

N° d’ordre NNT : 2025ECDL0018

THESE DE DOCTORAT DE L’ECOLE CENTRALE DE LYON
membre de l’Université de Lyon

Ecole Doctorale N°512
InfoMaths - Informatique et Mathématiques de Lyon

Spécialité de doctorat : Informatique

Soutenue publiquement le 16 juin 2025, par :

Rui YANG

Apprentissage continu adaptatif et efficace dans des environnements dynamiques

---

Devant le jury composé de :

Thierry CHATEAU, Professeur, Institut Pascal de l'Université Clermont Auvergne	Président
Céline HUDELOT, Professeure des universités, Mathématiques et Informatique pour la Complexité et les Systèmes (MICS)	Rapporteure
Ngoc Son VU, Maître de conférences HDR, Laboratoire Traitement de l'Information et des Systèmes à l'ENSEA (ETIS)	Rapporteur
Amaury HABRARD, Professeur, University Jean Monnet of Saint-Etienne	Examinateur
Céline TEULIÈRE, Maîtresse de conférences, Institut Pascal de l'Université Clermont Auvergne	Examinatrice
Liming CHEN, Professeur, LIRIS, Ecole Centrale de Lyon	Directeur de thèse
Emmanuel DELLANDRÉA, Maître de conférences HDR, LIRIS, Ecole Centrale de Lyon	Co-directeur de thèse
Matthieu GRARD, Docteur, Siléane	invité

Unité de recherche : LIRIS - Laboratoire d'InfoRmatique en Image et Systèmes d'information

Abstract

As data in real-world applications continuously evolve, the ability of artificial intelligence systems to learn incrementally while preserving previously acquired knowledge has become increasingly critical. However, deploying continual learning (CL) methods in practice is impeded by blurred task boundaries, severe data imbalance, and the high computational demands and data privacy concerns associated with large models. This thesis addresses these challenges through three core contributions, thus enhancing the feasibility and robustness of CL in dynamic environments. First, to manage blurred task boundaries, where data distributions often overlap, we propose a novel Distribution-Shift Incremental Learning (DS-IL) scenario. In this framework, an entropy-guided learning approach effectively leverages these overlaps to mitigate catastrophic forgetting without maintaining large memory buffers. In real-world scenarios, data imbalance is a common challenge that can significantly hinder the performance of learning systems. To address this issue, our second contribution introduces a Memory Selection and Contrastive Learning (MSCL) strategy. By actively sampling representative instances and coupling them with current data in a contrastive loss, the model better balances underrepresented classes and domains. This approach not only preserves crucial historical information but also maintains robust performance under significantly skewed data distributions. Finally, to alleviate the computational overhead of continually training diffusion models, particularly relevant in scenarios with data privacy constraints or prohibitive storage costs, we introduce a Multi-Mode Adaptive Generative Distillation (MAGD) framework. Using generative distillation, noisy intermediate representations, and exponential moving averages, this method enables efficient continual updates while preserving high-quality image generation and classification performance. Collectively, these contributions form a comprehensive framework for scalable, memory-efficient, and computationally tractable continual learning. Through effective knowledge retention, dynamic adaptation to imbalanced data, and resource-efficient generative replay, this thesis expands the applicability of CL methods to a wider range of real-world settings.
Keywords: computer vision, continual learning, knowledge distillation, deep learning

Résumé

Face à l'évolution continue des données dans les applications du monde réel, la capacité des systèmes d'intelligence artificielle à apprendre de manière incrémentale tout en préservant les connaissances acquises précédemment est devenue de plus en plus critique. Cependant, le déploiement des méthodes d'apprentissage continu (CL) dans la pratique est entravé par des frontières de tâches floues, un déséquilibre sévère des données, et les fortes exigences computationnelles et les préoccupations liées à la confidentialité des données associées aux grands modèles. Cette thèse aborde ces défis à travers trois contributions principales, améliorant ainsi la faisabilité et la robustesse du CL dans des environnements dynamiques. Premièrement, pour gérer les frontières de tâches floues, où les distributions de données se chevauchent souvent, nous proposons un nouveau scénario d'Apprentissage Incrémental du Changement de Distribution (DS-IL). Dans ce cadre, une approche d'apprentissage guidée par l'entropie exploite efficacement ces chevauchements pour atténuer l'oubli catastrophique sans maintenir de grands tampons de mémoire. Dans les scénarios réels, le déséquilibre des données est un défi commun qui peut entraver significativement la performance des systèmes d'apprentissage. Pour aborder ce problème, notre deuxième contribution introduit une stratégie de Sélection de Mémoire et d'Apprentissage Contrastif (MSCL). En échantillonnant activement des instances représentatives et en les couplant avec des données actuelles dans une perte contrastive, le modèle équilibre mieux les classes et les domaines sous-représentés. Cette approche préserve non seulement des informations historiques cruciales, mais maintient également une performance robuste sous des distributions de données considérablement biaisées. Enfin, pour alléger la charge computationnelle de la formation continue des modèles de diffusion, particulièrement pertinente dans des scénarios avec des contraintes de confidentialité des données ou des coûts de stockage prohibitifs, nous introduisons un cadre de Distillation Générative Adaptative Multi-Mode (MAGD). Utilisant la distillation générative, des représentations intermédiaires bruyantes, et des moyennes mobiles exponentielles, cette méthode permet des mises à jour continues efficaces tout en préservant une haute qualité de génération d'images et de performance de classification. Collectivement, ces contributions forment un cadre complet pour un

apprentissage continu, évolutif, efficace en mémoire et gérable computationnellement. À travers une rétention de connaissances efficace, une adaptation dynamique à des données déséquilibrées et une relecture générative efficiente en ressources, cette thèse étend l'applicabilité des méthodes de CL à un plus large éventail de paramètres du monde réel.
Mots clés: vision d'ordinateur, apprendissage d'incrémentale, distillation des connaissances, apprendissage en profondeur

Acknowledgements

I owe a profound debt of gratitude to my thesis director, Prof. Liming Chen, whose mentorship was invaluable to my studies and research. Prof. Chen not only guided me expertly through my PhD journey but also invested significant time and effort in improving my papers and sharpening my analytical skills. His dedication to excellence and his deep commitment to my academic and personal growth have left an indelible mark on my career.

I am equally thankful to HDR Emmanuel Dellandréa, whose encouragement and support have been pivotal in my research endeavors. His enthusiasm and deep knowledge consistently inspired me to push the boundaries of my work and strive for rigor and innovation in my field.

Special thanks are also due to PhD Matthieu Grard, whose insights and assistance were greatly appreciated throughout my research. His readiness to help and his meticulous attention to detail have significantly contributed to the success of my projects.

My heartfelt thanks also go to my wife, Yang Hao, for her enduring patience and companionship in France. Her presence has been a beacon of joy and comfort in my life. Additionally, I must express my deepest appreciation to my family, especially my mother, whose love and sacrifices have been the foundation of my successes. Their unwavering support has empowered me to pursue my dreams relentlessly.

Contents

Abstract iii

Résumé v

Acknowledgements vii

List of Acronyms xv

List of Figures xix

List of Tables xxiii

1 Introduction 1
- 1.1 Continual Learning . . . 3
- 1.2 Applications . . . 4
- 1.3 Contributions . . . 4
- 1.4 Publications . . . 6
- 1.5 Outline . . . 7

2 Literature Review 9
- 2.1 Deep Neural Networks . . . 9
  - 2.1.1 Offline Training . . . 9
- 2.2 Different Learning Paradigms . . . 10
- 2.3 Continual Learning . . . 12
  - 2.3.1 Continual Learning Scenarios . . . 13
  - 2.3.2 Datasets . . . 15
  - 2.3.3 Evaluation Metrics . . . 17
- 2.4 Continual Learning Methods . . . 18
  - 2.4.1 Regularization-based Methods . . . 18
  - 2.4.2 Architecture-Based Methods . . . 22
  - 2.4.3 Memory-based Methods . . . 23
- 2.5 Diffusion Models in Continual Learning . . . 34

2.5.1 Denoising Diffusion Probabilistic Models (DDPM) 34
2.5.2 Continual Learning Methods 36
2.6 Positioning 37

3 Continual Learning with Blurred Task Boundaries 41
3.1 Motivation 41
3.2 Distribution-Shift Incremental Learning (DS-IL) 42
3.2.1 From CIL to DS-IL 43
3.2.2 From DIL to DS-IL 44
3.3 Entropy-Guided Self-Regulated Learning without Forgetting 44
3.3.1 LWF (Learning without Forgetting) 45
3.3.2 The proposed ER-LWF approach 45
3.4 Experiments on Academic Datasets 47
3.4.1 Experimental protocols 47
3.4.2 Training Methods 50
3.4.3 Results 51
3.5 Experiments on FairWastes Dataset 53
3.5.1 FairWastes Dataset 53
3.5.2 Scenario 55
3.5.3 Results 55
3.6 Limitations 57
3.7 Conclusion 57

4 Online Continual Learning in both Balanced and Imbalanced Data Environments 59
4.1 Motivation 59
4.2 Preliminary and problem statement 60
4.3 Methodology 61
4.3.1 Feature-Distance based sample selection 61
4.3.2 Contrastive learning for better discriminative feature representation 65
4.4 Experiments and Results 67
4.4.1 Balanced benchmarks 67
4.4.2 Imbalanced benchmarks 68
4.4.3 Baselines and implementation details 69
4.4.4 Results on balanced benchmarks 70
4.4.5 Results on imbalanced scenarios 70
4.5 Experiments on FairWastes Dataset 71
4.5.1 Scenarios 71
4.5.2 Results 71
4.6 Ablation Study and Extensive Experiments 72

4.6.1 Ablation study . . . 72
4.6.2 The impact of σ in RBF kernel . . . 73
4.6.3 Running Time . . . 74
4.6.4 The impact of memory size . . . 74
4.6.5 Comprehensive Evaluation of Average Forgetting . . . 75
4.6.6 The distribution of our memory set . . . 76
4.6.7 Collaborative Learning with other memory-based methods . . . 77
4.6.8 Results on Balanced class-incremental learning scenario . . . 78
4.7 Conclusion . . . 78

5 Online Continual Learning of Diffusion Models **81**
5.1 Motivation . . . 81
5.2 Problem formulation . . . 83
5.3 Methodology . . . 84
5.3.1 Generative replay and Generative distillation . . . 84
5.3.2 Noisy Intermediate Generative Distillation (**NIGD**) . . . 85
5.3.3 SNR-Guided Generative Distillation (**SGGD**) . . . 88
5.3.4 EMA in Online Continual Learning . . . 89
5.3.5 Workflow and overall objective . . . 90
5.4 Experiments and Results . . . 91
5.4.1 Datasets . . . 91
5.4.2 Evaluation metrics and Methods . . . 91
5.4.3 Overall results . . . 92
5.5 Ablation Study and Extensive Experiments . . . 93
5.6 Ablation Study . . . 93
5.6.1 The Influence of Generation Steps . . . 94
5.6.2 Results Across Tasks . . . 94
5.7 Conclusion . . . 95

6 Conclusions and Perspectives **97**

A Continual Learning with Blurred Task Boundaries **101**
A.1 Details of the experimental protocol . . . 101
A.1.1 PACS dataset . . . 101
A.1.2 Cifar100 dataset . . . 101

B Online Continual Learning of Diffusion Models **107**
B.1 Demonstration . . . 107
B.2 Algorithm . . . 108

List of Acronyms

AA Average Accuracy

AF AverageForgetting

AI Artificial Intelligence

AIA Computer Vision

ASER Adversarial Shapley Value Experience Replay

BBCL Blurred Boundary Continual Learning

BiC Bias Correction

BWT Backward Transfer

CBRS Class-balancing reservoir sampling

CIL Class-Incremental Learning

CoPE Continual Prototype Evolution

CV Computer Vision

CV Online Continual Learning

DDGR Diffusion-based Generative Replay

DER Dynamically Expandable Representation

DiffClass Diffusion-Based Class Incremental Learning

DIL Domain-Incremental Learning

DS-IL Distribution-Shift Incremental Learning

DyTox Dynamic Token Expansion

EMA Exponential Moving Average

ER Experience Replay

ER-ACE Asymmetric Cross-Entropy

ER-LWF Entropy-Guided Self-Regulated Learning with- out Forgetting

EWC Elastic Weight Consolidation

FDBS Feature-Distance based sample selection

FID Fréchet Inception Distance

GD Generative Distillation

GR Generative Replay

GSA Gradient Self-Adaptation

GSS Gradient-based sample selection

GUIDE Guidance-based Incremental Learning with Diffusion Models

iCaRL Incremental Classifier and Representation Learning

ICL Imbalanced Continual Learning

Imb C-DIL Imbalanced Class and Domain-Incremental Learning

Imb CIL Imbalanced Class-Incremental Learning

Imb DIL Imbalanced Domain-Incremental Learning

KD Knowledge Distillation

KLD Kullback-Leibler Divergence

LWF Learning Without Forgetting

MAS Memory Aware Synapses

MIR Maximally Interfered Retrieval

ML Machine Learning

MSCL Memory Selection and Contrastive Representation Learning

NCM Nearest-Class-Mean Classifier

NIGD Noisy Intermediate Generative Distillation

OCM Online Continual Learning through Mutual Information Maximization

OCS Online Corset Selection

ODDL Online Discrepancy Distance Learning

OnPro Online Prototype Learning for Online Continual Learning

PASS Prototype Augmentation and Self-Supervision

PNNs Progressive Neural Networks

PODNet Pooled Outputs Distillation

PRS Partitioning Reservoir Sampling

RBF Radial Basis Function

RM Rainbow Memory

SCL Supervised Con- trastive Learning Loss

SCR Supervised Contrastive Replay

SDDR Stable Diffusion for Distillation and Replay

SGGD Signal-Guided Generative Distillation

SI Synaptic Intelligence

TFCL Task-Free Continual Learning

TIL Task-Incremental Learning

VCL Variational Continual Learning

WA Weight Alignement

List of Figures

1.1 The main settings of different learning paradigms are illustrated in the figure, inspired by [58]. 8

2.1 Illustration of three scenarios: Task-Incremental Learning (TIL), Class-Incremental Learning (CIL), and Domain-Incremental Learning (DIL). The main difference between TIL and CIL is that in CIL, the task ID is not provided during testing, making CIL more challenging. DIL focuses on the distribution shift of the input data, while the output labels remain unchanged. 14

2.2 In the general framework of memory-based methods, for each task k, we work with the current dataset D_k and the preceding memory set M_{k-1}. These are combined to update our model. After the update, we derive a new memory set M_k, which is then utilized for the next task. 24

3.1 Illustration of TIL in which the data have both class id and task id. Moreover, task id is provided for both training and testing. 44

3.2 Illustration of CIL and Smooth-CIL using three tasks with three classes. In the original CIL setting, each task only contains instances of one class. The Smooth-CIL breaks this limit, and each task may have multiple classes. Thus, the shift of $p(y)$ in our scenario is smoother. 45

3.3 Assume there are two domains, and each domain contains the same class labels. The classic DIL considers each domain as a single task. In Smooth-DIL, each task can contain two domains, but the distribution of different domains is different for each task. Therefore, from DIL to CIL, we can label data by using both the domain label and the class label. In this figure, there are four different labels, corresponding to four tasks. 46

3.4 Samples from the four domains of PACS dataset[31] 49

3.5 An example for the superclass *fish* of CIFAR-100 dataset [10], and it has five subclasses like: *aquarium fish*, *flatfish*, *ray*, *shark*, *trout*. In our experiments, each subclass corresponds to a domain. 50

3.6 Results related to the PACS dataset[31]. Testing accuracy (average on three runs) is provided after training on each task for different methods and different scenarios. 52

3.7 Results related to the CIFAR-100 dataset[10].Testing accuracy (average over three runs) is provided after training on each task for different methods and different scenarios. 53

3.8 Three images from three different sensors respectively 54

3.9 Cropping the original images for Classification purpose 54

3.10 The distribution of Dataset based on the different scenes 56

4.1 We illustrate domains using colors and categories with shapes. It shows models adapting to datasets with high inter-class similarity and intra-class variance, highlighting the challenge of differentiating closely related categories. 62

4.2 We illustrate domains using colors and categories with shapes.Our proposed MSCL involves mapping input data and a memory set into a shared feature space. Here, $\boldsymbol{D}_{i,j}$ represents the distance between input data x_i and data x_j in the memory set. We use the same indexing convention for other formulas. We calculate distances, $\boldsymbol{D}$ and $\boldsymbol{a}$, between input data and memory set, and then derive an importance weight matrix quantifying each input data representative importance w.r.t those in the memory set based on the analysis of their intra-class diversity or inter-class similarity in the feature space. These importance weights are combined with random selection to give birth to our Feature-Distance based Sample Selection (FDBS) which identifies the most representative input data points for storage into the memory set. Armed with this importance weight matrix, we proceed to craft a novel Contrastive Loss (SCL) aimed at refining the feature space by compacting intra-class data and creating greater separation among inter-class data. 63

4.3 RBF kernel values with different σ 73

4.4 Running Time of different methods on Blanced CIFAR-100. 75

4.5 The ratio of different domains within the memory set compared to the original scenario. 77

5.1 Illustration of Our Method. The **yellow region** represents **SGGD**, the **blue region** denotes **NIGD**, and the **red region** corresponds to training on the current batch $\boldsymbol{B}^k$. $\epsilon_{\theta_t^{k-1}}$ is our EMA-teacher, and ϵ_{θ^k} is our current model. 84
5.2 An illustration of the DDIM denoising process with S steps shows two approaches: using $S - i$ steps to directly generate $\boldsymbol{x}_{\tau_i}$, or first generating the original images $\boldsymbol{x}_0$, followed by adding noise to produce the noisy image $\hat{\boldsymbol{x}}_{\tau_i}$ at step τ_i. 86
5.3 Evaluation of $\boldsymbol{r}_i$ with 20 generation steps 86
5.4 $logSNR$ Across Time Steps in Fashion-MNIST and CIFAR-10 . . . 88
5.5 Evaluation of FID Score and KLD Across Tasks for Different Methods on Fashion-MNIST and CIFAR-10 (unconditional model) 95

A.1 Domain Distribution Across Different Tasks in the Domain Incremental Learning (DIL) Scenario on PACS. 102
A.2 Domain Distribution Across Different Tasks in the smooth Domain Incremental Learning (smooth-DIL) Scenario on PACS. 103
A.3 Class Distribution Across Different Tasks in the Class Incremental Learning (CIL) Scenario on PACS. 103
A.4 Class Distribution Across Different Tasks in the smooth Class Incremental Learning (smooth-CIL) Scenario on PACS. 103
A.5 Domain Distribution Across Different Tasks in the Domain Incremental Learning (DIL) Scenario on CIFAR-100. 104
A.6 Domain Distribution Across Different Tasks in the smooth Domain Incremental Learning (smooth-DIL) Scenario on CIFAR-100. 104
A.7 Class Distribution Across Different Tasks in the Class Incremental Learning (CIL) Scenario on CIFAR-100. 104
A.8 Class Distribution Across Different Tasks in the smooth Class Incremental Learning (smooth-CIL) Scenario on CIFAR-100. 105

List of Tables

2.1 Overall description of different continual learning scenarios. × means no, ✓ means yes, and ⋆ means optional. 15

2.2 Illustration of Evaluation Metrics in Continual Learning 17

2.3 Summary of recently proposed continual learning methods, categorized by settings, mechanisms, and techniques employed. In the table, "TIL," "CIL," and "OCL" correspond to the scenarios of task-incremental learning, class-incremental learning, and online continual learning, respectively. "Reg," "Mem," and "PI" represent three categories of continual learning methods: regularization-based, memory-based, and parameter isolation (architecture-based). The techniques include "DA" for data augmentation; "ConL" for contrastive learning loss; "KD" for knowledge distillation; "Gene" for generative replay; "PP" for post-processing; "EMA" for exponential moving average; and "Pt" for prototype-based methods. . . 33

3.1 Results on DIL, Smooth-DIL, CIL, and Smooth for the PACS dataset[31]. **AA** corresponds to Average Accuracy as Equation (2.11), while **BWT** stands for Backward Transfer defined in Equation (2.14).For both of them, the bigger, the better. The budget of memory-based methods is set at 400 images in total. The value here is the percentage and is the average accuracy of the model over all experiments after training on all tasks. We display the best without-memory method in **bold** font. Methods with an asterisk * use memory. 51

3.2 Results on the CIFAR-100 dataset[10]. The budget for memory-based methods is set at 1000. It uses the same setting as Table 3.1. 53

3.3 Original DataSet 55

3.4 Revised Dataset 55

3.5 Results on the FairWastes dataset. The memory budget for memory-based methods is set to 400. The same methods as in Table 3.1 are used. We report the final classification accuracy, averaged over three runs. 56

4.1 We report the results of our experiments conducted on **balanced** scenarios. We present the average accuracy(AA) as mean and standard deviation over five independent runs. 69
4.2 Results on our **imbalanced** scenarios. We present the average accuracy(AA) as mean and standard deviation over five independent runs. For PACS, the memory size was set to 1000, while for all other scenarios, the memory size was set to 5000. 69
4.3 Comparison of different methods in Domain-Class Inc and Class-Inc scenarios on FairWastes Dataset 72
4.4 Ablation studies on balanced CIFAR-100 and imbalanced DomainNet. We set the memory size to 5000. 73
4.5 Results of varing σ and τ on Blanaced CIFAR-100 with a memory size of 2000. 74
4.6 Comparison of different memory selection methods on Imb C-DIL PACS for three different memory sizes. We present the final accuracy as mean and standard deviation over five independent runs . . 75
4.7 Comparison average forgetting(AF) of different methods on balanced CIFAR-100, Imbalanced CIFAR-100, and Imbalanced class-domain PACS. We present the average accuracy(AF) as mean and standard deviation over five independent runs 76
4.8 Comparison of Memory Set Composition Across Methods in Imbalanced Domain-Incremental Learning (imb DIL) Scenario of PACS. We set the memory size as 1000. 77
4.9 Combining FDBS with Other Memory-Based Methods: Experiments on Balanced Split CIFAR-100 (Memory Size: 5000) and Imbalanced Class-Domain Incremental Learning on PACS (Memory Size: 1000).The final accuracy was presented as the mean and standard deviation over five independent runs. 78
4.10 Results for classic class-incremental learning on CIFAR-100. Results marked with '*' are obtained directly from [118]. The memory size is set to 2000. 79

5.1 Results Presented as Mean and Standard Deviation Over 5 Random Runs, with unconditional diffusion model 93
5.2 Results Presented as Mean and Standard Deviation Over 5 Random Runs, with class-conditioned diffusion model 93

5.3 Ablation Study on Fashion-MNIST 94
5.4 FID and KLD on CIFAR-10 Across Different Generation Steps . . . 94

Chapter 1

Introduction

Deep learning has become a cornerstone of modern artificial intelligence, allowing significant advancements in fields such as computer vision, natural language processing, and speech recognition. By leveraging large datasets and deep neural networks, deep learning models have achieved unprecedented performance levels, often surpassing human capabilities in specific tasks [20, 132]. These models have been instrumental in applications such as autonomous vehicles, robotics, medical diagnostics [65, 144, 21] that profoundly impact society and improve human lives.

Despite these successes, deploying deep learning models in real-world scenarios presents several critical challenges:

- **Substantial computational costs:** Deep learning models often require significant computational resources and time to train, which can hinder their applicability in fast-paced, real-world environments. In many applications, such as real-time decision-making in autonomous vehicles or adaptive user interfaces, there is a critical need for models that can be trained or updated quickly to reflect new data or changing conditions. The lengthy training times of large models like CLIP [113], which was trained on 256 V100 GPUs for 12 days using 600 million image/text pairs, are impractical for such scenarios. This limitation makes it challenging to deploy deep learning solutions that need to quickly adapt, reducing their effectiveness in dynamic environments where timely responses are essential.

- **Distribution shift and catastrophic forgetting:** In dynamic environments, data distributions change over time. When applying large foundation models on personal devices, the new data may differ significantly from the original training set. Due to limited computational resources, retraining the entire model is impractical. Simply fine-tuning the model with new data often leads to **catastrophic forgetting** [4, 5], where the model forgets previously learned knowledge, compromising overall performance.

- **Data privacy concerns:** Companies often keep their data confidential to protect user privacy and maintain a competitive advantage [22, 71]. This practice poses obstacles for independent researchers and developers, who may lack access to large-scale proprietary datasets. Consequently, they are limited to using their own collected data or open-source datasets, which can restrict the development and application of deep learning models.

To address these challenges, various approaches have been proposed:

- **Knowledge Distillation** involves transferring knowledge from a large, complex model to a smaller, more efficient one [18, 82]. This approach reduces computational costs by enabling the deployment of lightweight models that require less memory and computational power, making them suitable for real-time applications and devices with limited resources. However, it does not adapt the previously trained model to new data, limiting its ability to handle changing environments or distribution shifts.

- **Transfer Learning** leverages pre-trained models on large datasets to improve learning on a new, related task with less data [13, 77]. This method addresses both computational costs and data privacy concerns. By fine-tuning pre-trained models with smaller, task-specific datasets, developers can achieve good performance without extensive computational resources or access to large proprietary datasets. However, this approach often leads to **catastrophic forgetting** of previous tasks, as it does not consider maintaining performance on the original tasks after adaptation.

- **Online Learning** algorithms update models incrementally over a sequential stream of data [7, 110]. This method is more efficient compared to traditional offline training. However, online learning often encounters performance drops and typically assumes that the training data stream is independently and identically distributed (i.i.d.), which is usually not the case in real-world applications where data distributions change over time.

- **Meta Learning**, commonly known as "learning to learn," aims to equip models with the ability to quickly adapt to new tasks using only a small amount of data [28]. This paradigm focuses on training models that generalize from previous experiences to learn new tasks more efficiently. However, meta-learning does not explicitly consider maintaining performance on pre-trained tasks, which can result in forgetting previously acquired knowledge when adapting to new tasks.

Although significant advances have been made in these areas, none of the aforementioned approaches fully achieve the goal of developing an intelligent agent that

can learn from a dynamically changing environment while retaining previously acquired knowledge, all within limited computational resources and restricted access to past data. Therefore, in this thesis, we focus primarily on **continual learning**, integrating methods from other fields to address these challenges and enhance the practicality of continual learning algorithms in more realistic, real-world scenarios.

We briefly illustrate the different learning paradigms in Figure 1.1; the details are discussed in Section 2.2.

1.1 Continual Learning

Continual learning, also known as lifelong learning, is a machine learning paradigm where an intelligent agent learns continuously from a stream of data, adapting to new tasks and environments over time while retaining previously acquired knowledge[157, 73]. Unlike traditional learning methods that rely on static datasets and assume that all training data is available at once, continual learning models are designed to handle dynamic, non-stationary data distributions common in real-world scenarios.

The main challenges of continual learning include:

- **Limited Access to Previous Data**: In many cases, storing all past data is impractical due to memory limitations or privacy concerns. The model must learn new tasks without relying on the availability of previous training data, which complicates the retention of previously acquired knowledge.
- **Dynamic and Non-Stationary Environments**: Data distributions can change over time (distribution shift), and the model must adapt to these changes without prior knowledge of when or how the shifts occur.
- **Limited Computational Resources**: Continual learning often needs to be implemented on devices with constrained computational capabilities, such as mobile phones or embedded systems. Efficient algorithms are required to enable incremental learning without extensive retraining or excessive memory consumption.
- **Balancing Plasticity and Stability**: The model needs to remain plastic enough to learn new information while being stable enough to retain prior knowledge. Achieving this balance is a significant technical challenge[30].

In this thesis, we focus on advancing continual learning to address the aforementioned challenges, enhancing its practicality and effectiveness in real-world scenarios where environments are dynamic, resources are limited, and data privacy is paramount. We propose integrating methods from other fields with continual

learning to develop solutions that can learn adaptively over time without forgetting previous knowledge, all while operating within computational constraints and limited access to prior data.

1.2 Applications

This thesis is also part of the FairWastes project, which aims to automate one of the sorting steps to reduce workers' exposure to hazardous conditions. In this project, waste is collected from various regions and time periods, including items such as cardboard, paper, plastic, wood, etc. The waste is transported on a conveyor belt, where, traditionally, numerous workers manually sort through it. However, this method is both costly and poses significant health risks to workers. The project seeks to leverage various sensors to capture images and other relevant data, which are then processed and transmitted to a robotic arm to automate the sorting process. The variability in waste composition, influenced by regional and temporal factors, necessitates adaptive and robust solutions. This dynamic closely resembles a continual learning scenario, where data distribution shifts over time. In this thesis, we apply our methods to this real-world dataset and evaluate their performance. The details of the dataset are presented in Section 3.5, while the results of different methods are discussed in Section 3.5 and in Section 4.5.

1.3 Contributions

The contributions of this thesis are listed as follows:

- **Entropy-Guided Self-Regulated Learning Without Forgetting for Distribution-Shift Continual Learning with blurred task boundaries**: Recent research on CL has focused on Domain-Incremental Learning (DIL) or Class-Incremental Learning (CIL) with well-defined task boundaries and no overlap of data between tasks[73, 46, 95]. However, for real-life applications, e.g., waste sorting, robotic grasping, etc., the model needs to be constantly updated to fit new data. Additionally, there is usually an **overlap** between new and old data. Thus, **task boundaries may not be well-defined**, and a more smooth scenario is needed. Thus, we propose a more general scenario, namely Distribution-Shift Incremental Learning (DS-IL), which enables soft task boundaries with possible mixtures of data distributions over tasks and thereby subsumes the two previous CL scenarios: DIL and CIL are simply DS-IL. Moreover, given the increasingly greater importance of data privacy in real-life applications and, incidentally, data storage efficiency, we further introduce an entropy-guided self-regulated

distillation process **without memory**, which leverages data similarities between tasks with soft-boundaries. Evaluated on a variety of datasets, our proposed method outperforms or matches state-of-the-art continual learning methods.

- **Adaptive Class Aware Memory Selection and Contrastive Representation Learning for Robust Online Continual Learning in both Balanced and Imbalanced Data Environments**: Online Continual Learning (OCL) is a framework where models learn continuously from a stream of data without revisiting previously seen data. This is crucial for many real-life applications, *e.g.*, waste sorting, healthcare monitoring, and robotics, where data evolves over time. However, current state-of-the-art continuous learning methods struggle with dynamic and unbalanced data[127, 74, 126], often failing to adapt and leading to severe performance degradation. Thus, we introduce Memory Selection with Contrastive Learning (MSCL), an advanced approach to Continual Learning (CL) designed to tackle these challenges. MSCL integrates Feature-Distance Based Sample Selection (FDBS) for effective memory adaptation, emphasizing inter-class similarities and intra-class diversity, with a novel contrastive learning loss (SCL) for evolving data representation consolidation. Our extensive evaluations on datasets including CIFAR-100, Tiny-ImageNet, PACS, and DomainNet demonstrate that MSCL not only surpasses existing memory-based CL methods on data balanced scenarios, but also excels particularly in imbalanced scenarios, thereby establishing a novel state of the art in both balanced and imbalanced learning contexts.

- **Online Continual Learning of Diffusion Models: Multi-Mode Adaptive Generative Distillation**:Most state-of-the-art[124, 145] continual learning methods rely on storing real data in a memory set, an approach often infeasible due to privacy constraints. Generative models, particularly diffusion models, offer a high-fidelity alternative for replay but are typically used in a static, task-defined manner. In more realistic *online continual learning* (OCL) where new data arrive sequentially, the diffusion model itself can suffer from catastrophic forgetting, and generating replay samples becomes computationally expensive. Existing distillation techniques[143, 133] reduce generation steps but assume a fixed teacher model, making them unsuitable when the teacher must also adapt to shifting distributions. We propose Multi-Mode Adaptive Generative Distillation (**MAGD**), a novel framework for continually training diffusion models under distribution shifts, while reducing computation. MAGD integrates Noisy Intermediate Generative Distillation (**NIGD**), which leverages intermediate noisy samples to main-

tain generation quality, and SNR-Guided Generative Distillation (**SGGD**), which disentangles denoising from the generative process for efficient knowledge transfer. By combining these with an Exponential Moving Average (**EMA**) teacher model, MAGD mitigates catastrophic forgetting as new data arrive. Experiments on Fashion-MNIST, CIFAR-10, and CIFAR-100 show that MAGD reduces generation overhead by up to 25% relative to standard generative distillation and 92% compared to DDGR-1000, while maintaining generating quality. Furthermore, in class-conditioned diffusion models, MAGD outperforms memory-based methods in terms of classification accuracy.

1.4 Publications

Here is the list of publications published and under review during my thesis:

- [146]Rui Yang, Matthieu Grard, Emmanuel Dellandréa, Liming Chen. "When continual learning meets robotic grasp detection: a novel benchmark on the Jacquard dataset". Published in *the 18th International Conference on Computer Vision Theory and Applications (VISAPP), Feb 2023, Lisbon, Portugal.*

- [164]Rui Yang, Matthieu Grard, Emmanuel Dellandréa, Liming Chen. "Entropy-Guided Self-Regulated Learning Without Forgetting for Distribution-Shift Continual Learning with blurred task boundaries". Accepted by *5th International Conference on Robotics, Computer Vision, and Intelligent Systems, 2025, Porto, Portugal.*

- [159]Rui Yang, Matthieu Grard, Emmanuel Dellandréa, Liming Chen. "Imbalanced data robust online continual learning based on evolving class aware memory selection and built-in contrastive representation learning". Published in *IEEE International Conference on Image Processing (ICIP), 2024, Abu Dhabi, United Arab Emirates.*

- Rui Yang, Matthieu Grard, Emmanuel Dellandréa, Liming Chen. "Adaptive Class Aware Memory Selection and Contrastive Representation Learning for Robust Online Continual Learning in both Balanced and Imbalanced Data Environments". Under review in *IEEE Transactions on Knowledge and Data Engineering,2025.*

- Rui Yang, Matthieu Grard, Emmanuel Dellandréa, Liming Chen. "Fast Multi-Mode Adaptive Generative Distillation for Continually Learning Dif-

fusion Models". Under review in *IEEE International Conference on Image Processing (ICIP), 2025.*

1.5 Outline

The rest of this thesis is organized into five chapters:

- Chapter 2 provides a foundational overview of deep neural networks and explores various machine learning paradigms that intersect with the domain of continual learning (CL). It begins with a basic introduction to deep neural networks, and then discusses different machine learning fields. Following this, the chapter delves into the specific scenarios under which continual learning operates, the metrics used to evaluate continual learning systems, and offers a comprehensive review of state-of-the-art methods in the field. Lastly, the chapter briefly introduces diffusion models, setting the stage for their detailed discussion and application in Chapter 5.

- Chapter 3 presents our work on handling scenarios with blurred task boundaries, introducing the entropy-guided exemplar-free method, and demonstrating its effectiveness compared to baseline methods.

- In Chapter 4: Addresses the challenge of imbalanced online continual learning, introducing a memory-based method combining a specific memory selection process with a contrastive learning loss function, and evaluating its performance on various benchmarks.

- Chapter 5 focuses on continually training diffusion models, proposing multiple distillation techniques to reduce computational costs and mitigate forgetting during training, and demonstrating effectiveness in both traditional CL scenarios and online settings.

- Chapter 6 concludes by summarizing our work and contributions, and outlines potential future research directions and open questions for further advancements in continual learning.

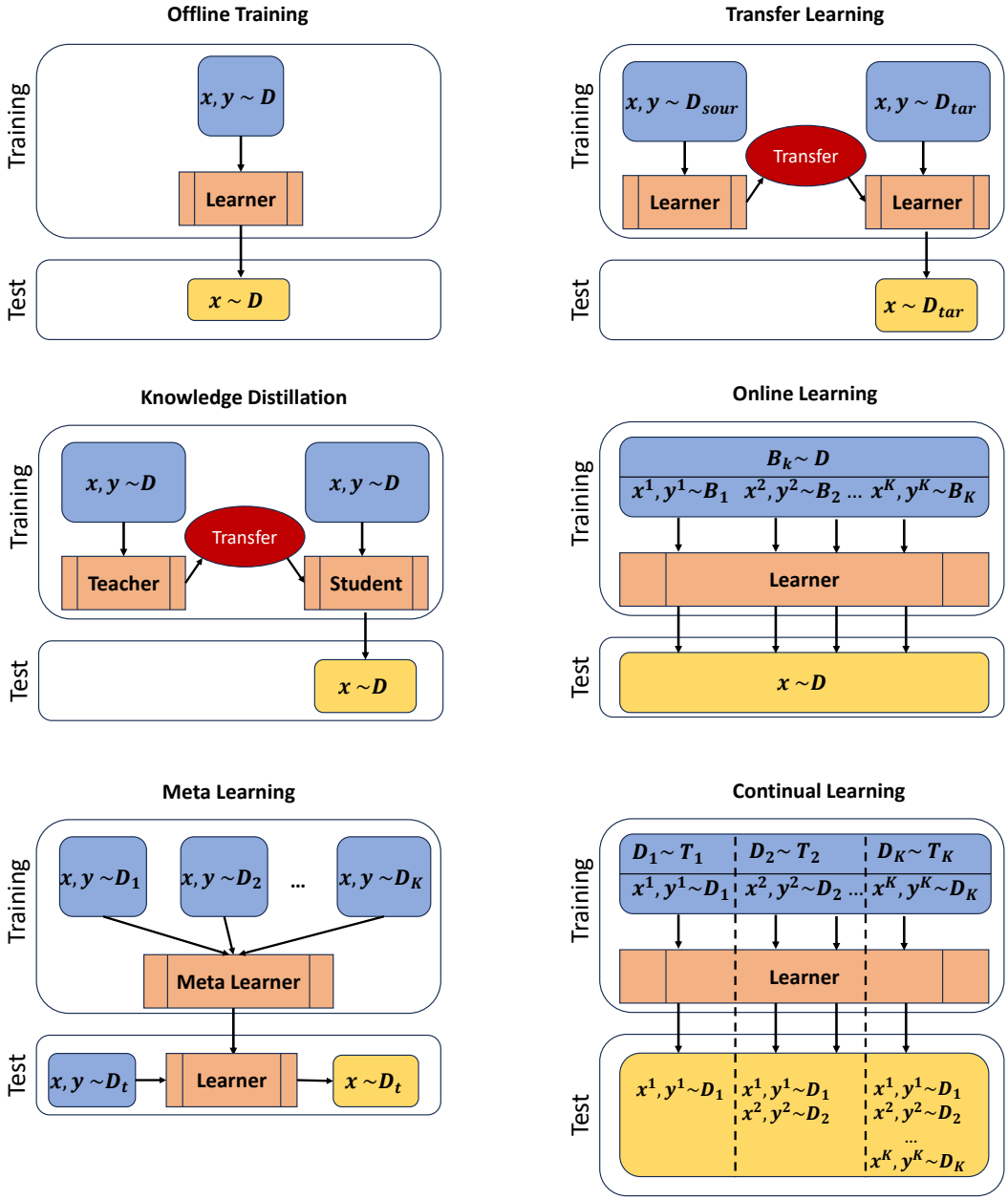


Figure 1.1: The main settings of different learning paradigms are illustrated in the figure, inspired by [58].

Chapter 2

Literature Review

2.1 Deep Neural Networks

Deep Neural Networks (DNNs) have revolutionized the field of machine learning by demonstrating exceptional abilities in tasks such as image recognition, natural language processing, and predictive analytics. Deep learning, a subset of machine learning, employs neural architectures with multiple layers of nonlinear processing units, allowing the learning of representations of data with multiple levels of abstraction [20]. Over recent years, various architectures have been proposed, including Convolutional Neural Networks (CNNs) [3], Residual Networks (ResNets) [17], and Transformers [37]. These networks are capable of discovering intricate structures in large datasets by leveraging the backpropagation algorithm, which guides how a machine should change its internal parameters to compute the representation in each layer from the representation in the previous layer [12].

2.1.1 Offline Training

Despite the substantial success of modern deep neural networks, they predominantly rely on offline training using mini-batch SGD updates. Given a dataset $\boldsymbol{D} = \{x_i, y_i\}_{i=1}^{N}$ and a model f_θ with parameters θ, the general objective in training, from the perspective of maximizing the likelihood, is formulated as:

$$\theta = \max_{\theta} \quad p(\boldsymbol{D}|\theta) \tag{2.1}$$

In offline training, it is assumed that each data pair (x, y) is sampled independently and identically distributed (i.i.d.) from $\boldsymbol{D}$. Thus, in supervised learning,

the likelihood can be expressed as:

$$\log p(\boldsymbol{D}|\theta) = \sum_{i=1}^{N} \log p(y_i|x_i, \theta) \tag{2.2}$$

The global objective can then be expressed as:

$$\mathcal{L}(\boldsymbol{D}, \theta) = -\frac{1}{N} \sum_{i=1}^{N} \log p(y_i|x_i, \theta) \tag{2.3}$$

During the training process, mini-batch SGD is utilized to approximate this global objective. For each iteration, a mini-batch $\boldsymbol{B}_k = \{x_i, y_i\}_{i=1}^{n_k}$ is selected. Thus, the objective for this mini-batch can be expressed as $\mathcal{L}_k(\boldsymbol{B}_k, \theta) = -\frac{1}{n_k} \sum_{i=1}^{n_k} \log p(y_i|x_i, \theta)$. Because each mini-batch is i.i.d., the expected value of the mini-batch $\mathcal{L}_k(\theta)$ is an unbiased estimator of the global objective $\mathcal{L}(\theta)$:

$$E[\mathcal{L}_k(\boldsymbol{B}_k, \theta)] = \mathcal{L}(\boldsymbol{D}, \theta) \tag{2.4}$$

The dataset's stability (i.e., no drift) and availability (i.e., the ability to review the data multiple times) in offline training allow for the repeated use of mini-batches, which in turn facilitates an accurate approximation of the expected loss. Overall, offline training relies fundamentally on two conditions:

- Each data pair (x, y) is sampled i.i.d. from $\boldsymbol{D}$.
- The training dataset is stable and available, enabling multiple reviews during training.

While offline training has been the cornerstone of deep neural network development, real-world applications often require models that continuously adapt or effectively leverage prior knowledge. To address these challenges, several learning paradigms have emerged. In the following, we review four key paradigms: Knowledge Distillation, Transfer Learning, Online Learning, and Meta-Learning. Each offers unique advantages and addresses different limitations inherent in traditional offline training.

2.2 Different Learning Paradigms

- **Knowledge Distillation** aims to transfer the knowledge from a larger, more complex model (referred to as the "teacher" model with parameters θ_{tea}) to a smaller, more efficient model (referred to as the "student" model with parameters θ_{stu}). Given a pre-trained teacher model and the original dataset

$\boldsymbol{D} = \{x_i, y_i\}_{i=1}^{N}$, the student model is trained to emulate the behavior of the teacher. The general objective of knowledge distillation can be formulated as minimizing the following loss function:

$$\mathcal{L}(\boldsymbol{D}, \theta_{stu}) = \frac{1}{N} \sum_{i=1}^{N} d(\phi_{\theta_{stu}}(x_i), f_{\theta_{tea}}(x_i)) \tag{2.5}$$

The function $d(.,.)$ measures the similarity between the outputs of the two models, which could be an L_1 or L_2 distance for regression tasks, or a Kullback-Leibler divergence loss for classification tasks where the outputs are probabilities [18]. This loss function encourages the student model to produce outputs that closely match those of the teacher model across the dataset. By doing so, the student learns to approximate the complex function represented by the teacher, leveraging the teacher's ability to generalize from the same input data while operating under potentially less computational overhead or memory usage.

- **Transfer Learning** seeks to leverage knowledge acquired from one domain or task (source) to enhance learning in a related but distinct domain or task (target). In this paradigm, we differentiate between the source domain with its distribution $\boldsymbol{D}_{source} = \{x_i^{Sou}, y_i^{Sou}\}_{i=1}^{N_{Sou}}$ where the model, denoted by $f_{\theta^{Sou}}$, is initially trained, and the target domain with distribution $\boldsymbol{D}_{target} = \{x_i^{Tar}, y_i^{Tar}\}_{i=1}^{N_{Tar}}$. Once the source model $f_{\theta^{Sou}}$ is trained on $\boldsymbol{D}_{source}$, the objective in transfer learning involves adapting this model to perform well on $\boldsymbol{D}_{target}$. The model for the target domain, represented by $f_{\theta^{Tar}}$, is initialized using the weights from θ^{Sou} and further trained to minimize the loss on the target data. The loss function for the target model is defined as:

$$\mathcal{L}(\boldsymbol{D}_{target}, \theta^{Tar}) = \frac{1}{N_{Tar}} \sum_{i=1}^{N_{Tar}} l(f_{\theta^{Tar}}(x_i^{Tar}), y_i^{Tar}) \tag{2.6}$$

where l is a loss function appropriate for the task (e.g., cross-entropy for classification or mean squared error for regression).

- **Online Learning** learns models in a sequential order and adapts to new data continuously as it arrives. In this paradigm, data arrives in a sequence of batches $\boldsymbol{B}_k = (x_i^k, y_i^k)_{i=1}^{n_k}$, for $k = 1, 2, 3, \ldots, K$, where n_k is the batch size for each time step k. The global objective, similar to offline training (Eq. 2.3), aims to minimize the cumulative loss over all observed data. The global objective in online learning can be formulated as minimizing the sum of losses

over time $\mathcal{L}(\boldsymbol{D}, \theta) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k(\boldsymbol{B}_k, \theta)$, where each $\mathcal{L}_k(\theta)$ is computed as:

$$\mathcal{L}_k(\boldsymbol{B}_k, \theta) = \frac{1}{n_k}\sum_{i=1}^{n_k} l(f_\theta(x_i^k), y_i^k) \tag{2.7}$$

In online learning, the stochastic nature of data presentation and the single-pass learning constraint mean that each batch $\boldsymbol{B}_k$ influences the model parameters θ once. This limitation complicates the approximation of the expected loss, $E[\mathcal{L}_k(\boldsymbol{B}_k, \theta)]$, since historical data is not retained for revisiting or recalibrating the model's parameters. As a result, the model must rely on the immediate feedback from each batch to perform updates and must be robust against variations in data quality and distribution.

- **Meta Learning** involves two levels of learning: the meta-training phase and the meta-testing phase. During meta-training, the model learns across multiple learning tasks, each potentially drawn from different distributions but sharing some commonality. The meta-testing phase evaluates the model's ability to adapt to new tasks, again using only a limited amount of data. Given a distribution of tasks $p(\mathcal{T})$, where each task $\mathcal{T}_k$ includes its own data distribution $p(\boldsymbol{D}_k)$. The meta-learning model trains across multiple tasks sampled from $p(\mathcal{T})$[28, 88]. The model parameters θ are updated in the meta-training phase to minimize the expected loss across tasks sampled from $p(\mathcal{T})$:

$$\theta^* = arg\min_\theta E_{\mathcal{T}_k \sim p(\mathcal{T})}[\mathcal{L}(\mathcal{T}_k, \theta)] \tag{2.8}$$

Here, $\mathcal{L}(\mathcal{T}_k, \theta)$ typically involves further adaptation on $\boldsymbol{D}_k$. During meta-testing, the effectiveness of θ^* is assessed on new tasks, using few examples to adapt θ^* to each new task.

2.3 Continual Learning

Distinct from other learning paradigms, continual learning uniquely focuses on developing models that can acquire new knowledge sequentially across a series of tasks while retaining previously learned knowledge. This paradigm addresses the critical challenge of **catastrophic forgetting**, where a model loses the knowledge it had learned from earlier tasks upon learning new ones. The goal of continual learning is to build systems that can update their knowledge continuously and adaptively.

Given a series of tasks as $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, where each task $\mathcal{T}_k$ is associated with a dataset $\boldsymbol{D}_k = \{x_i^k, y_i^k\}_{i=1}^{N_k}$. The overall objective in continual learning is to optimize the model's parameters θ over all the tasks:

$$\theta^* = arg\min_{\theta} \sum_{k=1}^{n} \mathcal{L}(\boldsymbol{D}_k, \theta) \tag{2.9}$$

However, due to the inherent limitations of continual learning, it is not feasible to revisit all previous datasets. Therefore, the objective during the training of task $\mathcal{T}_k$ can be formulated as follows using a memory set ($\boldsymbol{M}$):

$$\mathcal{L}_k = \mathcal{L}(\boldsymbol{D}_k, \theta) + \mathcal{L}(\boldsymbol{M}, \theta) \tag{2.10}$$

In continual learning, the memory set ($\boldsymbol{M}$) plays a crucial role. It is widely used to retain essential data from previous tasks, which may include real samples[24, 61, 60, 111] or synthetic data generated by models such as GANs[54, 75], VAEs[66], or diffusion models[134]. Typically, the parameters θ for the task $\mathcal{T}_k$ are initialized from the previous model with parameters θ^{k-1}, which was trained on prior tasks. This differs from offline training, where data ($\boldsymbol{D}_k$) is assumed to be i.i.d. and revisiting previous datasets is possible. In continual learning, the inability to review data from previous tasks often leads to catastrophic forgetting.

Compared to other learning paradigms, continual learning focuses on the retention of knowledge from previously trained models, which intersects the principles of **Knowledge Distillation**[24, 81, 64, 115] and **Transfer Learning**. It also involves updating the model in response to a continuous stream of data, relating closely to **Online Learning**[158]. Furthermore, continual learning strategies often overlap concepts from **meta learning**, especially in scenarios where only limited data from new tasks are available. However, continual learning emphasizes the retention of previously acquired knowledge, ensuring that the model remains proficient in previous tasks even after training on new ones.

2.3.1 Continual Learning Scenarios

Traditional Scenarios This category includes three primary scenarios: Task-Incremental Learning (TIL), Class-Incremental Learning (CIL), and Domain-Incremental Learning (DIL)[73, 46] as illustrated in Figure 2.1. We define a series of tasks as $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, where each task $\mathcal{T}_k$ is associated with a dataset $\boldsymbol{D}_k = \{x_i^k, y_i^k\}_{i=1}^{N_k}$, and C^k represents the unique class labels in $\boldsymbol{D}_k$. The index k serves as the task identity, indicating the origin of the data. In these scenarios, we typically assume that: 1) the task boundaries are well-defined, meaning the task identity is known during training; 2) the datasets are balanced, implying an equal number of data points across different classes or domains.

- **Task-Incremental Learning (TIL)**: In TIL, k is provided during both training and testing phases, which implies that the task origin of the data

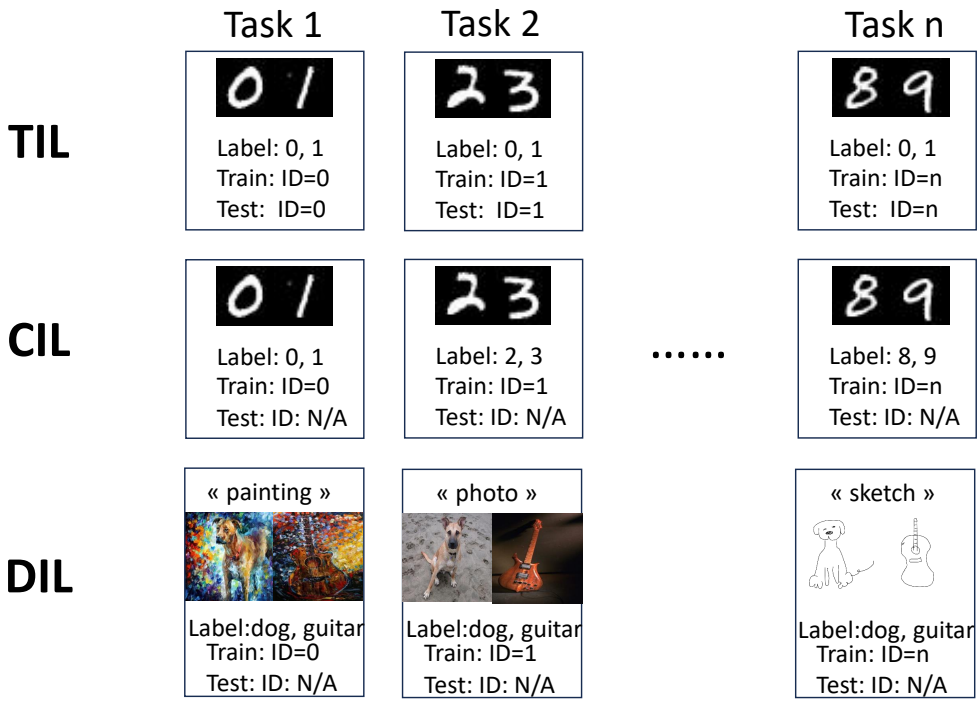


Figure 2.1: Illustration of three scenarios: Task-Incremental Learning (TIL), Class-Incremental Learning (CIL), and Domain-Incremental Learning (DIL). The main difference between TIL and CIL is that in CIL, the task ID is not provided during testing, making CIL more challenging. DIL focuses on the distribution shift of the input data, while the output labels remain unchanged.

is always known. Common architectures for this scenario often utilize task-specific components that are activated only for their respective tasks[23, 53].

- **Class-Incremental Learning (CIL)**: CIL extends TIL by enforcing that the class labels across different tasks do not overlap, i.e., $C^k \cap C^l = \emptyset$ for $k \neq l$. Moreover, while k is available during training, it is not during testing, making CIL more challenging than TIL.

- **Domain-Incremental Learning (DIL)**: Unlike CIL, DIL maintains a consistent label space across tasks, $C^k = C^l$ for $k \neq l$, but the input distribution varies between tasks. A common challenge in DIL is adapting to changes in the input distribution, such as in the dataset PACS[32], where each domain can be treated as a separate task. Here, k is not available during testing.

Advanced Scenarios These scenarios consider more realistic applications that do not conform to the traditional scenario constraints. Consequently, new scenarios have been proposed, such as:

- **Task-Free Continual Learning (TFCL)**[40] deviates from the traditional models by not defining task boundaries. Thus, k is unknown during both training and testing. Methods that depend on k during training, such as those cited in[24, 81, 41], are not applicable in TFCL.
- **Blurred Boundary Continual Learning (BBCL)**[56], which builds on the principles of TFCL, also does not identify k during training or testing. BBCL, however, requires that the transition between tasks be smoother.
- **Imbalanced Continual Learning (ICL)**[126, 104, 91], does not restrict the use of k. ICL focuses on handling datasets where the distribution of data among classes or domains is uneven, mirroring more closely real-world scenarios.
- **Online Continual Learning (OCL)**[59] treats data as a stream, $S_t = \{x_i^t, y_i^t\}_{i=1}^{n_t}$, where t is the time step, and n_t is the batch size at that step. Unlike previous scenarios, S_t can only be viewed once and is not reviewed multiple times as in offline training. OCL is often combined with other scenarios to create hybrid models such as Online Class-Incremental Learning (OCIL)[85] and Online Task-Free Continual Learning (OTFCL)[128, 161].

Table 2.1: Overall description of different continual learning scenarios. × means no, ✓ means yes, and ⋆ means optional.

Scenario	Training			Testing
	Task-id(k)	Online	Balance	Task-id(k)
TIL	✓	×	✓	✓
CIL	✓	⋆	✓	×
DIL	✓	⋆	⋆	×
TFCL	×	⋆	⋆	×
BBCL	⋆	⋆	⋆	×
ICL	⋆	⋆	×	×

2.3.2 Datasets

- **MNIST**[6] consists of 60,000 training images and 10,000 testing images of 10 handwritten digits, each of size 28×28. In continual learning, MNIST

is often used to create various benchmarks such as Split-MNIST, Rotated-MNIST, and Permuted-MNIST[73], which help in studying model robustness against task shifts and transformations.

- **FashionMNIST**[38] features Zalando's article images with a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 grayscale image, associated with one of 10 classes. This dataset serves as a more challenging alternative to MNIST in continual learning scenarios, offering a different set of visual features for tasks requiring fashion item recognition.

- **CIFAR-10**[11] comprises 60,000 32×32 RGB images in 10 classes, with each class containing 5,000 training images and 1,000 testing images. CIFAR-10 is widely used in continual learning to evaluate a model's ability to handle more complex image data across similar tasks.

- **CIFAR-100**[11] is similar to CIFAR-10 but with 100 classes, each containing 500 training images and 100 testing images. This dataset is beneficial for testing continual learning methods that must handle a higher degree of class granularity.

- **Mini-ImageNet**[26], a subset of ImageNet[9], includes 100 classes, each with 500 $3 \times 84 \times 84$ images for training and 100 images for testing. Mini-ImageNet is typically used in few-shot learning scenarios within continual learning, focusing on rapid adaptation to new tasks.

- **TinyImageNet**[19] downscales ImageNet to 200 classes, each with 500 $3 \times 64 \times 64$ images for training and 50 images for testing. This dataset tests continual learning algorithms' effectiveness across a broad range of classes and larger image sizes compared to Mini-ImageNet.

- **PACS**[32] contains images from four distinct domains: Art painting, Cartoon, Photo, and Sketch, each with the same seven classes. It includes 9,991 RGB images in total and is extensively used to assess domain generalization in continual learning frameworks.

- **DomainNet**[51] features images from six distinct domains (e.g., real, painting, clipart, quickdraw, infograph, and sketch), ranging from 48K to 172K images per domain (600K in total), categorized into 345 classes. This dataset challenges continual learning systems to maintain performance across extensive and diverse visual domains and classes.

2.3.3 Evaluation Metrics

Given n tasks denoted as $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, each task $\mathcal{T}_k$ is associated with a training set $\boldsymbol{D}_k^{\text{train}}$ and a test set $\boldsymbol{D}_k^{\text{test}}$. After training our model on $\mathcal{T}_k$, the model is evaluated on the test sets of all tasks. As detailed in Tab. 2.2, we use $R_{i,j}$ to represent the model's performance after training on $\mathcal{T}_i$ and testing on $\mathcal{T}_j$. This metric could be represented as classification accuracy for discriminative models, or as a FID score [29] for generative models.

Table 2.2: Illustration of Evaluation Metrics in Continual Learning

		Test On				
		$\mathcal{T}_1$	$\mathcal{T}_2$	...	$\mathcal{T}_{n-1}$	$\mathcal{T}_n$
After Training	$\mathcal{T}_1$	$R_{1,1}$	$R_{1,2}$		$R_{1,n-1}$	$R_{1,n}$
	$\mathcal{T}_2$	$R_{2,1}$	$R_{2,2}$		$R_{2,n-1}$	$R_{2,n}$
	...					
	$\mathcal{T}_{n-1}$	$R_{n-1,1}$	$R_{n-1,2}$		$R_{n-1,n-1}$	$R_{n-1,n}$
	$\mathcal{T}_n$	$R_{n,1}$	$R_{n,2}$		$R_{n,n-1}$	$R_{n,n}$

The common metrics used in continual learning are as follows:

- **Average Accuracy (AA)**[43] evaluates the model's performance on all trained tasks after training on task $\mathcal{T}_k$ as shown below:

$$AA_k = \frac{1}{k}\sum_{j=1}^{k} R_{k,j} \tag{2.11}$$

  In practice, the final average accuracy AA_n is used, which represents the performance after training on the final task $\mathcal{T}_n$ and testing on the test set of all tasks.

- **Average Incremental Accuracy (AIA)**[81], unlike AA, which represents the model's performance immediately after training on the current task, AIA considers the performance across all prior tasks.

$$AIA_k = \frac{1}{k}\sum_{i=1}^{k} AA_i = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{1}{i}\sum_{j=1}^{i} R_{i,j}\right) \tag{2.12}$$

- **Average Forgetting (AF)**[43] measures how much performance on a task has declined from its peak, which was observed in earlier evaluations:

$$AF_k = \frac{1}{k-1}\sum_{j=1}^{k-1} f_{j,k} \tag{2.13}$$

where $f_{j,k} = \max_{i \in \{1,\ldots,k-1\}}(R_{i,j} - R_{k,j})$.

- **Backward Transfer (BWT)**[34], unlike AF, evaluates the change in performance on a previous task j by calculating the difference between the performance when it was last trained and its current performance after learning subsequent tasks:

$$BWT_k = \frac{1}{k-1}\sum_{j=1}^{k-1}(R_{k,j} - R_{j,j}) \tag{2.14}$$

  In most cases, the best performance on the test set of task $\mathcal{T}_j$ is obtained immediately after training on $\mathcal{T}_j$. Therefore, $AF_k = -BWT_k$ for most experiments, indicating the magnitude of forgetting.

2.4 Continual Learning Methods

In this section, we present the state-of-the-art in continual learning, categorizing existing approaches into three primary groups: regularization-based methods, architecture-based methods, and memory-based methods. Although memory-based methods generally achieve the best performance in realistic and complex scenarios where the task identity is not provided during training, we begin by reviewing regularization-based methods from a Bayesian perspective to provide foundational understanding.

2.4.1 Regularization-based Methods

Regularization-based methods add explicit or implicit regularization terms to the loss function to constrain the update of the model parameters, thus helping to retain knowledge from previous tasks.

2.4.1.1 General Analysis from a Bayesian Perspective.

From a Bayesian standpoint, the posterior distribution of the model parameters θ after observing k tasks with datasets $\boldsymbol{D}_1, \ldots, \boldsymbol{D}_k$ can be expressed as

$$\log p(\theta|\boldsymbol{D}_1,\ldots,\boldsymbol{D}_k) = \underbrace{\log p(\theta|\boldsymbol{D}_1,\ldots,\boldsymbol{D}_{k-1})}_{\text{Previous posterior}} + \underbrace{\log p(\boldsymbol{D}_k|\theta)}_{\text{Current likelyhood}} - \underbrace{\log p(\boldsymbol{D}_k|\boldsymbol{D}_1,\ldots,\boldsymbol{D}_{k-1})}_{\textit{Constant}} \tag{2.15}$$

Assuming that the datasets are independent given θ, the term $\log p(D_k|D_1,\ldots,D_{k-1})$ is a constant with respect to θ and can be ignored during optimization. Thus, the posterior simplifies to:

$$\log p(\theta|\boldsymbol{D}_1,\ldots,\boldsymbol{D}_k) \propto \log p(\boldsymbol{D}_k|\theta) + \log p(\theta|\boldsymbol{D}_1,\ldots,\boldsymbol{D}_{k-1}) \tag{2.16}$$

This equation shows that updating the model parameters involves combining the likelihood of the current data $\log p(\boldsymbol{D}_k|\theta)$ with the prior knowledge encapsulated in the previous posterior $\log p(\theta|\boldsymbol{D}_1, \ldots, \boldsymbol{D}_{k-1})$. However, directly computing or storing the previous posterior is generally intractable. Therefore, most regularization-based continual learning methods focus on approximating this previous posterior. In the following, we explain the current methods from this perspective.

2.4.1.2 Explicit Regularization

Explicit regularization methods add quadratic constraints to the loss function to limit the updates of model parameters during training on new tasks. Specifically, they penalize changes in parameters by adding a term of the form $\sum_i \alpha_i(\theta_i^k - \theta_i^{k-1})^2$, where θ_i^k is the current parameter for the task k, θ_i^{k-1} is the parameter from previous tasks, and α_i is an importance weight that measures the importance parameter θ_i has to retain previous knowledge.

The origin of this quadratic term stems from a second-order Taylor expansion of the log-posterior distribution around the parameters obtained from previous tasks. Suppose that our model has l parameters denoted as $\theta = (\theta_1, \theta_2, \ldots, \theta_l)$. We approximate the previous posterior distribution as an independent multivariate Gaussian with mean μ_{k-1} and precision matrix (inverse of the covariance matrix) Λ_{k-1}^{-1} : $p(\theta|\boldsymbol{D}_1, \ldots, \boldsymbol{D}_{k-1}) \approx \mathcal{N}(\theta; \mu_{k-1}, \Lambda_{k-1}^{-1})$ Expanding the logarithm of this posterior around $\theta = \mu_{k-1}$ using a second-order Taylor expansion yields:

$$\log p(\theta|\mu_{k-1}, \Lambda_{k-1}^{-1}) \approx \log p(\mu_{k-1}|\mu_{k-1}, \Lambda_{k-1}^{-1}) - \frac{1}{2}(\theta - \mu_{k-1})^\top \Lambda_{k-1}(\theta - \mu_{k-1}) \quad (2.17)$$

Here, the first-order derivative term vanishes at $\theta = \mu_{k-1}$, because we assume the gradient at this point approximates zero. The Hessian matrix of the log-posterior, evaluated at $\theta = \mu_{k-1}$, is given by:

$$\mathcal{H}_{i,j} = \frac{\partial^2 \log p(\theta|\mu_{k-1}, \Lambda_{k-1}^{-1})}{\partial\theta_i \partial\theta_j}; \mathcal{H} = \quad -\Lambda_{k-1} \quad (2.18)$$

In practice, μ_{k-1} represents the parameters obtained after training on the previous tasks $(D_1, \ldots, D_{k-1})$.

- **Elastic Weight Consolidation (EWC)** [30] proposes using the Fisher Information Matrix (FIM) to approximate the Hessian of the log-posterior computed on task $k-1$. The FIM is defined as:

$$F_{k-1} = \mathbb{E}[\nabla_\theta \log p(D_{k-1}|\theta) \nabla_\theta \log p(D_{k-1}|\theta)^\top]|_{\theta=\mu_{k-1}} \quad (2.19)$$

where μ_{k-1} represents the model parameters after training on task $k-1$. This approach uses the squared gradients of the log-likelihood with respect to the parameters to estimate their importance. By adding a quadratic penalty term proportional to the parameter importance, EWC discourages significant changes to crucial parameters during training on new tasks.

- **Synaptic Intelligence (SI)** [39] calculates the importance of each parameter by accumulating its contribution to the loss reduction over time. The importance measure $\mathcal{H}_{i,i}$ for parameter θ_i is computed as:

$$\mathcal{H}_{i,i} = \sum_t \frac{\triangle\theta_i^t . g_i^t}{(\triangle\theta_i^t)^2 + \epsilon} \tag{2.20}$$

where $\Delta\theta_i^t$ is the change in parameter θ_i at time step t, g_i^t is the gradient of the loss with respect to θ_i at time step t, and ϵ is a small constant to stabilize the calculation. Unlike EWC, SI updates the importance weights online during training, eliminating the need for separate computations after each task.

- **Memory Aware Synapses (MAS)** [27] utilizes the sensitivity of the model's outputs to changes in parameters to compute importance weights, independent of the loss function. The importance $\mathcal{H}_{i,i}$ is estimated as:

$$\mathcal{H}_{i,i} = \mathbb{E}_{x\sim[D_1,\ldots,D_{k-1}]}\left[\left(\frac{\partial f_\theta(x)}{\partial\theta_i}\right)^2\right] \tag{2.21}$$

where $f_\theta(x)$ is the model's output for input x. The advantage of MAS over EWC and SI is its ability to use unlabeled data to estimate parameter importance, as it relies solely on the model's output responses rather than on loss computations that require ground-truth labels.

- Additional methods aim to refine the approximation of the Hessian matrix: **R-EWC**[48] performs a factorized rotation of the parameter space to diagonalize the FIM. This transformation captures complex parameter interactions and can lead to a more accurate estimation of parameter importance. **ALASSO**[68] introduces an asymmetric quadratic penalty that overestimates the importance on one side of the parameter space.

2.4.1.3 Implicit Regularization

Apart from explicitly constraining parameter updates using second-order Taylor expansions, another approach to preventing catastrophic forgetting is employing **online variational inference** to approximate the posterior probability

$p(\theta|\boldsymbol{D}_1, \ldots, \boldsymbol{D}_k)$ at $\mathcal{T}_k$. This approximation is formulated as: :

$$q_k(\theta) = \underset{q}{\text{argmin}} KL\left(q(\theta)\middle|\frac{1}{Z_k}q_{k-1}(\theta)p(\boldsymbol{D}_k|\theta)\right) \tag{2.22}$$

Where Z_k is a normalization constant. In practice, we often assume $q_k(\theta)$ follows a multivariate Gaussian distribution $\mathcal{N}(\mu_k, \Lambda_k^{-1})$ Several methods have been developed based on this variational inference framework:

- **Variational Continual Learning (VCL)** [72] proposes minimizing the following objective function:

$$\mathcal{L}_k = \mathbb{E}_{\theta\sim q_k(\theta)}[\log p(\boldsymbol{D}|\theta)] + KL(q_k(\theta)|q_{k-1}(\theta)) \tag{2.23}$$

  Here, $\mathbb{E}_{\theta\sim q_k(\theta)}[\log p(\boldsymbol{D}_k|\theta)]$ is the expected log-likelihood of the current dataset $\boldsymbol{D}_k$ under the approximate posterior $q_k(\theta)$, which corresponds to the cross-entropy loss in classification problems. The KL divergence term acts as a regularizer, encouraging the new posterior $q_k(\theta)$ to stay close to the previous posterior $q_{k-1}(\theta)$. Since both $q_k(\theta)$ and $q_{k-1}(\theta)$ are Gaussian distributions, the KL divergence can be computed in closed form, facilitating efficient optimization.

- **FOO-VB**[120] also employs Bayesian neural networks similar to VCL but introduces a key difference. FOO-VB incorporates Monte Carlo approximations not only during the inference phase, but also when updating the mean and variance of the network parameters. This approach enables an online update mechanism that is more suitable for continual learning scenarios, as it allows the model to adapt its parameters incrementally as new data arrives.

- There are also some methods continue in this direction such as **VAR-GP**[89] utilize a variational autoregressive Gaussian process to model temporal dependencies in sequential data, enhancing the model's ability to adapt over time. **FROMP**[96] applies Bayesian inference directly in the function space. **GRS**[92] adapts to non-stationary data.

Regularization-based methods generally have a clear mathematical foundation from the Bayesian perspective, as they incorporate prior knowledge into the learning process. These methods typically assume that the model parameters follow a multivariate Gaussian distribution. In explicit regularization, various approaches are used to approximate the Hessian matrix, which serves as the importance weight for each parameter. This approximation relies on a second-order Taylor expansion of the loss function around the critical point $\theta^\star$. Consequently, these methods are effective only under certain conditions: 1.The loss function's landscape must be

smooth around the critical point $\theta^\star$. This ensures that the second-order approximation is valid. 2.The data distribution between different tasks should not change dramatically. Significant shifts can invalidate the assumptions made during the approximation. However, in realistic scenarios, these conditions are often not met due to complex loss landscapes and substantial distributional shifts between tasks. Moreover, explicit regularization methods often employ Bayesian neural networks, which require higher computational costs. The assumption that parameters follow a multivariate Gaussian distribution may not hold in complex settings, limiting the practicality of these methods. Therefore, while regularization-based approaches are theoretically sound, their applicability in real-world continual learning scenarios is much constrained.

2.4.2 Architecture-Based Methods

Architecture-based methods are not discussed extensively in this thesis and are not used for comparison in our scenarios. Thus, we provide a relatively brief introduction to these methods. Architecture-based methods focus on either isolating parameters based on task identity and activating the corresponding parameters during inference, or dynamically expanding neural networks to learn new information. After training on the current task, the networks are then compressed. Several methods have been proposed in this direction.

- **Progressive Neural Networks (PNNs)** [25]: PNNs add a new neural network module for each new task while keeping previously learned modules fixed. This prevents interference between tasks and allows for knowledge transfer via lateral connections.

- **Piggyback** [49]: Piggyback leverages a single fixed-size model where separate weight masks are learned during training. Each mask activates the weights corresponding to a specific task.

- **Dynamically Expandable Representation (DER)** [118]: DER dynamically expands new feature extractors and encourages the network to use fewer neurons, promoting efficiency in representation.

- **Dynamic Token Expansion (DyTox)** [107]: DyTox dynamically expands a task-specific block in a transformer-based architecture, allowing the model to adapt to new tasks by adding tokens.

- **Online Discrepancy Distance Learning (ODDL)** [131]: Compared with previous methods, this approach also trains an auxiliary Variational Autoencoder (VAE) to generate images of the previous dataset. This enables

the evaluation of the discrepancy between generated images and the current dataset. Without relying on task identity, it uses this discrepancy evaluation to determine if the network should be expanded.

Most architecture-based methods require task identity during training, which is not convenient in realistic scenarios where task boundaries are not well-defined. Some methods propose specific conditions to determine if the distribution shifts to a new domain; however, in online continual learning where the data is non-stationary, it is difficult to make such decisions. Thus, in this thesis, we do not consider these methods.

2.4.3 Memory-based Methods

In this section, we primarily focus on memory-based methods, illustrated in Figure 2.2, which are among the most effective strategies for mitigating catastrophic forgetting. These methods concentrate on two main aspects: 1. **Memory Set Management**: This involves decisions on how to select, store, and replay memories. 2. **Utilization of the Memory Set**: This typically involves integrating the memory set with other approaches such as contrastive learning, knowledge distillation, and data augmentation to formulate new objectives. We will explore these aspects in detail in subsequent sections.

2.4.3.1 Memory Set Management

Effective management of the memory set begins with the crucial task of data point selection. Identifying the most critical data that retains knowledge and accurately represents previous distributions remains an underexplored area. Initially, we detail two prevalent memory selection strategies used in numerous memory-based methods: **Reservoir Sampling** and **Herding**.

- **Reservoir Sampling** often denoted as ER (Experience Replay), was initially proposed in [2] and later introduced to the field of Continual Learning in key studies such as [61] and [45]. This method enables random selection of data points without prior knowledge of the data stream's total length, which aligns perfectly with the demands of continual learning. The implementation of this algorithm is detailed in Algorithm 1. This is often considered as an important baseline.

- **Herding** is prominently utilized in iCaRL [24], which calculates the mean of features for each class after training on each task, selecting data points that are closest to this mean as shown in Algorithm 2. This method is

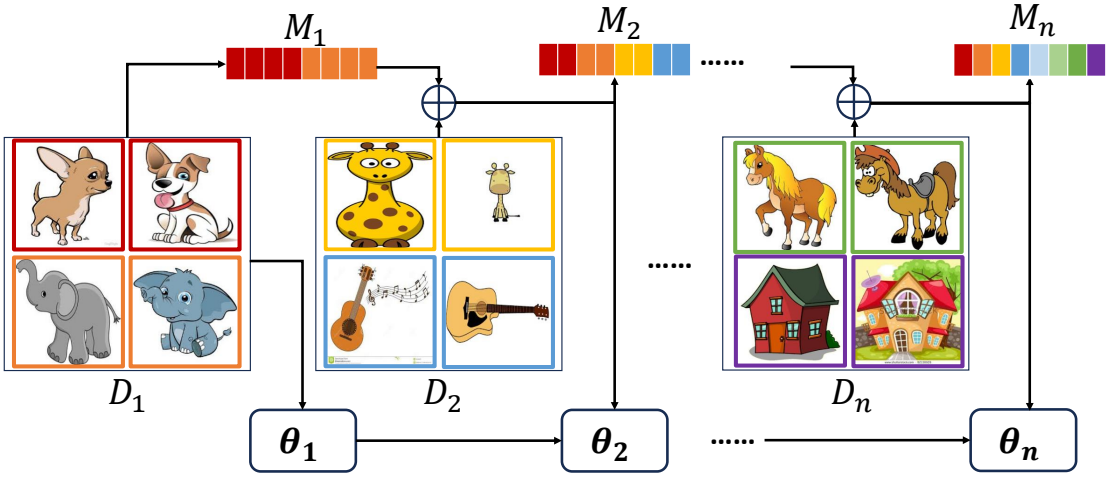


Figure 2.2: In the general framework of memory-based methods, for each task k, we work with the current dataset D_k and the preceding memory set M_{k-1}. These are combined to update our model. After the update, we derive a new memory set M_k, which is then utilized for the next task.

typically employed in class-incremental learning scenarios. However, its application becomes problematic in online scenarios where task boundaries are not clearly defined. In such cases, using all samples of each class to compute the class mean is not feasible, especially when multiple domains exist within a single class. Consequently, relying on a single class mean may be insufficient.

Other memory selection methods as:

- **Gradient-based sample selection (GSS)**[59] proposes to maximize the variance of gradient directions of the data samples in the replay buffer for data sample diversity but with no guarantee that the selected data are class representative. Furthermore, the replay buffer can be quickly saturated without any further update when local maximum of gradient variance is achieved.

- **Class-balancing reservoir sampling (CBRS)**[80] considers scenarios where the training data may be imbalanced, with unequal representation of classes, the application of Reservoir Sampling could result in a similarly imbalanced memory set in terms of class distribution. To address this, it is

proposed to implement random selection within each class while promoting a balanced memory set by preferentially discarding samples from the most populous class. However, this method does not account for the presence of multiple domains within a class and may also result in the selection of samples that are not the most representative of their respective classes.

- **Partitioning Reservoir Sampling (PRS)** [91] represents a variant of Reservoir Sampling. PRS calculates a target allocation for each class in the memory set based on the running frequency of each class. In contrast to Class-Balanced Reservoir Sampling (CBRS), which stores an equal number of samples for each class, PRS adjusts its strategy to reflect the training distribution. However, both methods share similar drawbacks.

- **Online Corset Selection (OCS)**[119] is a gradient-based memory selection method. This approach computes the gradient for each data point in the training batch, assessing minibatch similarity and sample diversity based on cosine similarity. Initially, OCS constructs a corset for training. After training, it selects samples from the corset that best represent the overall gradient characteristics of the corset in terms of cosine similarity for storage in the memory set. However, in a continual learning scenario where model parameters change over time, the gradients may only provide local information and prove less useful over prolonged training processes. Additionally, in imbalanced continual learning scenarios, OCS might overlook data from less-represented classes.

- **Rainbow Memory (RM)** [104] enhances the diversity of the memory set by estimating the uncertainty of an image. To calculate uncertainty, RM applies various data augmentations to a given image and then computes the variance in the model's outputs across the different augmented views. However, reliance on extensive data augmentation can significantly increase computational costs. Furthermore, while enhancing diversity, this approach may overlook important samples, such as those near the decision boundary, which are crucial for effective model training.

- **Coresets via Bilevel Optimization** [79] addresses the coreset selection problem in deep learning as a bilevel optimization problem. The outer objective seeks to find a coreset that minimizes the total loss across the entire dataset, while the inner objective aims to identify the best model within the coreset. Because it calculates the Hessian matrix during selection, this approach utilizes a proxy model (functions in a reproducing kernel Hilbert space associated with the Neural Tangent Kernel (NTK)). Although it demonstrates effectiveness in small coresets, the computational complexity

of $\mathcal{O}(n^2)$ increases with the size of the coreset, rendering it impractical for larger coresets.

- **Bilevel Coreset Selection via Regularization (BCSR)**The approach outlined in [136] leverages the framework of bilevel optimization to identify the coreset. Unlike [79], which employs a proxy model utilizing the Neural Tangent Kernel (NTK), BCSR directly incorporates the original model's architecture. It approximates the Hessian-inverse-vector product using a specific quadratic programming solution, which can reduce computational costs. However, the computational expense still increases rapidly with the number of parameters in the model.

Algorithm 1 Reservoir Sampling. $\mathcal{M}$ is the memory set, N_m is the memory budget, N is the number of examples observed so far, and B_t represents the current batch in time step t.

UpdateMemorySet$(\mathcal{M}, N_m, N, B_t)$
for i in range(len(B_t)) **do**
  $x_i, y_i = B_t[i]$
  if $N_m >$ len($\mathcal{M}$) **then**
    $\mathcal{M}$.append(x_i, y_i)
  else
    k = randint(0,N+i)
    if $k < N_m$ **then**
      $\mathcal{M}[k] = x_i, y_i$
    end if
  end if
end for
return $\mathcal{M}$

Besides the memory selection methods, there are also some methods study how to **retrieve samples** from the memory set during training. Here, we present some methods widely used in Continual leanring.

- **Maximally Interfered Retrieval (MIR)**[60] recognizes that certain samples are crucial for retaining previously learned information during training. In practice, it employs a strategy where a pseudo-update is performed on the current model using only the input batch. The method then evaluates the increase in loss between the current model and the model after the pseudo-update. Subsequently, it selects the top-K samples that most increase the loss for inclusion in the training of the current model. This technique is

Algorithm 2 Herding. $\mathcal{P}_c$ is the subset of memory set $\mathcal{M}$ for class c, N_m is the memory budget, N_c is the number of classes observed so far, ϕ_θ is our feature extractor with parameters θ,and X_c is the data of class c of the current task.

UpdateMemorySet$(\mathcal{M}, N_m, N_c, X_c, \phi_\theta)$
$\mathcal{P}_c = [\,]$
$n_p = \frac{N_m}{N_c}$
$\mu = \frac{\phi_\theta(X_c)}{len(X_c)}$ {Compute the mean of the features}
for i in range(n_p) **do**
  $p_i = \arg\min_{x \in X_c} ||\mu - \frac{1}{n_p}(\phi_\theta(x) + \sum_{j=1}^{i-1} \phi_\theta(p_j))||$
  $\mathcal{P}_c$.append(p_i)
end for
$\mathcal{M}$.append$(\mathcal{P}_c)$
return $\mathcal{M}$

straightforward to integrate with other memory-based methods, as it does not significantly increase computational costs. Typically, MIR enhances the performance compared to random sampling.

- **Adversarial Shapley Value Experience Replay (ASER)**[98] employ the concept of Shapley values, originally proposed by [1], to estimate the utility (importance) of each sample in the memory batch and retrieve the top-K important samples for training. Practically, it utilizes a KNN-based classifier to assess the overall performance impact of discarding a sample, which is then used to define the Shapley value of that sample. Due to the computational cost associated with the KNN-based classifier, especially with a large number of samples, the size of the retrieved samples is limited.

- **Reweighted Sampling** in [80], a method is proposed to assign a weight to each class in the memory set, where the weight is inverse to the total number of images in that class within the memory set. When retrieving samples from the memory set, the probability of each sample being chosen corresponds to its class weight. The benefit of this method is that it can provide a more balanced memory batch for training, especially when the memory set is imbalanced. This approach helps to ensure fair representation of all classes during the training process.

2.4.3.2 Utilization of the Memory Set

In addition to memory management techniques, an important question remains about how to effectively utilize these memory samples to retain previously learned

knowledge. We primarily discuss three aspects of using the memory set: 1. **Knowledge Distillation** 2. **Contrastive Learning** 3.**Bias in the Final Classification Layer**

Knowledge distillation is a powerful technique for preserving previously learned knowledge and is considered a form of functional regularization. By transferring knowledge from a larger or previous model to a current smaller or updated model, it helps to maintain performance consistency across model iterations.

- **Learning Without Forgetting (LWF)**[23] is the first work that introduce knowledge distillation into the continual learning field. In their original paper, it does not use the memory set. It uses a multi-head classfication layers corresponds to the task identity. For each training task it use the final output of the previous trained model as a target to constrained the current model's update. In rencent papers, they often combined knowledge distillation with the ER as a baseline which is often denoted as ER-distill or simply as ER. Further more, **EEIL**[41] use also the final layer's output to generate the distillation loss. however, it separetly distill knowlege by divideve the final output into old classes and new classed to solve the bias durning the continual learning. In rencent year, **AYT**[156] propose to update the batch normalization infomation of previous model by using the current training data not a fixed teacher model as previous work.

- **Pooled Outputs Distillation (PODNet)**[81] comparing with **LWF**, it not only considers the final layer's output, it also considers the intermediate layers' output and try to retain knowledge by align the intermediate layers' output of old and current model. This is a stronger constraint than simple distilation, if the distribution changes rapidly, it can hurt the model's plasticity to the new training data. **eTag**[151] continually refine the tarining process, it trains the feature extractor by distilling the output of the intermediates layer and use a lightweight generator to produce task-specific features to retain knowledge in the classfication layer.

- **Incremental Classifier and Representation Learning (iCaRL)**[24] use a distillation at the feature level. It combines a memory set selected by herding and a NCM classifier considred as an importance baseline for class-incremental learning. Otherwise, **LUCIR**[64] propose to use cosine normalization to the features, and then distill from the normalized features.

Contrastive Learning is increasingly employed in various memory-based methods, enhanced by data augmentation to improve representation learning during

training. This approach solidifies the distinctions and similarities among samples, thus enhancing the robustness of the learned representations. Below, we outline some notable methods:

- **Supervised Contrastive Replay (SCR)**[111] utilizes the **Supervised Contrastive Learning Loss (SCL)** from [90], combined with data augmentation. It integrates the Nearest Class Mean (NCM) classifier, achieving superior performance compared to traditional memory replay methods such as ER and MIR. However, the original form of SCL may face convergence issues in continual learning scenarios.

- **Continual Prototype Evolution (CoPE)** [93] employs a proxy-based contrastive learning loss to retain knowledge. It calculates a proxy for each class during training and updates these proxies using a moving average in online training. The method constructs a contrastive learning loss akin to SCL using these proxies. Moreover, **Proxy-based Contrastive Replay (PCR)** [140], another proxy-based method, replaces the positive and negative pairs in the supervised contrastive learning loss with learned proxies, facilitating faster convergence and reducing bias in the final classification layer.

- **Prototype Augmentation and Self-Supervision (PASS)**[121] stores a prototype for each class instead of actual images. During training, it distorts these prototypes by applying Gaussian noise and then directly feeds them to the classifier. To foster the learning of more transferable features across tasks, PASS also incorporates a self-supervised learning loss (SSL) with label augmentation[67]. This method achieves state-of-the-art performance in class-incremental learning among memory-free methods. However, it may not perform as well in other scenarios, such as domain incremental learning or more complex real-world situations where the class prototypes do not capture the full class information.

- **Online Continual Learning through Mutual Information Maximization (OCM)** [124] posits that cross-entropy (CE) primarily learns discriminative features for each task, which may not translate well to other tasks. To address this, it proposes maximizing the mutual information between past and current data. Various data augmentation techniques, such as local and global rotations, are used to construct an InfoNCE-like[50] contrastive loss. Despite its high computational cost, it serves as a strong baseline.

- **Online Prototype Learning for Online Continual Learning (OnPro)** [145] targets shortcut learning in online continual learning. It computes online prototypes for both original images and their augmented versions, constructing a contrastive loss based on the similarity between the original and

augmented prototypes to overcome shortcut learning. To further compact the feature space, it also employs the original SCL at the instance level. Utilizing similar data augmentation techniques as OCM, it achieves state-of-the-art performance, although at a significant computational expense.

Bias in the final classification layer was first identified by [74] who noted a model's propensity to predict recently seen classes by analyzing the confusion matrix of predictions. Subsequently, [76] examined the weight vectors of the final layer across different classes, finding that the norm of the weight vector for recently learned classes was significantly higher than that of older classes, indicating a bias toward newly learned classes. Recent studies, such as [78, 135], have attributed this bias primarily to the use of cross-entropy loss during training. The loss from cross-entropy at a data point (x, y) can be calculated as follows:

$$\mathcal{L}_{ce}(f_\theta(x), y) = -\sum_{i=1}^{C} y_{c_i} log(p_{c_i}) \quad ; \quad p_{c_i} = \frac{e^{o_i}}{\sum_j^C e^{o_j}} \tag{2.24}$$

Here, θ represents the model parameters and the output logits are $\theta(x) = [o_1, o_2, \ldots, o_C]$. The probability p_{c_i} corresponds to corresponds to the class c_i probability computed from the softmax function over C, the total number of classes in the final classification layer. The vector y is a one-hot vector where $y_{c_i} \in \{0, 1\}$. The gradient for each output logit is derived as:

$$\frac{\partial \mathcal{L}_{ce}(\theta(x), y)}{\partial o_i} = p_{c_i} - y_{c_i} \tag{2.25}$$

This formula reveals that for the target class, where $y_{c_i} = 1$, the gradient becomes $p_{c_i} - 1$ which is negative, effectively reducing the logit of the target class. Conversely, for other classes, the gradient is p_{c_i}, which is positive and increases the logit of non-target classes. During training, particularly in a class-incremental learning scenario where batches predominantly contain data from current classes, this mechanism introduces a strong bias. To address this phenomenon, several solutions have been proposed, which aim to mitigate the inherent bias introduced by cross-entropy loss.

- **Nearest-Class-Mean Classifier (NCM)**[14] is widely utilized in various continual learning methods such as[24, 111, 81, 102]. This classifier replaces the traditional fully-connected classification layer with an NCM, which leverages the features stored in the memory set. During inference, the NCM assesses the distance between the feature vector of the test data and the class means, which are either calculated during training or derived from the memory set. This approach effectively reduces the bias towards currently learned

classes, typically observed in the final layer of the network. However, the efficacy of the NCM largely depends on the quality of the class means. These means are usually computed from a memory set, which may not adequately represent the entire class if there is a domain shift within the class during training. This limitation highlights the need for careful management of the memory set to ensure that it remains representative of each class's features over time.

- **Weight Alignement (WA)** proposed by [76], is a post-processing method used during inference to mitigate class imbalance. It involves evaluating the average norms of the weights associated with old and new classes and calculating the ratio between them. This ratio is then used to adjust the output logits, thereby boosting the logits of older classes. While straightforward to implement, this method does not address imbalances within old classes themselves nor does it account for domain variations.

- **Post-Scaling**[109] is another post-processing technique that tracks the number of data instances encountered for different classes. It computes a bias correction term as $log(\frac{p_n(c)}{p_i(c)})$ where $p_i(c)$ represents the ratio of the sample number of class i to the total encountered images, and $p_n(c)$ represents the uniform distribution as $1/C$, where C is the total number of classes. This term is directly added to the output logits to enhance outputs for classes with fewer samples and reduce them for classes with more samples. Although this method is simple to implement and can adjust logits effectively, it does not aid in retaining learned features from underrepresented classes.

- **GDumB**[97] employs a greedy balancing sampler designed to maintain a balanced memory set across classes. Prior to inference, the method involves retraining the final classification layer exclusively with data from this balanced memory set, effectively reducing bias introduced by class imbalance. However, the overall performance of the classifier is highly contingent on the size and representativeness of the memory set. While GDumb simplifies the management of the memory set by ensuring balance, this approach can also be critiqued for its potential to oversimplify complex class distributions.

- **Bias Correction Layer** is from BiC[74] is another post-processing method, comparing with **GDumB**, it not retrain the final classification layer but add a liner correction layer trained only on a small balanced set.

- **Asymmetric Cross-Entropy (ER-ACE)**, proposed in [122], addresses the overlap of representations between old and new classes caused by the classic cross-entropy loss. It suggests using the following objective instead of

the classical cross-entropy loss:

$$\mathcal{L}_{ace} = \mathcal{L}_{ce}(\mathcal{B}_m, C_{old} \cup C_{curr}) + \mathcal{L}_{ce}(\mathbf{B}_{curr}, C_{curr}) \tag{2.26}$$

Here, $\mathcal{B}_m$ represents the batch sampled from the memory set, and $\mathcal{B}_{curr}$ denotes the current batch. The sets C_{old} and C_{curr} represent the previously seen classes and the classes in the current batch, respectively. When using the memory batch, this method activates all output logits to help separate features of current from old classes. Conversely, when using the current batch, it activates only the logits corresponding to the current classes to foster learning of discriminative features within new classes. This approach helps alleviate the label imbalance between current and previously seen classes.

- **Gradient Self-Adaptation (GSA)**[135] observes that the gradients of the final layers are imbalanced during online continual learning, stemming from data and cross-task class imbalances. It proposes computing the rate of accumulated positive and negative gradients and embedding this rate to adjust the cross-entropy loss. This method can effectively improve performance; however, its benefits are still limited when task boundaries are unknown.

2.4.3.3 Others

Several other methods that are not covered in the previous sections can also mitigate catastrophic forgetting, including:

- **Online Update with Moving Average**: This strategy is based on the assumption that knowledge is stored within the parameters of a neural network. When a previous model is adaptable to new data, the new knowledge is incorporated into the parameters of the new model. In an online continual learning scenario, defining a teacher model using distillation techniques can be challenging. It updates the teacher models by employing an exponential moving average (EMA) of the previous and current models after training a batch. This EMA-based teacher model, which contains both old and new knowledge, is considered effective. Unlike the classical EMA approach, **SDP**[139] utilizes two auxiliary models for a more refined update of the teacher model. Similarly, MKD[163] uses multiple auxiliary models combined with data augmentation to compute a KL-divergence loss. CLeWI[153] employs a novel interpolation technique proposed in [138] to merge two neural networks and updates the post-training statistics of the batch normalization layers.

- **Exemplar-free Methods:** Specifically, in the Class-Incremental Learning scenario, exemplar-free methods are predominantly prototype-based and

combined with the Nearest Class Mean (NCM) classifier. **SDC**[102] proposes storing the mean features of each class as prototypes and updates the existing prototypes by evaluating the average movement of the current sample in the feature space, both before and after the model update. Building on **SDC**, **ADC**[150] uses adversarial samples to calculate the average movement of previous prototypes in the feature space. **LDC**[162] introduces a learnable projector to adapt previous prototypes to the new feature space.

Table 2.3: Summary of recently proposed continual learning methods, categorized by settings, mechanisms, and techniques employed. In the table, "TIL," "CIL," and "OCL" correspond to the scenarios of task-incremental learning, class-incremental learning, and online continual learning, respectively. "Reg," "Mem," and "PI" represent three categories of continual learning methods: regularization-based, memory-based, and parameter isolation (architecture-based). The techniques include "DA" for data augmentation; "ConL" for contrastive learning loss; "KD" for knowledge distillation; "Gene" for generative replay; "PP" for post-processing; "EMA" for exponential moving average; and "Pt" for prototype-based methods.

Methods	Settings			Mechanisms			Techniques						
	TIL	CIL	OCL	Reg	Mem	PI	DA	ConL	KD	Gene	PP	EMA	Pt
EWC[30], SI[39], MAS[27] R-EWC[48], ALASSO[68],RWalk[70]	✓	✓	✓	✓									
VCL[72], VAR-GP[89], GRS[92] A-GEM[42], GEM[34], OGD[62]	✓	✓	✓	✓	✓								
MER[52], La-MAML[84]	✓	✓	✓		✓	✓							
ODDL[131],CN-DPM[94]	✓	✓	✓		✓	✓				✓			
Piggyback[49], HAT[53] PackNet[35], UCL[57], CLNP[63]	✓					✓							
PNN[25], DER[118], DyTox[107]	✓	✓			✓	✓			✓				
ER[61], GSS[59], MIR[60] CBRS[80], DSDM[129], OCS[119] CBO[79], BSCR[136], ASER[98]	✓	✓	✓		✓								
iCaRL[24], LUCIR[64], EEIL[41] PODNet[81], eTag[151],AYT[156]	✓	✓			✓				✓				
LwF-MC[23], LwM[44], DMC[103]	✓	✓	✓	✓					✓				
WA[76], PS[109], GDumB[97]	✓	✓	✓		✓						✓		
BiC[74]	✓	✓			✓				✓		✓		
SCR[111], ER-ACE[122] OCM[124], GSA[135]	✓	✓	✓		✓		✓	✓	✓				
SDC[102], ADC[150], LDC[162]	✓	✓		✓					✓				✓
PASS[121]	✓	✓	✓	✓				✓	✓			✓	✓
CoPE[93], OnPro[145], PCR[140]	✓	✓	✓		✓		✓	✓	✓			✓	✓
SDP[139],CLeWI[153],MKD[163]	✓	✓	✓		✓		✓		✓			✓	
MeRGAN[55], lifelongGAN[75], GD[142]	✓	✓			✓				✓	✓			
DDGR[134],DiffClass[155],GUIDE[149]	✓	✓			✓				✓	✓			

2.5 Diffusion Models in Continual Learning

Currently, the state-of-the-art performance is achieved by memory-based methods in continual learning scenarios. It means that a memory set is necessary. However, in some situations, the original data is not accessible due to privacy or safety issues. A successible strategy by using a generative model to reproduce the previous data is used in continual learning as in [55, 75, 116]. In recent years, diffusion models have attracted much more attention [86, 114, 106] in generating high-quality images. Some papers[142, 134, 155, 137] have started to study the potential of using diffusion models in continual learning. Here, we provide a brief introduction of diffusion models and current attempts of using diffusion models in continual learning scenarios.

2.5.1 Denoising Diffusion Probabilistic Models (DDPM)

The Denoising Diffusion Probabilistic Model (DDPM) is a generative model that transforms a gaussian distribution into a complex one through a series of learnable diffusion steps. This model comprises two primary processes: the forward process, which progressively adds noise to data, and the reverse process, which attempts to reconstruct the original data by removing the noise. Both processes can be modeled as Markov chains.

Given a data sample x_0 drawn from the real data distribution $q(x)$, the forward process introduces Gaussian noise over T timesteps, resulting in a sequence of increasingly noisy samples $x_1, \ldots, x_T$. The relationship between consecutive samples is defined by the equation:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \boldsymbol{I}) \tag{2.27}$$

where β_t are predefined noise levels at each timestep. The overall forward transition is the product of the transitions at each step:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{2.28}$$

In the model, we define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. Using these, the sample x_t at any timestep t can be expressed as:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \quad ; \quad q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, 1-\bar{\alpha}_t \boldsymbol{I}) \tag{2.29}$$

The noise schedule β_t is typically predefined, with earlier works like[87] using a linear schedule, while later improvements by [112] adopted a cosine-based schedule to optimize the negative-log-likelihood (NLL) of generated images.

During training, the model learns to predict the noise added at each timestep based on the noisy image, which is essential for the reverse process—iteratively reconstructing less noisy images until a denoised image is obtained.

Algorithm 3 Training

1: **repeat**
2: $x_0 \sim q(x_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$
5: Take gradient descent step on $\nabla_\theta \|\epsilon - \epsilon_\theta(\sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon)\|^2$
6: **until** converged

In the reverse process of DDPM, the goal is to reconstruct the original image from its noised version by effectively reversing the forward diffusion process. However, directly computing the conditional probability $p(x_{t-1}|x_t)$ is not straightforward as it is not explicitly defined in the forward process. Using Bayesian inference, the conditional distribution $p(x_{t-1}|x_t, x_0)$ can be derived as follows:

$$p(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}|\hat{\mu}_t(x_t, x_0), \hat{\beta}_t \boldsymbol{I}) = p(x_t|x_{t-1}, x_0)\frac{p(x_{t-1}|x_0)}{p(x_t|x_0)} \tag{2.30}$$

Here, $p(x_t|x_{t-1}, x_0)$, $p(x_{t-1}|x_0)$, and $p(x_t|x_0)$ are Gaussian distributions, and their means and variances are known from the forward process. The expression for $\hat{\mu}_t$ and $\hat{\beta}_t$ can be computed as follows:

$$\hat{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t) \quad ; \quad \hat{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{2.31}$$

where α_t and $\bar{\alpha}_t$ are parameters from the forward process as defined previously, and ϵ_t is the estimated noise. The variance $\hat{\beta}_t \boldsymbol{I}$ adjusts as we step backwards, reflecting the decreasing uncertainty as we approach the original data.

The reverse process is iterative, where at each step, the model estimates the less noisy image x_{t-1} from a noisier image x_t using the model's learned parameters. This process continues until the original image is reconstructed from the noised image at x_T

In the DDPM, the variance σ_t^2 of the noise added in the reverse reconstruction process is typically set equal to β_t. In the Denoising Diffusion Implicit Model (DDIM)[99], σ_t is instead set to zero, which makes the reverse process deterministic and thus more computationally efficient.

Although diffusion models are capable of generating high-quality images, their reverse process often requires substantial computational resources, limiting their

Algorithm 4 Sampling

1: $x_T \sim \mathcal{N}(0, \boldsymbol{I})$
2: **for** t $= T, \ldots, 1$ **do**
3: $\quad z \sim \mathcal{N}(0, \boldsymbol{I})$ if $t > 1$, else $z = 0$
4: $\quad x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sigma_t z$
5: **end for**
6: **return** x_0

practical applications. Several methods have been proposed to reduce the computational cost of the reverse process. For instance, DDIM uses a deterministic reverse process, which can significantly reduce the number of reverse steps required. Another approach is **Progressive Distillation**[130], which distills trained deterministic samplers into new models with halved sampling steps. The student model, initialized from the teacher model, denoises towards a target where one student DDIM step corresponds to two teacher steps, as opposed to using the original sample x_0 as the denoising target. Through iterative progressive distillation, the number of sampling steps can be halved in each iteration. Additionally, **Consistency Models**[143] learn to map any intermediate noisy data points $x_t, t > 0$, on the diffusion sampling trajectory, back to its origin x_0 directly.

2.5.2 Continual Learning Methods

In the context of continual learning, diffusion models are increasingly being applied as an alternative to traditional memory buffers during the training on new data. These models leverage their generative capabilities to recapitulate previous knowledge, thereby alleviating issues related to catastrophic forgetting.

- **Diffusion-based Generative Replay (DDGR)**[134] utilizes a class-conditioned generative model to substitute the traditional memory set. Prior to each task, DDGR employs the previously trained diffusion model to generate a number of synthetic samples equivalent to the current training dataset. Both the diffusion model and the classifier are then retrained using these synthetic samples alongside real data. Although this method leverages the high quality and diversity of images produced by diffusion models to achieve state-of-the-art performance in class-incremental learning, it does not address the computational costs associated with full generation steps of diffusion models.

- **Stable Diffusion for Distillation and Replay (SDDR)**[137] incorporates a pre-trained stable diffusion model to generate supplementary samples during training. This straightforward application of diffusion models in

continual learning shows consistent improvement in performance. However, because the pre-trained model remains fixed, it cannot adapt to shifts in data distribution during training, posing challenges in dynamic real-world scenarios.

- **Diffusion-Based Class Incremental Learning (DiffClass)**[155] maintains a distinct diffusion model for each task, utilizing these models to generate data from previous tasks. When training a new task, it employs multi-distribution matching to augment synthetic images. Despite its effectiveness, this method requires expanding the architecture with each new task, which may be impractical in environments with limited storage.

- **Generative Distillation**[142] utilizes a previously trained diffusion model as a teacher to generate noisy images, which are then used to distill knowledge into the current model. This strategy allows for updates in boundary-free scenarios and leverages synthetic image generation prior to each task. However, the simple distillation approach does not fully exploit the capabilities of diffusion models, limiting the potential performance improvements.

- **Guidance-based Incremental Learning with Diffusion Models (GUIDE)**[149] employs a class-conditional diffusion model to generate samples that potentially lie on decision boundaries, thus enhancing classification accuracy. This approach introduces class information during the generation process, similar to earlier methods that stored real images[104]. While GUIDE improves classification performance, the need for continual updates of the diffusion model remains a challenge for enhancing the overall model performance.

2.6 Positioning

In this thesis, we tackle several pressing challenges within the realm of continual learning (CL), specifically in contexts where traditional approaches falter due to blurred task boundaries, data imbalance, data privacy, and computational inefficiency. Our three primary contributions each address distinct aspects of these challenges, collectively pushing forward the state of CL by introducing novel methodologies that are both scalable and practical.

Traditionally, the CL community has focused on scenarios such as Task-Incremental Learning (TIL), Class-Incremental Learning (CIL), and Domain-Incremental Learning (DIL), which assume well-defined task boundaries and significant shifts in data distribution without any overlap between tasks. However, in dynamic real-world applications like waste sorting, data collected from different

regions and seasons are not entirely distinct, often displaying considerable overlap. This overlap indicates that task boundaries in such real-world scenarios are inherently blurred, which challenges traditional CL paradigms.

To better accommodate these real-world complexities, we propose a novel scenario named **Distribution-Shift Incremental Learning (DS-IL)**. This framework facilitates soft task boundaries with potential mixtures of data distributions across tasks, reflecting the more nuanced and intertwined nature of real-world data. Existing methods typically depend on maintaining a memory set or necessitate expanding the architecture for each new task, approaches that significantly increase both computational costs and storage requirements [81, 34, 24, 25, 118].

To address these issues, particularly the problem of forgetting in blurred boundary scenarios, we introduce an entropy-guided method that eschews the need for a memory set. By using entropy as an indicator to assess whether the previously trained model recognizes current samples, which indicates overlap, we adaptively adjust the knowledge distillation process. This method leverages the inherent data overlaps to refine model training dynamically, effectively mitigating forgetting without the overhead associated with traditional approaches. Our experiments demonstrate that this entropy-guided approach significantly alleviates forgetting in DS-IL scenarios, as detailed in Chapter 3.

In our initial experiments, when applying our entropy-guided approach to complex scenarios where data distributions shift rapidly, such as transitioning from real images to sketches, the method proved inadequate. This limitation highlighted the necessity of a memory set to manage drastic changes in data characteristics effectively. Current memory-based methods, often evaluated using balanced datasets like CIFAR-10, CIFAR-100, and ImageNet, tend to underperform in imbalanced continual learning scenarios [24, 61, 59, 98, 60]. These methods struggle primarily due to their inability to manage the imbalance across different classes and domains effectively. Specifically, models tend to develop a bias towards dominant categories, and similarly, the memory sets mirror these biases. Another critical issue is the insufficient acquisition of knowledge from underrepresented categories, leading to poor performance on the test set.

To address these challenges, we propose a refined approach focusing on two key aspects: 1) effective management of the memory set, and 2) strategic utilization of the memory set. Our method first relies on feature-based similarities between memory sets and current samples. These similarities enable the selection of more representative samples that enhance the inter-class similarity and intra-class variance within the memory set, thus creating a more balanced training environment. Furthermore, to augment the generality and robustness of our model, we incorporate the memory set alongside current samples to generate a contrastive learning loss. This approach not only preserves past knowledge but also enhances the

model's adaptability and performance across varied and evolving data distributions.

Our experiments demonstrate that this method significantly outperforms existing approaches in scenarios characterized by imbalanced data distributions. The comprehensive details and empirical validations of our approach are presented in Chapter 4.

While memory-based methods have achieved state-of-the-art performance in various continual learning scenarios, modern challenges associated with data privacy and commercial competition restrict their practicality. Prominent AI companies such as OpenAI, Google, and Microsoft have opted for closed-source approaches, which complicate the storage of real images in memory sets. As a result, leveraging generative models to synthesize training samples presents a viable alternative. Previous studies have employed Generative Adversarial Networks (GANs) to generate historical data samples [54, 75]; however, these models often suffer from rapid degradation in image quality during continual training. This degradation is primarily due to the accumulation of errors as the model is trained sequentially across tasks, leading to significant deterioration in both image generation and classification performance.

In response to these issues, diffusion models have gained significant attention for their ability to generate high-quality images. Recent approaches in the community have started integrating diffusion models into continual learning frameworks [154, 134, 152, 155]. These methods typically utilize diffusion models as replay buffers to enhance classification accuracy. However, they often overlook critical challenges such as the computational costs associated with generating images and the complexities involved in continually training diffusion models.

Our final contribution addresses these gaps by focusing on how to train diffusion models continually and efficiently, which is crucial for their application in realistic settings. We introduce a novel framework that incorporates strategies such as Noisy Intermediate Generative Distillation (NIGD), Signal-Guided Generative Distillation (SGGD), and Exponential Moving Average (EMA). These strategies enable the efficient and effective continuous training of diffusion models within an online continual learning framework. By optimizing the training process, our approach reduces computational overhead and minimizes the error accumulation over successive tasks, thereby maintaining the quality of generated images and the model's classification accuracy.

Collectively, these contributions not only address specific technical hurdles but also synergize to form a robust framework for continual learning. This framework enhances the scalability and applicability of continual learning systems across a variety of domains, making it a valuable asset for practical deployment in dynamic and resource-constrained environments.

Chapter 3

Continual Learning with Blurred Task Boundaries

In this section, we firstly consider a specific realistic scenario where the task boundaries are blurred and there is an overlapping between tasks. In this scenario, we propose an entropy-guided distillation method which use the entropy information as an indicator to find the part of overlapping. and distill knowledge from the overlapping to overcome catastrophic forgetting. This work is presented in our published paper "Entropy-Guided Self-Regulated Learning Without Forgetting for Distribution- Shift Continual Learning with blurred task boundaries" [164].

3.1 Motivation

Continual Learning (CL) has emerged as a critical avenue in machine learning, enabling models to adapt to evolving information without discarding previously acquired knowledge. This capacity is indispensable for applications in dynamic environments, such as autonomous vehicles and healthcare monitoring systems, where data distributions shift over time. Traditional CL methods [24, 81, 41, 74, 64] generally assume clear and discrete task boundaries, such as TIL, CIL, and DIL [73, 46], and perform well under controlled conditions where tasks are strictly separated. However, these assumptions do not hold in some real-world scenarios where data from multiple tasks often overlaps, leading to blurred or non-existent task divisions.

Recent efforts to address this challenge, including the NIC scenario [33] and the Continuous Task Agnostic model [120], remain limited in scope. NIC focuses on new classes and instances within a single dataset, restricting its applicability, while the Continuous Task Agnostic model handles only linear task shifts, overlooking the more complex patterns of data evolution that arise in practice. Consequently,

many studies confine DIL and CIL to isolated domains and neglect their inherent interconnectedness.

In practical settings such as waste sorting and robotic navigation, the same objects or environments can recur in slightly different forms across multiple sessions. These repeated yet shifted distributions strain conventional CL systems, which often suffer from catastrophic forgetting the inadvertent erosion of previously learned information when new data is introduced. Existing solutions further require large memory buffers or specialized model architectures, increasing computational costs and raising privacy concerns.

To address these shortcomings, we propose a new CL framework called Distribution-Shift Incremental Learning (DS-IL). DS-IL is designed to accommodate overlapping data distributions across tasks, thereby facilitating smoother transitions that better mirror real-world conditions. Our approach integrates an entropy-guided self-regulated knowledge distillation procedure, which eliminates reliance on stored past data for guidance and mitigates both storage and privacy issues. By using entropy measurements to identify similarities between newly encountered samples and previously learned tasks, the model dynamically adjusts its learning priorities and preserves relevant knowledge.

The shift to DS-IL represents a significant advancement in the field of CL, aligning more closely with the complex and unpredictable nature of real-world data. This approach not only enhances the model's ability to learn continually in a more natural and less constrained manner but also reduces the computational overhead associated with traditional methods. The introduction of DS-IL, tested on benchmarks like PACS[31] and CIFAR-100[10], demonstrates promising results in maintaining high performance across continually evolving data landscapes without the need for extensive memory storage or computational resources.

3.2 Distribution-Shift Incremental Learning (DS-IL)

For simplicity without loss of generality, let us consider only two tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ with their data $\boldsymbol{D}_1$ for $\mathcal{T}_1 : \{x_i, y_i\}_{i \in 1:n_1}$ drawn from a joint distribution $p_1(x, y)$, and $\boldsymbol{D}_2$ for $\mathcal{T}_2 : \{x_i, y_i\}_{i \in 1:n_2}$ drawn from a joint distribution $p_2(x, y)$. Let l be the loss function (cross-entropy for classification). Let ϕ be the output of the model, and θ its parameters. Then, we can construct the objectives for $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively:

$$\mathcal{T}_1 : \mathcal{L}_1(\boldsymbol{D}_1, \theta) = \sum_{i=1}^{n_1} l(\phi_\theta(x_i), y_i)) \tag{3.1}$$

$$\mathcal{T}_2 : \mathcal{L}_2(\boldsymbol{D}_2, \theta) = \sum_{i=1}^{n_2} l(\phi_\theta(x_i), y_i)) \tag{3.2}$$

Gradient descent, such as Stochastic Gradient Descent (SGD), can be used to update the parameters θ to minimize the objective function. If $p_1(x, y)$ equals $p_2(x, y)$, $\mathcal{L}_1$ approximates $\mathcal{L}_2$. Thus, when training only with $\boldsymbol{D}_2$, $\mathcal{L}_1$ will not increase and the model will not forget $\mathcal{T}_1$. However, if $p_1(x, y)$ is very different from $p_2(x, y)$, the gradient direction of $\mathcal{L}_1$ may violate the gradient direction of $\mathcal{L}_2$ when updating the parameters only with the gradients of $\mathcal{L}_2$. Therefore, the objective function $\mathcal{L}_1$ will increase, and the phenomenon of *catastrophic forgetting* will occur.

From a Bayesian perspective, the joint distribution $p(x, y)$ can be decomposed as follows:

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x) \tag{3.3}$$

$p(x)$ and $p(y)$ denote the probability density of the input and output, respectively. $p(x|y)$ denotes the conditional probability of an output given an input. $p(x|y)$ denotes the conditional probability of an input given an output. For continual learning, the shift of these four terms is primarily involved, *e.g.*, $p(x)$ mainly changes in Domain-Incremental Learning (DIL), whereas $p(y)$ mainly changes in Class-Incremental Learning (CIL). We can create different scenarios by adjusting the proportion of these four terms.

From this data distribution shift perspective, it is easy to interpret DIL or CIL as DS-IL and extend them to Smooth-DIL or CIL when task transitions are blurred and data distributions mixed up. Thus, DS-IL can be regarded as a continual learning scenario in which both the input distribution $p(x)$ and the label distribution $p(y)$ change across tasks, which also induces changes in the conditional distributions $p(y \mid x)$ and $p(x \mid y)$.

3.2.1 From CIL to DS-IL

For a classic CIL, each new task only contains a few new classes. Thus, the density probability of the previous classes equals zero, and some new output variables y appear. The CIL setting can be considered to be an extreme case of the DS-IL setting where data labels are only available for the novel classes of the second task $\mathcal{T}_2$. Thus, $p(y)$ changes rapidly. By smoothing the change of $p(y)$, we can increase the similarity between tasks and generate a smoother and more flexible version of CIL, as shown in Figure 3.2.

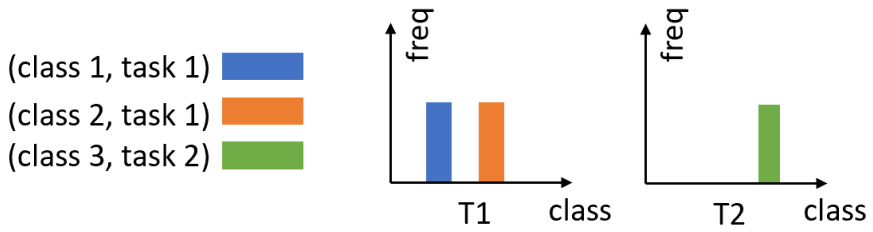


Figure 3.1: Illustration of TIL in which the data have both class id and task id. Moreover, task id is provided for both training and testing.

3.2.2 From DIL to DS-IL

The classic DIL defines each domain as a single task with well-defined task boundaries. This means that the domain shift is discontinuous. Thus, we can quickly transform a DIL into a CIL by labeling the data with domain and class labels as shown in Figure 3.3, thereby interpreting DIL as DS-IL. However, in many real-life applications, there are no clear and well-defined task boundaries with evident domain labels. Task transitions can be fuzzy, and data distributions of different tasks can overlap and mix up, resulting in a Smooth-DIL as shown in Figure 3.3, which is enabled by DS-IL.

3.3 Entropy-Guided Self-Regulated Learning without Forgetting

To ensure simplicity without loss of generality, we consider the same hypotheses as in Section 3.2. There are only two tasks. From Equation (3.1) and Equation (3.2), we can derive the joint objective for both task 1 and task 2 as:

$$\begin{aligned} \mathcal{L}_{total} &= \mathcal{L}_1 + \mathcal{L}_2 \\ &= \sum_{D_1} l(\phi_\theta(x_i), y_i)) + \sum_{D_2} l(\phi_\theta(x_j), y_j)) \end{aligned} \tag{3.4}$$

When training on $\mathcal{T}_2$, if we could store all the data from $\mathcal{T}_1$, then we can construct the objective $\mathcal{L}_1$. However, for CL, we can only use a small part or none of $\boldsymbol{D}_1$ (dataset for $\mathcal{T}_1$). Thus, the core problem is how to approximate $\mathcal{L}_1$.

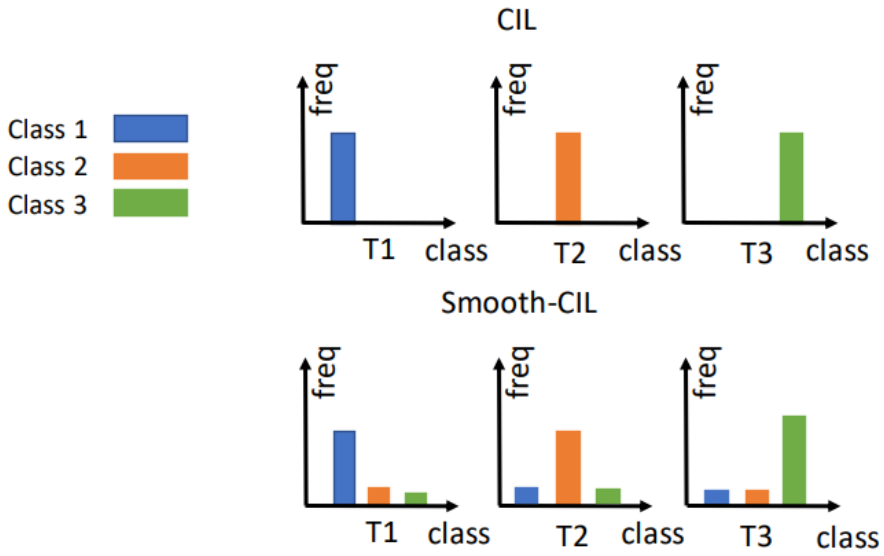


Figure 3.2: Illustration of CIL and Smooth-CIL using three tasks with three classes. In the original CIL setting, each task only contains instances of one class. The Smooth-CIL breaks this limit, and each task may have multiple classes. Thus, the shift of $p(y)$ in our scenario is smoother.

3.3.1 LWF (Learning without Forgetting)

LWF[23] proposes to use the previous model to generate a soft label and replaces the cross-entropy loss with a distillation loss. In this work, to fit our scenario, we only use the concept of knowledge distillation to represent LWF. θ^* represents the parameters learned from $\mathcal{T}_1$; l_{lwf} corresponds to the distillation loss; $\boldsymbol{M}$ is the memory set which is from $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$. Then, the new objective is as follows:

$$\hat{\mathcal{L}}_{total} = \hat{\mathcal{L}}_1 + \mathcal{L}_2 \tag{3.5}$$

$$\hat{\mathcal{L}}_1 = \sum_{M} l_{lwf}(\phi_\theta(x_i), \phi_{\theta^*}(x_i)) \tag{3.6}$$

LWF uses distillation loss to approximate the previous objective function $\mathcal{L}_1$.

3.3.2 The proposed ER-LWF approach

As we discussed in Section 3.2, the distribution shift between tasks is not always rigid and discontinuous. Thus, some current data may be similar to previous data. Here, we use $\boldsymbol{D}_1 \cap \boldsymbol{D}_2$ to represent the set of similar data. It then becomes crucial to find this intersection without access to $\boldsymbol{D}_1$. We propose to use the output of

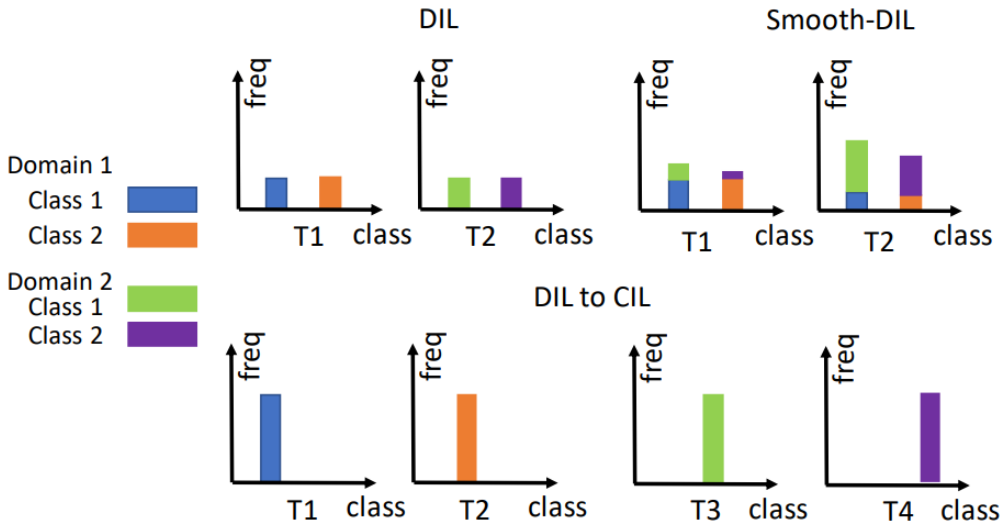


Figure 3.3: Assume there are two domains, and each domain contains the same class labels. The classic DIL considers each domain as a single task. In Smooth-DIL, each task can contain two domains, but the distribution of different domains is different for each task. Therefore, from DIL to CIL, we can label data by using both the domain label and the class label. In this figure, there are four different labels, corresponding to four tasks.

the previously learned model to verify if the data are similar. Let EI denote the entropy. Then, EI for given data, is defined as follows:

$$EI(x, y) = EI(p(y|\theta^*, x)) = EI(softmax(\phi(y|\theta^*, x))) \tag{3.7}$$

The closer EI is to zero, the more familiar and confident the model is about the input data, and the more corresponding data is valuable for reconstructing the $\mathcal{L}_1$. On the contrary, if EI is large, output distribution is close to discrete uniform distribution. This means that the model hardly classifies the given input data.

Rather than looking for similar data in $\boldsymbol{D}_1 \cap \boldsymbol{D}_2$, we propose to use EI to modulate distillation loss. Given an N-class classifier, the upper-bound of EI over its outputs corresponds to discrete uniform distribution:

$$EI_{upper} = -\sum_{1}^{N}(\frac{1}{N}log(\frac{1}{N})) = log(N) \tag{3.8}$$

$$0 \leqslant EI(x, y) \leqslant log(N) \tag{3.9}$$

Thus, we can define our $Entropy - Guided$ weight (EG-weight) for given data as follows:

$$w_{eg}(x, y) = 1 - \frac{EI(x, y)}{log(N)} \tag{3.10}$$

Then, the previous objective function could be approximated as:

$$\hat{\mathcal{L}}_1 = \sum_{\boldsymbol{D}_2} w_{eg}(x_i, y_i) l_{lwf}(\phi_\theta(x_i), \phi_{\theta^*}(x_i)) \quad , \quad (x_i, y_i) \in \boldsymbol{D}_2 \tag{3.11}$$

From Equation (3.11), we only use the current data $\boldsymbol{D}_2$ and our EG-weight to approximate the previous objective function. The benefits of this formulation are threefold:

- Memory is not necessary;
- $w_{eg}(x, y)$ can verify if a model is familiar with data from $\boldsymbol{D}_2$ or not. If the model is familiar with data, then $w_{eg}(x, y)$ will be close to 1, else it will be close to 0;
- It can be used for online updates.

The corresponding algorithm, ER-LWF, is detailed in Algorithm 5.

3.4 Experiments on Academic Datasets

In this section, we first present our experimental protocols under DS-IL in Section 3.4.1 and then introduce the methods we use in Section 3.4.2. Finally, the results are discussed in Section 3.4.3.

3.4.1 Experimental protocols

As we discussed in Section 3.2, we can generate a Smooth-CIL and a Smooth-DIL for classic CIL and DIL, respectively. In this section, we continue this idea and conduct experiments on two datasets: PACS and CIFAR-100. The details of the different setting is shown in Appendix A.1.

PACS is an image dataset originally for data generalization, as shown in Figure 3.6. It consists of four domains: photo, art painting, cartoon, and sketch. Each domain contains seven classes. For simplicity, each image is resized to 64×64 for all scenarios, and 20% of each domain is used to construct our test set. We repeated each experiment three times.

Algorithm 5 ER-LWF for continual learning

Train($\theta, \boldsymbol{D}, n, \mathcal{T}$)
$\boldsymbol{D}$: # dataset; θ: # model parameters;
$\mathcal{T}$: # of tasks; N: # of batches
for t = 1:T **do**
  for n = 1:N **do**
    sample data (x_n, y_n) from D_t;
    # CE: Cross-Entropy loss;
    $l_{current} = CE(\phi_\theta(x_n), y_n)$;
    if t >1 **then**
      # KD: knowledge distillation loss;
      # θ^*: previous model parameters;
      $l_{lwf} = KD(\phi_\theta(x_n), \phi_{\theta^*}(x_n))$
      # $w_{eg}(x_n, y_n)$: EG-weight;
      $l_{ER-LWF} = w_{eg}(x_n, y_n) * l_{lwf}$
    else
      $l_{ER-LWF} = 0$
    end if
    $l_{total} = l_{current} + l_{ER-LWF}$
    $Update(l_{total}, \theta)$
  end for
end for

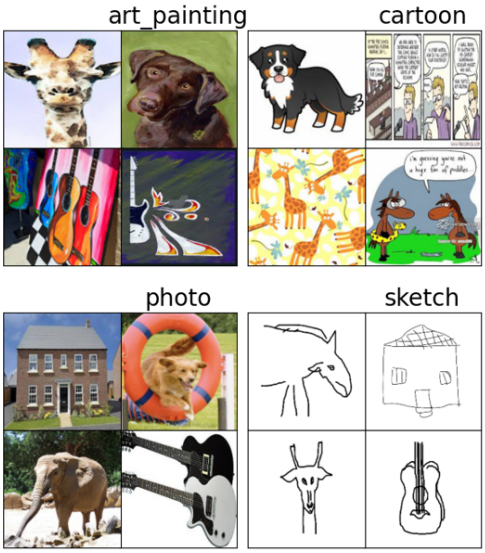


Figure 3.4: Samples from the four domains of PACS dataset[31]

First, for DIL, the dataset is divided into four tasks. Each task has one domain. The learning order arbitrarily chosen is: $art-painting \rightarrow cartoon \rightarrow photo \rightarrow sketch$.

We then generate six tasks containing all seven categories for Smooth-DIL. However, the four domains are mixed up with different proportions over the six tasks, and also the domain ratio is randomly generated. To compensate for this randomness, we generate three experiments with varying domain ratios.

In our CIL setting under the DS-IL perspective, each task contains all categories but with one dominant class and only a few exemplars of the others. Since PACS consists of seven object categories, we construct seven tasks by assigning each class as the dominant one in turn.

Analogous to Smooth-DIL, we define Smooth-CIL by splitting the data into six tasks where the category ratios vary gradually. The category proportions are randomly generated, and we repeat three independent experiments with different random seeds to ensure robustness.

CIFAR-100 [10] has 20 superclasses, where each superclass contains five classes. Each class comprises 600 images, of which 500 are used for training and 100 for testing. To use the dataset in our experiments, we need to create a notion of the domain in CIFAR-100. For example, there is a superclass *fish* that corresponds to five classes: *aquarium fish*, *flatfish*, *ray*, *shark*, *trout*. Each can be seen as a domain of *fish*, as shown in Figure 3.5. Thus, we use the superclass label as our new class label and the original class label as the domain label. Finally, the original

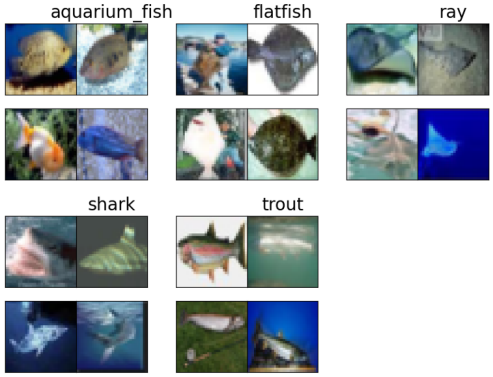


Figure 3.5: An example for the superclass *fish* of CIFAR-100 dataset [10], and it has five subclasses like: *aquarium fish*, *flatfish*, *ray*, *shark*, *trout*. In our experiments, each subclass corresponds to a domain.

CIFAR-100 is transformed into a dataset containing 20 classes and five domains. For simplicity, we keep the same training set and test set as the original version of CIFAR-100. We use the original image size that is 32*32*3. Each experiment is repeated three times.

Then, the rest is similar to the PACS dataset. For DIL, each superclass has five domains (subclasses). Thus, we generate five tasks for DIL. Then, we generate eight tasks containing different domain distributions for Smooth-DIL. We created ten tasks for CIL, where each task has two dominant classes. Finally, for Smooth-CIL, we produced eight tasks containing different class label distributions. We provide the details in the supplementary material to ensure completeness.

3.4.2 Training Methods

We use the standard PyTorch[69] implementation of ResNet-18[17] in both protocols. Because our method does not use memory, we mainly compare our approach to other regularization-based methods (online-EWC, SI, LWF without memory). We also test ER and LWF as the baselines of memory-based methods. FOO-VB[120], which targets the task agnostic CL scenario, is also tested for comparison.

We use **Cumulative** and **Fine-tuning** as the upper bound and lower bound, respectively. **Cumulative** means that all of the data seen so far are available for each training. **Fine-tuning** means that the previous model parameters are used as the initial parameters for the next task. Furthermore, the memory budget is set to 400 for PACS and to 1000 for CIFAR-100.

To ensure a fair comparison, we turn off the data augmentation for all experiments. We rely on the Adam optimizer[15], which is re-initialized for each task. All

hyperparameters are selected by a grid search on the validation set. Our network is trained on a single RTX 3070 8GB GPU and an i7-10700 8-core CPU.

3.4.3 Results

Table 3.1: Results on DIL, Smooth-DIL, CIL, and Smooth for the PACS dataset[31]. **AA** corresponds to Average Accuracy as Equation (2.11), while **BWT** stands for Backward Transfer defined in Equation (2.14).For both of them, the bigger, the better. The budget of memory-based methods is set at 400 images in total. The value here is the percentage and is the average accuracy of the model over all experiments after training on all tasks. We display the best without-memory method in **bold** font. Methods with an asterisk * use memory.

Methods	DIL		Smooth-DIL		CIL		Smooth-CIL	
	AA(%)↑	BWT(%)↑	AA(%)↑	BWT(%)↑	AA(%)↑	BWT(%)↑	AA(%)↑	BWT(%)↑
Cumulative(upper)	54.2 ± 1.3	0.8 ± 0.1	59.5 ± 1.1	6.3 ± 0.3	58.0 ± 1.6	14.7 ± 0.8	58.91 ± 1.4	7.8 ± 0.5
F.T. (lower)	30 ± 3.5	-38.9 ± 4.2	54.2 ± 2.5	4.5 ± 0.5	30.0 ± 2.9	4.7 ± 0.5	50.6 ± 2.6	2.4 ± 0.3
Online-EWC[43]	29.6 ± 2.8	-39.7 ± 4.5	54.8 ± 1.8	$4.6 \pm \pm 0.8$	30.7 ± 2.5	5.4 ± 0.6	50.8 ± 2.5	2.3 ± 0.3
SI[39]	$\mathbf{31.6 \pm 2.5}$	-38.5 ± 4.1	55.8 ± 1.7	$\mathbf{5.6 \pm 0.8}$	30 ± 2.8	4.3 ± 0.6	51.1 ± 2.2	2.7 ± 0.4
FOO-VB[120]	29.6 ± 2.5	$\mathbf{-21.3 \pm 2.7}$	58.5 ± 1.3	5.3 ± 0.6	38.5 ± 2.5	7.5 ± 0.8	55.4 ± 2.5	3.8 ± 0.6
LWF[23]	27.7 ± 3.1	-26.8 ± 2.5	57.4 ± 1.5	5.5 ± 0.7	29.5 ± 2.7	8.2 ± 1.2	55 ± 2.4	5.4 ± 0.5
ER*[61]	44.3 ± 1.8	-12.1 ± 1.5	54.4 ± 1.7	4.1 ± 0.6	37.8 ± 2.3	4.1 ± 0.6	52 ± 1.6	2.8 ± 0.3
LWF(memory)*[23]	47.7 ± 1.7	-7.1 ± 1.1	57.9 ± 1.1	5.9 ± 0.5	39.8 ± 2.2	4.2 ± 0.5	56 ± 1.5	5.3 ± 0.3
ER-LWF (ours)	30.2 ± 2.5	-32.5 ± 3.2	$\mathbf{58.9 \pm 1.3}$	$\mathbf{5.6 \pm 0.5}$	$\mathbf{38.8 \pm 2.5}$	$\mathbf{10.7 \pm 1.0}$	$\mathbf{56.1 \pm 1.8}$	$\mathbf{5.5 \pm 0.4}$

3.4.3.1 PACS results

Table 3.1 and Figure 3.6 show the results on the PACS dataset[31]. The results on the Smooth-CIL and Smooth-DIL show that our ER-LWF outperforms all other methods, including even the memory-based ones. In the smooth version, the similarities between different tasks are magnified. Thus, current data are more likely to reconstruct the previous loss function. That is why our method performs well and can even reach the upper bound.

From the DIL column in Table 3.1, we find that all regularization-based methods fail in this scenario, and that their performance is close to the lower bound (Fine-tuning), including our ER-LWF approach. In contrast, memory-based approaches perform better.

For the results on the PACS CIL, our ER-LWF performs best compared with other regularization-based methods, and its performance is very close to that of LWF with memory. However, there is still a big gap between the upper bound and other methods. This means that only replaying the previous data is not enough, and that the loss function or the gradient requires more constraints.

To conclude, our method does not work on DIL but performs well on the others. Furthermore, it consistently performs better than the LWF without memory.

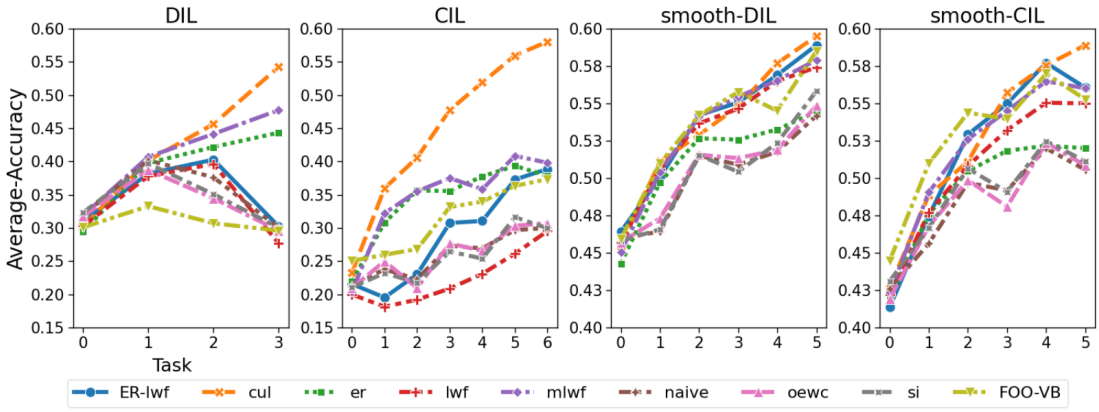


Figure 3.6: Results related to the PACS dataset[31]. Testing accuracy (average on three runs) is provided after training on each task for different methods and different scenarios.

3.4.3.2 CIFAR-100 results

The results on the CIFAR-100 dataset are presented in Table 3.2 and Figure 3.7. As for Smooth-DIL, CIL, and Smooth-CIL, similar phenomena to the PACS experiments can be observed. Our proposed ER-LWF outperforms all other regularization-based methods, and its performance is even better than memory-based methods for smooth scenarios.

For the DIL setting with CIFAR-100, we do not observe the same result as for PACS. The LWF family's methods all perform well. This can be explained by the way we define the DIL scenario with CIFAR-100. As discussed in Section 3.4.1, CIFAR-100 has 20 superclasses, and each superclass contains five subclasses. In the proposed DIL protocol, each subclass is considered to be a domain, thereby leading to slight differences between tasks. As a result, we can reconstruct the previous objective more efficiently compared with the PACS DIL.

We also find that FOO-VB, which initially targets the task agnostic CL scenario (an extremely smooth scenario), is close to our method for the Smooth-DIL and Smooth-CIL scenarios. However, our method performs much better when data distribution shifts more rapidly, as shown in the DIL and CIL plots of Figure 3.7.

Compared to LWF, we designed EG-weight Equation (3.10) to self-regulate the original LWF loss. This can give more weight to similar data and less weight to unrelated data. Thus, the model can have more plasticity when the current data are different and more stability when the current data are similar. This is confirmed by the results obtained on the CIFAR-100 and PACS, from which we can observe that our method consistently outperforms or matches the performance of LWF without memory.

Table 3.2: Results on the CIFAR-100 dataset[10]. The budget for memory-based methods is set at 1000. It uses the same setting as Table 3.1.

Methods	DIL		Smooth-DIL		CIL		Smooth-CIL	
	AA(%)↑	BWT(%)↑	AA(%)↑	BWT(%)↑	AA(%)↑	BWT(%)↑	AA(%)↑	BWT(%)↑
Cumulative(upper)	38.8 ± 1.0	-1.4 ± 0.2	41.7 ± 1.2	8.8 ± 0.5	42.9 ± 1.1	14.5 ± 0.7	41.4 ± 1.2	9.2 ± 0.5
F.T. (lower)	28.4 ± 1.5	-26.5 ± 1.2	35.3 ± 1.3	4.7 ± 0.6	15.5 ± 2.9	1.7 ± 0.5	33.1 ± 1.8	3.7 ± 0.5
Online-EWC[43]	30.3 ± 1.6	-30.7 ± 1.5	36.1 ± 1.3	4.6 ± 0.8	15.6 ± 3.1	2.2 ± 0.8	34.4 ± 1.8	4.2 ± 0.4
SI[39]	31.0 ± 1.4	-30.5 ± 1.6	37.7 ± 1.5	6.7 ± 0.8	15.5 ± 2.8	1.4 ± 0.6	33.1 ± 2.3	3.2 ± 0.5
FOO-VB[120]	31.4 ± 1.5	-29.9 ± 1.3	41.4 ± 1.6	7.5 ± 1.1	18.5 ± 2.2	3.4 ± 1.0	40.2 ± 1.5	6.2 ± 0.6
LWF[23]	36.9 ± 1.8	-19.6 ± 1.0	39.8 ± 1.4	5.9 ± 0.7	23.9 ± 2.5	5.9 ± 0.9	38.0 ± 1.6	5.2 ± 0.6
ER*[61]	31.3 ± 1.2	-23.0 ± 1.1	36.2 ± 1.2	4.4 ± 0.5	24.1 ± 1.5	3.4 ± 0.5	36.2 ± 1.5	5.5 ± 0.3
LWF(memory)*[23]	36.7 ± 1.3	-4.8 ± 0.5	39.8 ± 1.2	5.9 ± 0.6	26.0 ± 1.8	7.5 ± 0.6	38.3 ± 1.4	6.9 ± 0.6
ER-LWF (ours)	$\mathbf{37.0 \pm 1.5}$	$\mathbf{-12.6 \pm 0.8}$	$\mathbf{41.4 \pm 1.2}$	$\mathbf{7.9 \pm 0.8}$	$\mathbf{25.8 \pm 2.0}$	$\mathbf{8.2 \pm 0.6}$	$\mathbf{40.3 \pm 1.4}$	$\mathbf{7.9 \pm 0.5}$

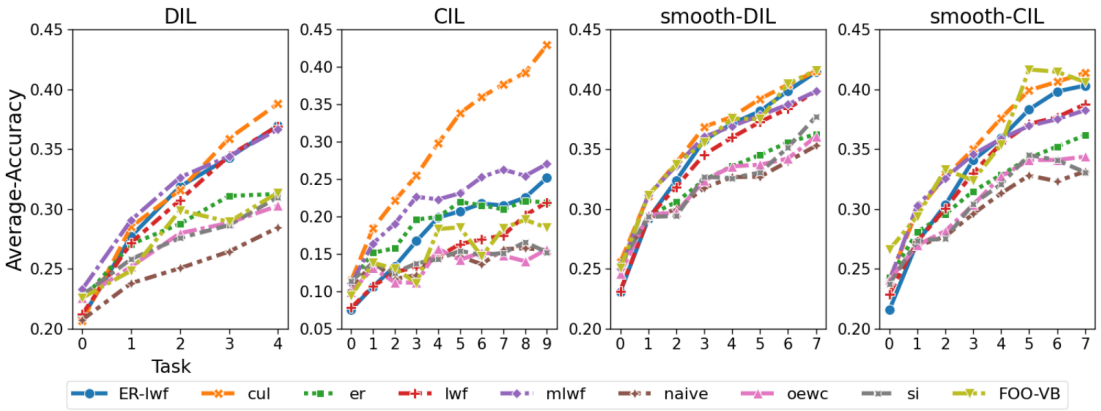


Figure 3.7: Results related to the CIFAR-100 dataset[10].Testing accuracy (average over three runs) is provided after training on each task for different methods and different scenarios.

3.5 Experiments on FairWastes Dataset

3.5.1 FairWastes Dataset

The original FairWaste Dataset comprises two folders: one for training and one for testing, each containing images captured from various sensors, as illustrated in Figure 3.8. Each image is accompanied by an object detection label in the COCO format [16]. For classification tasks, we initially segmented the original images into multiple sub-images using the mask labels, ensuring that each sub-image contained only one object, as depicted in Figure 3.9.

After performing image segmentation, we structured the dataset as outlined in Table 1, which contains 13 distinct material categories. The dataset exhibits an imbalanced distribution of objects: materials such as bois, carton, plâtre, and film are overrepresented, while others, including pile, condensateur, polystyrène,

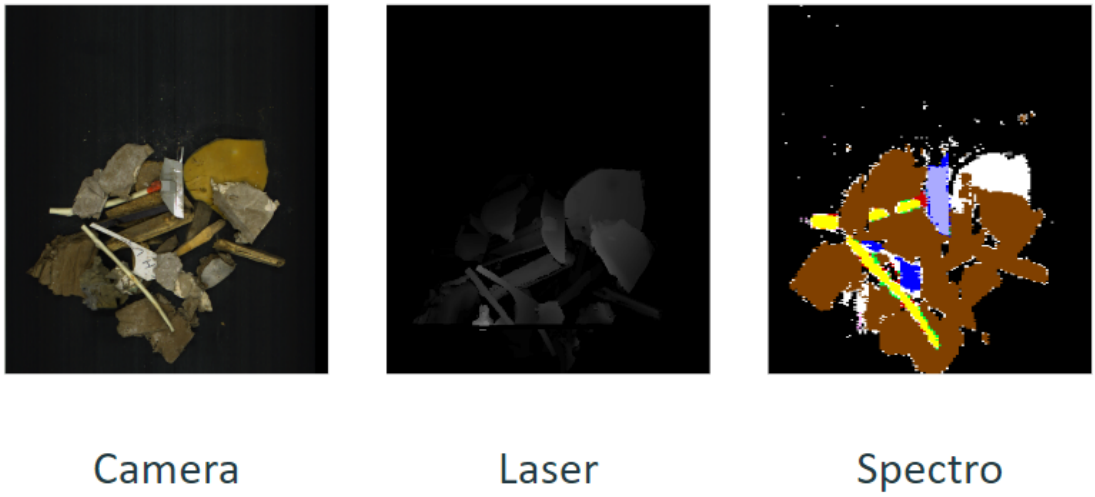


Figure 3.8: Three images from three different sensors respectively

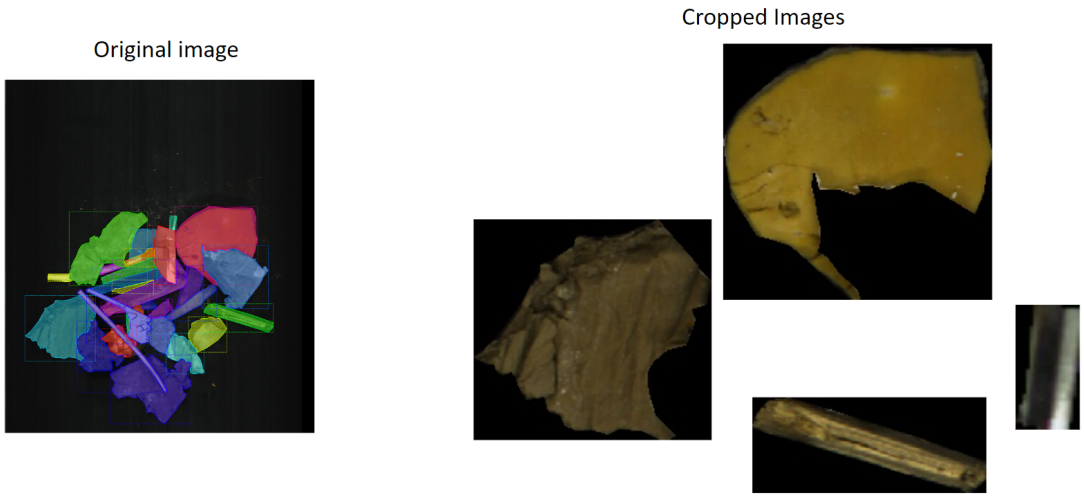


Figure 3.9: Cropping the original images for Classification purpose

textile, and gravat, are underrepresented. Additionally, some materials appear exclusively in the training set, whereas others are only present in the test set. To enhance the dataset's usability, we have restructured it to achieve a more balanced distribution. Specifically, the test set now contains a number of samples for each material that is proportional to its overall occurrence in the dataset. All experiments reported in this study have been conducted on this revised version of the dataset.

Material	Train	Test
Bois	3382	2900
Carton	1188	241
Plâtre	1012	52
Plastique	852	853
Film	406	1
Ferreux	295	21
Mousse	269	451
Papier	215	218
Pile	186	0
Condensateur	101	0
Polystyrène	0	98
Textile	0	21
Gravat	0	12

Table 3.3: Original DataSet

Material	Train	Test
Bois	6047	200
Carton	1216	200
Plâtre	894	200
Plastique	1491	200
Mousse	617	100
Ferreux	259	50
Film	353	50
Papier	385	50
Pile	186	50
Condensateur	51	50
Polystyrène	88	10
Textile	16	5
Gravat	7	5

Table 3.4: Revised Dataset

3.5.2 Scenario

In the FairWaste Dataset, data are derived from various scenes, which can be interpreted as distinct domains as illustrated in Figure 3.10. We have designated each 'acquisition' as an individual task where the domain and the classed varies from different tasks. Thus, we have 26 distinct tasks in total. This scenario is denoted as Domain-Class-Inc in this thesis.

3.5.3 Results

The results are shown in Table 3.5. In this real-world scenario, different 'acquisitions' are considered as a single task, and different tasks have overlaps. The results demonstrate that our method can even outperform memory-based methods such as ER and LWF with memory.

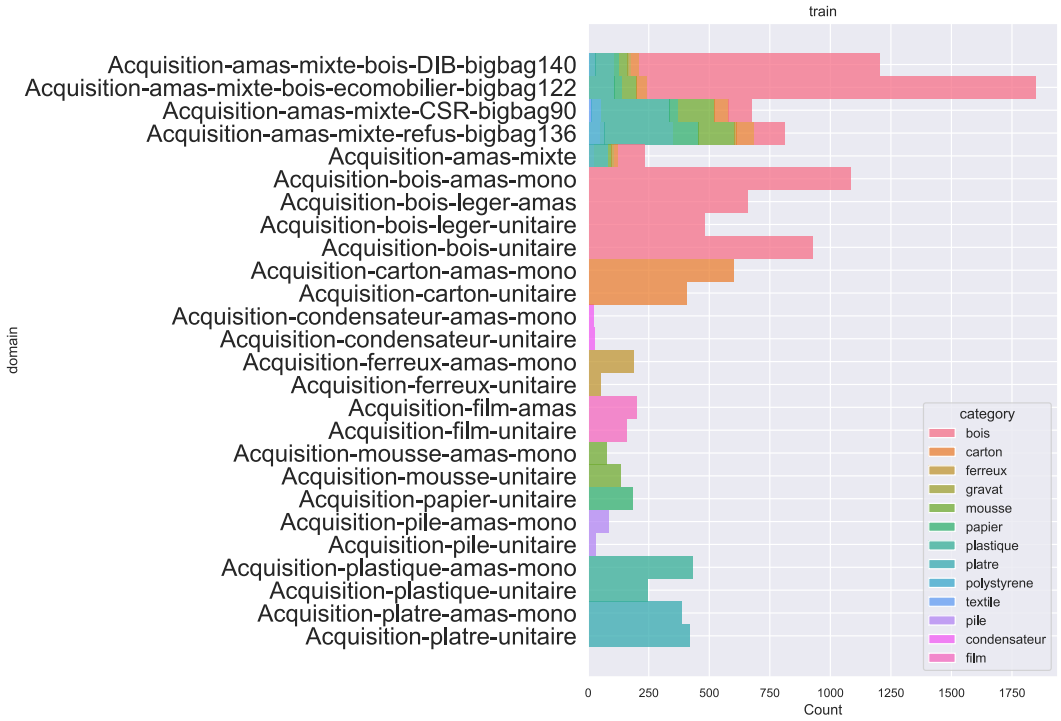


Figure 3.10: The distribution of Dataset based on the different scenes

Table 3.5: Results on the FairWastes dataset. The memory budget for memory-based methods is set to 400. The same methods as in Table 3.1 are used. We report the final classification accuracy, averaged over three runs.

Methods	Cumulative	F.T.	Online-EWC	SI	FOO-VB	LWF	ER*	LWF*(memory)	**Ours**
AA(%)↑	81.5±1.5	37.4±3.3	39.2±4.2	40.1±3.5	42.6±2.9	39.7 ± 3.1	48.8±2.4	52.2±2.2	**53.5**±2.7

3.6 Limitations

In this work, to better fit real-life applications, we make an assumption that task boundaries are blurred, in other words their data distributions overlap or at least have a certain degree of similarity. Smooth-CL setting, *i.e.*, Smooth-DIL and CIL, and regular CL setting, *i.e.* DIL and CIL, correspond to high-level and low-level similarity, respectively. However, if the task boundaries are well-defined with no overlapping of domains, then our method may fail since it makes use of the current data and the previous model output to approximate the previous loss function. If the previous model cannot recognize or provide a meaningful output, the EG-weight Equation (3.10) will be close to zero. Then, our method degenerates to **Fine-tuning**. In our experiments, the most difficult CL scenario is the DIL setting on the PACS dataset. Since we assign each domain to a single task in DIL, the similarity between the data of different domains may be very slight when the gap between different domains is too large. In this situation, we cannot find similar data from the current task, meaning that the previous loss function cannot be reconstructed. From the DIL plot of Figure 3.6, we can see that model performance degenerates rapidly in the final task. In our experiments, the last task of the DIL corresponds to *sketch*, which is the domain that is most different. We think that this is what accounted for this poor performance.

3.7 Conclusion

In this section, we propose a new continual learning paradigm, namely Distribution-Shift Incremental Learning (DS-IL), which, by considering soft task boundaries as encountered in real-world applications, subsumes traditional Domain-Incremental Learning (DIL) and Class-Incremental Learning (CIL) scenarios. Furthermore, we propose ER-LWF, a novel CL method which extends LWF to deal with CL under the DS-IL setting where there are no well-defined task boundaries. It uses the entropy information of the previously learned model's output on the current data to self-regulate the original knowledge distillation loss. This EG-weight can provide the model with more stability when the current data are similar to the previous data, or with more plasticity otherwise. Based on the results of the PACS and CIFAR-100 datasets, we show that our proposed CL method, without memory, consistently outperforms the classic LWF (without memory) and can reach the upper bound for the Smooth-CL scenarios. Additionally, on the FairWastes dataset, our method continues to outperform other baselines.

Chapter 4

Online Continual Learning in both Balanced and Imbalanced Data Environments

Because our previous method struggled when data distributions shifted rapidly, this chapter addresses a more challenging scenario: imbalanced online continual learning, where data distributions change quickly and task boundaries are not defined. We present our work on this topic, which is partially published in "Imbalanced data robust online continual learning based on evolving class aware memory selection and built-in contrastive representation learning"[159] and "Adaptive Class Aware Memory Selection and Contrastive Representation Learning for Robust Online Continual Learning in both Balanced and Imbalanced Data Environments".

4.1 Motivation

Continual learning (CL) seeks to enable models to learn from data streams over extended periods without retaining access to previously encountered samples. However, when new information arrives, catastrophic forgetting can occur, erasing previously acquired knowledge [30, 39]. Most traditional CL methods—including regularization-based [30, 39, 27], parameter isolation-based [25, 117], and rehearsal-based approaches [61, 59, 119] assume either clearly defined task boundaries or relatively balanced data distributions. Such assumptions simplify the learning process, but rarely hold in practical applications.

By contrast, real-world data streams are often non-stationary and highly imbalanced [74, 126], and they may include domain shifts that introduce entirely new feature distributions over time [131, 31, 51]. In scenarios where the model receives

continuous input without predefined tasks—sometimes referred to as task-free online continual learning (OCL) [40]—the learning algorithm must dynamically adjust class boundaries and representations. Meanwhile, existing rehearsal-based methods commonly rely on strategies such as random reservoir sampling [61] or herding [24], which overlook how imbalanced or evolving data distributions affect both class diversity and inter-class boundaries.

Additionally, while contrastive learning [90] has proven effective for improving representation quality, many CL approaches do not fully leverage sample selection to maximize the benefit of replay. They typically focus on augmentations and instance-level similarities within a mini-batch while underestimating how a carefully selected memory buffer can help adapt previously learned representations to new data. This gap becomes especially problematic when classes are introduced gradually or domains change abruptly, limiting the model's ability to refine its learned boundaries in an online manner.

To address these challenges, we propose a memory-based online CL method that emphasizes class and domain-aware sample selection alongside contrastive representation consolidation. Our method, **Memory Selection and Contrastive Representation Learning (MSCL)**, continually updates a diverse memory set that captures the evolving complexity of class boundaries and domain shifts, while a novel contrastive loss aligns new samples with historical knowledge. Through this combination of adaptive memory selection and enhanced representation learning, MSCL aims to mitigate catastrophic forgetting and thrive in non-stationary, imbalanced streaming environments.

4.2 Preliminary and problem statement

We consider the setting of online task-free continual learning. The learner receives non-stationary data stream $\mathbb{O}$ through a series of data batches denoted as $\mathbb{S}_t^{str} = (x_i, y_i)_{i=1}^{N_b}$ at time step t. Here, (x_i, y_i) represents an input data and its label, respectively, and N_b denotes the batch size. The learner is represented as $f(\cdot; \boldsymbol{\theta}) = g \circ F$, where g represents a classifier and F denotes a feature extractor. We define a memory set as $\mathbb{S}^{mem} = (x_j, y_j)_{j=1}^{M}$, where M is the memory size. We use the function $l(\cdot, \cdot)$ to denote the loss function. The global objective from time step 0 to T can be computed as follows:

$$l^* = \sum_{t=0}^{T} \sum_{(x_i, y_i) \in \mathbb{S}_t^{str}} l(f(x_i; \boldsymbol{\theta}), y_i) \tag{4.1}$$

However, within the setting of online continual learning, the learner does not have access to the entire data at each training step but only the current data batch

and those in the memory set if any memory. Therefore, the objective at time step T can be formulated as follows:

$$l_T = \underbrace{\sum_{\mathbb{S}_T^{str}} l(f(x_i; \boldsymbol{\theta}_{T-1}), y_i)}_{\text{current loss}} + \underbrace{\sum_{\mathbb{S}^{mem}} l(f(x_j; \boldsymbol{\theta}_{T-1}), y_j)}_{\text{replay loss}} \tag{4.2}$$

As a result, to enable online continual learning without catastrophic forgetting, one needs to minimize the gap between l^* and l^T:

$$\min(l^* - l_T) = \min(\sum_{t=0}^{T-1} \sum_{\mathbb{S}_t^{str} \setminus \mathbb{S}^{mem}} l(f(x_i; \boldsymbol{\theta}_{T-1}), y_i)) \tag{4.3}$$

In this paper, we are interested in memory-based online CL. Our objective is to define a strategy which carefully selects data samples to store in the memory set and continuously refines data representation to minimize the gap as shown in Equation (4.3).

4.3 Methodology

The proposed method, denoted as **MSCL**, consists of two main components, namely Feature-distance based sample selection (FDBS) (sect.4.3.1) and contrastive learning for better discriminative feature representation (sect.4.3.2). The whole algorithm is sketched in algo.6.

4.3.1 Feature-Distance based sample selection

In the context of imbalanced online domain and class continual learning scenarios, models need to contend with at least two types of distribution shifts: correlation shift and diversity shift. In classification problems, these distribution shifts can result in increased inter-class similarity and intra-class variance, ultimately leading to catastrophic forgetting. Current memory selection methods (e.g., ER [36], CBRS [80], GSS [59], OCS [119]) are unable to effectively address both of these challenges simultaneously. To tackle this issue, we introduce our feature-based method, referred to as Feature-Based Dissimilarity Selection (FDBS). FDBS encourages the model to select data points that are the most dissimilar within a class and the most similar between different classes. This strategy aims to enhance both inter-class similarity and intra-class variance within the memory set. Consequently, FDBS helps to narrow the gap between the memory set and the true data distribution, as highlighted in Equation 4.3.

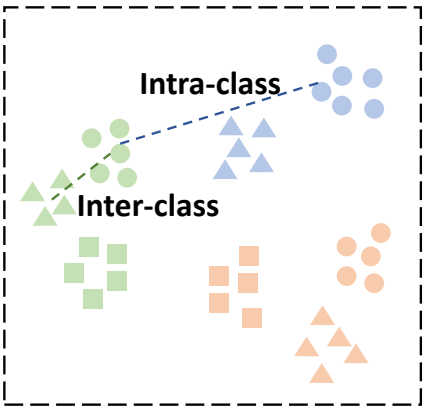


Figure 4.1: We illustrate domains using colors and categories with shapes. It shows models adapting to datasets with high inter-class similarity and intra-class variance, highlighting the challenge of differentiating closely related categories.

Let M denote the memory size and K the number of data samples so far streamed. Let p denote our projection head. The current batch size is set to N_b, and the sampled memory batch size is N_m. When the learner receives a batch of data $\mathbb{S}^{str}$ from the stream $\mathbb{O}$, we check for each new data sample x_i in $\mathbb{S}^{str}$ whether the memory set is full. If it is not full, we can directly store x_i. However, if the memory set is full, we need to evaluate the importance weight w_i of the new data sample x_i to determine whether it is worth storing. The key to this process is to keep the memory set aware of intra-class diversity and inter-class boundaries based on the feature distances between the new data sample x_i and the memory set. It involves the following three main steps:

- We sample a batch of data, denoted as $\mathbb{S}^m$, from the memory set with size N_m. Generate double-view augmentations for both batches. $\mathbb{S}^m_{doub}$ contains both the original images and the augmented views from the memory batch. Apply the same notation to $\mathbb{S}^{str}_{doub}$. To get the features, we use $z(x) = p \circ F(x)$.

- We then calculate the feature distance, denoted as $\boldsymbol{D}$ (refer to Equation (4.6)), between every data point in the set $\mathbb{S}^{str}_{doub}$ and each data sample stored in $\mathbb{S}^m_{doub}$. Subsequently, we identify the minimum distance between the input data and the memory set for each input data sample, resulting in the vector $\boldsymbol{d}^{str}$ as defined in Equation (4.4)

$$\boldsymbol{D}_{i,j} = dist\left\{z(x_i), z(x_j)\right\}_{(x_i \in \mathbb{S}^{str}_{doub}; x_j \in \mathbb{S}^m_{doub})} \tag{4.4}$$

- Subsequently, we compute $\boldsymbol{D}^{mem}$, as in Equation (4.6), the feature distance between every data in $\mathbb{S}^m_{doub}$ and $\mathbb{S}^{mem}$, and the minimum distance for each

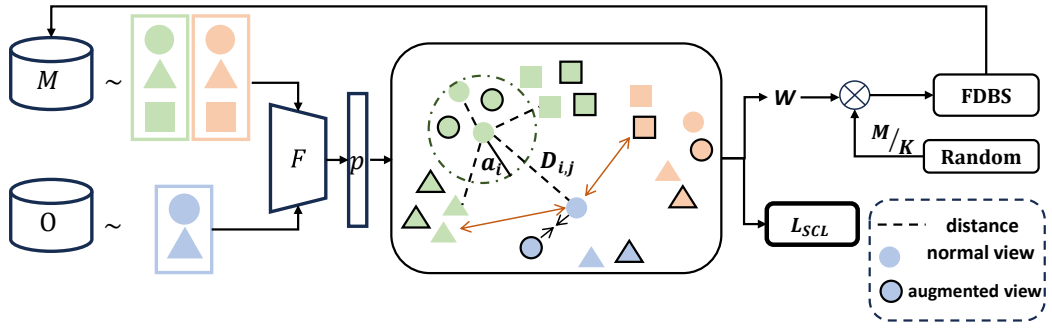


Figure 4.2: We illustrate domains using colors and categories with shapes.Our proposed MSCL involves mapping input data and a memory set into a shared feature space. Here, $\boldsymbol{D}_{i,j}$ represents the distance between input data x_i and data x_j in the memory set. We use the same indexing convention for other formulas. We calculate distances, $\boldsymbol{D}$ and $\boldsymbol{a}$, between input data and memory set, and then derive an importance weight matrix quantifying each input data representative importance w.r.t those in the memory set based on the analysis of their intra-class diversity or inter-class similarity in the feature space. These importance weights are combined with random selection to give birth to our Feature-Distance based Sample Selection (FDBS) which identifies the most representative input data points for storage into the memory set. Armed with this importance weight matrix, we proceed to craft a novel Contrastive Loss (SCL) aimed at refining the feature space by compacting intra-class data and creating greater separation among inter-class data.

data point in the memory set in $\boldsymbol{d}^{mem}$, as shown in Equation (4.5). We then calculate $\boldsymbol{a}$ as in Equation (4.7) a weighted average distance from a data point in the memory set to all other points, using a RBF kernel as in Equation (4.7) to weight the distances. We aim to assign higher weight to closer distances.

$$\boldsymbol{D}^{mem}_{i,j} = dist\left\{z(x_i), z(x_j)\right\}_{(x_i \in \mathbb{S}^m_{doub}, x_j \in \mathbb{S}^{mem})} \tag{4.5}$$

$$\boldsymbol{d}^{str}_i = min(\boldsymbol{D}_{i,:}); \boldsymbol{d}^{mem}_i = min(\boldsymbol{D}^{mem}_{i,j \neq i}) \tag{4.6}$$

- By computing the difference between $\boldsymbol{a}$ and $\boldsymbol{D}$, we can derive an **importance weight** for each new data. This weight is subsequently combined with the reservoir sampling coefficient to determine the probability of selecting the new data point.

$$\boldsymbol{\alpha}_{i,j} = e^{-\frac{\left\|\boldsymbol{D}^{mem}_{i,j} - \boldsymbol{d}^{mem}_{i}\right\|^2}{2\sigma^2}}; \; \boldsymbol{a}_i = \frac{\sum_{j\neq i}^{M} \boldsymbol{D}^{mem}_{i,j}\boldsymbol{\alpha}_{i,j}}{\sum_{j\neq i}^{M} \boldsymbol{\alpha}_{i,j}} \tag{4.7}$$

Importance weight is the core concept of our proposed method. It serves to assess the significance of a new data sample with respect to the memory set, with a focus on promoting diversity among previously encountered intra-class data while also considering the potential closeness to inter-class boundaries. Specifically, we calculate this importance weight, as defined in Equation (4.9), to capture the influence of each data point in the memory set on an input data sample. This influence is determined by whether they belong to the same class, as illustrated in Figure 4.2. Our approach is based on the intuitive notion that when two points, x_i and x_j, are closer in proximity, the impact of x_j on x_i becomes more pronounced. To achieve this, we employ a Radial Basis Function (RBF) kernel, as expressed in Equation (4.8). This kernel ensures that the influence of distant points diminishes rapidly. Additionally, we use the sign function, as shown in Equation (4.8), to assign a value of 1 if the classes are the same and -1 otherwise.

When comparing a new data sample x_i with a memory set data point x_j, we consider two scenarios based on their class labels. If they share the **same class label**, as shown in Figure 4.2, and if the feature distance $\boldsymbol{D}_{i,j}$ significantly exceeds $\boldsymbol{a}_j$, it implies a substantial difference between x_i and x_j. In this case, we assign $\boldsymbol{W}_{i,j}$ a value greater than 1, promoting the selection of x_i for storage. However, when x_i and x_j have **different class labels**, we aim to store data points near decision boundaries to capture closer class boundaries caused by increased inter-class similarities. We achieve this by setting $\boldsymbol{W}_{i,j}$ using Equation (4.9) with the sign function returning -1. If $\boldsymbol{a}_j$ significantly surpasses $\boldsymbol{D}_{i,j}$, it implies that despite their different labels, x_i closely resembles x_j, motivating us to store x_i. Conversely, if $\boldsymbol{a}_j$ is substantially smaller than $\boldsymbol{D}_{i,j}$, it suggests that the model can readily distinguish between x_i and x_j, leading us to exclude x_i from storage. When $\boldsymbol{D}_{i,j}$ is approximately equal to $\boldsymbol{a}_j$, we consider x_i as a typical data point close to x_j, leading $\boldsymbol{W}_{i,j}$ to approach 1, resulting in a random selection.

$$\boldsymbol{\beta}_{i,j} = e^{-\frac{\left\|\boldsymbol{D}_{i,j} - \boldsymbol{d}^{str}_{i}\right\|^2}{2\tau^2}}; sgn(y_i, y_j) = \begin{cases} 1 \text{ if } y_i = y_j \\ -1 \text{ if } y_i \neq y_j \end{cases} \tag{4.8}$$

$$\boldsymbol{W}_{i,j} = e^{sgn(y_i,y_j)\frac{\boldsymbol{D}_{i,j}-\boldsymbol{a}_j}{\boldsymbol{D}_{i,j}+\boldsymbol{a}_j}\boldsymbol{\beta}_{i,j}} (y_i \in \mathbb{S}^{str}_{doub}; y_j \in \mathbb{S}^{m}_{doub}) \tag{4.9}$$

To take into account the influence of all data points in the memory set on a new input data point for its importance weight, we directly multiply the impact of each memory point as shown in Equation (4.10).

To get the final probability p_i for a new data sample x_i to be chosen for storage in memory, we introduce the reservoir sampling. Given a fixed memory size M and the number of data samples observed so far in the data stream, denoted as K, M/K represents the probability of each data sample being randomly selected. We then use the importance weight $\boldsymbol{w}_i$ to adjust the probability of the new data sampled x_i being selected, as shown in Equation (4.10). This allows us to handle imbalanced data and retain a certain level of randomness.

$$\boldsymbol{w}_i = \frac{\sum_{j=1}^{2N_m} \boldsymbol{W}_{i,j}}{2N_m} \quad ; \quad p_i = min(\boldsymbol{w}_i \frac{M}{K}, 1) \tag{4.10}$$

4.3.2 Contrastive learning for better discriminative feature representation

Our Feature-Distance Based Sample Selection (FDBS) can effectively store the most representative samples during training. However, the latent space of our memory set may not be compact, potentially degrading our classification performance. To address this issue, we introduce the use of contrastive learning loss. Previous methods, such as OnPro[145] and CaSSLe[108], have already employed supervised contrastive learning[90] to learn instance-wise representations:

$$\begin{aligned} L_{\text{SUP}} = & \sum_{i=1}^{2N_b} \frac{1}{|I_i^b|} \sum_{j \in I_i^b} \log \left(\frac{\exp(\text{sim}(z_i^b, z_j^b)/\tau_{sc})}{\sum_{k \neq i} \exp(\text{sim}(z_i^b, z_k^b)/\tau_{sc})} \right) \\ & + \sum_{i=1}^{2N_m} \frac{1}{|I_i^m|} \sum_{j \in I_i^b} \log \left(\frac{\exp(\text{sim}(z_i^m, z_j^m)/\tau_{sc})}{\sum_{k \neq i} \exp(\text{sim}(z_i^m, z_k^m)/\tau_{sc})} \right) \end{aligned} \tag{4.11}$$

Where N_b and N_m represent the number of training data in the current batch and the batch sampled from the memory set, respectively. I_i is the set of positive samples for z_i. This equation separately computes the supervised contrastive loss for current data and data from the memory set. However, it overlooks the distance between the memory set and current data. To address this issue, we propose the use of an importance weight to compute a specific contrastive learning loss.

The importance weight $\boldsymbol{W}_{i,j}$, derived from Equation (4.9), measures feature space similarity between data points and is differentiable. Inspired by contrastive learning's goal to distinguish between similar (positive) and dissimilar (negative) sample pairs. IWL aims to decrease inter-class similarity and intra-class variance, serving as an adversarial element to memory selection and compacting the feature space for better memory selection. For a data batch of size N_b, we select a mini-batch from the memory set of size N_m, and compute L_{IWL} as per Equation (4.12),

Algorithm 6 Train a batch at time step t

Input: F, g ,$\mathbb{S}^{mem}$, $\mathbb{S}^{str}$, n, K, $\mathbf{Z}^{mem}$ stores the features of the memory set, N_b is the current batch size, and N_m is the memory batch size.

for n steps **do**
  sample batch $I, \mathbf{X}^m, \boldsymbol{y}^m$ of size N_m from $\mathbb{S}^{mem}$ $\{I$: the index of the samples in $S^{mem}\}$
  $\mathbf{X}^{str}, \boldsymbol{y}^{str} = \mathbb{S}^{str}$
  $\mathbf{X}^m_{doub} = cat(aug(\mathbf{X}^m), \mathbf{X}^m)$
  $\mathbf{X}^{str}_{doub} = cat(aug(\mathbf{X}^{str}), \mathbf{X}^{str})\{aug()$ denotes stochastic data augmentation$\}$
  $\mathbf{Z}^m, \hat{\boldsymbol{y}}^m = p \circ F(\mathbf{X}^m_{doub}), g \circ F(\mathbf{X}^m_{doub})$
  $\mathbf{Z}^{str}, \hat{\boldsymbol{y}}^{str} = p \circ F(\mathbf{X}^{str}_{doub}), g \circ F(\mathbf{X}^{str}_{doub})$
  $\alpha = 0.1 + 0.9 * 0.999^t$ {replay-scheduling coefficient}
  Current Loss : $L_{cur} = \ell(\hat{\boldsymbol{y}}^{str}, \boldsymbol{y}^{str})$
  Replay Loss : $L_r = \ell(\hat{\boldsymbol{y}}^m, \boldsymbol{y}^m)$
  Update $\mathbf{Z}^{mem}[I] = \mathbf{Z}^m[: N_m]$
  $\boldsymbol{D}^{mem} = dist(\mathbf{Z}^m, \mathbf{Z}^{mem})$ as Equation (4.5)
  Compute $\boldsymbol{a}$ based on Equation (4.7)
  $\boldsymbol{D} = dist(\mathbf{Z}^{str}, \mathbf{Z}^m)$ as Equation (4.4)
  Compute $\boldsymbol{w}$ based on Equation (4.9) and Equation (4.10)
  $L_{IWL} = L_{IWL}(\boldsymbol{w})$ as Equation (4.12)
  $L_{SUP} = L_{SUP}(\mathbf{X}^{str}, \boldsymbol{y}^{str}, \mathbf{X}^{mem}, \boldsymbol{y}^{mem})$ as Equation (4.11)
  Total Loss : $L = \alpha L_{cur} + (1-\alpha)L_r + L_{IWL} + L_{SUP}$
  Update: F, g : Adam.step( )
  FDBS($\mathbb{S}^{mem}$,$\mathbb{S}^{str}_t$,$\boldsymbol{w}$,$\boldsymbol{D}$,M,K,$\mathbf{Z}^{mem}$) as shown in Algorithm 7
end for

optimizing $\boldsymbol{W}_{i,j}$ to align data points with matching class labels closer and separate those with differing labels.

$$L_{IWL} = \frac{\sum_{i=1}^{2N_m} \sum_{j=1}^{2N_b} log(\boldsymbol{W}_{i,j})}{\sum_{i=1}^{2N_m} \sum_{j=1}^{2N_b} \boldsymbol{\beta}_{i,j}} \tag{4.12}$$

Thus, our total contrastive learning loss is:

$$L_{SCL} = L_{SUP} + L_{IWL} \tag{4.13}$$

Algorithm 7 FDBS at time step t

Input: $\mathbb{S}^{mem}$, $\mathbb{S}^{str}$, $\boldsymbol{w}$, $\boldsymbol{D}$, M, K, $\mathbf{Z}^{mem}$

$\mathbf{X}^{mem}, \boldsymbol{y}^{mem} = \mathbb{S}^{mem}$;
for each data $i, (x_i, y_i)$ in $\mathbb{S}_t^{str}$ **do**
  $K = K + 1$
  if $len(\mathbb{S}^{mem}) < M$ **then**
    store (x_i, y_i) in $\mathbb{S}^{mem}$
  else
    $p = \boldsymbol{w}_i * M/K$
    $r = random.rand()$
    if $r < p$ or $y_i \notin \mathbb{S}^{mem}$ **then**
      $c = most_frequent(\boldsymbol{y}^{mem})$
      $I = index(\boldsymbol{y}^{mem} == c)$
      $k = random.choice(I)$
      $\mathbf{X}^{mem}[k], \boldsymbol{y}^{mem}[k] = x_i, y_i$;
      $\mathbf{Z}^{mem}[k] = Z(x_i)$
    else
      ignore (x_i, y_i)
    end if
  end if
end for

4.4 Experiments and Results

We introduce balanced CL benchmarks in sect.4.4.1, define imbalanced ones in sect.4.4.2, describe the baselines and implementations details in sect.4.4.3, and present the experimental results both on balanced scenarios in sect.4.4.4 and imbalanced ones in sect.4.4.5.

4.4.1 Balanced benchmarks

Building upon previous research [73, 59, 81], we utilize three well-established Continual Learning (CL) benchmarks: Split Mini-ImageNet, Split CIFAR-100, and PACS. For Split CIFAR-100, we partition the original CIFAR-100 dataset [10] into ten subsets, with each subset representing a distinct task comprising ten classes. For Split Mini-ImageNet[26], we partition the original Mini-ImageNet dataset into 10 subsets, with each subset representing a distinct task comprising ten classes. As for PACS [31], it encompasses four domains: photo, art painting, cartoon, and sketch. Each domain consists of the same seven classes. In our experiments, we treat each domain as an individual task, resulting in a total of four tasks. Notably,

due to significant differences between images in each domain, one can observe a notable increase in inter-class variance within this dataset.

4.4.2 Imbalanced benchmarks

Existing CL benchmarks, with uniform class and domain distributions, fail to test CL methods on non-stationary, imbalanced data. Thus, we've created benchmarks specifically to assess CL methods' robustness to data imbalance.

4.4.2.1 Imbalanced Class-Incremental Learning (Imb CIL).

To establish an imbalanced Class-incremental scenario for split CIFAR-100 and split mini-ImageNet, we build upon the approach introduced by [80]. Unlike traditional benchmarks that distribute instances equally among classes, we induce class imbalance by utilizing a predefined ratio vector, denoted as $\mathbf{r}$, encompassing five distinct ratios: $(10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^{0})$. In this setup, for each run and each class, we randomly select a ratio from $\mathbf{r}$ and multiply it by the number of images corresponding to that class. This calculation determines the final number of images allocated to the class, thus establishing our imbalanced class scenario. We maintain the remaining conditions consistent with the corresponding balanced scenario.

4.4.2.2 Imbalanced Domain-incremental Learning (Imb DIL)

We adapt the PACS dataset, encompassing four domains, and follow an approach akin to our Imbalanced Class-Incremental method. For each domain, we randomly select a ratio from $\mathbf{r}$, multiply it with the image count of the domain, thereby maintaining a balanced class count within the imbalanced domain.

4.4.2.3 Imbalanced Class and Domain Incremental Learning (Imb C-DIL).

We further refine the PACS dataset to generate an imbalanced class-domain incremental scenario, which mirrors a more realistic data setting. This scenario involves randomly selecting a ratio from $\mathbf{r}$ for each class and domain, and multiplying it with the count of instances for that class within the domain. This operation yields $4 * 7$ values for PACS, resulting in a diverse number of data points across different classes and domains. This approach accentuates the growth of inter-class similarity and intra-class variance. Because both the class and domain are already imbalanced in the original **DomainNet**[51], we directly use its original format to generate the imbalanced scenario. We adhere to a sampling without replacement

strategy for data stream generation. Once data from a pair of class and domain is exhausted, we transition to the next pair.

Table 4.1: We report the results of our experiments conducted on **balanced** scenarios. We present the average accuracy(AA) as mean and standard deviation over five independent runs.

	Mini-ImageNet			CIFAR-100			PACS		
F.T.	4.2 ± 0.2			4.4±0.2			20.6±0.2		
i.i.d. Off	52.5 ± 0.1			49.8±0.3			59.6±0.1		
	M=1k	M=2k	M=5k	M=1k	M=2k	M=5k	M=0.1k	M=0.2k	M=0.5k
ER	10.1±0.7	13.2±0.8	16.5±1.8	11.0±0.7	14.2±0.5	20.2±0.9	36.1±1.2	38.6±1.4	39.8±1.5
GSS	10.2±0.6	13.1 ± 1.2	14.2±0.9	10.3 ± 0.5	13.3 ± 0.5	17.5 ± 1.2	35.8 ± 2.8	37.8 ± 3.2	38.7 ± 2.2
CBRS	10.3±0.8	13.5 ± 0.9	16.4±2.1	11.0 ± 0.6	14.5 ± 0.8	20.5± 0.8	36.3 ± 1.1	38.8 ± 1.6	40.1 ± 1.7
MIR	10.7±0.7	14.8 ± 1.1	17.5±1.5	11.5 ± 0.4	15.1 ± 0.5	21.7 ± 0.9	37.6 ± 0.9	40.2 ± 0.8	43.2 ± 1.2
OCS	10.8±0.5	15.1 ± 1.1	17.8±1.6	11.4 ± 0.5	14.8 ± 0.8	21.3 ± 0.9	36.8 ± 0.7	39.6 ± 0.7	42.2 ± 1.1
OnPro	21.2±0.4	30.5 ± 0.5	34.5±0.8	26.6 ± 0.5	30.6 ± 0.8	36.6 ± 0.8	36.3 ± 1.3	40.5 ± 1.3	41.4 ± 1.5
MSCL(ours)	**24.7**±0.4	**33.9** ± 0.5	**36.9**±0.9	**27.5** ± 0.4	**31.2** ± 0.7	**37.5** ± 0.8	**38.8** ± 0.9	**42.7** ± 1.1	**45.8** ± 1.3

Table 4.2: Results on our **imbalanced** scenarios. We present the average accuracy(AA) as mean and standard deviation over five independent runs. For PACS, the memory size was set to 1000, while for all other scenarios, the memory size was set to 5000.

Scenarios	**Imb CIL**		**Imb DIL**	**Imb C-DIL**	
	CIFAR-100	Mini-ImageNet	PACS	PACS	DomainNet
Fine Tunning	3.1 ± 0.3	3.5 ± 0.2	15.5 ± 1.3	14.3 ± 1.2	2.3 ± 0.6
i.i.d. Offline	41.6 ± 0.5	43.1 ± 0.6	46.3 ± 0.4	46.1 ± 0.9	37.2 ± 0.7
ER	7.1 ± 0.8	8.2 ± 1.3	25.6 ± 2.1	22.4 ± 1.3	6.2 ± 0.6
GSS	8.3 ± 0.7	7.9 ± 0.5	24.4 ± 1.7	20.2 ± 2.1	5.1 ± 0.4
CBRS	10.2 ± 0.4	11.3 ± 0.6	25.9 ± 1.5	23.6 ± 1.7	6.1 ± 0.6
MIR	7.5 ± 0.9	8.9 ± 0.3	25.8 ± 2.1	22.2 ± 2.5	6.4 ± 0.4
OCS	11.6 ± 0.6	12.3 ± 0.4	27.1 ± 1.4	24.7 ± 1.3	8.4 ± 0.7
OnPro	22.3 ± 0.5	15.8 ± 0.7	27.1 ± 1.7	25.5 ± 1.4	11.2 ± 0.9
MSCL(Ours)	**24.8**±0.6	**17.2**±0.4	**31.2**±0.8	**30.6**±0.7	**12.4**±0.7

4.4.3 Baselines and implementation details

As the proposed FDBS is a memory-based online CL method, we compare it primarily against other memory-centric techniques such as Experience Replay (ER) [61], Gradient-Based Sample Selection (GSS) [59], Class-Balancing Reservoir Sampling (CBRS) [80], Maximally Interfering Retrieval (MIR) [60], and Online Coreset Selection(OCS)[119]. Online Prototype Learning(OnPro) [145] achieved the SOTA performance in the Class-incremental scenario over Cifar-100 and Mini-ImageNet.

We compare Fine-tuning (F.T.), where pre-existing model parameters are used as starting points for new tasks without additional data, against i.i.d. offline training, a method that grants complete access to the dataset, allowing multiple data reviews for maximum performance. In this comparison, FT represents the lower bound of performance, while offline training serves as the upper bound. Our method introduces Feature-Distance Based Sampling (FDBS) for choosing samples and Contrastive Learning Loss for better representation learning. We test the effectiveness of FDBS combined with L_{SCL} in our experiments.

We adopt a reduced ResNet-18 architecture similar to that used in [61]. We maintain a fixed batch size of 20 for the incoming data stream, with one update steps per batch. We set the σ value in our radial basis function (RBF) kernel at 0.1, and the τ value in Equation (4.9) at 1.0. Our approach's performance is evaluated across the balanced and imbalanced benchmarks through five independent runs, from which we compute the average accuracy.

4.4.4 Results on balanced benchmarks

Results for balanced scenarios are shown in Table 4.1. In class incremental learning (CIL) scenarios such as split Mini-ImageNet and CIFAR-100, classical methods like ER, CBRS, and GSS do not perform well with low memory sizes. This is because, with a low memory size relative to the training data size, these methods heavily bias towards the memory data. As the memory size increases, the performance of these methods significantly improves. OnPro, which uses rich data augmentation and evaluates the class mean for each update, performs very well in these scenarios. In comparison, our method uses a more representative selection strategy and incorporates a comprehensive contrastive loss, leading to consistent improvements in results. In domain incremental learning (DIL) scenarios such as PACS, OnPro does not perform as well as in CIL scenarios, because the class mean loses its significance across multiple domains. However, our method still achieves the best results. Our memory selection strategy aims to increase intra-class variance, leading to greater diversity in the memory and improved performance. Additionally, MSCL maintains stable performance with lower standard deviations, indicating more reliable and consistent results. This robustness, combined with its superior accuracy, highlights MSCL's efficiency and reliability in handling various memory sizes and datasets.

4.4.5 Results on imbalanced scenarios

Table 4.2 displays the experimental results in the imbalanced settings. For imbalanced CIL scenarios, the CBRS method, which maintains an equal count of images from each class in memory, outperforms the basic ER approach. Meanwhile,

OCS, by continuously evaluating data batch gradients, filters noise and selects more representative data, shining particularly in imbalanced contexts. However, our method stands out, consistently leading in all imbalanced tests. As scenarios evolve from Imb DIL to Imb C-DIL, other methods' accuracy drops significantly, but FDBS maintains robust performance. Its strength lies in using feature-distance to fine-tune memory selection, preserving class boundaries and boosting intra-class diversity.

4.5 Experiments on FairWastes Dataset

4.5.1 Scenarios

The details of the dataset are discussed in Section 3.5. We follow the same preprocessing methods for this dataset in this chapter. The original FairWastes dataset is neither balanced across classes nor domains, allowing us to easily generate two imbalanced scenarios as follows:

- **Domain and Class Incremental Learning:** In the FairWaste Dataset, data are derived from various scenes, which can be interpreted as distinct domains. We have designated each 'acquisition' as an individual task, creating a scenario of imbalanced domain and class incremental learning. Thus, we have 26 distinct tasks in total.
- **Class Incremental Learning:** The dataset encompasses 13 unique materials, each representing a distinct task in our class incremental learning framework. To more accurately mimic real-world conditions, we construct the initial task to include all classes, utilizing 30 percent of the total data. Consequently, our model undertakes 14 tasks in total, sequentially introduced to replicate the incremental learning process.

For both scenarios, we feed the data as an **online stream**, which differs from the scenario described in Section 3.5.

4.5.2 Results

For all scenarios, we adopt the standard ResNet-18 [17]architecture implemented in PyTorch[69]. The replay buffer size is configured as 400. We maintain a fixed batch size of 20 for the incoming data stream, with five update steps per batch. The results are shown in Table 4.3

For the Class incremental scenario, the CBRS method, which maintains an equal count of images from each class in memory, outperforms the basic ER approach. Meanwhile, OCS, by continuously evaluating data batch gradients, filters

noise and selects more representative data, shining particularly in imbalanced contexts. However, our FDBS method stands out, consistently leading in all scenarios. As scenarios evolve from Class-Inc to Domain-Class-Inc, methods' (such as ER,GSS, and CBRS) accuracy drops significantly, but ours maintains robust performance. Its strength lies in using feature-distance to fine-tune memory selection, preserving class boundaries and boosting intra-class diversity. This advantage is amplified when paired with the IWL, reinforcing the benefits seen in balanced scenarios.

Table 4.3: Comparison of different methods in Domain-Class Inc and Class-Inc scenarios on FairWastes Dataset

Scenarios	Domain-Class Inc	Class-Inc
Fine-Tuning	33.4± 3.7	57.5± 2.8
i.i.d. Offline	81.5± 1.5	81.5± 1.5
ER	42.8±2.5	61.8±2.2
GSS	26.7±3.6	59.9±2.9
CBRS	48.5±2.9	62.5±2.0
MIR	71.2±1.8	67.5±1.5
OCS	69.3±2.1	67.1±1.8
Ours	**73.8**±1.4	**72.6**±1.2

4.6 Ablation Study and Extensive Experiments

We conduct an ablation study in Section 4.6.1, discuss the impact of σ and τ of RBF kernel in Section 4.6.2, compare the running time of different methods in Section 4.6.3, assess the impact of memory size in Section 4.6.4, evaluate average forgetting in Section 4.6.5, illustrate the distribution of our memory set in Section 4.6.6, explore the integration of our method with other methods in Section 4.6.7, and finally present the results of the methods in the classical class incremental scenario in Section 4.6.8.

4.6.1 Ablation study

Our method comprises two key components: the memory selection method **FDBS** for memory adaptation and the contrastive learning loss L_{SCL}, as detailed in Equation (4.13), for evolving data representation consolidation. Tab.4.4 highlights the contributions and effectiveness of each component. As can be seen there, memory adaptation by FDBS and data representation consolidation through L_{SCL} prove

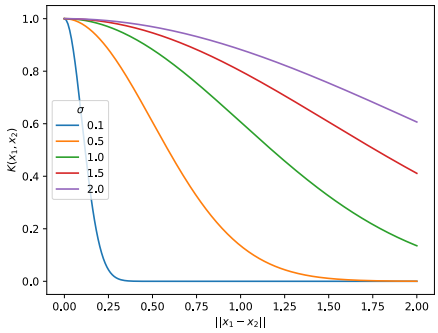


Figure 4.3: RBF kernel values with different σ

to be both useful and complementary, with FDBS consistently enhancing performance, especially in imbalanced scenarios, while L_{SCL} appears further critical.

Table 4.4: Ablation studies on balanced CIFAR-100 and imbalanced DomainNet. We set the memory size to 5000.

Method	Balanced CIFAR-100	Imb DomainNet
F.T.	4.4 ± 0.2	2.3 ± 0.6
w/o L_{SCL}	22.1 ± 1.2	7.8 ± 0.8
w/o FDBS	34.7 ± 0.9	9.5 ± 0.9
MSCL	**37.5** $\pm$ 0.8	**12.4** $\pm$ 0.7

4.6.2 The impact of σ in RBF kernel

The Radial Basis Function (RBF) kernel is a widely used kernel function in machine learning, defined as [8]:

$$K(x_1, x_2) = exp(-\frac{||x_1 - x_2||^2}{2\sigma^2}) \tag{4.14}$$

In our implementation, we normalize the feature vectors x_1 and x_2 such that $||x_1|| = 1$ and $||x_2|| = 1$. With this normalization, the squared Euclidean distance between x_1 and x_2 satisfies $||x_1 - x_2|| \in [0, 2]$, since the maximum distance occurs when x_1 and x_2 are in opposite directions.

To illustrate the effect of σ on the kernel values, we plot $K(x_1, x_2)$ for different values of σ over the range $||x_1 - x_2|| \in [0, 2]$:

As shown in Figure 4.3, the parameter σ controls the radius of influence of the kernel function. When σ is small, the kernel value $K(x_1, x_2)$ is significant only

Table 4.5: Results of varing σ and τ on Blanaced CIFAR-100 with a memory size of 2000.

$\tau \setminus \sigma$	0.1	0.5	1.0	1.5
0.1	30.5	28.7	28.9	30.1
0.5	29.6	28.2	28.0	28.8
1.0	**31.5**	29.1	29.5	30.6
1.5	30.6	30.3	29.6	30.5

when x_1 and x_2 are very close; for larger distances, the kernel value approaches zero rapidly. Conversely, a larger σ results in a broader influence, allowing more distant points to contribute meaningfully to the kernel value.

To evaluate the impact of σ and another parameter τ on our method, we conducted experiments using a memory size of 2000 on the Balanced CIFAR-100 dataset. The results are summarized in Table 4.5.

In our framework, σ is the parameter in Equation (4.7), which determines the number of points that significantly contribute to the calculation of the average distance a from a data point in the memory set to other points. A smaller σ means that only nearby points have a substantial impact on a, effectively focusing on local neighborhoods.

Similarly, τ is the parameter in Equation (4.8), which influences the calculation of the importance weights w. A larger τ allows for contributions from more distant points when computing these weights.

Our experimental results indicate that setting $\sigma = 0.1$ and $\tau = 1.0$ yields the best performance. This suggests that when calculating the average distance a within the memory set, it is beneficial to focus on the nearest points, as distant points may introduce noise or irrelevant information. However, when computing the importance weights w, considering the influence of all points in the memory set (achieved by a larger τ) is advantageous.

4.6.3 Running Time

In this section, we evaluate the overall running time of our method on the Balanced CIFAR-100 scenario with a memory size of 5K. The results are presented in Figure 4.4. Our method shows only a minor increase in running time compared to ER and MIR, while achieving significantly better performance.

4.6.4 The impact of memory size

We compare our FDBS with other memory selection methods by adjusting the size of the memory set. The experiments were conducted using the imbalanced class-

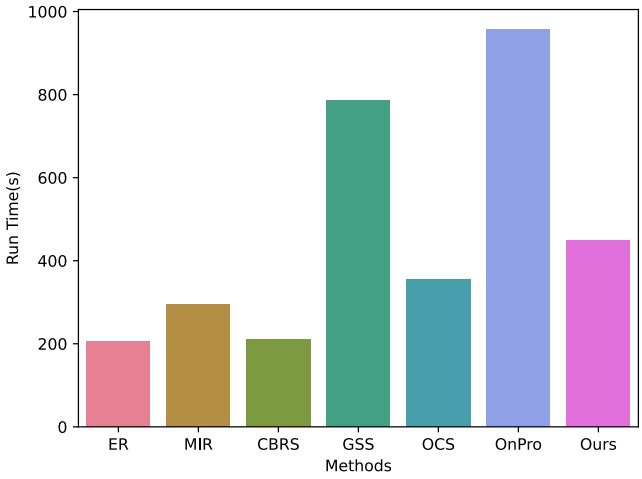


Figure 4.4: Running Time of different methods on Blanced CIFAR-100.

domain incremental scenario of PACS, and the results are presented in Table 4.6.

The experimental results show consistent performance improvements for our proposed FDBS method across all memory sizes tested. Our method outperforms all other memory selection methods in each case, with the magnitude of the improvement being more pronounced for larger memory sizes.

Table 4.6: Comparison of different memory selection methods on Imb C-DIL PACS for three different memory sizes. We present the final accuracy as mean and standard deviation over five independent runs

	Memory size			
Methods	100	200	500	1000
ER	16.4±2.3	18.3±2.5	20.4± 1.8	22.4± 1.3
GSS	15.7±1.6	16.6±1.9	18.2±2.3	20.2±2.1
CBRS	17.2±2.1	19.1±2.1	21.6±1.5	23.6±1.7
OCS	18.3±1.8	21.4±2.2	22.7±1.6	24.7±1.3
FDBS	**19.7**±1.9	**23.5**±2.6	**24.7**±2.0	**26.8**±2.2

4.6.5 Comprehensive Evaluation of Average Forgetting

We use the metric known as Average ForgettingEquation (2.13) to measure the extent of knowledge forgotten after training. We compare our method with different approaches across three typical scenarios: balanced CIFAR-100, imbalanced

CIFAR-100, and imbalanced class and domain PACS. For the experiments, we set the memory size to 5k for CIFAR-100 and 1k for PACS.

Table 4.7: Comparison average forgetting(AF) of different methods on balanced CIFAR-100, Imbalanced CIFAR-100, and Imbalanced class-domain PACS. We present the average accuracy(AF) as mean and standard deviation over five independent runs

	CIFAR-100	Imb CIFAR-100	Imb C-DIL PACS
F.T.	53.2 ± 2.8	27.5 ± 1.6	35.5 ± 2.2
ER	40.8 ± 3.5	22.7 ± 1.3	23.9 ± 1.5
GSS	38.2 ± 2.3	23.5 ± 1.8	25.7 ± 1.4
CBRS	37.4 ± 3.1	17.8 ± 1.1	22.8 ± 1.5
MIR	35.6 ± 1.8	22.3 ± 1.5	23.5 ± 1.9
OCS	22.5 ± 1.5	16.5 ± 0.9	21.4 ± 1.4
OnPro	16.3 ± 1.4	14.7 ± 0.9	20.4 ± 1.4
MSCL(ours)	**15.4** ± 1.1	**13.6** ± 0.8	**17.5** ± 0.9

Table 4.7 demonstrates that, in both balanced and imbalanced scenarios, our method achieves the lowest forgetting and has a lower standard deviation. This indicates that our method is better at retaining learned knowledge while adapting to new information.

4.6.6 The distribution of our memory set

To gain deeper insights into the efficacy of our memory selection method, we examine the distribution of our memory set. Our experiments focus on the challenging task of imbalanced Domain-Incremental Learning using the PACS dataset, which comprises four distinct domains (e.g., photo, art painting, cartoon, and sketch). Following training, we analyze the distribution of our memory set, shedding light on how our method has shaped the representation of critical data points within this dynamic learning environment. The results of this analysis are presented in Table 4.8, while the ratios of different domains within the memory set generated by various methods are shown in Figure 4.5.

Methods such as ER and CBRS opt for random image selection, aiming to maintain a distribution akin to the original dataset. In contrast, our method prioritizes increasing intra-class diversity, thereby influencing a more balanced distribution of stored images. This approach plays a crucial role in improving the overall performance of continual learning. Additionally, the integration of our Contrastive Learning Loss (SCL) further enhances the feature space consolidation within our memory set. This refinement proves instrumental in effectively capturing images

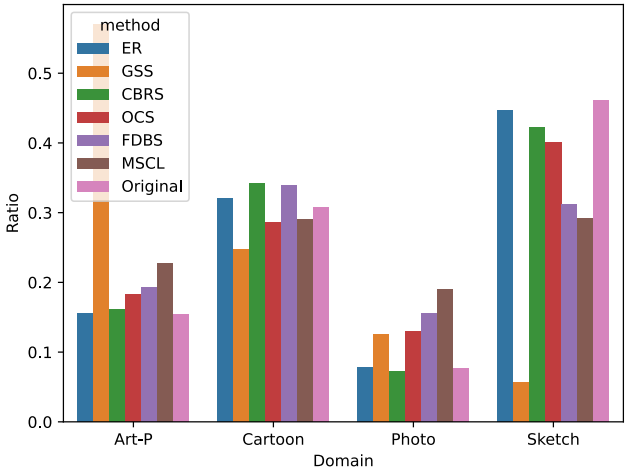


Figure 4.5: The ratio of different domains within the memory set compared to the original scenario.

from minority domains, contributing to a more robust and balanced representation of data.

Table 4.8: Comparison of Memory Set Composition Across Methods in Imbalanced Domain-Incremental Learning (imb DIL) Scenario of PACS. We set the memory size as 1000.

Methods /Domains	Photo	Art Painting	Cartoon	Sketch
Our Scenario	500	1000	2000	3000
ER	78	155	320	447
GSS	125	570	248	57
CBRS	73	162	342	423
OCS	130	183	286	401
FDBS(ours)	156	193	339	312
MSCL(Ours)	190	227	291	292

4.6.7 Collaborative Learning with other memory-based methods

In our evaluation, we consider three notable continual learning methods, PodNet[81] and AFC[125]. We integrate our Feature-Distance Based Sample Selection (FDBS) method instead of their primary memory selection method, which

was originally either random or based on herding. We also introduce our novel contrastive learning loss **SCL**. Our experiments encompass two distinct scenarios: Balanced CIFAR-100 and the imbalanced Class-Domain Incremental Learning (imb C-DIL) of PACS. The results of these experiments are presented in Table 4.9. Remarkably, our method consistently enhances the performance of these continual learning methods both on balanced and imbalanced scenarios.

Table 4.9: Combining FDBS with Other Memory-Based Methods: Experiments on Balanced Split CIFAR-100 (Memory Size: 5000) and Imbalanced Class-Domain Incremental Learning on PACS (Memory Size: 1000).The final accuracy was presented as the mean and standard deviation over five independent runs.

Methods	Split-CIFAR100	Imb C-DIL PACS
PodNet	19.5 ± 1.4	20.4 ± 1.1
PodNet + **MSCL**	**25.6** ± 2.3	**29.5**± 0.8
AFC	19.4 ± 1.7	21.5 ± 1.2
AFC + **MSCL**	25.4 ± 2.6	27.6± 0.9

4.6.8 Results on Balanced class-incremental learning scenario

We have further evaluated the effectiveness of our proposed approach in the context of classic balanced class-incremental learning. In this scenario, the task boundary is well-defined, and for each task, we employ offline training for multiple epochs. For this purpose, we conducted an experiment named **Cifar 100-B0** as detailed in [118]. In this experiment, we partitioned the original Cifar 100 dataset into 10 and 20 distinct tasks, with each task encompassing a set of 5 distinct classes. The memory size is set as 2000. The result is presented in Table 4.10. Even in the classic class-incremental learning scenario, our proposed method can still significantly improve the previous state-of-the-art method.

4.7 Conclusion

This research advances continual learning (CL) by tackling critical hurdles—namely blurred task boundaries and severe data imbalance that limit real-world deployment. The Memory Selection and Contrastive Learning (MSCL) method presented here offers a robust means of adapting to imbalanced data streams without sacrificing historical knowledge. Through a combination of feature-distance based sampling and novel contrastive learning, MSCL effectively

Table 4.10: Results for classic class-incremental learning on CIFAR-100. Results marked with '*' are obtained directly from [118]. The memory size is set to 2000.

Methods	10 steps	20 steps
iCaRL*[24]	65.2 ± 1.0	61.2 ± 0.8
BiC*[74]	68.8 ± 1.2	66.4 ± 0.3
PodNet*[81]	58.0 ± 1.3	53.9 ± 0.8
AFC[125]	61.2 ± 1.4	54.7 ± 0.8
WA*[76]	69.4 ± 0.3	67.3 ± 0.2
MSCL(ours)	**72.5** ± 0.4	**70.5** ± 0.5

manages intra-class diversity and inter-class similarity, resulting in a discriminative representation essential for on-the-fly adaptation in dynamic environments.

Importantly, this second contribution complements the other two pillars of the thesis. While the first contribution focuses on mitigating catastrophic forgetting under moderate distribution shifts using an entropy-guided approach, MSCL is adept at handling more pronounced and imbalanced shifts where memory replay is necessary. In turn, the third contribution alleviates computational overhead through generative distillation, further extending CL's applicability to resource-constrained settings. Together, these three methods form a comprehensive framework that enhances CL's viability for a wide spectrum of real-world scenarios emphasizing scalability, memory efficiency, and computational feasibility.

Chapter 5

Online Continual Learning of Diffusion Models

In the previous chapter, we introduced a memory-based approach to address the imbalanced online continual learning scenario. However, many real-world applications impose strict data privacy or safety constraints that prevent access to the original data, thereby making the use of a memory buffer infeasible. To overcome this constraint, we explore the use of diffusion models as an alternative for generating representative samples. Nevertheless, diffusion models themselves are susceptible to catastrophic forgetting when trained continually. Consequently, our objective extends beyond employing them as pseudo replay buffers for previously encountered data distributions to the continual training of diffusion models. This work is partially presented in "Online Continual Learning of Diffusion Models: Multi-Mode Adaptive Generative Distillation".

5.1 Motivation

Building upon the previous contributions in this thesis, where we addressed data imbalance using memory-based strategies and mitigated catastrophic forgetting via careful memory selection, an additional constraint emerges in many real-world applications: strict data privacy and safety requirements. In such settings, models often cannot store or revisit the original training data, making replay buffers infeasible. Consequently, generative replay using large-scale generative models stands out as an appealing alternative for preserving learned knowledge without exposing sensitive information.

When employing generative replay, the quality of synthesized samples is paramount. Among various generative models, diffusion models have emerged as a state-of-the-art method for generating high-fidelity images [87, 106]. Recent

works have successfully applied diffusion models to class-incremental learning scenarios, treating them primarily as generators to recreate older classes [134, 155, 137]. However, these approaches typically assume well-defined task boundaries: before training each new task, they rely on the diffusion model to generate sufficient replay samples, often overlooking computational constraints. Consequently, they are not readily applicable to online continual learning (OCL), where data arrive sequentially without clear task demarcations. In OCL, two key challenges arise. First, diffusion models are computationally expensive to sample from because they rely on iterative denoising steps that can run into the hundreds or thousands. Efficiently generating replay samples is therefore crucial. Second, because no clear task boundaries exist, models must be updated with each incoming data batch. Both the classifier and the diffusion model face the risk of catastrophic forgetting, highlighting the need for an efficient update mechanism that preserves previously learned knowledge.

To reduce the computational cost in the generation of diffusion models, several techniques have been proposed to reduce computational overhead by transferring noise-prediction knowledge from a large teacher model to a more efficient student [130, 143, 133]. These distillation methods can drastically cut the number of sampling steps needed to generate images, thereby lowering computational costs. However, they frequently assume that the teacher diffusion model is fixed and that the original data remain accessible. Such assumptions break down in OCL scenarios, where the dataset is no longer available and the teacher model itself must adapt to new data continuously. Repeatedly retraining both teacher and student models in each new learning phase quickly becomes prohibitively expensive.

To address these shortcomings, we introduce Online Multi-Mode Adaptive Generative Distillation (MAGD)—a novel framework designed to surmount the principal obstacles in continually training diffusion models under online conditions. MAGD unifies generative replay and knowledge distillation, enabling efficient knowledge transfer across sequential tasks. This integration tackles longstanding challenges in continual learning, including deteriorating image quality, excessive computational costs, and class imbalance. Our method contains three components:

- **Noisy Intermediate Generative Distillation (NIGD)**: We propose an enhanced distillation strategy that leverages both intermediate noisy images ($\boldsymbol{x}_\tau$) directly generated from the reverse process and noisy images ($\hat{\boldsymbol{x}}_\tau$) obtained by adding noise to the final image ($\boldsymbol{x}_0$). The intermediate noisy images capture detailed local information, while the latter provide richer global context. This complementary approach enhances the stability of the distillation process in online continual learning (OCL) scenarios, maximizing data utility without increasing computational overhead. By preserving

image quality across successive tasks, NIGD significantly improves overall training efficiency.

- **SNR-Guided Generative Distillation (SGGD)**: Inspired by recent studies on the denoising capabilities of diffusion models [123, 147], we use an SNR-based threshold to dynamically select among current data, generated samples, or Gaussian noise to replay, minimizing the frequency of full generation cycles and cutting computational costs without losing performance

- **Preserving Learned Knowledge with EMA**: We employ an Exponential Moving Average (EMA) strategy to update our diffusion model, drawing inspiration from techniques[163, 83] used in online continual learning and self-supervised contrastive learning. By updating the teacher model through EMA, we effectively maintain previously learned knowledge while adapting to new data. This approach enhances the knowledge distillation process in our framework, allowing the model to better handle the challenges of online continual learning scenarios.

In summary, our contributions are as follows:

- We introduce a novel method, **Online Multi-Mode Adaptive Generative Distillation (MAGD)**, which targets the continual learning of diffusion models in OCL. This method comprises three key components: Noisy Intermediate Generative Distillation (NIGD), SNR-Guided Generative Distillation (SGGD), and Exponential Moving Average (EMA). MAGD significantly reduces computational costs while maintaining or enhancing generation performance.

- Empirical evaluations on Fashion-MNIST, CIFAR-10, and CIFAR-100 show that MAGD reduces overall generation steps (e.g., 10 for Fashion-MNIST, 25 for CIFAR) while preserving or even improving performance. Notably, it achieves a 25% reduction in computation compared to standard distillation and a 92% saving over methods using 1000-step denoising (DDGR-1000), all while producing higher-quality generated samples and strong classification performance.

5.2 Problem formulation

Following recent work in continual learning [127, 111], we explore the online continual learning scenario where a model learns from an online data stream, with each sample presented only once. Formally, we define a data stream at time step k

as $\boldsymbol{B}^k = \{(x_i, y_i)\}_{i=1}^{N_b}$, where each pair (x_i, y_i) corresponds to an input data point and its associated label, respectively, and N_b represents the batch size.

The diffusion model is denoted by ϵ_θ, comprising T generative steps. We define the online continual learning algorithm A as follows:

$$A^k : (\epsilon_{\theta^{k-1}}, \boldsymbol{B}^k, \mathcal{M}^{k-1}) \rightarrow (\epsilon_{\theta^k}, \mathcal{M}^k) \tag{5.1}$$

At each time step k, the model receives a small batch $\boldsymbol{B}^k$. It updates based on this current batch $\boldsymbol{B}^k$ and data in $\mathcal{M}^{k-1}$, where $\mathcal{M}^k$ represents the memory set that stores either true images or generated images at time step k. In this study, we employ a diffusion model to generate images to replace the memory set $\mathcal{M}$, thereby eliminating the need to store true images.

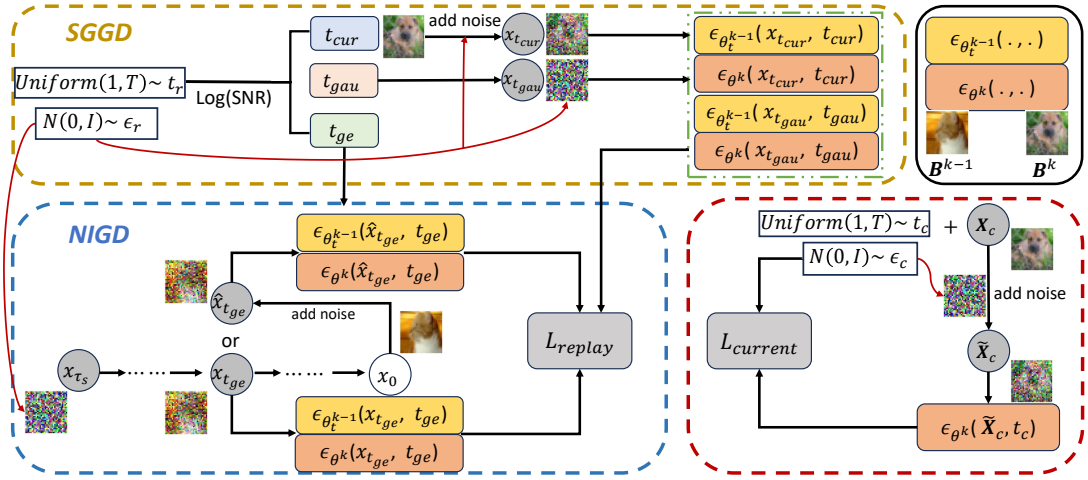


Figure 5.1: Illustration of Our Method. The **yellow region** represents **SGGD**, the **blue region** denotes **NIGD**, and the **red region** corresponds to training on the current batch $\boldsymbol{B}^k$. $\epsilon_{\theta_t^{k-1}}$ is our EMA-teacher, and ϵ_{θ^k} is our current model.

5.3 Methodology

5.3.1 Generative replay and Generative distillation

In this section, we explore the mechanisms of Generative Replay (DGR) and its advanced variant, Generative Distillation (DGR-distill), which are detailed in Algorithm 8. We consider both Generative Replay (DGR) and Generative Distillation (DGR-distill) [142] as our baselines, which are among the most commonly used strategies applying generative models to continual learning algorithms. Before training on a new batch $\boldsymbol{B}^k$, they first use the previous noise prediction model $\epsilon_{\theta^{k-1}}$

to generate a memory batch $\mathbf{X}_r$. We then add noise ϵ_r corresponding to the diffusion step t_r to obtain the noisy images $\widetilde{\mathbf{X}}_r$. The two methods differ only in the target used for the distillation loss. In DGR, it uses the known added noise as the prediction target : $\mathcal{L}_{DGR} = MSE(\epsilon_r, \epsilon_{\theta^k}(\widetilde{\mathbf{X}}_r, t_r))$. However, in DGR-distill, it uses the previous model's output as target : $\mathcal{L}_{DGR-distill} = MSE(\epsilon_{\theta^{k-1}}(\widetilde{\mathbf{X}}_r, t_r), \epsilon_{\theta^k}(\widetilde{\mathbf{X}}_r, t_r))$

Algorithm 8 Train diffusion model at step k

Input: θ_t^{k-1} is the previous teacher model parameters, θ^{k-1} is the previous model parameters, $\boldsymbol{B}^k$, N_b is the batch size, n is the number of iterations, λ is the updating speed in EMA.

Get current data $\mathbf{X}_c, \boldsymbol{y}_c$ of size N_b from $\boldsymbol{B}^k$
$\theta_s^k = \theta^{k-1}$ {Initialize the current model}
for n steps **do**
  $t_c, t_r \sim Uniform(\{1, \ldots, T\})$
  $\epsilon_c, \epsilon_r \sim \mathcal{N}(0; \boldsymbol{I})$
  $\widetilde{\mathbf{X}}_c = \sqrt{\bar{\alpha}_{t_c}}\mathbf{X}_c + \sqrt{1 - \bar{\alpha}_{t_c}}\epsilon_c$
  $\mathbf{X}_r$ = DDIM(ϵ_r,θ_t^{k-1}) {Generate Images from previous diffusion model}
  $\widetilde{\mathbf{X}}_r = \sqrt{\bar{\alpha}_{t_r}}\mathbf{X}_r + \sqrt{1 - \bar{\alpha}_{t_r}}\epsilon_r$
  $\mathcal{L}_{current} = MSE(\epsilon_{\theta^k}(\mathbf{X}_c, t_c), \epsilon_c)$ {current loss}
  if method == "DGR" **then**
    $\mathcal{L}_{replay}$ = MSE($\epsilon_{\theta^k}(\widetilde{\mathbf{X}}_r, t_r)$,$\epsilon_r$)
  else if method == "DGR-distill" **then**
    $\mathcal{L}_{replay}$ = MSE($\epsilon_{\theta^k}(\widetilde{\mathbf{X}}_r, t_r)$,$\epsilon_{\theta_t^{k-1}}(\widetilde{\mathbf{X}}_r, t_r)$)
  end if
  $\mathcal{L}_{total} = \mathcal{L}_{current} + \mathcal{L}_{replay}$
  $\mathcal{L}_{total}$.backward()
  Update θ^k
end for
θ_t^k = (1 - λ)θ_t^{k-1} + λ θ^k {Update teacher model}

5.3.2 Noisy Intermediate Generative Distillation (NIGD)

To efficiently generate images, we utilize a DDIM scheduler [99], which operates over a selected subset of steps $\{\tau_1, \tau_2, \ldots, \tau_s\}$ from the total number of steps T, thereby reducing the number of necessary steps to S. As discussed in Section 5.3.1, current continual learning methods that apply diffusion models use either DGR or DGR-distill. Both approaches initially generate the original images $\boldsymbol{x}_0$ using the full S steps, then add noise to produce $\hat{\boldsymbol{x}}_{\tau_i}$ at specified time steps τ_i, as illustrated in Figure 5.2. However, if we utilize only $\hat{\boldsymbol{x}}_{\tau_i}$ for distillation during the generation

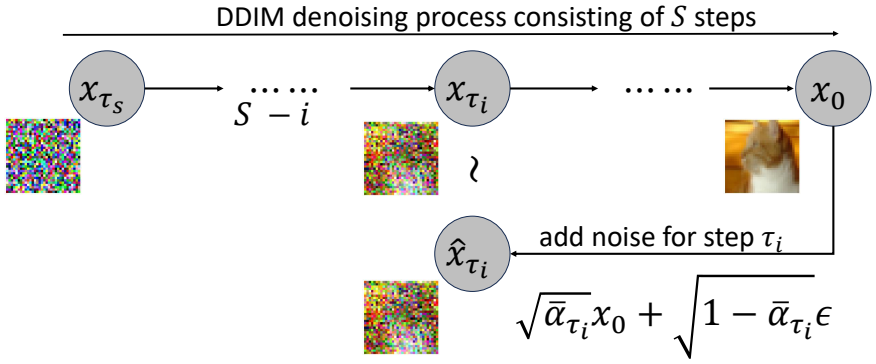


Figure 5.2: An illustration of the DDIM denoising process with S steps shows two approaches: using $S - i$ steps to directly generate $\boldsymbol{x}_{\tau_i}$, or first generating the original images $\boldsymbol{x}_0$, followed by adding noise to produce the noisy image $\hat{\boldsymbol{x}}_{\tau_i}$ at step τ_i.

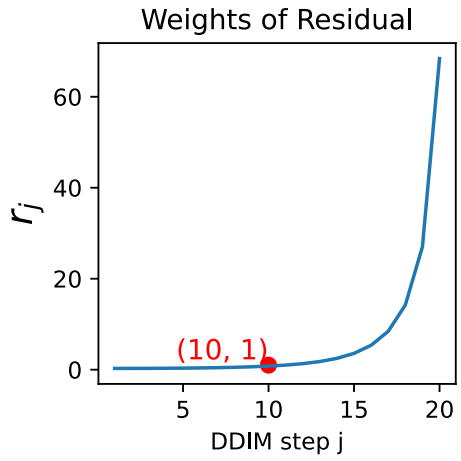


Figure 5.3: Evaluation of $\boldsymbol{r}_i$ with 20 generation steps

process, the diffusion model can directly generate noisy images at time step denoted by $\boldsymbol{x}_{\tau_i}$. These noisy images can also be useful for distillation as they require only $S-i$ generation steps. We then compute the differences between them and explain their utility for distillation.

$$\boldsymbol{x}_{\tau_i} = \sqrt{\bar{\alpha}_{\tau_i}} * \frac{\boldsymbol{x}_{\tau_{i+1}} - \sqrt{1-\bar{\alpha}_{\tau_{i+1}}}\epsilon_\theta(\boldsymbol{x}_{\tau_{i+1}})}{\sqrt{\bar{\alpha}_{\tau_{i+1}}}} + \sqrt{1-\bar{\alpha}_{\tau_i}}\epsilon_\theta(\boldsymbol{x}_{\tau_{i+1}}) \tag{5.2}$$

After completing S steps of the reverse process, we obtain the generated image denoted as $\boldsymbol{x}_0$. In the forward process, the distribution $q(\boldsymbol{x}_{\tau_i}|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{\tau_i}; \sqrt{\bar{\alpha}_{\tau_i}}\boldsymbol{x}_0, (1-\bar{\alpha}\tau_i)\boldsymbol{I})$ describes how the image $\boldsymbol{x}_0$ transitions to its noisy versions. Specifically, we derive the noisy images $\hat{\boldsymbol{x}}_{\tau_i}$ directly from $\boldsymbol{x}_0$.

$$\hat{\boldsymbol{x}}_{\tau_i} = \sqrt{\bar{\alpha}_{\tau_i}}\boldsymbol{x}_0 + \sqrt{(1-\bar{\alpha}_{\tau_i})}\epsilon \tag{5.3}$$

We can then derive the difference between $\hat{\boldsymbol{x}}_{\tau_i}$ and $\boldsymbol{x}_{\tau_i}$ (full demonstration is detailed in the Appendix B.1) :

$$\hat{\boldsymbol{x}}_{\tau_i} - \boldsymbol{x}_{\tau_i} = \sum_{j=i}^{1}(\boldsymbol{r}_j\epsilon_\theta(\boldsymbol{x}_{\tau_j})) \tag{5.4}$$

$$\boldsymbol{r}_j = \sqrt{\bar{\alpha}_{\tau_i}}\left(\sqrt{\frac{1-\bar{\alpha}_{\tau_{j-1}}}{\bar{\alpha}_{\tau_{j-1}}}} - \sqrt{\frac{1-\bar{\alpha}_{\tau_j}}{\bar{\alpha}_{\tau_j}}}\right) \tag{5.5}$$

From Equation (5.4), we observe that the difference between the noisy image $\hat{\boldsymbol{x}}_{\tau_i}$, derived from the generated $\boldsymbol{x}_0$ by adding noise, and the directly generated noisy image $\boldsymbol{x}_{\tau_i}$ depends solely on the generation steps from τ_i to τ_1. As illustrated in Figure 5.3, we evaluate the values of $\boldsymbol{r}_j$ when $\tau_i = 500$ $(i = 10)$ for 20 steps of DDIM. We find that for all $j < 10$, the residuals $\boldsymbol{r}_j$ are smaller than 1 and significantly lower than $\boldsymbol{r}_j$ for $j > 10$. This indicates that the residual component is relatively weaker compared to the noisy image $\boldsymbol{x}_{\tau_i}$.

In our continual learning scenario, we employ the previously trained diffusion model at time $k-1$, denoted as θ^{k-1}, as our teacher model. Our objective is to train the new model θ^k by distilling knowledge from the teacher model. Therefore, for any given τ_i, we require:

$$\epsilon_{\theta^k}(\boldsymbol{x}_{\tau_{i-1}}|\boldsymbol{x}_{\tau_i}, \tau_i) = \epsilon_{\theta^{k-1}}(\boldsymbol{x}_{\tau_{i-1}}|\boldsymbol{x}_{\tau_i}, \tau_i) \tag{5.6}$$

During the generation process, the previously trained model generates $\boldsymbol{x}_{\tau_i}$ without access to $\hat{\boldsymbol{x}}_{\tau_i}$. However, $\boldsymbol{x}_{\tau_i}$ is crucial for distilling knowledge from the previous model, as it retains more localized information.

We demonstrate that in a S-step DDIM generation process, both the directly generated noisy image $\boldsymbol{x}_{\tau_i}$ and the noisy image $\hat{\boldsymbol{x}}_{\tau_i}$ obtained by adding noise to $\boldsymbol{x}_0$

are valuable for distillation. The former helps retain more local information, while the latter preserves more global details, as shown in Figure 5.2.

This study suggests two methods of obtaining a noised image, for any given diffusion step, τ_i as shown in Figure 5.2:

1. **Two-Stage Approach**: Generate $\boldsymbol{x}_0$ using S steps, then add noise for step τ_i to get $\hat{\boldsymbol{x}}_{\tau_i}$.

2. **Direct Approach**: Directly generate $\boldsymbol{x}_{\tau_i}$ using $S - i$ steps.

In practice, we distill knowledge from both the intermediate noisy images and the two-stage noisy images produced during the inverse process.

5.3.3 SNR-Guided Generative Distillation (SGGD)

Research by [123] discovered that a diffusion model operates in two distinct phases based on the time steps (t): as a denoiser for refining corrupted images into final samples when t is small, and as a generator for creating images from noise when t is larger. Their research shows robust generalization across datasets such as CIFAR-10 and CelebA, particularly in the early stages of diffusion (when $(t/T < 0.1)$), as illustrated in Fig. 3 of [123]

The use of solely generated images for training in continual learning scenarios, as discussed in [36], [47], and [134], leads to progressive degradation in image quality. To counter this, we propose utilizing the early-stage denoising capabilities of diffusion models to distill knowledge directly from current training data, rather than generated images. This approach yields several benefits: (1) Enhanced image clarity. (2) Preservation of knowledge from earlier stages. (3) Reduced computational cost by eliminating the need for image generation in the initial steps.

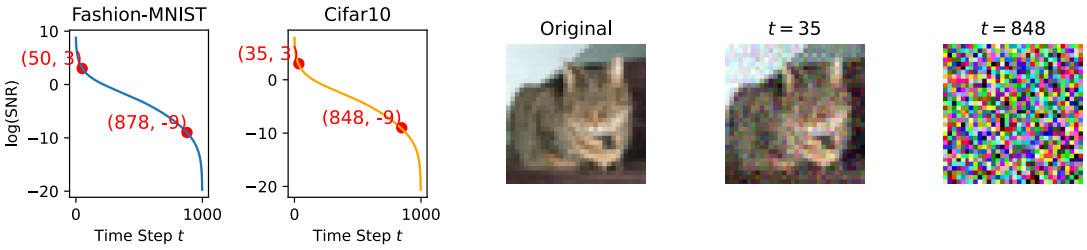


Figure 5.4: $logSNR$ Across Time Steps in Fashion-MNIST and CIFAR-10

To find the turning point t_c of the time step before which current training data can be effectively used, we calculate the Signal-to-Noise Ratio (SNR) along with

the time step. This measurement assesses the relative amplitude of the added noise compared to the original image. We use the same formula as in [123]:

$$SNR(\boldsymbol{x}_0, t) = \frac{\bar{\alpha}_t \boldsymbol{x}_0^2}{1 - \bar{\alpha}_t} \tag{5.7}$$

where $\boldsymbol{x}_0$ is the original image. The SNR quantifies the amplitude ratio between the original image and noise. Research by [123] demonstrates that a $log(SNR) = 3$ serves as a reliable threshold, which does not negatively impact the FID of generated images. The critical time steps, t_{low}, are determined as 50 for Fashion-MNIST and 35 for CIFAR-10, as shown in Figure 5.4.

As the time step increases and $log(SNR)$ becomes significantly negative, indicating a strong dominance of noise over signal, the diffusion model's input approximates Gaussian noise. In such scenarios, distilling knowledge from Gaussian noise becomes crucial. We utilize a rescaled schedule, as suggested by [141], where a $log(SNR) = -9$ marks the input as nearly indistinguishable from noise. The identified transition points, t_{high}, are 878 for Fashion-MNIST and 848 for CIFAR-10, detailed in Figure 5.4.

In the yellow region of Figure 5.1, we propose selecting images for distillation based on the training step t_r and two thresholds: t_{low} and t_{high}. Specifically:

- If $t_r < t_{low}$, images are selected from the current batch for distillation.
- If $t_r > t_{high}$, Gaussian noise is directly used for distillation.
- Otherwise, noisy images are generated from previous model

To manage this process, $log(SNR)$ for each image batch is calculated. Additionally, the thresholds t_{low} and t_{high} are dynamically updated using a moving average formula based on each training batch. This adaptive approach minimizes the need for manual tuning of these parameters and reduces the overall number of images that need to be generated by approximately 20%, without compromising performance outcomes.

5.3.4 EMA in Online Continual Learning

In an Online Continual Learning setting, unlike classic class-incremental learning where task boundaries allow the use of a previously trained model as a static teacher model [81, 74, 24], no such boundaries exist. This necessitates a different approach to updating our teacher model dynamically. We adopt the concept of Exponential Moving Average (EMA), as utilized in [83, 163], to update our teacher model at time k:

$$\theta_t^k = (1-\lambda)\theta_t^{k-1} + \lambda\theta^k \tag{5.8}$$

Here, θ^k is our current diffusion model, updated with the latest training batch. θ_t represents our teacher model. The parameter λ is a hyperparameter that controls the rate of update; for this study, we set λ to 0.01. This setting ensures that the teacher model gradually integrates new knowledge while maintaining stability over time, a crucial aspect in the absence of clear task boundaries.

5.3.5 Workflow and overall objective

Our method's workflow is illustrated in Figure 5.1. At the training step k, we first initialize the current model ϵ_{θ^k} using the parameters from the previous model $\epsilon_{\theta^{k-1}}$. We then update ϵ_{θ^k} by processing the current batch B^k and distilling knowledge from the EMA-based teacher $\epsilon_{\theta_t^{k-1}}$. Because the previous batch B^{k-1} is not retained, it is unavailable for training at this step. The current batch B^k comprises images ($\mathbf{X}_c$), labels ($\mathbf{Y}_c$).

The process starts by randomly selecting time steps t_r, and generating Gaussian noise ϵ_r. According to the resampling guidelines in **SNR-Guided Generative Distillation (SGGD)** Section 5.3.3, we categorize t_r into three types: t_{cur} for replaying current images $\boldsymbol{x}_{t_{cur}}$, t_{gau} for replaying Gaussian noise $\boldsymbol{x}_{t_{gau}}$, and t_{ge} for replaying generated noisy images. For instances categorized under t_{ge}, we employ **Noisy Intermediate Generative Distillation (NIGD)** to produce half of the images as directly noisy images $\boldsymbol{x}_{t_{ge}}$ and the other half as two-stage noisy images $\hat{\boldsymbol{x}}_{t_{ge}}$. By combining all types of noisy images for replay, we can construct our noisy memory batch as $\widetilde{\mathbf{X}}_r$. The replay loss is then calculated based on these categorizations.

$$\mathcal{L}_{replay} = MSE(\epsilon_{\theta^k}(\widetilde{\mathbf{X}}_r, t_r), \epsilon_{\theta_t^{k-1}}(\widetilde{\mathbf{X}}_r, t_r)) \tag{5.9}$$

Next, we sample time steps t_c and noise ϵ_c. We then pass the current noisy training data $(\widetilde{\mathbf{X}}_c, \epsilon_c)$ through our current model to obtain:

$$\mathcal{L}_{current} = MSE(\epsilon_{\theta^k}(\widetilde{\mathbf{X}}_c, t_c), \epsilon_c) \tag{5.10}$$

Finally, the overall objective is formulated as:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{current} + (1-\alpha)\mathcal{L}_{replay} \tag{5.11}$$

where α is a hyperparameter controls the balance between the current loss and the replay loss. After updating our current model ϵ_{θ^k}, we then use Equation (5.8) to update our EMA-based teacher. Typically, the value of α is decreased as training progresses to reduce the updates on the current batch, thereby enhancing the

model's stability. In the context of online continual learning, we update α dynamically with the following formula:

$$\alpha = 0.1 + 0.4 * 0.995^k \tag{5.12}$$

where k represents the current training step. This formulation ensures that λ gradually decreases over time from 0.5 to 0.1, providing a smooth transition from focusing on the current batch to maintaining stability through replay.

Then, we present the algorithm of our method in Algorithm 9.

5.4 Experiments and Results

5.4.1 Datasets

In this paper, we use three well-used datasets in online continual learning such as **Fashion-MNIST**,**CIFAR-10**, and **CIFAR-100**. For all datasets, we use half of classes as the zero task T_0 where we train our model offline and the model can be considered as a well-trained initial model. Then, we feed the half of the dataset in on online stream.

5.4.2 Evaluation metrics and Methods

In this paper, we assess image quality using the Fréchet Inception Distance (**FID**), which is calculated between generated images and a test set from previously encountered tasks. To evaluate the model's capability to generate balanced batches, we compute the Kullback-Leibler Divergence (**KLD**) between the uniform distribution and the predicted class distribution of the generated images. Additionally, we measure the average classification accuracy (**AA**) for class-conditional diffusion models.

We explore two types of diffusion models in our experiments: unconditional and class-conditioned. The references for diffusion models include [87, 21].

For unconditional diffusion model, We compare our method primarily with deep generative approaches such as **DGR**, **DGR with distillation** [142], and **DDGR** [134], along with the memory-based method ER [61], as well as Fine-tuning and Joint-training as lower and upper bound, respectively.

- **F.T. (Fine-tuning)**: Fine-tunes only on the current task (lower bound).
- **J.T. (Joint-training)**: Trains on all encountered tasks jointly (upper bound).

Additionally, we utilize **DDGR-1000**, which involves 1000 full generation steps and provides state-of-the-art performance but at a high computational cost, thereby acting as a secondary upper bound. We also measure training time for all methods on a **1 × NVIDIA A100 40GB**, using **DGR-distill** as the baseline for comparison.

For class-conditional diffusion model, we can compare the final classification accuracy with two memory-free methods as **BIR**[100] and PASS[121], and one memory-based methods: **PCR**[140].

For **Fashion-MNIST**, we deploy a small UNet [87, 21] with 10 DDIM steps. For **CIFAR-10** and **CIFAR-100**, a medium-sized UNet with 25 DDIM steps is used. Both **ER** and **PCR** method employs a memory buffer of size 1000. We employ EMASection 5.3.4 for all diffusion models.

5.4.3 Overall results

In Table 5.1 and Table 5.2, we present the results, summarized as mean and standard deviation over five random runs for both unconditional and class-conditioned diffusion models.

Unconditional Diffusion Model Results: In Table 5.1, our method significantly outperforms the classical DGR-distill, demonstrating substantial improvements in both FID and KLD scores across various datasets. We observe an improvement range of approximately 4.5 to 5.3 in FID scores, alongside superior KLD performance, while concurrently achieving a 25% reduction in computational costs. Notably, our approach matches the performance of the more computationally intensive DDGR-1000 model, which requires up to 1000 generation steps. Specifically, for Fashion-MNIST, our method equals the FID and KLD scores obtained by DDGR-1000 with just 10 generation steps, and it requires only 25 steps for CIFAR-10 and CIFAR-100. Moreover, our method reduces the computational demand by 92% compared to DDGR-1000, underscoring its efficiency and potential for practical applications requiring lower resource utilization.

Class-Conditioned Diffusion Model Results: In Table 5.2, when examining the class-conditioned diffusion model, our method does not just compete on FID scores but also demonstrates a clear advantage in classification accuracy. It consistently surpasses the performance of basic DGR-distill and closely approaches, and in some metrics exceeds, that of DDGR-1000. Remarkably, our approach outperforms the memory-based method PCR, even with a significantly larger memory buffer.

These results validate the effectiveness of our comprehensive generative distillation strategy, which substantially elevates the benchmarks set by DGR-distill. By efficiently leveraging fewer generation steps, our method not only curtails computational expenses but also enhances the generative quality of models across diverse datasets.

Table 5.1: Results Presented as Mean and Standard Deviation Over 5 Random Runs, with unconditional diffusion model

	Fashion-MNIST			CIFAR-10			CIFAR-100		
	FID↓	KLD↓	Time↓	FID↓	KLD↓	Time↓	FID↓	KLD↓	Time↓
F.T.	95.5 ± 10.2	4.75 ±1.81	×0.15	73.5±5.8	3.83±1.15	×0.08	85.2 ± 6.9	8.55 ± 3.37	×0.08
DDGR-1000	19.2 ± 2.5	0.09 ± 0.01	×20.5	37.8 ± 3.4	0.15 ± 0.02	×8.75	42.6 ± 3.5	0.9 ± 0.31	×8.75
J.T.	14.7 ± 1.5	0.07 ± 0.01	×0.15	27.3±2.1	0.11±0.01	×0.08	32.4 ±2.5	0.61 ± 0.13	×0.08
ER	25.9±3.9	0.38±0.15	×0.15	50.5±5.9	0.35±0.13	×0.08	52.6±5.4	2.2 ± 0.75	×0.08
DGR	90.5 ± 10.5	1.15 ±0.23	×0.91	75.3 ± 6.6	1.55 ± 0.58	×0.95	80.8 ± 8.2	3.6 ± 0.87	×0.95
DGR-distill	24.8 ± 3.4	0.17 ± 0.08	0.8h ×1	46.3 ± 6.0	0.28 ± 0.14	1.5h ×1	48.5 ± 5.6	1.5 ± 0.41	1.5h ×1
Ours	20.3 ± 2.2	0.10 ± 0.04	×0.75	41.3 ± 4.6	0.17 ± 0.09	×0.72	42.4 ± 4.8	1.05 ± 0.35	×0.72

Table 5.2: Results Presented as Mean and Standard Deviation Over 5 Random Runs, with class-conditioned diffusion model

	CIFAR-10			CIFAR-100		
	FID↓	AA↑	Time↓	FID↓	AA↑	Time↓
F.T.	58.5±7.8	11.2 ± 0.2	× 0.07	65.2 ± 8.9	4.5 ±0.3	× 0.07
DDGR-1000	31.5 ± 1.6	45.7 ± 1.2	× 9.21	35.8 ± 2.3	28.8 ± 0.9	× 9.21
J.T.	26.3±1.8	73.5 ± 0.5	× 0.07	30.5 ±2.2	67.4 ± 0.3	× 0.07
ER	42.5±5.1	31.8 ± 2.5	× 0.07	52.6±5.4	18.9 ± 3.1	× 0.07
DGR-distill	39.3 ± 4.2	38.8 ± 3.8	1.7h × 1	44.5 ± 4.5	24.3 ± 2.4	1.7h × 1
BIR	-	30.5 ± 3.7	-	-	16.3 ± 3.8	-
PASS	-	37.8 ± 2.5	-	-	20.5 ± 2.7	-
PCR	-	40.5 ± 3.2	-	-	25.7 ± 2.2	-
Ours	34.7 ± 3.5	42.1 ± 2.1	× 0.71	38.2 ± 3.7	27.5 ± 1.3	× 0.71

5.5 Ablation Study and Extensive Experiments

5.6 Ablation Study

Our method incorporates two innovative components: NIGD and SGGD. SGGD utilizes three types of replay images: Gaussian noise, current images, and generative images. NIGD employs two methods to generate noisy images from the diffusion model: a two-stage approach and a direct approach. We use the DGR-distill backbone with unconditional diffusion models and apply 10 generation steps

for Fashion-MNIST. Our results demonstrate that both SGGD and NIGD enhance the quality of the generated images while reducing computational costs.

Table 5.3: Ablation Study on Fashion-MNIST

	FID↓	KLD↓	Time↓
w. Generative (two-stage)	24.8 ± 3.4	0.17 ±0.08	0.8h × 1
w. Current	56.9 ± 3.8	0.33 ± 0.12	×0.25
w. Gaussian	35.4 ± 2.6	0.76 ± 0.09	×0.25
w. SGGD	22.4±3.8	0.15±0.08	×0.85
w. Generative (direct)	23.8 ± 5.8	0.15 ± 0.09	×0.63
w. NIGD	21.5 ± 2.7	0.13 ±0.05	×0.83
Ours	20.1 ± 2.2	0.10 ± 0.03	×0.75

5.6.1 The Influence of Generation Steps

In this section, we analyze the impact of varying generation steps on our method (using the unconditional diffusion model). As shown in Table 5.4, our method consistently outperforms the baseline (DGR-distill) by a large margin in both FID and KLD across all generation steps. Notably, our method with only 25 steps achieves FID scores close to those of DGR-distill with 100 steps. Furthermore, our method with 50 steps closely matches DDGR-1000, achieving FID scores of 39.4 **vs** 37.8 and KLD of 0.15 **vs** 0.15.

Table 5.4: FID and KLD on CIFAR-10 Across Different Generation Steps

Steps	5		10		25		50		100	
	FID↓	KLD↑	FID↓	KLD↑	FID↓	KLD↑	FID↓	KLD↑	FID↓	KLD↑
DGR-distill	56.4 ± 8.7	0.55 ± 0.15	50.5±7.4	0.35±0.15	46.3±6.0	0.28±0.14	44.1 ±5.7	0.22 ±0.09	42.5±3.9	0.16 ±0.06
Ours	52.5± 7.9	0.31 ± 0.11	46.9± 7.1	0.25 ±0.10	41.3 ± 4.6	0.18 ±0.09	39.4 ± 4.3	0.15 ±0.06	37.1±3.4	0.14 ± 0.05

5.6.2 Results Across Tasks

Here, we present the results of different methods using an unconditional diffusion model on Fashion-MNIST and CIFAR-10 across various tasks, as illustrated in Figure 5.5. The results show that our method consistently outperforms the baseline. Unlike ER and DGR-distill, our method also enhances the model's performance as it learns new tasks, demonstrating greater stability during the learning process.

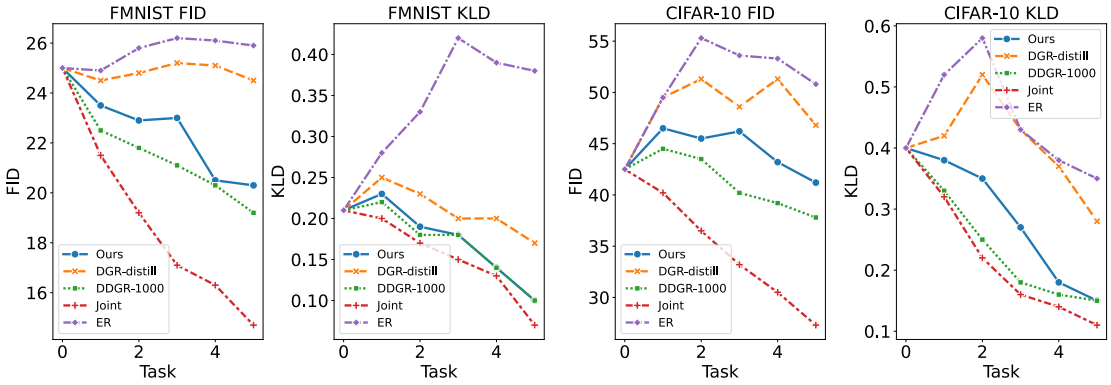


Figure 5.5: Evaluation of FID Score and KLD Across Tasks for Different Methods on Fashion-MNIST and CIFAR-10 (unconditional model)

5.7 Conclusion

In this chapter, we introduced the Multi-Mode Adaptive Generative Distillation (MAGD) approach to mitigate catastrophic forgetting and reduce computational costs in online continuous training of diffusion models. By integrating NIGD, SGGD, and EMA, our method maintains high-quality image generation while reducing computational expenses by up to 25% compared to basic DGR-distill and 92% compared to DDGR-1000. In class-conditioned models, MAGD significantly outperforms basic DGR-distill and surpasses memory-based methods in terms of classification accuracy, demonstrating its potential as a viable alternative to traditional memory buffers.

These results underscore the broader goals of this thesis: developing practical, efficient, and privacy-conscious continual learning solutions. By eliminating the need to store original data and focusing on a lightweight generative replay mechanism, MAGD advances the prospects for deploying diffusion models in real-world, rapidly evolving environments. In the future, we will investigate the scalability of our framework to more complex datasets, explore synergies with other advanced CL techniques, and delve deeper into optimizing the trade-offs among data fidelity, model size, and computational efficiency.

Chapter 6

Conclusions and Perspectives

In this thesis, we address the complex challenge of continual learning from streaming, evolving data under strict computational and privacy constraints, aiming to preserve previously acquired knowledge. Our research contributes to three fundamental aspects of continual learning (CL): operating in environments with blurred task boundaries and overlapping distributions, adapting to highly imbalanced online data streams, and reducing the computational burden and privacy issues associated with training diffusion models continually.

In Chapter 3, we introduce the Distribution-Shift Incremental Learning (DSIL) framework, which acknowledges that task boundaries are often indistinct, and data distributions may overlap in practical settings. We propose an entropy-guided self-regulated distillation process that effectively utilizes soft task boundaries without relying on extensive memory buffers. This method demonstrates significant efficacy in managing moderate distribution shifts, indicating that task overlap can be utilized to mitigate catastrophic forgetting. However, we also find that in scenarios with substantial distribution shifts, reliance solely on no-memory strategies may be inadequate, thus underscoring the potential necessity for hybrid or memory-based approaches.

Chapter 4 discusses the Memory Selection with Contrastive Learning (MSCL) framework, which merges a class-aware memory selection mechanism with contrastive representation learning to handle both balanced and imbalanced data streams effectively. This framework dynamically selects representative instances in a resource-efficient way while employing contrastive loss to enhance robust and evolving feature representations. Our experiments across various datasets validate the effectiveness of this approach, particularly in severe data imbalances.

Chapter 5 addresses the computational challenges of training large diffusion models in an ongoing manner. We present the Multi-Mode Adaptive Generative Distillation (MAGD) strategy, which facilitates efficient continual updates with minimal knowledge loss. This framework utilizes generative distillation across

noisy intermediate representations and employs exponential moving averages to preserve high-fidelity generative capabilities. Notably, MAGD significantly reduces the computational overhead and mitigates privacy and storage concerns, as evidenced by our empirical results that show sustained high-quality performance in image generation and classification compared to traditional replay-based methods.

Collectively, these innovations advance continual learning towards greater applicability in real-world, resource-limited, and privacy-sensitive scenarios. This thesis demonstrates that by strategically integrating adaptive distillation, targeted memory selection, and contrastive learning, it is feasible to harmonize the demands of knowledge retention, computational efficiency, and model adaptability. Our research contributes to the foundational efforts in advancing autonomous, incrementally learning systems in areas such as computer vision and robotics. It also suggests potential pathways for integrating large-scale foundation models and multi-modal data streams into continual learning frameworks.

For future work, I believe there are two core directions to explore: model generalization in continual learning and the representation and utilization of knowledge within artificial neural networks.

Enhancing **model generalization in continual learning** is a critical challenge given the dynamic nature of real-world data. In traditional supervised learning, especially within class-incremental scenarios[74, 81], the introduction of new classes often leads to significant shifts in data distribution, causing models to overfit on recent information and subsequently forget previously acquired knowledge. Recent approaches have leveraged contrastive learning, extensive data augmentation, and self-supervised techniques to extract invariant features that remain robust despite these distributional changes[111, 104, 124, 145]. Notably, self-supervised methods[108, 148] have shown promise in reducing catastrophic forgetting, suggesting that the learned features are more transferable across tasks. However, our current understanding of the underlying mechanisms remains limited, and there is a clear need for further exploration of dynamic regularization strategies, meta-learning frameworks, and theoretical analyses of the representation spaces in neural networks. Such research could provide deeper insights into balancing the stability-plasticity trade-off, thereby enhancing the model's ability to integrate new information while preserving existing knowledge.

A second important aspect concerns the **representation and utilization of knowledge** within artificial neural networks. Most continual learning approaches define knowledge as either raw data stored[24, 61] in memory buffers for replay, or as static model parameters[118, 107] inherited from previous training iterations. While techniques such as parameter freezing and knowledge distillation help mitigate forgetting, they oversimplify the multifaceted nature of human cognition, which relies on latent, abstract constructs that can be selectively activated

depending on the context. Exploring alternative forms of knowledge representation, such as structured frameworks like concept bottleneck models[160], knowledge graphs[101, 105], or neural-symbolic architectures, offers a promising avenue for capturing this complexity. Additionally, incorporating adaptive parameter allocation strategies and dynamic knowledge activation mechanisms, potentially inspired by biological memory consolidation processes, could further enhance the retention and transfer of learned information without succumbing to catastrophic forgetting. Such advancements would not only improve continual learning performance but also contribute to the development of more interpretable and flexible artificial intelligence systems.

Appendix A

Continual Learning with Blurred Task Boundaries

A.1 Details of the experimental protocol

A.1.1 PACS dataset

This section presents different tasks' domain distribution and class distribution for the four scenarios: DIL, smooth-DIL, CIL, and smooth-CIL. For better visualization, we map the domain label and class label as follows:

- **Class labels** : "dog": 0, "elephant": 1, "giraffe": 2, "guitar": 3, "horse": 4, "house": 5, "person": 6;
- **Domain labels** : "art_painting": "A", "cartoon": "C", "photo": "P", "sketch": "S".
- **DIL:** The domain distribution of DIL is shown in Figure A.1.
- **smooth-DIL:** We randomly generate multiple experiments. The domain distribution for one smooth-DIL experiment is shown in Figure Figure A.2.
- **CIL:** Same as DIL, the class distribution of CIL is shown in Figure A.3.
- **smooth-CIL:** We randomly generate multiple experiments. The class distribution for one smooth-DIL experiment is shown in Figure A.4.

A.1.2 Cifar100 dataset

In cifar100 dataset, there are 20 superclasses and each superclass contains five subclasses. In our experiments, we consider the superclass label as class-label and

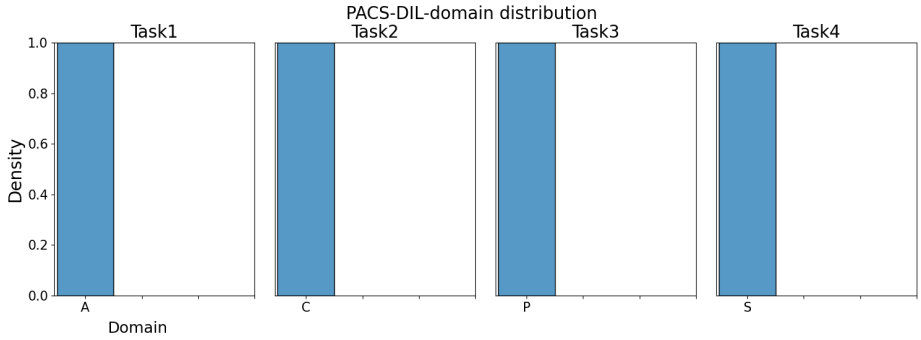


Figure A.1: Domain Distribution Across Different Tasks in the Domain Incremental Learning (DIL) Scenario on PACS.

the subclass label as domain-label. For better visualization, we map the domain label and class label as follows:

- **Class labels**: aquatic_mammals':0, 'fish':1, 'flowers':2, 'food_containers':3, 'fruit_and_vegetables':4, 'household_electrical_devices':5, 'household_furniture':6, 'insects':7, 'large_carnivores':8, 'large_man-made_outdoor_things':9, 'large_natural_outdoor_scenes':10, 'large_omnivores_and_herbivores':11, 'medium_mammals':12, 'non-insect_invertebrates':13, 'people':14, 'reptiles':15, 'small_mammals':16, 'trees':17, 'vehicles_1':18, 'vehicles_2':19.

- **Domain labels**: Because one superclass has five subclasses, we generate five domains for our experiments. For each domain, we randomly select one subclass from a superclass. Thus, each domain contains 20 subclasses from the different superclass. For simplicity, we use the number as (0,1,2,3,4) to represent the domain label.

- **DIL:** The domain distribution of DIL is shown in Figure A.5.

- **smooth-DIL:** We randomly generate multiple experiments. The domain distribution for one smooth-DIL experiment is shown in Figure Figure A.6.

- **CIL:** Same as DIL, the class distribution of CIL is shown in Figure A.7.

- **smooth-CIL:** We randomly generate multiple experiments. The class distribution for one smooth-DIL experiment is shown in Figure A.8.

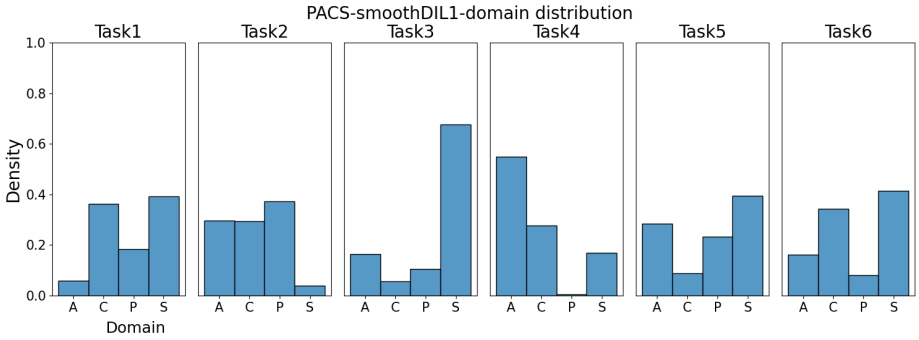


Figure A.2: Domain Distribution Across Different Tasks in the smooth Domain Incremental Learning (smooth-DIL) Scenario on PACS.

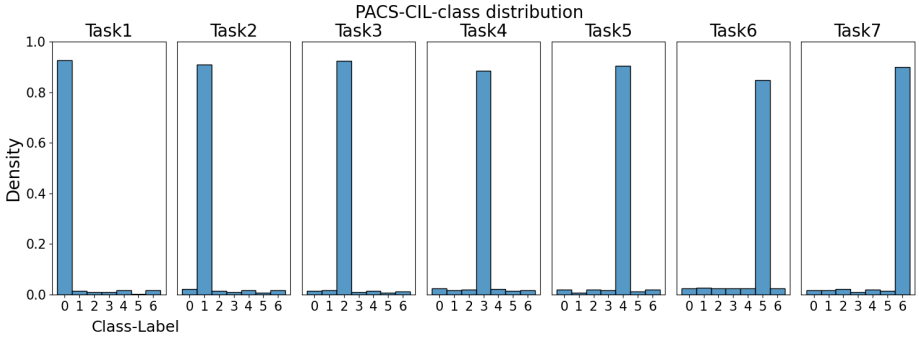


Figure A.3: Class Distribution Across Different Tasks in the Class Incremental Learning (CIL) Scenario on PACS.

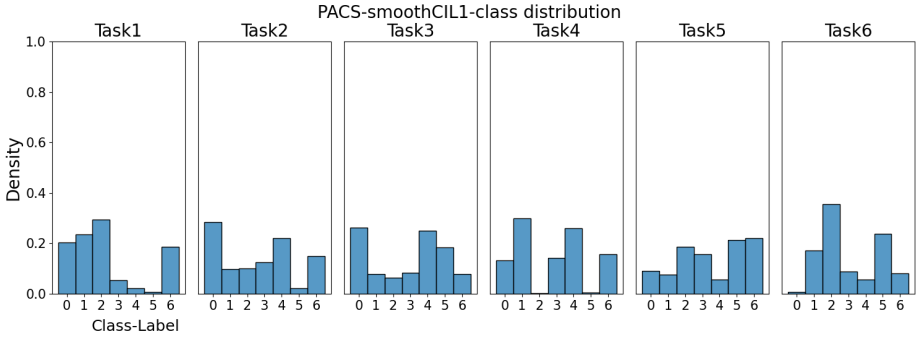


Figure A.4: Class Distribution Across Different Tasks in the smooth Class Incremental Learning (smooth-CIL) Scenario on PACS.

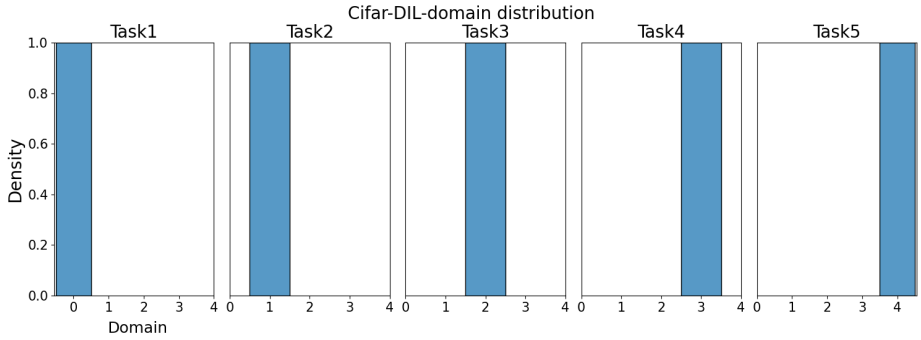


Figure A.5: Domain Distribution Across Different Tasks in the Domain Incremental Learning (DIL) Scenario on CIFAR-100.

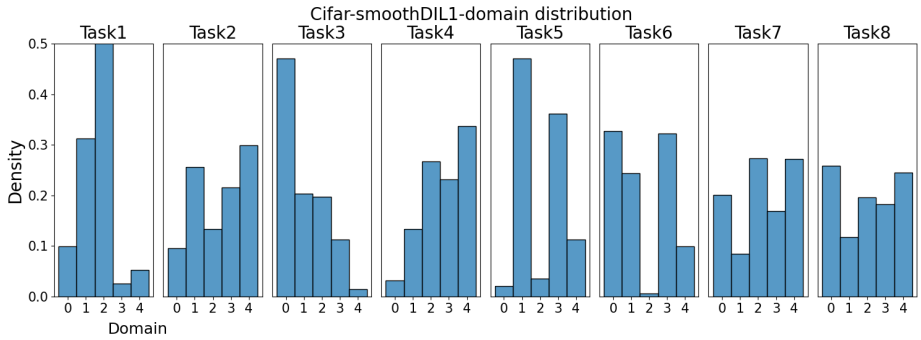


Figure A.6: Domain Distribution Across Different Tasks in the smooth Domain Incremental Learning (smooth-DIL) Scenario on CIFAR-100.

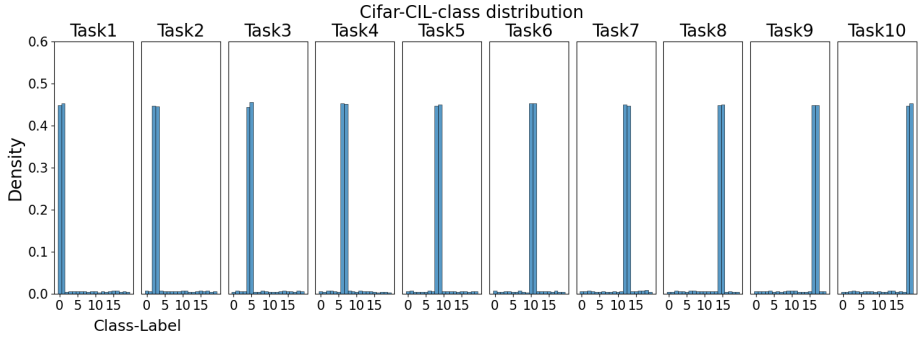


Figure A.7: Class Distribution Across Different Tasks in the Class Incremental Learning (CIL) Scenario on CIFAR-100.

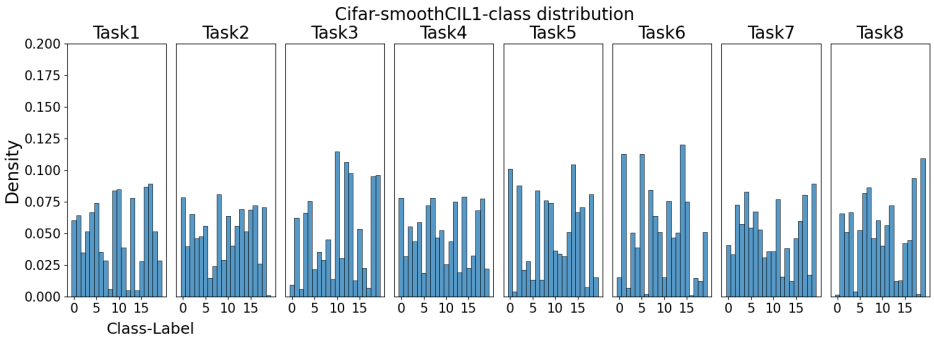


Figure A.8: Class Distribution Across Different Tasks in the smooth Class Incremental Learning (smooth-CIL) Scenario on CIFAR-100.

Appendix B

Online Continual Learning of Diffusion Models

B.1 Demonstration

In this section, we demonstrate the difference between the directly generated noisy image and the noisy image generated from the original image.

Firstly, we have the noisy images $\hat{\boldsymbol{x}}_{\tau_i}$ generated from $\boldsymbol{x}_0$.

$$\hat{\boldsymbol{x}}_{\tau_i} = \sqrt{\bar{\alpha}_{\tau_i}}\boldsymbol{x}_0 + \sqrt{(1-\bar{\alpha}_{\tau_i})}\epsilon \tag{B.1}$$

We can also obtain the noisy image at step τ_i from the image at the previous step τ_{i+1} using the model ϵ_θ, as shown in Equation (B.2).

$$\boldsymbol{x}_{\tau_i} = \sqrt{\bar{\alpha}_{\tau_i}} * \frac{\boldsymbol{x}_{\tau_{i+1}} - \sqrt{1-\bar{\alpha}_{\tau_{i+1}}}\epsilon_\theta(\boldsymbol{x}_{\tau_{i+1}})}{\sqrt{\bar{\alpha}_{\tau_{i+1}}}} + \sqrt{1-\bar{\alpha}_{\tau_i}}\epsilon_\theta(\boldsymbol{x}_{\tau_{i+1}}) \tag{B.2}$$

We can then reformulate Equation (B.2) as follows:

$$\boldsymbol{x}_{\tau_i} = k_{\tau_{i+1}}\boldsymbol{x}_{\tau_{i+1}} + l_{\tau_{i+1}}\epsilon_\theta(\boldsymbol{x}_{\tau_{i+1}}) \tag{B.3}$$

where $k_{\tau_{i+1}} = \sqrt{\frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_{i+1}}}}$, and $l_{\tau_{i+1}} = \sqrt{1-\bar{\alpha}_{\tau_i}} - k_{\tau_{i+1}}\sqrt{1-\bar{\alpha}_{\tau_{i+1}}}$

For a DDIM process comprising S steps, we have:

$$\begin{aligned}
\boldsymbol{x}_{\tau_{s-1}} &= k_{\tau_s}\boldsymbol{x}_{\tau_s} + l_{\tau_s}\epsilon_\theta(\boldsymbol{x}_{\tau_s}) \\
\boldsymbol{x}_{\tau_{s-2}} &= k_{\tau_{s-1}}\boldsymbol{x}_{\tau_{s-1}} + l_{\tau_{s-1}}\epsilon_\theta(\boldsymbol{x}_{\tau_{s-1}}) \\
&\dots \\
\boldsymbol{x}_{\tau_i} &= k_{\tau_{i+1}}\boldsymbol{x}_{\tau_{i+1}} + l_{\tau_{i+1}}\epsilon_\theta(\boldsymbol{x}_{\tau_{i+1}})
\end{aligned} \tag{B.4}$$

Starting from the initial step, with $\tau_s = 999$ for a total of 1000 steps, $\boldsymbol{x}_{\tau_s}$ represents random noise. Based on the recurrence relation, we obtain:

$$\begin{aligned}\boldsymbol{x}_{\tau_i} = \sqrt{\frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_s}}}\epsilon + \sqrt{\frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_{s-1}}}} l_{\tau_s}\epsilon_\theta(\boldsymbol{x}_{\tau_s}) + \\ \cdots \\ + \sqrt{\frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_{i+1}}}} l_{\tau_{i+2}}\epsilon_\theta(\boldsymbol{x}_{\tau_{i+2}}) + l_{\tau_{i+1}}\epsilon_\theta(\boldsymbol{x}_{\tau_{i+1}})\end{aligned} \tag{B.5}$$

$$\begin{aligned}\boldsymbol{x}_0 = \sqrt{\frac{\bar{\alpha}_0}{\bar{\alpha}_{\tau_s}}}\epsilon + \sqrt{\frac{\bar{\alpha}_0}{\bar{\alpha}_{\tau_{s-1}}}} l_{\tau_s}\epsilon_\theta(\boldsymbol{x}_{\tau_s}) + \\ \cdots \\ + \sqrt{\frac{\bar{\alpha}_0}{\bar{\alpha}_{\tau_1}}} l_{\tau_2}\epsilon_\theta(\boldsymbol{x}_{\tau_2}) + l_{\tau_1}\epsilon_\theta(\boldsymbol{x}_{\tau_1})\end{aligned} \tag{B.6}$$

By introduce Equation (B.6) into Equation (B.1) and minus Equation (B.5), we derive:

$$\hat{\boldsymbol{x}}_{\tau_i} = \boldsymbol{x}_{\tau_i} + \sum_{j=i}^{1}(\boldsymbol{r}_j\epsilon_\theta(\boldsymbol{x}_{\tau_j})) \tag{B.7}$$

where:

$$\boldsymbol{r}_j = \sqrt{\frac{\bar{\alpha}_{\tau_i}}{\bar{\alpha}_{\tau_{j-1}}}} l_{\tau_j} = \sqrt{\bar{\alpha}_{\tau_i}}\left(\sqrt{\frac{1-\bar{\alpha}_{\tau_{j-1}}}{\bar{\alpha}_{\tau_{j-1}}}} - \sqrt{\frac{1-\bar{\alpha}_{\tau_j}}{\bar{\alpha}_{\tau_j}}}\right) \tag{B.8}$$

B.2 Algorithm

Algorithm 9 Train diffusion model at step k (Ours)

Input: θ_t^{k-1},θ^{k-1}, $\boldsymbol{B}^k$, N_b is the batch size, n is the number of iterations, t_{low} is the transition point for current batch, and t_{high} is the transition point for Gaussian noise, λ is the updating speed in EMA.

Get current data $\mathbf{X}_c, \boldsymbol{y}_c$ of size N_b from $\boldsymbol{B}^k$
$\theta^k = \theta^{k-1}$
for n steps **do**
  $t_c, t_r \sim Uniform(\{1, \dots, T\})$
  $\epsilon_c, \epsilon_r \sim \mathcal{N}(0; \boldsymbol{I})$
  $\hat{t}_{low} = \{t | log(SNR(\mathbf{X}_c, t)) = 3\}$
  $\hat{t}_{high} = \{t | log(SNR(\mathbf{X}_c, t)) = -9\}$
  $\widetilde{\mathbf{X}}_c = \sqrt{\bar{\alpha}_{t_c}}\mathbf{X}_c + \sqrt{1 - \bar{\alpha}_{t_c}}\epsilon_c$ {Add noise to the current training batch}
  $\widetilde{\mathbf{X}}_r = []$
  for i, t in enumerate(t_r) **do**
    if $t < t_{low}$ **then**
      $x_r = \mathbf{X}_c[i]$
      $\widetilde{x}_r = \sqrt{\bar{\alpha}_t}x_r + \sqrt{1 - \bar{\alpha}_t}\epsilon_r$ {Add noise to the current batch}
      $\widetilde{\mathbf{X}}_r$ add $\widetilde{x}_r$ {Add Current}
    else if $t > t_{high}$ **then**
      $\widetilde{\mathbf{X}}_r$ add $\epsilon_r[i]$ {Add Gaussian}
    else
      tau = random.random()(random value from 0 to 1.)
      if $tau < 0.5$ **then**
        $\boldsymbol{x}_t$ = DDIM($\epsilon_r[i]$,θ_t^{k-1}) {Get noisy images from previous diffusion model}
        $\widetilde{\mathbf{X}}_r$ add $\boldsymbol{x}_t$ {Add directly noisy image}
      else
        $\boldsymbol{x}_0$ = DDIM($\epsilon_r[i]$,θ_t^{k-1}) {Get original images from previous diffusion model}
        $\hat{\boldsymbol{x}}_t = \sqrt{\bar{\alpha}_{t_r}}\boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_{t_r}}\epsilon_r[i]$ {Get noisy image from the original image.}
        $\widetilde{\mathbf{X}}_r$ add $\hat{\boldsymbol{x}}_t$ {Add two-stage noisy image}
      end if
    end if
  end for
  $\mathcal{L}_{current} = MSE(\epsilon_{\theta^k}(\mathbf{X}_c, t_c), \epsilon_c)$
  $\mathcal{L}_{replay} = MSE(\epsilon_{\theta^k}(\widetilde{\mathbf{X}}_r, t_r), \epsilon_{\theta_t^{k-1}}(\widetilde{\mathbf{X}}_r, t_r))$
  $\mathcal{L}_{total} = \alpha\mathcal{L}_{current} + (1 - \alpha)\mathcal{L}_{replay}$
  $\mathcal{L}_{total}$.backward()
  Update θ^{k-1}
  $\alpha = 0.1 + 0.4 * 0.995^k$
  $t_{low} = 0.999t_{low} + 0.001\hat{t}_{low}$
  $t_{high} = 0.999t_{high} + 0.001\hat{t}_{high}$
end for
θ_t^k = (1 - λ)θ_t^{k-1} + λ θ^k {Update teacher model}

Bibliography

[1] Lloyd S. Shapley. "Notes on the n-Person Game — II: The Value of an n-Person Game". In: *None*. 1951.

[2] Jeffrey Scott Vitter. "Random sampling with a reservoir". In: *ACM Trans. Math. Softw.* 11 (1985), pp. 37–57.

[3] Yann LeCun et al. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Neural Information Processing Systems*. 1989.

[4] Michael McCloskey and Neal J. Cohen. "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". In: *Psychology of Learning and Motivation* 24 (1989).

[5] James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." In: *Psychological review* 102 3 (1995), pp. 419–457.

[6] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86 (1998), pp. 2278–2324.

[7] Léon Bottou. "On-line Learning and Stochastic Approximations". In: *On-Line Learning in Neural Networks*. Ed. by DavidEditor Saad. Publications of the Newton Institute. Cambridge University Press, 1999, pp. 9–42.

[8] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. "A Primer on Kernel Methods". In: MITPRESS, July 2004, pp. 35–70. ISBN: 9780262256926. DOI: 10.7551/mitpress/4057.003.0004.

[9] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255.

[10] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *arxiv* (2009).

[11] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: *None* (2009).

[12] Léon Bottou. "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *International Conference on Computational Statistics*. 2010.

[13] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22 (2010), pp. 1345–1359.

[14] Thomas Mensink et al. "Distance-Based Image Classification: Generalizing to New Classes at Near-Zero Cost". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2013), pp. 2624–2637.

[15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR* abs/1412.6980 (2014).

[16] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755.

[17] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 770–778.

[18] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: *ArXiv* abs/1503.02531 (2015).

[19] Ya Le and Xuan S. Yang. "Tiny ImageNet Visual Recognition Challenge". In: *arxiv*. 2015.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *MICCAI* abs/1505.04597 (2015).

[22] Reza Shokri and Vitaly Shmatikov. "Privacy-preserving deep learning". In: *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2015, pp. 909–910. DOI: 10.1109/ALLERTON.2015.7447103.

[23] Zhizhong Li and Derek Hoiem. "Learning without Forgetting". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (2016), pp. 2935–2947.

[24] Sylvestre-Alvise Rebuffi et al. "iCaRL: Incremental Classifier and Representation Learning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 5533–5542.

[25] Andrei A. Rusu et al. "Progressive Neural Networks". In: *ArXiv* abs/1606.04671 (2016).

[26] Oriol Vinyals et al. "Matching Networks for One Shot Learning". In: *Neural Information Processing Systems*. 2016.

[27] Rahaf Aljundi et al. "Memory Aware Synapses: Learning what (not) to forget". In: *ICCV* abs/1711.09601 (2017).

[28] Chelsea Finn, P. Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *International Conference on Machine Learning*. 2017.

[29] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Neural Information Processing Systems*. 2017.

[30] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3521–3526. DOI: 10.1073/pnas.1611835114.

[31] D. Li et al. "Deeper, Broader and Artier Domain Generalization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2017, pp. 5543–5551. DOI: 10.1109/ICCV.2017.591.

[32] Da Li et al. "Deeper, Broader and Artier Domain Generalization". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 5543–5551.

[33] Vincenzo Lomonaco and Davide Maltoni. "CORe50: a New Dataset and Benchmark for Continuous Object Recognition". In: *CoRR* abs/1705.03550 (2017). arXiv: 1705.03550.

[34] David Lopez-Paz and Marc'Aurelio Ranzato. "Gradient Episodic Memory for Continual Learning". In: *Neural Information Processing Systems*. 2017.

[35] Arun Mallya and Svetlana Lazebnik. "PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), pp. 7765–7773.

[36] Hanul Shin et al. "Continual Learning with Deep Generative Replay". In: *Neural Information Processing Systems*. 2017.

[37] Ashish Vaswani et al. "Attention is All you Need". In: *Neural Information Processing Systems*. 2017.

[38] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[39] Friedemann Zenke, Ben Poole, and Surya Ganguli. "Continual Learning Through Synaptic Intelligence". In: *Proceedings of machine learning research* 70 (2017), pp. 3987–3995.

[40] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. "Task-Free Continual Learning". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 11246–11255.

[41] Francisco Manuel Castro et al. "End-to-End Incremental Learning". In: *European Conference on Computer Vision*. 2018.

[42] Arslan Chaudhry et al. "Efficient Lifelong Learning with A-GEM". In: *ArXiv* abs/1812.00420 (2018).

[43] Arslan Chaudhry et al. "Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence". In: *ECCV* abs/1801.10112 (2018).

[44] Prithviraj Dhar et al. "Learning Without Memorizing". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 5133–5141.

[45] Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan. "Memory Efficient Experience Replay for Streaming Learning". In: *2019 International Conference on Robotics and Automation (ICRA)* (2018), pp. 9769–9776.

[46] Yen-Chang Hsu, Yen-Cheng Liu, and Zsolt Kira. "Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines". In: *ArXiv* abs/1810.12488 (2018).

[47] Timothée Lesort et al. "Generative Models from the perspective of Continual Learning". In: *2019 International Joint Conference on Neural Networks (IJCNN)* (2018), pp. 1–8.

[48] Xialei Liu et al. "Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting". In: *2018 24th International Conference on Pattern Recognition (ICPR)* (2018), pp. 2262–2268.

[49] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. "Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights". In: *European Conference on Computer Vision*. 2018.

[50] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation Learning with Contrastive Predictive Coding". In: *ArXiv* abs/1807.03748 (2018).

[51] Xingchao Peng et al. "Moment Matching for Multi-Source Domain Adaptation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2018), pp. 1406–1415.

[52] Matthew Riemer et al. “Learning to Learn without Forgetting By Maximizing Transfer and Minimizing Interference”. In: *ArXiv* abs/1810.11910 (2018).

[53] Joan Serrà et al. “Overcoming catastrophic forgetting with hard attention to the task”. In: *International Conference on Machine Learning*. 2018.

[54] Chenshe Wu et al. “Memory Replay GANs: learning to generate images from new categories without forgetting”. In: *Conference on Neural Information Processing Systems (NIPS)*. 2018.

[55] Chenshen Wu et al. “Memory Replay GANs: learning to generate images from new categories without forgetting”. In: *Neural Information Processing Systems*. 2018.

[56] Chen Zeno et al. “Task Agnostic Continual Learning Using Online Variational Bayes”. In: *arXiv: Machine Learning* (2018).

[57] Hongjoon Ahn et al. “Uncertainty-based Continual Learning with Adaptive Regularization”. In: *NeurIPS* (2019). eprint: 1905.11614.

[58] Rahaf Aljundi. “Continual Learning in Neural Networks”. In: *ArXiv* abs/1910.02718 (2019).

[59] Rahaf Aljundi et al. “Gradient based sample selection for online continual learning”. In: *Neural Information Processing Systems*. 2019.

[60] Rahaf Aljundi et al. “Online Continual Learning with Maximally Interfered Retrieval”. In: *NeurIPS* abs/1908.04742 (2019).

[61] Arslan Chaudhry et al. “Continual Learning with Tiny Episodic Memories”. In: *ICML* abs/1902.10486 (2019).

[62] Mehrdad Farajtabar et al. “Orthogonal Gradient Descent for Continual Learning”. In: *ArXiv* abs/1910.07104 (2019).

[63] Siavash Golkar, Micheal Kagan, and Kyunghyun Cho. “Continual Learning via Neural Pruning”. In: *Real Neurons & Hidden Units: Future directions at the intersection of neuroscience and artificial intelligence @ NeurIPS 2019*. 2019.

[64] Saihui Hou et al. “Learning a Unified Classifier Incrementally via Rebalancing”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 831–839.

[65] Sampo Kuutti et al. “A Survey of Deep Learning Applications to Autonomous Vehicle Control”. In: *IEEE Transactions on Intelligent Transportation Systems* 22 (2019), pp. 712–733.

[66] Anna Kuzina, Evgenii Egorov, and Evgeny Burnaev. "BooVAE: A scalable framework for continual VAE learning under boosting approach". In: *Neural Information Processing Systems* abs/1908.11853 (2019).

[67] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. "Self-supervised Label Augmentation via Input Transformations". In: *International Conference on Machine Learning*. 2019.

[68] Dongmin Park et al. "Continual Learning by Asymmetric Loss Approximation With Single-Side Overestimation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 3334–3343.

[69] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[70] Jathushan Rajasegaran et al. "Random Path Selection for Continual Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[71] Mohammad Al-Rubaie and J. Morris Chang. "Privacy-Preserving Machine Learning: Threats and Solutions". In: *IEEE Security and Privacy* 17.2 (2019), pp. 49–58. DOI: 10.1109/MSEC.2018.2888775.

[72] Siddharth Swaroop et al. "Improving and Understanding Variational Continual Learning". In: *ArXiv* abs/1905.02099 (2019).

[73] Gido M. van de Ven and Andreas Savas Tolias. "Three scenarios for continual learning". In: *ArXiv* abs/1904.07734 (2019).

[74] Yue Wu et al. "Large Scale Incremental Learning". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 374–382.

[75] Mengyao Zhai et al. "Lifelong GAN: Continual Learning for Conditional Image Generation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 2759–2768.

[76] Bowen Zhao et al. "Maintaining Discrimination and Fairness in Class Incremental Learning". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 13205–13214.

[77] Fuzhen Zhuang et al. "A Comprehensive Survey on Transfer Learning". In: *CoRR* abs/1911.02685 (2019). arXiv: 1911.02685.

[78] Hongjoon Ahn et al. "SS-IL: Separated Softmax for Incremental Learning". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2020), pp. 824–833.

[79] Zalán Borsos, Mojm'ir Mutn'y, and Andreas Krause. "Coresets via Bilevel Optimization for Continual Learning and Streaming". In: *ArXiv* abs/2006.03875 (2020).

[80] Aristotelis Chrysakis and Marie-Francine Moens. "Online Continual Learning from Imbalanced Data". In: *International Conference on Machine Learning*. 2020.

[81] Arthur Douillard et al. "PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning". In: *European Conference on Computer Vision*. 2020.

[82] Jianping Gou et al. "Knowledge Distillation: A Survey". In: *International Journal of Computer Vision* 129 (2020), pp. 1789–1819.

[83] Jean-Bastien Grill et al. "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning". In: *ArXiv* abs/2006.07733 (2020).

[84] Gunshi Gupta, Karmesh Yadav, and Liam Paull. "La-MAML: Look-ahead Meta Learning for Continual Learning". In: *ArXiv* abs/2007.13904 (2020).

[85] Jiangpeng He et al. "Incremental Learning in Online Scenario". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 13923–13932.

[86] Jonathan Ho, Ajay Jain, and P. Abbeel. "Denoising Diffusion Probabilistic Models". In: *ArXiv* abs/2006.11239 (2020).

[87] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *NeurIPS* (2020).

[88] Timothy M. Hospedales et al. "Meta-Learning in Neural Networks: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (2020), pp. 5149–5169.

[89] Sanyam Kapoor, Theofanis Karaletsos, and Thang D. Bui. "Variational Auto-Regressive Gaussian Processes for Continual Learning". In: *ArXiv* abs/2006.05468 (2020).

[90] Prannay Khosla et al. "Supervised contrastive learning". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[91] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. "Imbalanced Continual Learning with Partitioning Reservoir Sampling". In: *European Conference on Computer Vision*. 2020.

[92] Richard Kurle et al. “Continual Learning with Bayesian Neural Networks for Non-Stationary Data”. In: *International Conference on Learning Representations*. 2020.

[93] Matthias De Lange and Tinne Tuytelaars. “Continual Prototype Evolution: Learning Online from Non-Stationary Data Streams”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2020), pp. 8230–8239.

[94] Soochan Lee et al. “A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning”. In: *ArXiv* abs/2001.00689 (2020).

[95] Marc Masana et al. “Class-Incremental Learning: Survey and Performance Evaluation on Image Classification”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2020), pp. 5513–5533.

[96] Pingbo Pan et al. “Continual Deep Learning by Functional Regularisation of Memorable Past”. In: *Neurips* abs/2004.14070 (2020).

[97] Ameya Prabhu, Philip H. S. Torr, and Puneet Kumar Dokania. “GDumb: A Simple Approach that Questions Our Progress in Continual Learning”. In: *European Conference on Computer Vision*. 2020.

[98] Dongsub Shim et al. “Online Class-Incremental Continual Learning with Adversarial Shapley Value”. In: *AAAI Conference on Artificial Intelligence*. 2020.

[99] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising Diffusion Implicit Models”. In: *ICLR* abs/2010.02502 (2020).

[100] Gido M. van de Ven, Hava T. Siegelmann, and Andreas Savas Tolias. “Brain-inspired replay for continual learning with artificial neural networks”. In: *Nature Communications* 11 (2020).

[101] Chen Wang, Yuheng Qiu, and Sebastian A. Scherer. “Lifelong Graph Learning”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 13709–13718.

[102] Lu Yu et al. “Semantic Drift Compensation for Class-Incremental Learning”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 6980–6989.

[103] Junting Zhang et al. “Class-incremental Learning via Deep Model Consolidation”. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2020), pp. 1120–1129.

[104] Jihwan Bang et al. “Rainbow Memory: Continual Learning with a Memory of Diverse Samples”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 8214–8223.

[105] Angel Andres Daruna et al. “Continual Learning of Knowledge Graph Embeddings”. In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 1128–1135.

[106] Prafulla Dhariwal and Alex Nichol. “Diffusion Models Beat GANs on Image Synthesis”. In: *NeurIPS* abs/2105.05233 (2021).

[107] Arthur Douillard et al. “DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 9275–9285.

[108] Enrico Fini et al. “Self-Supervised Models are Continual Learners”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 9611–9620.

[109] Chen He, Ruiping Wang, and Xilin Chen. “A Tale of Two CILs: The Connections between Class Incremental Learning and Class Imbalanced Learning, and Beyond”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2021), pp. 3554–3564.

[110] Steven C.H. Hoi et al. “Online learning: A comprehensive survey”. In: *Neurocomputing* 459 (2021), pp. 249–289. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.04.112.

[111] Zheda Mai et al. “Supervised Contrastive Replay: Revisiting the Nearest Class Mean Classifier in Online Class-Incremental Continual Learning”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2021), pp. 3584–3594.

[112] Alex Nichol and Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. In: *ICML* abs/2102.09672 (2021).

[113] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *International Conference on Machine Learning*. 2021.

[114] Robin Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 10674–10685.

[115] Pravendra Singh et al. “Rectification-based Knowledge Retention for Continual Learning”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 15277–15286.

[116] Sakshi Varshney et al. “CAM-GAN: Continual Adaptation Modules for Generative Adversarial Networks”. In: *Neural Information Processing Systems*. 2021.

[117] Vinay Kumar Verma et al. "Efficient Feature Transformations for Discriminative and Generative Continual Learning". In: *CVPR* abs/2103.13558 (2021). arXiv: 2103.13558.

[118] Shipeng Yan, Jiangwei Xie, and Xuming He. "DER: Dynamically Expandable Representation for Class Incremental Learning". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 3013–3022.

[119] Jaehong Yoon et al. "Online Coreset Selection for Rehearsal-based Continual Learning". In: *ICLR* abs/2106.01085 (2021).

[120] Chen Zeno et al. "Task-Agnostic Continual Learning Using Online Variational Bayes With Fixed-Point Updates". In: *Neural Computation* 33.11 (2021), pp. 3139–3177. DOI: 10.1162/neco_a_01430.

[121] Fei Zhu et al. "Prototype Augmentation and Self-Supervision for Incremental Learning". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 5867–5876. DOI: 10.1109/CVPR46437.2021.00581.

[122] Lucas Caccia et al. "New Insights on Reducing Abrupt Representation Change in Online Continual Learning". In: *ICLR* abs/2203.03798 (2022).

[123] Kamil Deja et al. "On Analyzing Generative and Denoising Capabilities of Diffusion-based Deep Generative Models". In: *Neurips 2022*. 2022. eprint: 2206.00070 (cs.LG).

[124] Yiduo Guo, Bing Liu, and Dongyan Zhao. "Online Continual Learning through Mutual Information Maximization". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 8109–8126.

[125] Minsoo Kang, Jaeyoo Park, and Bohyung Han. "Class-Incremental Learning by Knowledge Distillation with Adaptive Feature Consolidation". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 16050–16059.

[126] Xialei Liu et al. "Long-Tailed Class Incremental Learning". In: *ECCV* abs/2210.00266 (2022).

[127] Zheda Mai et al. "Online continual learning in image classification: An empirical survey". In: *Neurocomput.* 469.C (Jan. 2022), pp. 28–51. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.10.021.

[128] Julien Pourcel, Ngoc-Son Vu, and Robert M. French. "Online Task-free Continual Learning with Dynamic Sparse Distributed Memory". In: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXV*. Tel Aviv, Israel: Springer-Verlag, 2022, pp. 739–756. ISBN: 978-3-031-19805-2. DOI: 10.1007/978-3-031-19806-9_42.

[129] Julien Pourcel, Ngoc-Son Vu, and Robert M. French. "Online Task-free Continual Learning with Dynamic Sparse Distributed Memory". In: *ECCV*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 739–756.

[130] Tim Salimans and Jonathan Ho. "Progressive Distillation for Fast Sampling of Diffusion Models". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[131] Fei Ye and Adrian G. Bors. "Task-Free Continual Learning via Online Discrepancy Distance Learning". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022.

[132] OpenAI Josh Achiam et al. "GPT-4 Technical Report". In: *Arxiv*. 2023.

[133] David Berthelot et al. "TRACT: Denoising Diffusion Models with Transitive Closure Time-Distillation". In: *ICML* abs/2303.04248 (2023).

[134] R. Gao and Weiwei Liu. "DDGR: Continual Learning with Deep Diffusion-based Generative Replay". In: *International Conference on Machine Learning*. 2023.

[135] Yiduo Guo, Bing Liu, and Dongyan Zhao. " Dealing with Cross-Task Class Discrimination in Online Continual Learning ". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2023, pp. 11878–11887. DOI: 10.1109/CVPR52729.2023.01143.

[136] Jie Hao, Kaiyi Ji, and Mingrui Liu. "Bilevel Coreset Selection in Continual Learning: A New Formulation and Algorithm". In: *Neural Information Processing Systems*. 2023.

[137] Quentin Jodelet et al. "Class-Incremental Learning using Diffusion Model for Distillation and Replay". In: *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)* (2023), pp. 3417–3425.

[138] Keller Jordan et al. "REPAIR: REnormalizing Permuted Activations for Interpolation Repair". In: *The Eleventh International Conference on Learning Representations*. 2023.

[139] Hyun-woo Koh et al. "Online Boundary-Free Continual Learning by Scheduled Data Prior". In: *International Conference on Learning Representations*. 2023.

[140] Huiwei Lin et al. "PCR: Proxy-Based Contrastive Replay for Online Class-Incremental Continual Learning". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 24246–24255.

[141] Shanchuan Lin et al. "Common Diffusion Noise Schedules and Sample Steps are Flawed". In: *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2023), pp. 5392–5399.

[142] Sergi Masip et al. "Continual Learning of Diffusion Models with Generative Distillation". In: *arXiv preprint arXiv:2311.14028* (2023).

[143] Yang Song et al. "Consistency Models". In: *ICML* (2023).

[144] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. "Artificial intelligence, machine learning and deep learning in advanced robotics, a review". In: *Cognitive Robotics* 3 (2023), pp. 54–70. ISSN: 2667-2413. DOI: https://doi.org/10.1016/j.cogr.2023.04.001.

[145] Yujie Wei et al. "Online Prototype Learning for Online Continual Learning". In: *ICCV 2023* (2023). eprint: 2308.00301.

[146] Rui Yang et al. "When continual learning meets robotic grasp detection: a novel benchmark on the Jacquard dataset". In: *18th International Conference on Computer Vision Theory and Applications (VISAPP)*. Lisbon, Portugal: SCITEPRESS - Science and Technology Publications, Feb. 2023. DOI: 10.5220/0011874700003417.

[147] Michal Zajac et al. "Exploring Continual Learning of Diffusion Models". In: *arxiv* (2023).

[148] Andrea Cossu et al. "Continual pre-training mitigates forgetting in language and vision". In: *Neural Networks* 179 (2024), p. 106492. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2024.106492.

[149] Bartosz Cywiński et al. *GUIDE: Guidance-based Incremental Learning with Diffusion Models*. 2024. arXiv: 2403.03938 [cs.LG].

[150] Dipam Goswami et al. "Resurrecting Old Classes with New Data for Exemplar-Free Continual Learning". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024), pp. 28525–28534.

[151] Libo Huang et al. "eTag: Class-Incremental Learning via Embedding Distillation and Task-Oriented Generation". In: *AAAI Conference on Artificial Intelligence*. 2024.

[152] Junsu Kim et al. " SDDGR: Stable Diffusion-Based Deep Generative Replay for Class Incremental Object Detection ". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2024, pp. 28772–28781. DOI: 10.1109/CVPR52733.2024.02718.

[153] Jedrzej Kozal et al. " Continual Learning with Weight Interpolation ". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2024, pp. 4187–4195. DOI: 10.1109/CVPRW63382.2024.00422.

[154] Jinglin Liang et al. "Diffusion-Driven Data Replay: A Novel Approach to Combat Forgetting in Federated Class Continual Learning". In: *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XXX*. Milan, Italy: Springer-Verlag, 2024, pp. 303–319. ISBN: 978-3-031-73403-8. DOI: 10.1007/978-3-031-73404-5_18.

[155] Zichong Meng et al. "DiffClass: Diffusion-Based Class Incremental Learning". In: *ECCV* abs/2403.05016 (2024).

[156] Filip Szatkowski et al. "Adapt Your Teacher: Improving Knowledge Distillation for Exemplar-free Continual Learning". In: *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2024), pp. 1966–1976.

[157] L. Wang et al. "A Comprehensive Survey of Continual Learning: Theory, Method and Application". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.08 (2024), pp. 5362–5383. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2024.3367329.

[158] Hongwei Yan et al. "Orchestrate Latent Expertise: Advancing Online Continual Learning with Multi-Level Supervision and Reverse Self-Distillation". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024), pp. 23670–23680.

[159] Rui Yang et al. "Imbalanced data robust online continual learning based on evolving class aware memory selection and built-in contrastive representation learning". In: *IEEE International Conference on Image Processing (ICIP)*. Abou Dabi, United Arab Emirates, 2024.

[160] Sin-Han Yang, Tuomas Oikarinen, and Tsui-Wei Weng. "Concept-Driven Continual Learning". In: *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856.

[161] Fei Ye and Adrian G. Bors. "Online Task-Free Continual Generative and Discriminative Learning via Dynamic Cluster Memory". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 26202–26212. DOI: 10.1109/CVPR52733.2024.02476.

[162] Alex Gomez-Villa et al. "Exemplar-Free Continual Representation Learning via Learnable Drift Compensation". In: *Computer Vision – ECCV 2024*. Cham: Springer Nature Switzerland, 2025, pp. 473–490. ISBN: 978-3-031-72667-5.

[163] Nicolas Michel et al. "Rethinking momentum knowledge distillation in online continual learning". In: *Proceedings of the 41st International Conference on Machine Learning*. ICML'24. Vienna, Austria: JMLR.org, 2025.

[164] Rui Yang et al. "Entropy-Guided Self-Regulated Learning Without Forgetting for Distribution-Shift Continual Learning with blurred task boundaries". In: *5th International Conference, ROBOVIS 2025*. Feb. 2025.