are used for inference. *Unlike traditional catastrophic forgetting, which occurs due to changes in data distribution,* ***growth-induced forgetting*** *is a distinct phenomenon. It arises not from new data but from increasing the model's parameter capacity—for example, by adding new neurons or layers to a previously trained model. This growth can disrupt the model's ability to retain or effectively use the knowledge learned before the growth.* A similar phenomenon is observed in neuroscience, where increasing neurogenesis in the brain after memory formation can promote forgetting, referred to as neurogenesis-based forgetting [16].

Some existing works in continual learning [94, 101, 42] adopt different model growth strategies and randomly initialize them to enhance adaptability. However, they fail to recognize the presence of growth-induced forgetting resulting from improper model growth. These methods are mostly task-specific, requiring manual task identification during inference to avoid forgetting, and not applicable for domain or class incremental scenarios; the rest are task-agnostic methods adopting improper model growth and initialization strategies that may further lead to growth-induced forgetting. Nevertheless, the lack of explicit acknowledgment of the growth-induced forgetting issue poses challenges in identifying suitable model growth approaches. This not only limits comprehensive control of forgetting but also hinders full utilization of model growth.

To our knowledge, this study is the first to identify and systematically investigate the phenomenon of growth-induced forgetting in continual learning. Our work provides a comprehensive analysis aimed at developing effective strategies for model growth

that expand the model's capacity while minimizing the risk of forgetting previously learned knowledge. Our study finds that layer expansion, which widens layers without impacting model functionality, can increase the sparsity of the model gradient in subsequent learning [65]. This feature distinguishes it from alternative model growth strategies, such as lateral connections and in-depth growth.

The impacts of these growth methods on growth-induced forgetting differ: Layer expansion expands parameters in width, which are computed collectively during inference; Lateral connections here refer to the addition of lateral layers, with the values from the lateral layers added to the main pathway; In-depth growth involves adding new hidden layers to a neural network, introducing new computations for preceding and succeeding layers' parameters. To promote both adaptability and knowledge retention with model growth, we have identified two critical elements: gradient and parameter sparsity. Drawing inspiration from neurogenesis [16, 18], we propose a novel sparse model expansion growing (SparseGrow) approach. SparseGrow utilizes layer expansion and gradient gating to improve gradient sparsity. With expanded layers, only a small portion of the model's parameters need to be updated when training for a new task. Meanwhile, gradient gating protects important parameters by using gradient masks. This sparsified gradient allows for targeted updates on the parameters while maintaining critical ones, ensuring network stability and preventing forgetting caused by growth. Furthermore, SparseGrow enhances parameter sparsity through both sparsity in training and initialization. Sparse training helps maintain a compact network without compromising accuracy, allowing the network to retain parameters that can adapt to new tasks. By initializing expanded layers with partially zero-valued parameters that are tailored to the dataset distribution, we aim to carefully control the model's plasticity. This approach enhances adaptability to new data and helps reduce forgetting that can occur due to model growth. In contrast, zero initialization can hinder updates, while random initialization may introduce unwanted interference.

To validate our findings, this study conducted extensive experiments across different task-agnostic settings using domain and class incremental datasets with varying numbers of tasks. The results highlight the importance of gradient and parameter sparsity during layer expansion. Our approach effectively addresses the issue of growth-induced forgetting while demonstrating adaptability and knowledge retention for incremental tasks. The main contributions of this paper are threefold:

- Our study reveals growth-induced forgetting, a novel variant of catastrophic forgetting observed when a model's capacity expands. This discovery fills a gap in existing research and contributes to the ongoing research on catastrophic forgetting.
- This work presents an in-depth study comparing various model growth strategies, revealing the potential of layer expansion to retain model knowledge through increased gradient sparsity.
- This work highlights the critical roles of gradient and parameter sparsity in continual learning. We propose SparseGrow, a novel method that effectively mitigates growth-induced forgetting via layer expansion, gradient gating, and sparse initialization/training during model growth. Our study shows that SparseGrow effectively maintains adaptability and knowledge retention in task-agnostic continual learning scenarios, supported by experimental validation in both domain and class-incremental settings.

6.2 Related Works

6.2.1 Growing Approaches

At present, there are three kinds of model growth methods. One is the lateral connection in width, the second is layer expansion in width, the third is growing in

depth.

Lateral Connection

Lateral connection grows new modules or layers in parallel with existing layers and allows connections with previous layers.

For instance, Progressive Neural Nets (PNNs) [80] statically grow the architecture with randomly initialized modules while retaining lateral connections to previously frozen modules. However, this method is limited to specific simple networks. Schwarz et al. [81] use randomly initialized active lateral columns (1x1 conv) to learn new tasks by connecting them to lateral columns that store previous knowledge. They also distill knowledge from the active column to knowledge base layers. This method is applicable to Conv2d layers but is limited to specifically designed simple networks and lacks generality. Zhang et al. [105] adopt AutoML-based model growing with both lateral connections and in-depth growth. They also use knowledge distillation to compress the model after learning a task. This method is applied to complex networks like VGG. However, using AutoML or similar methods often requires defining the layers for growth during the initial model definition, limiting its applicability. In-depth and lateral growth can also lead to more forgetting. Ardywibowo et al. [6] propose VariGrow, which detects if a new task is arriving through an energy-based novelty score. If the novelty score is high and the sample is "detected" as a new task, VariGrow grows a new expert module to handle it. Otherwise, the sample is assigned to one of the existing experts that is most familiar with it. Unlike other methods, VariGrow is task-agnostic continual learning and does not require task identification during inference, making it suitable for class and domain incremental continual learning. However, this method can prevent the update of old modules when encountering similar data. In Hu et al. [31], given a new task t, a new branch called the task t expert is added while freezing existing experts. This is achieved

by introducing dense connections between the intermediate layers of the task expert networks. Li et al. [50] propose a hybrid solution for lateral connection and layer expansion, involving three operations: 'new', 'adaptation', and 'reuse'. The 'new' operation introduces a randomly initialized 3x3 layer trained from scratch. In the 'reuse' strategy, existing frozen weights are reused. The 'adaptation' strategy adds a 1x1 convolution layer in parallel to a 3x3 convolution layer, keeping the original 3x3 kernel fixed while learning the parameters of the 1x1 adapter. However, this hybrid solution is limited to specially designed Conv2d networks and lacks generality.

Layer Expansion

DEN [102] uses layer expansion in a top-down manner, growing every layer if the loss does not meet a threshold. Similar to DEN, Hung et al. [32] expand the number of filters (weights) for new tasks. Moreover, they adopt gradual pruning to compact the model. However, different from our sparse growing method, their pruning is not data-driven, and their method requires manual help of task IDs during inference, making it unsuitable for domain incremental or class incremental learning. Ostapenko et al. [67] introduce Dynamic Generative Memory (DGM) and expand the same number of neurons used in a layer in the generator of a GAN for the scalability of rehearsal. The scope of rehearsal is expanded. Geng et al. [20] expand the hidden size H_k for the k-th task from H_{k-1} by the pruning ratio of task $k-1$. After random initialization, on-data finetuning is performed for the newly added parameters. Yang et al. [101] grow a randomly initialized expanded filter and concatenate it to the network. Xu et al. [96] adaptively expand each layer of the network when the t-th task arrives. This method is also applicable for simple convolutional networks and fully-connected networks. Yan et al. [98] expand the model with new parameters by creating a separate feature extractor F_t for incoming data and taking a uniform distribution as the prior distribution. This method is also applicable for class incremental learning.

In-Depth Growth

Some non-continual learning methods, such as Wen et al. [94] and Yuan et al. [104], increase the depth of the neural network in the hidden layer to achieve faster convergence and efficiency. Yan et al. [98] also increase the depth of the neural network in the hidden layer to achieve faster convergence and efficiency. They use random initialization, which can help escape from a bad starting point. Yuan et al. [104] propose a budget-aware growing process that starts from a small, simple seed architecture and dynamically grows and prunes both layers and filters to make the network wider and deeper. They also adopt the initialization of ResNet or VGG.

In the field of continual learning, a few papers also utilize in-depth growth to increase model capacity: Kozal et al. [42] add new layers on top of existing ones. Zhang et al. [105] adopt AutoML-based model growing in both width and depth by adding lateral layers and in-depth layers. They also use knowledge distillation to compress the model after learning a task. Nevertheless, adding new layers in depth within the neural network can have a profound impact on the model's structure compared to growing in width, leading to more severe growth-induced forgetting and altering the learned features of previously acquired tasks. This effect undermines the network's ability to retain prior acquired knowledge and, therefore, is not suitable for continual learning.

6.2.2 Overcoming growth-induced forgetting

In this section, we explore the related works in overcoming growth-induced forgetting.

growth-induced forgetting is currently primarily defined and addressed in the field of neural architecture search (NAS). Existing methods in NAS mainly employ regularization techniques. For instance, Benyahia et al. [8] introduce a statistically-justified weight plasticity loss that regularizes the learning of a model's shared parameters

according to their importance for the previous models. Zhang et al. [109] define growth-induced forgetting and propose a Weight Plasticity Loss (WPL) to reduce this forgetting in One-Shot NAS, which aims to maximize the posterior probability.

While the problem of growth-induced forgetting caused by model growth is currently only defined in the neural architecture search area, it has also been indirectly addressed in the context of continual learning. Several studies in continual learning have developed methods to handle model growth and maintain accuracy on previous tasks, albeit without explicitly recognizing or referring to the phenomenon of growth-induced forgetting.

Here, we focus on task-agnostic continual learning, excluding task-specific continual learning, which relies on manual labelling of task IDs to assist machines in selecting the model's inference region, thereby bypassing growth-induced forgetting.

Existing task-agnostic continual learning approaches typically employ regularization and rehearsal methods to mitigate the forgetting problem encountered by models. For example, Madaan et al. [59] build on top of the distillation family of techniques and modify it for a new setting where a weaker model takes the role of a teacher. They propose Quick Deep Inversion (QDI) to recover prior task visual features to enhance distillation.

Zhang et al. [105] use RWC to alleviate forgetting while employing knowledge distillation for compression. Schwarz et al. [81] use Elastic Weight Consolidation (EWC) to overcome forgetting caused by adding an active column. Ostapenko et al. [67] expand the same number of neurons used in a layer in the generator of GAN for the scalability of rehearsal. Ardywibowo et al. [6] propose a different method called VariGrow. It detects if a new task is arriving through an energy-based novelty score. If the novelty score is high and the sample is "detected" as a new task, VariGrow will grow a new expert module to be responsible for it. Otherwise, the sample will be assigned to one of the existing experts who is the most "familiar" with it. However,

this method has a drawback in that when encountering similar data, the old modules cannot be updated with new data, which hinders knowledge reuse and propagation between modules and is not parameter-efficient.

Similar to our approach's task-agnostic freezing method, Yan et al. [98] freeze the previously learned representation and augment it with additional feature dimensions from a new learnable feature extractor.

6.3 Problem Formulation

We consider the problem of achieving better adaptation of new data by growing model's capacity while also overcoming growth-induced forgetting in task-agnostic scenarios, encompassing domain-incremental and class-incremental learning settings.

Given an incremental sequence of non-iid datasets $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_t\}$ where $\mathcal{D}_t$ is the dataset at time t and a model trained with previous datasets, denoted as $f(\cdot; \theta_t)$ where θ_t represents the model parameters (weights and biases) at time t.

The assumption is that when accessing the current dataset $\mathcal{D}_t$ at time t, it is impractical or impossible to access any previous or future dataset.

The objective is to grow the model capacity to enhance the accuracy of the current dataset $\mathcal{D}_t$ and future datasets while minimizing the degradation of the performance on the previous dataset, as compared to a scenario where the model does not grow. This can be formally represented as:

$$\text{minimize } J(\theta_t) = \mathcal{L}(\mathcal{D}_t; f(\cdot; \theta_t)) + \lambda \sum_{i=1}^{t-1} \mathcal{L}(\mathcal{D}_i; f(\cdot; \theta_i))$$

where $J(\theta_t)$ is the objective function at time t, $\mathcal{L}(\cdot)$ represents the loss function, λ is a regularization parameter to control the degradation of the performance on previous datasets, the first term $\mathcal{L}(\mathcal{D}_t; f(\cdot; \theta_t))$ denotes the loss on the current dataset, and the

second term $\sum_{i=1}^{t-1} \mathcal{L}(\mathcal{D}_i; f(\cdot; \theta_i))$ represents the cumulative loss on previous datasets.

6.4 Methodology

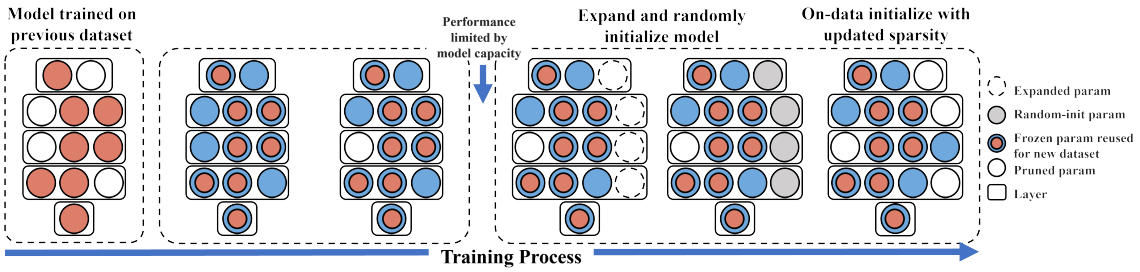


Figure 6.2: The training flow of SparseGrow involves sparse neural expansion at the structural level and on-data initialization at the weight level at the conclusion of the training process. Initially, model expansion and random initialization are employed to increase model capacity for improved adaptation to new data. Subsequently, growth-induced forgetting is addressed through frozen sparse training and on-dataset fine-tuning. Notably, while 0-initialization of grown parameters prevents growth-induced forgetting, it also stops the update of these parameters. In contrast, the case differs for pruned parameters initialized to 0 value.

6.4.1 Increased Gradient Sparsity

Increasing the sparsity of model training gradients is an important way to reduce model forgetting[65]. When training on new tasks, updating only a small number of model parameters helps stabilize the network and reduces the forgetting of previously learned tasks. Common methods for regularization and structure can limit significant changes in model parameters, which reduces the variability of model gradients and thus minimizes forgetting. The techniques of layer expansion and gradient gating that we utilize also aim to prevent catastrophic forgetting by increasing the sparsity of

model gradients. As the network grows wider, the gradients become sparser, meaning fewer parameters need to be adjusted, which helps minimize forgetting of previously learned tasks.

Layer Expansion

Layer expansion is an effective growth method that minimizes forgetting while allowing for more general and detailed development without the need for manual selection of growth locations. Unlike lateral connections or extensive growth methods, this approach maintains the functionality of the model and is particularly well-suited for complex neural networks.

In this approach, we consider a neural network with a set of parameters denoted as W_l for $1 \leq l \leq C$, where W_l represents the parameter matrix at layer l and C is the total number of layers. Each layer in the neural network—such as Conv2D, MLP, or Batch Norm—undergoes an expansion technique.

Given an original weight tensor $W \in \mathbb{R}^{(c_o \times c_i)}$ and an original bias tensor $b \in \mathbb{R}^{(c_o)}$, where c_o is the number of output channels and c_i is the number of input channels, these tensors are expanded into W_{exp} to accommodate the desired expansion.

$$W_{exp} \in \mathbb{R}^{(c_i+m, c_o+n)}; \quad b_{exp} \in \mathbb{R}^{(c_o+n)}. \tag{6.1}$$

In this equation, m and n denote the expanded channel numbers. The expanded layer l now has $c_i + m$ input channels and $c_o + n$ output channels. Weights from the existing layer are transferred to their original positions within the new layer, while the expanded portion is initialized. The preservation and pruning masks are updated accordingly. After the expansion, the weights and biases of the expanded layer are updated as follows:

$$W_{exp}[:c_o, :c_i] = W$$
$$b_{exp}[:c_o] = b \tag{6.2}$$

Expansion across the entire model follows a systematic rule ensuring the alignment of input and output channels across consecutive layers. This expansion logic, consistent with structures like ResNet blocks and skip connections, maintains information flow by expanding involved layers while upholding the correspondence between input and output channels. Each layer (l ranging from 1 to C) is expanded by adding n extra channels, aligning input and output sizes between adjacent layers:

$$c_o^{(l)} \to c_o^{(l)} + n; \quad c_i^{(l+1)} \to c_i^{(l+1)} + n \tag{6.3}$$

This gradual growth in capacity allows the network to capture more complex patterns and representations as the number of channels expands, while maintaining the necessary consistency for effective information propagation.

Gradient Gating

During training, gradients only influence non-preserved parameters, while preserved parameters remain fixed.

The gradient gating mechanism can be described by the following equation:

$$\nabla W_{\text{gated}} = (1 - P) \odot \nabla W \tag{6.4}$$

In this equation, ∇W_{gated} represents the gated gradients, while ∇W contains the original gradients. The variable P is the preservation mask, where a value of 1 indicates that the gradient is preserved, and a value of 0 indicates that it is trainable.

This approach is designed to preserve specific knowledge within the model, which is

particularly useful in scenarios that require task-agnostic adaptation while maintaining previously learned information.

After training on each dataset, a preservation mask P is generated by aggregating the important learned parameters.

Since unimportant parameters are pruned during sparse training and initialization, the preservation mask is computed as follows:

$$P_{ij} = I\left(|W_{ij}| > 0\right), \quad \forall i \in [1, c_o], \forall j \in [1, c_i]$$

In this equation, I is the indicator function, c_o is the number of output channels, c_i is the number of input channels, and W_{ij} represents the jth weight associated with the ith output neuron.

To prevent gradients from affecting preserved parameters in expanded layers, we implement gradient gating as follows:

$$W^*_{exp} = \arg\min_{W} L(D; W_{exp}) \odot (1 - P)$$

In this equation, $L(D; W)$ denotes the loss function applied to the current dataset, and W^*_{exp} is the optimal value of the expanded weights. The $\odot$ operation performs element-wise gradient gating, where zero entries in $(1 - P)$ ensure that updates do not occur for the preserved weights. Additionally, the preservation mask will also be expanded.

6.4.2 Increased Parameter Sparsity

Introducing increased parameter sparsity in continual learning aims to better control model plasticity, improve generalization, and enhance model adaptability. By leveraging data-driven pruning techniques, these methods provide a way to enhance

parameter sparsity.

Sparse Training and Sparse Initialization are innovative strategies designed to increase parameter sparsity as tasks evolve. By utilizing data-driven pruning, these methods enable the network to dynamically adjust its parameter sparsity based on the specific characteristics of the data being processed. This approach not only promotes a more efficient allocation of model resources but also improves the network's resilience to catastrophic forgetting.

Sparse Training

The sparsification mechanism operates inherently within each layer, automatically regulating the levels of sparsity during training while maintaining the model's performance.

Sparsification is achieved through a sparse mask M_{ij} determined by trainable thresholds $t \in R^{c_o}$:

$$M_{ij} = I\left(|W_{ij}| > t_i\right), \quad \forall i \in [1, c_o], \forall j \in [1, c_i] \tag{6.5}$$

The threshold mechanism operates on a neuron basis for fully-connected layers (using threshold t_i for each output neuron) and on a filter basis for convolutional layers (using threshold t_i for each output channel).

Thresholding enables fine-grained sparsification while maintaining structural integrity. During forward passes, sparse operations are performed using the following matrix multiplication: $W \odot M$. The threshold parameters t are dynamically optimized during training through a dual-phase process. To promote high sparsity in neural networks, a regularization term is introduced as follows:

$$\mathcal{R} = \sum_{i=1}^{c_o} e(-t_i), \quad L_s = \sum_{i=1}^{C} \mathcal{R}_i \tag{6.6}$$

In this context, $e(-x)$ acts as a threshold regularizer that penalizes the threshold values t from becoming excessively large or too small. The training loss function integrates L_s to enable the training of a sparse neural network directly using back-propagation.

The complete optimization objective combines the task loss with the sparsity regularization:

$$W^*, t^* = \arg\min_{W,t} \left[L(D; W) + \alpha L_s\right] \odot (1 - P) \tag{6.7}$$

Here, α controls the trade-off between sparsity and training accuracy. The exponential regularization $e(-x)$ progressively penalizes low thresholds, prevents threshold explosion through asymptotic decay, and allows for layer-wise adaptation using independent t vectors.

When training on sequential datasets, the sparsification process is reinitialized for each new task. This approach allows new sparsity patterns to be formed for novel data while retaining the parameter configurations learned from previous tasks through the preservation mask P. Importantly, the threshold parameters will also be expanded during the process of layer expansion.

Sparse Initialization

Sparse initialization involves employing random initialization with a sparse on-dataset warm-up phase. While random initialization of grown parameters enables better adaptation to new data, it will cause higher growth-induced forgetting. We further apply on-dataset sparse initialization to fill the gap of randomly initialized grown parameters to the learned data pattern, therefore ensuring less forgetting.

Proper initialization of new parameters is crucial for adaptability and less forgetting, with zero-init hindering its update and random-init causing heavy growth-induced forgetting. To address this, we propose a simple yet effective approach: random and then on-data initialize the expanded part for several epochs to adapt it to the current distribution at the final phase of training. The random initialization for a the expanded weight parameter $W_{exp}[c_o, :]$, the corresponding bias b_{exp} and previous mentioned threshold vector t_{exp} in a neural network layer is defined as:

$$\begin{aligned} W_{exp}[c_o, :]_{\text{init}} &\sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right) \\ t_{exp}[c_o, :]_{\text{init}}, \quad b_{exp}[c_o, :]_{\text{init}} &\sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right) \end{aligned} \tag{6.8}$$

where $\mathcal{N}$ denotes the normal distribution, and n_{in} represents the number of input units. We then selectively warm-up the randomly initialized layer with the current dataset using sparse training to adapt it to the current distribution. Sparse initialization involves updating the expanded parameters W_{exp} and b_{exp} using the current dataset by minimizing the sparse expanded loss function $L\left(D; W\right) + \alpha\ L_s$. During this, we update the expanded parameters by computing their gradients with respect to the sparse expansion loss and freezing technique, thus applying an optimization algorithm:

$$W^*_{exp}, t^*_{exp} = \arg\min_{W,t} [L\left(D; W_{exp}\right) + \alpha\ L_s] \odot (1 - P) \tag{6.9}$$

This process allows the expanded part to adapt to the learned distribution, preventing the overwriting of previously learned knowledge while incorporating the expanded parameters' capacity for better adaptation and faster learning.

6.5 Experimental Setup

6.5.1 Datasets

The domain-incremental datasets we use include the Permuted MNIST, FreshStale, and DomainNet. For evaluation in a class-incremental setting, we use the class-incremental MNIST.

Permuted MNIST is a variation of the MNIST dataset in which the pixel positions of the images are randomly permuted, with no shared features among different permutations. Permuted MNIST includes an infinite number of permutations in a domain-incremental setting. Each dataset consists of 60,000 images for training and 10,000 for testing.

Class-incremental MNIST In Class-incremental MNIST, the MNIST dataset is partitioned into distinct groups of classes. These groups may contain an uneven number of classes or class overlaps.

FreshStale [36] dataset comprises a total of 14,683 images of six different types of fruits and vegetables, including apples, bananas, bitter gourds, capsicums, oranges, and tomatoes. Each image in the dataset is classified as either fresh or stale. The total size of the dataset is approximately 2GB.

DomainNet [70] dataset consists of image data from six domains, each with a different amount of data, including real photos, painting, clipart, infograph, quickdraw, and sketch. There are 48K - 172K images (600K in total) categorized into 345 classes per domain.

6.5.2 Evaluation Metrics

For a principled evaluation, we adopt the following evaluation metrics[55]:

- Average Accuracy: $AAC = \frac{1}{T}\sum_{i=1}^{T} R_{T,i}$
- Backward Transfer: $BWT = \frac{1}{T-1}\sum_{i=1}^{T-1}(R_{T,i} - R_{i,i})$
- Forward Transfer: $FWT = \frac{1}{T-1}\sum_{i=2}^{T-1} R_{i-1,i} - \bar{b_i}$

where $R_{T,i}$ represent the test accuracy of each dataset after all datasets are learned. $R_{i,j}$ is the test accuracy on dataset t_j after observing the last sample from dataset t_i. Letting $\bar{b}$ represent the vector of test accuracy at random initialization.

The primary evaluation criterion is the average accuracy (AAC), with higher values indicating better performance. When AAC is the same, larger BWT or FWT is superior. We assess parameter efficiency through parameter calculations (Params).

6.5.3 Experiment Settings

All baselines are applied on ResNet-18 as base network for the experiments. To guarantee completely reproducible results, we set seed value as 5 for the random function of Numpy, python Random, Pytorch, Pytorch Cuda, and set Pytorch backends Cudnn benchmark as False, with Deterministic as True, configuring PyTorch to avoid using nondeterministic algorithms for some operations, so that multiple calls to those operations, given the same inputs, will produce the same result.

6.5.4 Baselines

Since most of the existing work on model growth mentioned in the related work section are implemented on specially designed networks, for a fair comparison, we abstract their model growth method and apply it to the same base network.

For baselines without model growth, we use the following classic and recent baselines: 1) SGD [10]: A naïve model trained with direct stochastic gradient descent. 2) EWC

[41]: A regularization technique in continual learning that uses diagonal elements of Fisher Information Matrix to constrain the weights of the neural network and avoid catastrophic forgetting. 3) LwF [51]: A rehearsal-based method that uses knowledge distillation to preserve previously learned knowledge along with training on new tasks. 4) PRE-DFKD [9]: A recently proposed rehearsal strategy that rehearses the model using the data-free knowledge distillation through the distribution of the previously observed synthetic samples from a Variational Autoencoder. 5) PackNet [61]: A method that also using pruning technique. Different from our sparse neural expansion, it uses static pruning and has no model growth. It prunes a certain percentage of neurons, knowing the number of tasks to learn. Here we implement it into task-agnostic settings. 6) AdaptCL [114]: A recently published task-agnostic method that uses dynamic pruning and freezing but no model growth.

For comparison of different model growth methods, we implement the following as baselines. Here, we implement all three growth methods with a uniform increase of the model's Conv2d and MLP layers by the smallest unit: 1) SGD+LayerExp: A method that implements Layer Expansion (LayerExp) on base network that is equipped with simple SGD. With each expansion, the input sizes and channels of Conv2d and MLP layers increase. The expanded parameters are randomly initialized. 2) SGD+IDGrow: Applying In-Depth Growth (IDP) to base networks with simple SGD. During in-depth growth, we add one MLP or Conv2d layer below the same existing one, with randomly initialized parameters.

3) SGD+LatConn: Employing Lateral Connections (LatConn) on base networks with SGD. For model growth, each MLP or Conv2d layer will add a randomly initialized lateral layer, and both parameters will be connected to the following layer. 4) SGD+LayerExp+ODInit: A model with layer expansion and o-dataset initialization for evaluation of the initialization strategy.

For baselines with model growth, we implement the following task-agnostic CL baselines with layer expansion, for fair comparison of the ability to overcome growth-

induced forgetting: 1) EWC+LayerExp: We implement this version of EWC with layer expansion. After expansion, the Fisher Information Matrix will expand accordingly. 2) LwF+LayerExp: A method that combines layer expansion and LwF. After expansion, the knowledge distills from the expanded model. 3) PRE-DFKD+LayerExp: We apply layer expansion with PRE-DFKD to base networks.

6.6 Results

6.6.1 Comparison of Model Growth Methods

Table 6.1: Performance evaluation of different model growth methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and number of used parameters (Param $\times 10^7$) after observing four domains of Permuted MNIST.

Method	AAC↑	BWT↑	FWT↑	Params↓
No Growth	0.666	-0.424	0.014	1.117
Lateral Connection	0.424	-0.747	0.050	2.216
In-Depth Growth	0.373	-0.814	0.133	1.993
Layer Expansion	**0.679**	**-0.406**	0.014	1.131

To compare the effects of different model growth methods on the overall performance of the model, including average accuracy, knowledge transfer primarily consisting of catastrophic forgetting and growth-induced forgetting issues, as well as parameter efficiency, we contrast three model growth methods—layer expansion, lateral connections, in-depth growth, and no growth—on a domain incremental permuted MNIST dataset comprising four domains. The results are summarized in Table 6.1.

The results indicate that, compared to not growing the model, layer expansion not

only minimize growth-induced forgetting but also reduces catastrophic forgetting caused by non-iid data. It facilitates backward knowledge transfer, improving from -0.424 to -0.406, consequently boosting the model's overall accuracy from 0.666 to 0.679, with only 1.2% increase in parameters. Conversely, other model growth methods lead to pronounced growth-induced forgetting, thereby lowering the model's average accuracy. Additionally, since both methods require full-layer expansion to ensure uniform growth of MLP and Conv2d layers, the model's parameter count nearly doubles.

6.6.2 Method Scalability and Adaptability

We comprehensively compare performances of all baselines, including existing methods with and without model growth to domain-incremental permuted MNIST datasets.

Figure 6.3 illustrates the fluctuation in average accuracy with increasing numbers of tasks or domains for different baseline methods. A comparison between SparseGrow and its NoLE version reveals that as the number of domains increases, the benefits of model growth also increase, highlighting the effectiveness of proper model growth in enhancing model scalability. Among different model growth techniques, layer expansion demonstrates a consistent improvement in model accuracy, while the effects of lateral connections are unstable, and in-depth growth mostly leads to negative impacts.

Referring to methods incorporating LayerExp with rehearsal and regularization methods, it is observed that the AAC of the LayerExp versions is even lower than those without LayerExp, suggesting that for this type of dataset, such rehearsal and regularization methods are not suitable for direct integration with layer expansion. However, combining simple SGD directly with LayerExp can enhance the model's AAC. The fusion of our model with LayerExp performs the best, even surpassing the stability of SGD.

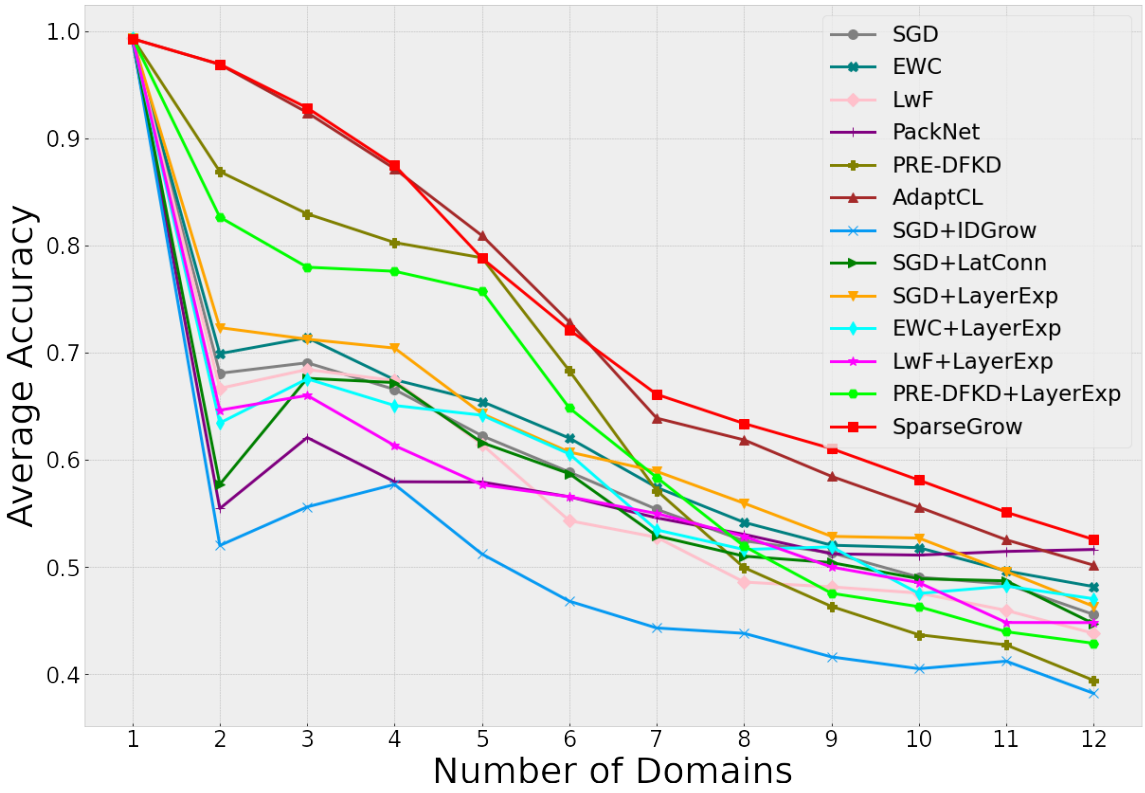


Figure 6.3: Average accuracy comparison of continual learning methods (EWC, LwF, PackNet, PRE-DFKD, and AdaptCL) alongside different model growth techniques (IDGrow, LatConn, LayerExp) and several CL strategies incorporating LayerExp across an increasing number of observed domains. Notably, rehearsal methods like LwF and PRE-DFKD exhibit an initial decline when combined with LayerExp, suggesting potential unsuitability for direct combination with expansion, possibly leading to more MF. Yet, the later positive effects of model capacity enhancement from expansion seem to surpass the negative impact of MF as domains increase. SparseGrow has demonstrated superior performance in knowledge retention, with its effectiveness improving as domain number rises.

In the case of PackNet, the model's average accuracy remains almost constant as the number of domains increases. Introducing on-data initialization to all baselines incorporating layer expansion results in improved AAC across the board, suggesting

the effectiveness of this approach.

It is noted that LwF's average accuracy falls below that of SGD, although this discrepancy is not observed on other datasets. This prompts the speculation that knowledge distillation methods may not be suitable for permuted MNIST, a dataset lacking inter-domain similarity.

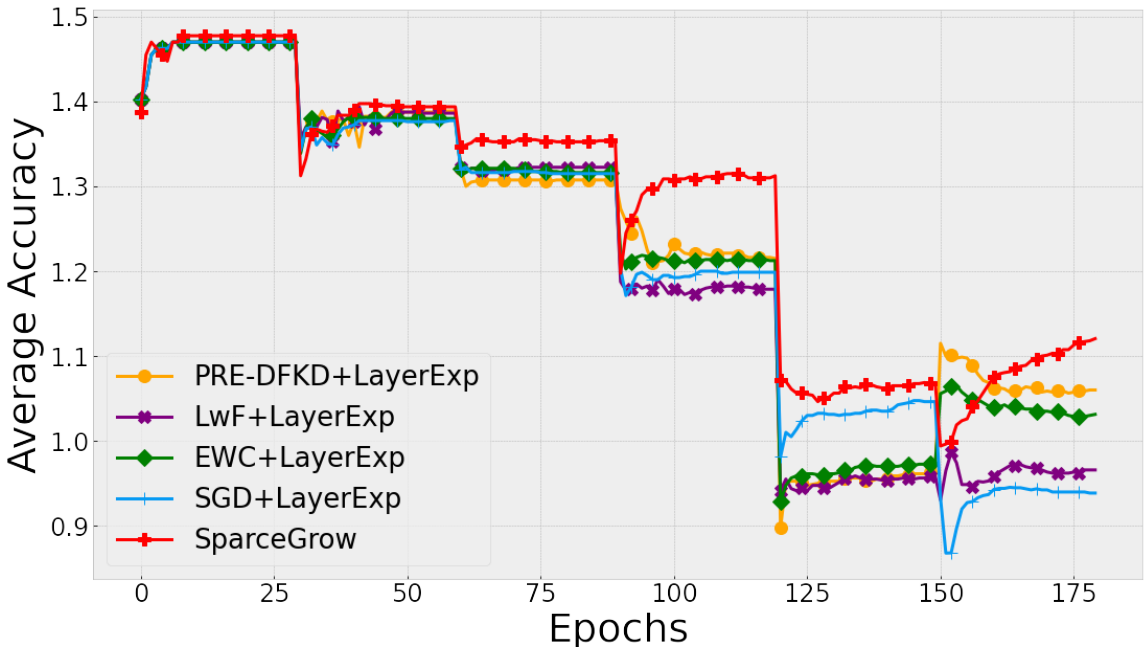


Figure 6.4: Epoch-wise average accuracy of observed domains using different continual learning methods with layer expansion on FreshStale datasets with six sequential domains.

Table 6.2 presents a comparison of the AAC, BWT, and FWT metrics of the three categories of continual learning models after learning from 12 domains continuously. SparseGrow, compared to SGD without CL methods, significantly reduces the model's overall forgetting issues, boosting BWT by 64% and AAC by 15%, showcasing the scalability of the approach. Layer expansion demonstrates improvements combining with SGD, PRE-DFKD, and SparseGrow in terms of AAC, BWT, and FWT across the 12 datasets. PackNet, utilizing static pruning, exhibits certain advantages with a

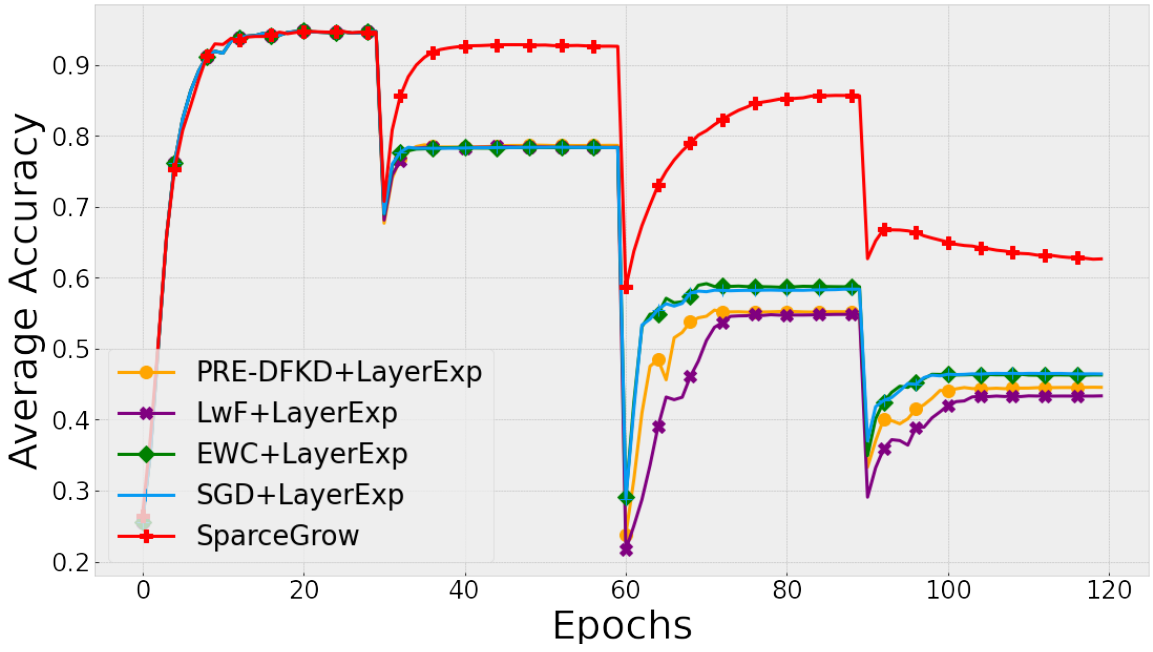


Figure 6.5: Epoch-wise average accuracy of observed domains using different continual learning methods with layer expansion on DomainNet datasets with four sequential domains.

larger number of datasets; however, due to its requirement of known dataset sizes, it is unattainable in CL scenarios, rendering comparisons unfair and making it unsuitable for continuously growing datasets.

6.6.3 Evaluation of Overcoming growth-induced forgetting

To compare the effectiveness of these continual learning (CL) methods across different types of data and their ability to reduce growth-induced forgetting, we evaluated the impact of these methods in suppressing growth-induced forgetting with layer expansion on the FreshStale and DomainNet datasets.

As shown in Figure 6.4, the baseline methods exhibited increased Average Accuracy Change (AAC) on the FreshStale dataset, indicating that these CL methods can miti-

gate the effects of layer expansion-induced growth-induced forgetting to some extent. Among them, SparseGrow consistently demonstrated the most stable reduction in model forgetting, achieving the highest AAC.

In Figure 6.5, SparseGrow demonstrates consistent effectiveness in overcoming growth-induced forgetting, outperforming other methods in effectively mitigating the growth-induced forgetting induced by layer expansion when compared to SGD.

The results presented in Table 6.3 indicate that existing regularization and rehearsal-based methods show some effectiveness in addressing growth-induced forgetting, but their performance across different datasets lacks robustness. In contrast, SparseGrow consistently minimizes the impact of model growth-induced growth-induced forgetting across various data types.

6.6.4 Class-Incremental Learning Setting Results

To assess the performance of our method in diverse task-agnostic settings, the methods were also evaluated in challenging class-incremental learning scenarios, as depicted in Table 6.4. This setting presents a formidable challenge for continual learning, where similar inputs lead to distinct classes assigned to different output layers, often resulting in rapid forgetting of previously learned classes within a single epoch.

Given the heightened intensity of forgetting and the scale of the datasets, techniques such as rehearsal and regularization methods struggle to effectively maintain accuracy, exhibiting only marginal superiority over SGD. In contrast, SparseGrow excels in preserving accuracy significantly, highlighting its potential to mitigate growth-induced forgetting and catastrophic forgetting in task-agnostic environments.

6.7 Chapter Summary

In this chapter, our study is the first to highlight the crucial issue of growth-induced forgetting caused by improper model growth in task-agnostic continual learning. We have identified layer expansion as a promising method to boost adaptability and knowledge retention. Our proposed SparseGrow approach is specifically designed to address these challenges. Through experimental validation, we have demonstrated the effectiveness of our method in mitigating growth-induced forgetting and improving knowledge retention for incremental tasks. In future research, we will focus on optimizing the timing of model growth and leveraging techniques such as neural architecture search to further enhance the model's adaptability and overall performance with new data.

Algorithm 3 Increasing Gradient Sparsity

Require: Parameter matrix W, threshold vector t, sequence of datasets $\{D_0, D_1, \ldots\}$, expansion size n, sparsity hyperparameter α

for dataset $D_k \in \{D_0, D_1, \ldots\}$ **do**
  for epoch **do**
    if final epochs **and** expansion phase **then**
      for layer in model **do**
        Create expanded parameter matrix $W_{exp} \in R^{(c_o+n, c_i+n)}$
        Initialize them and threshold $W_{exp}[c_o, :]$init
        Copy old weight $W_{exp}[: c_o, : c_i] = W$
      end for
    end if
    for training step t **do**
      Compute loss: $L = L(D; W_{exp}) + \alpha L_r$
      if $k == 0$ **then**
        Gradient decent with sparse parameter: $W^*_{exp}, t^*_{exp} = \arg\min[L(D; W_{exp}) + \alpha L_s]$
      else if $k > 0$ **then**
        Sparse gradient decent with sparse parameters: $W^*_{exp}, t^*_{exp} = \arg\min_{W,t} [L(D; W_{exp}) + \alpha L_s] \odot (1 - P)$
      end if
    end for
  end for
  Update preservation mask: $P = I\left(|W_{exp}| > 0\right)$
end for

Algorithm 4 Increasing Parameter Sparsity

Require: Parameter matrix W, sequence of datasets $\{D_0, D_1, \ldots\}$, threshold vector t, sparsity hyperparameter α

for dataset $D_k \in \{D_0, D_1, \ldots\}$ **do**
  for layer $\ell \in$ model **do**
    Initialize threshold $t \leftarrow t_0$
  end for
  for epoch **do**
    for training step t **do**
      for layer in model **do**
        Update sparsity masks: $M = I(|W| > t)$
        Update sparsified weights: $W = W \odot M$
      end for
      Compute loss: $L = L(D; W) + \alpha L_s$
    end for
  end for
end for

Table 6.2: Comprehensive performance evaluation of different continual learning methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), and forward knowledge transfer (FWT) on twelve domains of Permuted MNIST. Here we have three groups of comparison: 1) Simple baselines, and 2) SGD with model growth methods IDGrow, LatConn and LayerExp, 3) Baselines with LayerExp,.

Method	AAC↑	BWT↑	FWT↑
SGD	0.4556	-0.5739	0.0493
EWC	0.4816	-0.5454	0.0638
LwF	0.3976	-0.6387	0.0540
PackNet	0.5163	-0.5049	0.0659
PRE-DFKD	0.3941	-0.6392	0.0774
AdaptCL	0.5016	-0.2532	0.0865
SGD+LatConn	0.4474	-0.6799	0.0779
SGD+IDGrow	0.3822	-0.6532	0.1418
SGD+LayerExp	0.4632	-0.5658	0.0737
SGD+LayerExp+ODInit	0.4683	-0.5598	0.0742
EWC+LayerExp	0.4703	-0.5571	0.0597
LwF+LayerExp	0.3941	-0.6426	0.0680
PRE-DFKD+LayerExp	0.4064	-0.6248	0.0759
SparseGrow	**0.5256**	**-0.2066**	**0.0974**

Table 6.3: Performance evaluation of continual learning methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), and forward knowledge transfer (FWT) on the domain-incremental datasets of FreshStale and DomainNet.

Method	FreshStale			DomainNet		
	AAC↑	BWT↑	FWT↑	AAC↑	BWT↑	FWT↑
SGD+LayerExp	0.617	-0.450	0.027	0.464	-0.657	0.109
EWC+LayerExp	0.694	-0.360	0.039	0.463	-0.661	0.109
LwF+LayerExp	0.641	-0.424	0.009	0.434	-0.699	0.099
PRE-DFKD+LayerExp	0.703	-0.200	0.089	0.446	-0.475	0.090
SparseGrow	**0.737**	-0.273	0.049	**0.624**	**-0.283**	0.091

Table 6.4: Comparison of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and each dataset's test accuracy using different continual learning methods in the most challenging class-incremental setting.

Method	AAC↑	BWT↑	FWT↑	Test Accuracy↑			
				Class 0,1,2	2,3,4,5	0,3,6,7	6,8,9
SGD+LayerExp	0.309	-0.917	0.188	0.000	0.000	0.240	0.996
EWC+LayerExp	0.309	-0.917	0.188	0.000	0.000	0.240	0.996
LwF+LayerExp	0.309	-0.917	0.187	0.000	0.000	0.239	0.997
PRE-DFKD+LayerExp	0.330	-0.839	0.181	0.100	0.000	0.336	0.883
SparseGrow	**0.412**	**-0.623**	0.165	**0.279**	**0.081**	0.307	0.981

Table 6.5: verage accuracy comparison of continual learning methods (EWC, LwF, PackNet, PRE-DFKD, and AdaptCL) alongside different model growth techniques (IDGrow, LatConn, LayerExp) and several CL strategies incorporating LayerExp across an increasing number of observed domains.

Method	Test Accuracy of Number of Domains Learned											
	1	2	3	4	5	6	7	8	9	10	11	12
SGD	0.993	0.681	0.691	0.666	0.622	0.589	0.554	0.524	0.513	0.490	0.484	0.456
EWC	0.994	0.699	0.714	0.675	0.654	0.620	0.574	0.542	0.520	0.518	0.496	0.482
LwF	0.994	0.609	0.626	0.616	0.556	0.530	0.492	0.471	0.442	0.427	0.390	0.398
PackNet	0.992	0.554	0.621	0.579	0.579	0.565	0.546	0.530	0.512	0.511	0.514	0.516
PRE-DFKD	0.994	0.869	0.829	0.803	0.789	0.683	0.571	0.499	0.463	0.437	0.427	0.394
AdaptCL	0.993	0.969	0.924	0.872	0.809	0.728	0.639	0.619	0.584	0.556	0.525	0.502
SGD+IDGrow	0.993	0.520	0.556	0.577	0.512	0.468	0.443	0.438	0.416	0.405	0.412	0.382
SGD+LatConn	0.994	0.577	0.676	0.672	0.616	0.587	0.529	0.510	0.504	0.489	0.487	0.447
SGD+LayerExp	0.993	0.723	0.712	0.704	0.643	0.607	0.589	0.559	0.529	0.527	0.496	0.463
EWC+LayerExp	0.994	0.634	0.675	0.651	0.642	0.605	0.535	0.516	0.519	0.475	0.482	0.470
LwF+LayerExp	0.994	0.588	0.602	0.555	0.519	0.485	0.470	0.428	0.423	0.418	0.401	0.380
PRE-DFKD+LayerExp	0.994	0.826	0.780	0.776	0.758	0.636	0.522	0.463	0.417	0.417	0.412	0.406
SparseGrow	0.993	0.969	0.928	0.875	0.788	0.721	0.661	0.634	0.610	0.581	0.551	0.526

Table 6.6: Performance evaluation of continual learning methods in terms of average accuracy (AAC), backward knowledge transfer (BWT), forward knowledge transfer (FWT), and number of used parameters on the Large-Scale, Diverse Binary-Class Food Quality Dataset.

	AAC↑	BWT ↑	FWT ↑	Test Accuracy↑					
				Apple	Orange	Banana	Tomato	Gourd	Capsicum
SGD+LayerExp	0.617	-0.450	0.027	0.590	0.630	0.415	0.440	0.635	0.994
EWC+LayerExp	0.694	-0.360	0.039	0.555	0.445	0.760	0.555	0.850	1.000
LwF+LayerExp	0.641	-0.424	0.009	0.685	0.630	0.445	0.400	0.690	0.996
PRE-DFKD+LayerExp	0.703	-0.200	0.089	0.635	0.660	0.790	0.435	0.695	1.000
SparseGrow	0.737	-0.273	0.049	0.825	0.350	0.680	0.740	0.905	0.924

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

The objective of this thesis is to investigate the potential for enabling scalable continual learning in neural networks. A number of new challenges emerge. Firstly, the distribution shifts among sequential datasets cause catastrophic forgetting, which limits the accuracy that can be retained in continual learning. It is thus imperative to address the issue of catastrophic forgetting in order to enable scalable continual learning. Secondly, real-life data is inherently heterogeneous, with varying sizes, similarities and complexities across datasets. It is therefore essential for neural network models to demonstrate robustness against heterogeneous datasets in order to enable scalable continual learning. Finally, to scale up to incremental data, neural network models must enhance their long-term adaptability in the context of limited capacity and unlimited data amounts. Consequently, it is necessary to investigate model growth strategies that can enhance model long-term adaptability over incremental new data while maintaining appropriate knowledge retention.

In this thesis, I tackle these challenges, identify a new problem, and develop algorithms that facilitate scalable continual learning in neural networks. I then apply these

algorithms to a diverse range of real-world applications such as type incremental food freshness detection, generalization of object detection, and multi-domain adaptation of traffic prediction, as outlined below.

In particular, an overview of existing work on scalable continual learning is provided in Chapter 2. In this review, I provide an overview of existing taxonomies for addressing challenges related to catastrophic forgetting, bias, heterogeneity, and model growth strategies.

In Chapter 3, I put forward a memory-efficient approach for IoT devices to adapt incrementally to domain shifts and overcome catastrophic forgetting in a fixed neural network, which I have named E-DomainIL. The proposed method involves freezing the learned parameters and reusing them at a later stage of training, thus avoiding interference between different domains. E-DomainIL does not require task labels or storing masks, as it utilises all parameters during inference. Furthermore, data-driven pruning is employed to adjust the parameter ratio according to the dataset, thereby maintaining a balance between accuracy and parameter efficiency. In Chapter 4, I introduce FedDistill, a novel distillation framework for Continual non-IID Federated Learning that addresses the challenge of imbalanced local data distributions without requiring extra communications. In Chapter 5, I propose AdaptCL, a novel adaptive continual learning method for addressing heterogeneity in sequential datasets. AdaptCL employs fine-grained data-driven pruning to adapt to variations in data complexity and dataset size. It also utilises task-agnostic parameter isolation to mitigate the impact of varying degrees of catastrophic forgetting caused by differences in data similarity. Through a two-pronged case study approach, I evaluate AdaptCL on both datasets of MNIST Variants and DomainNet, as well as datasets from the specific realm of food quality. The latter includes both large-scale, diverse binary-class food datasets and few-shot, multi-class food datasets. Across all these scenarios, AdaptCL consistently exhibits robust performance, demonstrating its flexibility and general applicability in handling heterogeneous datasets. In Chapter 6, I identify the

issue of growth-induced forgetting (GIFt) and conduct an in-depth study on its root cause. Among model growth strategies, layer expansion stands out as a promising approach for mitigating GIFt, as it allows for widening layers without affecting model functionality. Nevertheless, the direct adoption of layer expansion also presents certain challenges. Furthermore, there is a lack of data-driven control and initialisation of expanded parameters, which would enable a more balanced approach to adaptability and knowledge retention. This chapter proposes sparse model growth (SparseGrow), which is designed to address the issue of GIFt while enhancing adaptability to new data. SparseGrow utilises data-driven sparse layer expansion to regulate the efficient utilisation of parameters during model growth, thereby reducing GIFt from excessive growth and functionality alterations. Furthermore, it combines sparse growth with on-data initialisation at the late-stage of training to create partially zero-valued expansions that align with the learned distribution, thus enhancing retention and adaptability. To further minimise forgetting, freezing is applied by calculating the sparse mask, allowing data-driven preservation of important parameters.

7.2 Future Directions

While most existing research on continual learning primarily focuses on catastrophic forgetting, the vision for continual learning should extend far beyond this limitation. Continual learning serves as a pathway towards achieving Artificial General Intelligence (AGI), drawing inspiration from the human brain's capacity for continuous learning. The goal is for neural networks to truly engage in ongoing learning: autonomously expanding their knowledge based on external information, flexibly adjusting the weights of old and new knowledge, and applying this knowledge across diverse domains.

Therefore, the challenges facing continual learning are more extensive than just catastrophic forgetting. Here are some promising and personally intriguing future direc-

tions that I have considered:

Self-Supervised Continual Learning of Pre-trained Large Language Models Large language models present a compelling scenario for continual learning. Existing large language models lack access to recent news updates, a significant drawback. Moreover, these models interact with users solely through prompts, losing track of previous interactions after refreshes, hindering personalization. This gives rise to two research scenarios: 1) Self-Supervised Continual Learning on News Data, and 2) Self-Supervised Continual Learning with Prompts. These scenarios pose several significant challenges: a) Data quality control and self-labeling: Current continual learning methods still rely on manual data input, even if they exhibit strong knowledge retention, making their learning appear less truly continual. A major breakthrough would involve greater autonomy in data quality control and self-labeling for new data. b) Knowledge preservation of pre-trained models: Conventional continual learning requires specific operations from the initial model, such as recording crucial parameters or sparse training for preserving space, which are impractical on pre-trained models, introducing additional challenges. c) Balancing the importance of old and new knowledge: The significance of old and new knowledge varies across different contexts. For instance, after training a language model, if we desire to refine its conversational etiquette, retaining the impact of new quality-controlled data becomes crucial. Conversely, when continuously updating a pre-trained model with news and prompts, preserving pre-trained knowledge takes precedence to prevent new data from overriding it. d) Change of data structure: Continuous learning news and prompt data formats may differ from training data formats, such as dialogues versus non-dialogue formats, presenting another challenge to overcome.

Additionally, continual learning can be applied in reinforcement learning to help an agent adapt to varied environmental settings. It can also be integrated with neural architecture search to optimize initial models and continually refine existing models based on new data.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940, 2020.

[2] Tameem Adel, Han Zhao, and Richard E Turner. Continual learning with adaptive weights (claw). *arXiv preprint arXiv:1911.09514*, 2019.

[3] Manal Aljahdali, Ahmed M Abdelmoniem, Marco Canini, et al. Flashback: Understanding and mitigating forgetting in federated learning. *arXiv preprint arXiv:2402.05558*, 2024.

[4] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[5] Michael L Anderson. Neural reuse: A fundamental organizational principle of the brain. *Behavioral and brain sciences*, 33(4):245–266, 2010.

[6] Randy Ardywibowo, Zepeng Huo, Zhangyang Wang, Bobak J Mortazavi, Shuai Huang, and Xiaoning Qian. Varigrow: Variational architecture growing for task-agnostic continual learning based on bayesian novelty. In *International Conference on Machine Learning*, pages 865–877. PMLR, 2022.

[7] Cem Arslan, Dogra Puneet, Ajanthan Thalaiyasingam, et al. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision (ECCV)*, pages 532–547. Springer, 2018.

[8] Yassine Benyahia, Kaicheng Yu, Kamil Bennani Smires, Martin Jaggi, Anthony C Davison, Mathieu Salzmann, and Claudiu Musat. Overcoming multi-model forgetting. In *International Conference on Machine Learning*, pages 594–603. PMLR, 2019.

[9] Kuluhan Binici, Shivam Aggarwal, Nam Trung Pham, Karianto Leman, and Tulika Mitra. Robust and resource-efficient data-free knowledge distillation by generative pseudo replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6089–6096, 2022.

[10] Léon Bottou et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[11] Ajesh Koyatan Chathoth, Abhyuday Jagannatha, and Stephen Lee. Federated intrusion detection for iot with heterogeneous cohort privacy. *arXiv preprint arXiv:2101.09878*, 2021.

[12] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[13] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[14] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

[15] Zhixuan Chu, Ruopeng Li, Stephen Rathbun, and Sheng Li. Continual causal inference with incremental observational data. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 3430–3439. IEEE, 2023.

[16] Ronald L Davis and Yi Zhong. The biology of forgetting—a perspective. *Neuron*, 95(3):490–503, 2017.

[17] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[18] Wei Deng, James B Aimone, and Fred H Gage. New neurons and new memories: how does adult hippocampal neurogenesis affect learning and memory? *Nature reviews neuroscience*, 11(5):339–350, 2010.

[19] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[20] Binzong Geng, Fajie Yuan, Qiancheng Xu, Ying Shen, Ruifeng Xu, and Min Yang. Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 517–523, Online, August 2021. Association for Computational Linguistics.

[21] Bishwas Ghimire and Danda B Rawat. Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things. *IEEE Internet of Things Journal*, 9(11):8229–8249, 2022.

[22] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019.

[23] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[24] Yuchen Guo, Xueyang Tang, and Tao Lin. Fedbr: Improving federated learning on heterogeneous data via local learning bias reduction. In *International Conference on Machine Learning*, pages 12034–12054. PMLR, 2023.

[25] Peng Han, Xiaoyang Shi, and Jian Huang. Fedal: Black-box federated knowledge distillation enabled by adversarial learning. *arXiv preprint arXiv:2311.16584*, 2023.

[26] LIU Hanmo, DI Shimin, LI Haoyang, LI Shuangyin, CHEN Lei, and ZHOU Xiaofang. Effective data selection and replay for unsupervised continual learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1449–1463. IEEE, 2024.

[27] Jiangpeng He and Fengqing Zhu. Online continual learning for visual food classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2337–2346, 2021.

[28] Jiangpeng He and Fengqing Zhu. Exemplar-free online continual learning. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 541–545. IEEE, 2022.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Wenpeng Hu, Qi Qin, Mengyu Wang, Jinwen Ma, and Bing Liu. Continual learning by using information of each class holistically. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7797–7805, 2021.

[31] Zhiyuan Hu, Yunsheng Li, Jiancheng Lyu, Dashan Gao, and Nuno Vasconcelos. Dense network expansion for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11858–11867, 2023.

[32] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[33] Ferenc Huszár. On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*, 2017.

[34] Xiaozhou Ji, Jing Tian, Haozhao Zhang, et al. Joint device selection and bandwidth allocation for cost-efficient federated learning in industrial internet of things. *IEEE Internet of Things Journal*, 10(10):9148–9160, 2023.

[35] Michael V Johnston. Plasticity in the developing brain: implications for rehabilitation. *Developmental disabilities research reviews*, 15(2):94–101, 2009.

[36] Richard Joseph, Naren Khatwani, Rahul Sohandani, Raghav Potdar, and Adithya Shrivastava. Food aayush: Identification of food and oils quality. In *Integrated Emerging Methods of Artificial Intelligence & Cloud Computing*, pages 71–78. Springer, 2021.

[37] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, et al. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[38] Zixuan Ke, Bing Liu, and Xingchang Huang. Continual learning of a mixed sequence of similar and dissimilar tasks. *Advances in Neural Information Processing Systems*, 33:18493–18504, 2020.

[39] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.

[40] Geeho Kim, Jinkyu Kim, and Bohyung Han. Fedacg: Federated learning with adaptive client selection and global model update. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12385–12394. IEEE, 2024.

[41] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[42] Jedrzej Kozal and Michal Wozniak. Increasing depth of neural networks for life-long learning. *Information Fusion*, 98:101829, 2023.

[43] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[44] Gihun Lee, Minchan Jeong, Youngbin Shin, et al. Preservation of the global knowledge by not-true distillation in federated learning. *Advances in Neural Information Processing Systems*, 35:38461–38474, 2022.

[45] Soobee Lee, Minindu Weerakoon, Jonghyun Choi, Minjia Zhang, Di Wang, and Myeongjae Jeon. Carousel memory: Rethinking the design of episodic memory for continual learning. *arXiv preprint arXiv:2110.07276*, 2021.

[46] Honglin Li, Payam Barnaghi, Shirin Enshaeifar, and Frieder Ganz. Continual learning using bayesian neural networks. *IEEE transactions on neural networks and learning systems*, 32(9):4243–4252, 2020.

[47] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022.

[48] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10713–10722. IEEE, 2021.

[49] Tian Li, Anit Kumar Sahu, Manzil Zaheer, et al. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

[50] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934. PMLR, 2019.

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

[52] Tao Lin, Lingjing Kong, Sebastian U Stich, et al. Ensemble distillation for robust model fusion in federated learning. *arXiv preprint arXiv:2006.07242*, 2020.

[53] Jianzhao Liu, Wei Zhou, Xin Li, Jiahua Xu, and Zhibo Chen. Liqa: Lifelong blind image quality assessment. *IEEE Transactions on Multimedia*, pages 1–16, 2022.

[54] Junjie Liu, Zhe Xu, Runbin Shi, Ray CC Cheung, and Hayden KH So. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. *arXiv preprint arXiv:2005.06870*, 2020.

[55] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

[56] Zhilin Lu, Hongyu Pan, Yiqin Dai, et al. Federated learning with non-iid data: A survey. *IEEE Internet of Things Journal*, 11(11):19188–19209, 2024.

[57] Min Luo, Fei Chen, Dapeng Hu, et al. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

[58] Rui Ma, Qingbo Wu, King Ngi Ngan, Hongliang Li, Fanman Meng, and Linfeng Xu. Forgetting to remember: A scalable incremental learning framework for cross-task blind image quality assessment. *IEEE Transactions on Multimedia*, pages 1–12, 2023.

[59] Divyam Madaan, Hongxu Yin, Wonmin Byeon, Jan Kautz, and Pavlo Molchanov. Heterogeneous continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15985–15995, 2023.

[60] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[61] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[62] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[63] Brendan McMahan, Eider Moore, Daniel Ramage, et al. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[64] Matias Mendieta, Taojiannan Yang, Pu Wang, et al. Local learning matters: Rethinking data heterogeneity in federated learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8387–8396. IEEE, 2022.

[65] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide neural networks forget less catastrophically. In *International conference on machine learning*, pages 15699–15717. PMLR, 2022.

[66] Alexander Ororbia, Ankur Mali, C Lee Giles, and Daniel Kifer. Continual learning of recurrent neural networks by locally aligning distributed representations. *IEEE Transactions on Neural Networks and Learning Systems*, 31(10):4267–4278, 2020.

[67] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11321–11329, 2019.

[68] Gyeong-Moon Park, Sahng-Min Yoo, and Jong-Hwan Kim. Convolutional neural network with developmental memory for continual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 32(6):2691–2705, 2020.

[69] Alvaro Pascual-Leone, Amir Amedi, Felipe Fregni, and Lotfi B Merabet. The plastic human brain cortex. *Annu. Rev. Neurosci.*, 28:377–401, 2005.

[70] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[71] Automatic Differentiation In Pytorch. Pytorch, 2018.

[72] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for incremental learning. *Advances in Neural Information Processing Systems*, 2019.

[73] Ashish Rauniyar et al. Federated learning for medical applications: A taxonomy, current trends, challenges, and future research directions. *IEEE Internet of Things Journal*, 11(5):7374–7398, 2024.

[74] Leonardo Ravaglia, Manuele Rusci, Davide Nadalini, Alessandro Capotondi, Francesco Conti, and Luca Benini. A tinyml platform for on-device continual learning with quantized latent replays. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(4):789–802, 2021.

[75] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[76] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[77] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[78] Andrea Rosasco, Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, and Davide Bacciu. Distilled replay: Overcoming forgetting through synthetic samples. In *Continual Semi-Supervised Learning: First International Workshop, CSSL 2021, Virtual Event, August 19–20, 2021, Revised Selected Papers*, pages 104–117. Springer, 2022.

[79] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):651–663, 2018.

[80] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[81] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[82] Hyejun Seo, Jihong Park, Seungeun Oh, et al. Federated knowledge distillation. In *Machine Learning and Wireless Communications*, page 457. Cambridge University Press, 2022.

[83] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[84] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9630–9638, 2021.

[85] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[86] Neta Shoham, Tomer Avidor, Aviv Keren, et al. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

[87] Changlin Song, Divya Saxena, Jiannong Cao, and Yuqing Zhao. Feddistill: Global model distillation for local model de-biasing in non-iid federated learning. *arXiv preprint arXiv:2404.09210*, 2024.

[88] Ge Song and Xiaoyang Tan. Real-world cross-modal retrieval via sequential learning. *IEEE Transactions on Multimedia*, PP:1–1, 06 2020.

[89] Jingyuan Sun, Shaonan Wang, Jiajun Zhang, and Chengqing Zong. Distill and replay for continual language learning. In *Proceedings of the 28th international conference on computational linguistics*, pages 3569–3579, 2020.

[90] Jian Tang, Xiaomin Ding, Dan Hu, et al. Fedrad: Heterogeneous federated learning via relational adaptive distillation. *Sensors*, 23(14):6518, 2023.

[91] Selvarajah Thuseethan, Sutharshan Rajasegarar, and John Yearwood. Deep continual learning for emerging emotion recognition. *IEEE Transactions on Multimedia*, 24:4367–4380, 2022.

[92] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, et al. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[93] Liyuan Wang, Bo Lei, Qian Li, Hang Su, Jun Zhu, and Yi Zhong. Triple-memory networks: A brain-inspired method for continual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[94] Wei Wen, Feng Yan, Yiran Chen, and Hai Li. Autogrow: Automatic layer growing in deep convolutional networks. In *Proceedings of the 26th ACM SIGKDD*

International Conference on Knowledge Discovery & Data Mining, pages 833–841, 2020.

[95] Chen Xu, Zonghang Hong, Mingkai Huang, et al. Acceleration of federated learning with alleviated forgetting in local training. *arXiv preprint arXiv:2203.02645*, 2022.

[96] Ju Xu and Zhanxing Zhu. Reinforced continual learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[97] Zhe Xu and Ray CC Cheung. Accurate and compact convolutional neural networks with trained binarization. *arXiv preprint arXiv:1909.11366*, 2019.

[98] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3014–3023, 2021.

[99] Yuguang Yan, Chen Ming Feng, Mengting Ye, et al. Rethinking client drift in federated learning: A logit perspective. *arXiv preprint arXiv:2308.10162*, 2023.

[100] Guanglei Yang, Enrico Fini, Dan Xu, Paolo Rota, Mingli Ding, Tang Hao, Xavier Alameda-Pineda, and Elisa Ricci. Continual attentive fusion for incremental learning in semantic segmentation. *IEEE Transactions on Multimedia*, pages 1–1, 2022.

[101] Li Yang, Sen Lin, Junshan Zhang, and Deliang Fan. Grown: Grow only when necessary for continual learning. *arXiv preprint arXiv:2110.00908*, 2021.

[102] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[103] Felix Yu, Wenchang Zhang, Zongsheng Qin, et al. Fed2: Feature-aligned federated learning. In *27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2066–2074. ACM, 2021.

[104] Xin Yuan, Pedro Henrique Pamplona Savarese, and Michael Maire. Growing efficient deep networks by structured continuous sparsification. In *International Conference on Learning Representations*, 2021.

[105] Jie Zhang, Junting Zhang, Shalini Ghosh, Dawei Li, Jingwen Zhu, Heming Zhang, and Yalin Wang. Regularize, expand and compress: Nonexpansive continual learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 854–862, 2020.

[106] Lan Zhang, Yuguang Luo, Yan Bai, et al. Federated learning for non-iid data via unified feature learning and optimization objective alignment. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4400–4408. IEEE, 2021.

[107] Lei Zhang, Xiang Lei, Yang Shi, et al. Federated learning for iot devices with domain generalization. *IEEE Internet of Things Journal*, 10(11):9622–9633, 2023.

[108] Lin Zhang, Li Shen, Liang Ding, et al. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. *arXiv preprint arXiv:2203.09249*, 2022.

[109] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven Su. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 7809–7818, 2020.

[110] Borui Zhao, Quan Cui, Renjie Song, et al. Decoupled knowledge distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11953–11962. IEEE, 2022.

[111] Hanbin Zhao, Hui Wang, Yongjian Fu, Fei Wu, and Xi Li. Memory-efficient class-incremental learning for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5966–5977, 2021.

[112] Yue Zhao, Meng Li, Li Lai, et al. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[113] Yuqing Zhao, Divya Saxena, and Jiannong Cao. Memory-efficient domain incremental learning for internet of things. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, pages 1175–1181, 2022.

[114] Yuqing Zhao, Divya Saxena, and Jiannong Cao. Adaptcl: Adaptive continual learning for tackling heterogeneity in sequential datasets. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[115] Yuqing Zhao, Divya Saxena, Jiannong Cao, Xiaoyun Liu, and Changlin Song. Overcoming growth-induced forgetting in task-agnostic continual learning. *arXiv preprint arXiv:2408.10566*, 2024.