

ONLINE CONTINUAL LEARNING IN IMAGE CLASSIFICATION

by

Zheda Mai

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

Abstract

Online Continual Learning in Image Classification

Zheda Mai

Master of Applied Science

Graduate Department of Mechanical and Industrial Engineering

University of Toronto

2021

Online continual learning (CL) in image classification studies the problem of learning to classify images from an online stream of data and tasks, where tasks may include data with new classes or nonstationarity. One of the key challenges of CL for neural networks is to avoid catastrophic forgetting, forgetting old tasks in the presence of more recent tasks.

This thesis starts with a comparative empirical survey of the state-of-the-art methods and recently proposed tricks to better understand their relative advantages and the settings where they work the best. We note that the methods' performance highly depends on the experimental setups, including CL setting, dataset and memory buffer size. Also, we find that memory-replay-based methods with simple tricks can produce performance levels that bring online CL much closer to the ultimate goal of matching offline training.

Given the insights from the survey, we provide a simple but effective approach that combines memory replay, CL tricks summarized in the survey and common image classification techniques, including transfer learning, data augmentation and deeper network architectures to prepare CL for real-world applications. The proposed method shows remarkable performance in different metrics, including accuracy, RAM usage and run time, etc., in various realistic settings and demonstrates the practicality of CL algorithms.

As we can see above, memory replay techniques have shown exceptional promise in online CL, but the best method for selecting which buffered images to replay is still an open question. To this end, we propose a novel Adversarial Shapley value scoring method that scores memory data samples according to their ability to preserve latent decision boundaries for previously observed classes (to maintain learning stability and avoid forgetting) while interfering with latent decision boundaries of current classes being learned (to encourage plasticity and optimal learning of new class boundaries).

Overall, this thesis has made three meaningful contributions for online CL in image classification, including comprehensively reviewing and evaluating recent approaches, addressing the challenges for bringing online CL into more realistic applications, and proposing a novel memory buffer management method that shows the state-of-the-art performance when memory buffer size is small.

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor Prof. Scott Sanner for his continual support of my master study. I am really grateful to him for bringing me into the research world, helping me with incredibly valuable insights and providing me with a huge amount of time to discuss and develop my research ideas. I wouldn't have been able to proceed without your guidance.

I wish to thank the financial, academic and technical support provided by the University of Toronto and LG AI Research for my master study.

I want to thank my fellow labmates in the Data-Driven Decision-Making Lab for enjoyable chats and informative lab meetings. In particular, I would like to pay special regards to Ga Wu, Jihwan Jeong, Ruiwen Li, Kai Luo for the sleepless nights we were working together before deadlines, for the stimulating discussions, and for all the fun we have had in the last sixteen months.

I also want to thank Hyunwoo Kim and Jongseong Jang from LG AI Research for their insightful comments and suggestions.

I must express my very profound gratitude to my family for providing me with unfailing support and constant encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Last but not least, I would like to thank my friends, especially Vickie Ruan and Luyuan Chen, who work in different fields but were always willing to hear me out when I struggled during my master study.

Statement of Contributions

1. Chapter 2 of this thesis is a collaboration with Ruiwen Li, Jihwan Jeong, David Quispe and Professor Scott Sanner from the University of Toronto, Dr. Hyunwoo Kim from the LG AI Research. Ruiwen implemented several compared methods and tricks, while Jihwan helped edit the writing, and David runs some experiments. This chapter has been incorporated into a paper that will be submitted to the Neurocomputing Journal.
2. Chapter 3 is joint work with Dr. Hyunwoo Kim from the LG AI Research, Jihwan Jeong and Professor Scott Sanner from the University of Toronto. Dr. Kim provided insightful suggestions, and Jihwan helped edit the writing. The work presented in Chapter 3 is the winning solution of the *1st Continual Learning in Computer Vision Challenge*, organized within the *CLVision* workshop at CVPR 2020.
3. Chapter 4 is a collaboration with Jihwan Jeong, Dongsub Shim and Professor Scott Sanner from the University of Toronto, Dr. Hyunwoo Kim and Dr. Jongseong Jang from the LG AI Research. Dongsub provided a general idea and some initial work. Jihwan and I worked closely together on further developing the approach and deploying it under the online class incremental setting. In the last weeks of this work, I focused on empirical validation while Jihwan worked on providing theoretical justification for the proposed approach. This chapter has been incorporated into a paper that published at the AAAI Conference on Artificial Intelligence (AAAI 2021).

Contents

| | | |
|---|---|----------|
| 1 | Introduction | 1 |
| 1.1 | General Introduction | 1 |
| 1.2 | Main Contributions | 2 |
| 1.3 | Outline | 3 |
| 2 | Comparative Empirical Study of Online Continual Learning | 4 |
| 2.1 | Motivation | 4 |
| 2.2 | Related Work | 5 |
| 2.3 | Online Class/Domain Incremental Learning | 5 |
| 2.3.1 | Problem Definition | 5 |
| 2.3.2 | Metrics | 7 |
| 2.4 | Hyperparameter Tuning for Online Continual Learning | 8 |
| 2.5 | Overview of Continual Learning Techniques | 9 |
| 2.6 | Compared Methods | 10 |
| 2.6.1 | Regularization-based methods | 10 |
| 2.6.2 | Memory-based Methods | 12 |
| 2.6.3 | Parameter-isolation-based Methods | 14 |
| 2.7 | Tricks for Memory-based Methods in the Online Class Incremental Setting | 15 |
| 2.7.1 | Error Analysis of Memory-based Methods | 16 |
| 2.7.2 | Biased Fully Connected Layer | 17 |
| 2.7.3 | Compared Tricks | 18 |
| 2.8 | Experiments | 20 |
| 2.8.1 | Experiment Settings | 20 |
| Datasets | 20 | |
| Baselines | 21 | |
| Other settings | 21 | |
| 2.8.2 | Performance Comparison in the Online Class Incremental Setting | 22 |
| Regularization-based Methods | 22 | |
| Memory-based Methods | 22 | |
| Parameter-isolation-based Methods | 23 | |
| Performance on Other Metrics | 23 | |
| 2.8.3 | Effectiveness of Tricks | 25 |
| 2.8.4 | Performance Comparison in the Online Domain Incremental Setting | 26 |

| | | |
|--------------------------|---|-----------|
| 2.8.5 | Overall Comments for Methods and Tricks | 28 |
| 2.9 | Emerging Directions in Online Continual Learning | 28 |
| 2.10 | Conclusion | 30 |
| 3 | Continual Learning for Real-World Applications | 34 |
| 3.1 | Introduction | 34 |
| 3.2 | CLVision Continual Learning Competition | 35 |
| 3.2.1 | Dataset | 35 |
| 3.2.2 | Scenarios | 36 |
| 3.2.3 | Evaluation Metrics | 36 |
| 3.3 | Batch-level Experience Replay with Review | 37 |
| 3.3.1 | General Approach | 37 |
| 3.3.2 | Architecture and Training Details | 37 |
| 3.3.3 | Final Solutions | 39 |
| 3.4 | Experiments | 39 |
| 3.4.1 | Multi-Task New Classes (MT-NC) | 39 |
| 3.4.2 | New Instances (NI) | 40 |
| 3.4.3 | New Instances and Classes (NIC) | 40 |
| 3.4.4 | Official Competition Result | 40 |
| 3.5 | Conclusion | 41 |
| 4 | Adversarial Shapley value Experience Replay | 42 |
| 4.1 | Introduction | 42 |
| 4.2 | Online Class Incremental Learning | 44 |
| 4.2.1 | Problem Definition | 44 |
| 4.2.2 | Experience Replay Methods | 44 |
| 4.3 | Efficient Computation of Shapley Value via KNN Classifier | 44 |
| 4.3.1 | Shapley Value (SV) for Machine Learning | 45 |
| 4.3.2 | Efficient KNN Shapley Value Computation | 45 |
| 4.4 | Adversarial Shapley Value Experience Replay (ASER) | 46 |
| 4.4.1 | ASER Memory Retrieval | 46 |
| 4.4.2 | KNN Shapley Value Memory Update | 48 |
| 4.5 | Experiments | 48 |
| 4.5.1 | Experiment Settings | 49 |
| Datasets | | 49 |
| Baselines | | 49 |
| Other settings | | 49 |
| 4.5.2 | Comparative Performance Evaluation | 50 |
| 4.6 | Conclusion | 52 |
| 5 | Conclusion | 53 |
| 5.1 | Summary of Contributions | 53 |
| 5.2 | Future Directions | 54 |
| Bibliography | | 55 |

| | |
|--|-----------|
| I Appendix | 69 |
| Appendix A Survey Experiment Details | 70 |
| Appendix A.1 Dataset Detail | 70 |
| Appendix A.2 Implementation Details | 70 |
| Appendix B Additional Experiments and Results | 71 |
| Appendix B.1 More Results for OCI Setting | 71 |
| Appendix B.2 OCI Tricks on Split Mini-ImageNet | 71 |
| Appendix B.3 More Results for ODI Setting | 72 |

List of Tables

| | | |
|------|---|----|
| 2.1 | Three CL scenarios | 6 |
| 2.2 | Examples of the three CL scenarios | 7 |
| 2.3 | Accuracy matrix to demonstrate evaluate metrics | 8 |
| 2.4 | Summary of recently proposed CL methods | 11 |
| 2.5 | Error analysis for A-GEM, ER and MIR | 15 |
| 2.6 | Summary of dataset statistics | 19 |
| 2.7 | Example images of different nonstationary types in NS-MiniImageNet | 21 |
| 2.8 | Average accuracy (end of training) for the OCI setting | 22 |
| 2.9 | Performance of compared tricks for the OCI setting | 25 |
| 2.10 | Running time of different tricks applying to ER | 25 |
| 2.11 | The Average Accuracy (end of training) for the ODI setting | 26 |
| 2.12 | Overall comments for compared methods | 32 |
| 2.13 | Overall comments for compared tricks | 33 |
| 3.1 | The details of the data preprocessing steps. | 39 |
| 3.2 | Multi-Task-NC Performance Comparison | 40 |
| 3.3 | NI Performance Comparison | 40 |
| 3.4 | NIC Performance Comparison | 40 |
| 3.5 | Final ranking of the competition | 41 |
| 4.1 | Average Accuracy | 50 |
| 4.2 | Average Forgetting | 50 |
| A.1 | Summary of dataset statistics | 70 |
| A.2 | Hyperparameter grid for the compared methods. | 71 |
| B.1 | Performance of compared tricks for the OCI setting on Split Mini-ImageNet | 72 |

List of Figures

| | | |
|-----|---|----|
| 2.1 | Error analysis for three memory-based methods, A-GEM, ER and MIR | 16 |
| 2.2 | The average accuracy measured by the end of each task for the OCI setting | 23 |
| 2.3 | Average Accuracy, Forgetting, Running Time, Forward Transfer and Backward Transfer for the OCI setting | 24 |
| 2.4 | Comparison of various tricks for the OCI setting on Split CIFAR-100 | 24 |
| 2.5 | Average Accuracy, Forgetting, Running Time, Forward Transfer and Backward Transfer for the ODI setting | 27 |
| 3.1 | Example images of the 50 objects in <i>CORe50</i> | 35 |
| 3.2 | DenseNet-161 Architecture | 38 |
| 3.3 | Example of pre-processing | 38 |
| 4.1 | 2D t-SNE visualization of CIFAR-100 data embeddings | 43 |
| 4.2 | Average accuracy on observed tasks | 51 |
| B.1 | The average accuracy measured for the OCI setting on CIFAR-100 with three memory sizes | 71 |
| B.2 | The average accuracy measured for the OCI setting on Mini-ImageNet with three memory sizes | 72 |
| B.3 | The average accuracy measured for the OCI setting on CORe50-NC with three memory sizes | 72 |
| B.4 | Comparison of various tricks for the OCI setting on Split Mini-ImageNet | 73 |
| B.5 | The average accuracy measured for the ODI setting on Mini-ImageNet-Noise with three memory sizes. | 73 |
| B.6 | The average accuracy measured by the end of each task for the ODI setting on Mini- ImageNet-Occlusion with three memory sizes. | 73 |
| B.7 | The average accuracy measured by the end of each task for the ODI setting on CORe50-NI with three memory sizes. | 73 |

List of Abbreviations

| | |
|----------|--|
| A-GEM : | Average Gradient Episodic Memory |
| ASER : | Adversarial Shapley Value Experience Replay |
| CL : | Continual Learning |
| CN-DPM : | Continual Neural Dirichlet Process Mixture |
| CNN : | Convolutional Neural Network |
| EWC : | Elastic Weight Consolidation |
| ER : | Experience Replay |
| KDC : | Knowledge Distillation with Classification Loss |
| LB : | Labels Trick |
| LwF : | Learning without Forgetting |
| GDumb : | Greedy Sampler and Dumb Learner |
| GEM : | Gradient Episodic Memory |
| GSS : | Gradient based Sample Selection |
| iCaRL : | Incremental Classifier and Representation Learning |
| i.i.d. : | Independent and Identically Distributed |
| KNN : | K-Nearest Neighbors |
| NCM : | Nearest Class Mean |
| NI : | New Instances |
| NIC : | New Instances and Classes |
| MI : | Multiple Iterations Trick |
| MIR : | Maximally Interfered Retrieval |
| MT-NC : | Multi-Task New Classe |
| OCI : | Online Class Incremental |

ODI : Online Domain Incremental

RL : Reinforcement Learning

SGD : Stochastic Gradient Descent

SS : Separated Softmax Trick

SV : Shapley Value

RV : Review Trick

Chapter 1

Introduction

1.1 General Introduction

With the ubiquity of personal smart devices and image-related applications, a massive amount of image data is generated daily. While image-based deep neural networks have demonstrated exceptional advances in recent years [138], incrementally updating a neural network with a nonstationary data stream results in *catastrophic forgetting* (CF) [109, 45], the inability of a network to perform well on previously seen data after updating with recent data. For this reason, conventional deep learning tends to focus on offline training, where each mini-batch is sampled i.i.d from a static dataset with multiple epochs over the training data. However, to accommodate changes in the data distribution, such a training scheme requires entirely retraining the network on the new dataset, which is inefficient and sometimes infeasible when previous data are not available due to storage limits or privacy issues.

Continual Learning (CL) studies the problem of learning from a non-i.i.d stream of data, with the goal of preserving and extending the acquired knowledge. A more complex and general viewpoint of CL is the stability-plasticity dilemma [16, 111] where stability refers to the ability to preserve past knowledge and plasticity denotes the fast adaptation of new knowledge. Following this viewpoint, CL seeks to strike a balance between learning stability and plasticity. Since CL is often used interchangeably with lifelong learning [23, 162] and incremental learning [130, 18], for simplicity, we will use CL to refer to all concepts mentioned above.

Most early CL approaches consider task-incremental setting [150], where data arrives one task at a time and the model can utilize task-ID during both training and inference time [77, 91, 103]. Specifically, a common practice in this setting is to assign a separate output layer (head) for each task; then the model just needs to classify labels within a task, which is known as multi-head evaluation [18]. However, this setting requires additional supervisory signals at inference time—namely the task-ID—to select the corresponding head, which obviates its use when the task label is unavailable.

This thesis focuses on two realistic, but challenging settings, known as ***Online Class Incremental*** (OCI) and ***Online Domain Incremental*** (ODI), where a model needs to learn from an online stream of data (each sample is seen only once) and the incoming data may include new classes (OCI) or data nonstationarity (ODI). In contrast to the task incremental setting, these settings adopt the single-head evaluation [18], where the model needs to classify all labels without task-ID. These settings are based on the practical CL desiderata proposed recently [87, 26, 37] and it has received much attention in the

past year [6, 8, 85].

To keep focus, we only consider the supervised classification problem in computer vision. Although CL is also studied in reinforcement learning [134, 116, 71] and more recently in unsupervised learning [129], the image classification problem is still the main focus for many CL researchers.

In this thesis, we address three important questions in online CL:

- Over the past few years, a large range of methods and tricks have been introduced to address the online CL problem, but there is limited consensus in the literature on experimental setups and datasets. Although most papers show that their methods surpass others in one specific setting, one open question is: *what are the relative advantages of these approaches and the settings where they work best?*
- The primary evaluation of CL in the last few years has been centred around accuracy-related metrics. However, this may lead to a biased conclusion without accounting for the scalability of these techniques over an increasing number of tasks and more complex settings [30]. *When considering more practical metrics, including accuracy, total run time, RAM usage, and more realistic CL datasets and settings, which method works the best?*
- Methods with a memory buffer have shown to be successful and efficient for the online class incremental setting [8, 6]. Since the memory buffer is the only place to store data from previous tasks, a vital but open question for these methods is *how to retrieve memory samples and update the memory buffer when new data arrives?*

1.2 Main Contributions

This thesis aims to study the online CL in image classification that requires a neural network to learn continually from an online stream of non-i.i.d data and accumulate the acquired knowledge without forgetting. Specifically, we try to address the three crucial problems we mentioned in the previous section, and the contributions of this thesis can be summarized as follows:

- (1) We provide a practical comparative survey of the state-of-the-art methods and recently proposed tricks to better understand their relative advantages and the settings where they work the best. We note that the methods' performance highly depends on the experimental setups, including CL setting, dataset and memory buffer size. Also, we find that memory-replay-based methods with simple tricks can produce performance levels that bring online CL much closer to the ultimate goal of matching offline training.
- (2) We provide a simple but effective approach that combines memory replay and commonly used techniques in image classification including, transfer learning, data augmentation and deeper network architectures. The proposed method shows remarkable performance in different metrics, including accuracy, RAM usage and run time, etc., in various realistic settings and demonstrates the practicality of CL algorithms. Our solution based on this method won the CLVision Continual Learning challenge at Computer Vision and Pattern Recognition conference, CVPR 2020.
- (3) We propose a novel memory-based method called Adversarial Shapley value Experience Replay (ASER) that leverages Shapley value (SV) to determine the contribution of memory samples to

learning performance. We introduce an adversarial perspective of SV for CL memory retrieval that scores memory data samples according to their ability to preserve latent decision boundaries for previously observed classes (to maintain learning stability and avoid forgetting) while interfering with latent decision boundaries of current classes being learned (to encourage plasticity and optimal learning of new class boundaries). Through extensive experiments on three commonly used benchmarks in the CL literature, we demonstrate that ASER provides competitive or improved performance compared to state-of-the-art memory-based methods in the OCI setting, especially when the memory buffer size is small.

1.3 Outline

The thesis proceeds as follows. Chapter 2 provides a practical comparative survey of the state-of-the-art methods and recently proposed tricks. We start with the review of recent surveys covering the advances of CL, showing the lack of empirical survey of online CL. Next, we present a general introduction to online CL, including the formal definition, evaluation metrics, hyperparameter tuning protocol. After that, we provide an overview of the general approaches in CL and summarize the recently proposed methods. Then we review the methods and tricks we compared, followed by a systematic evaluation of them in various experimental settings and datasets.

Chapter 3 introduces a simple but effective approach that won the CLVision Continual Learning challenge at Computer Vision and Pattern Recognition conference, CVPR 2020. We begin by introducing the competition, including dataset, scenarios and evaluation framework. We then explain our proposed approach, dubbed Batch-level Experience Replay with Review. This chapter ends with empirical results in all the four competition tracks and final competition result.

In Chapter 4, we propose a novel memory-based method called Adversarial Shapley value Experience Replay (ASER) that leverages Shapley value (SV) to determine the contribution of memory samples to learning performance. We start with an introduction to SV for machine learning and an efficient KNN SV computation. Then we explain how we use SV to manage the memory buffer of memory-based methods, followed by detailed experiments showing that the proposed method provides competitive or improved performance compared to state-of-the-art memory-based methods in the OCI setting, especially when the memory buffer size is small.

Finally, in Chapter 5, we summarize the main contributions of this thesis and discuss a few possible future research directions.

Chapter 2

Comparative Empirical Study of Online Continual Learning

Over the past few years, a broad range of methods and tricks have been introduced to address the online CL problem, but due to the plethora of experiment settings, a fair comparison of the state-of-the-art methods remains challenging. To this end, we begin the contributions of the thesis with a comprehensive empirical study of a broad range of methods and tricks to better understand the relative advantages of different approaches and the settings where they work best.

2.1 Motivation

Many existing CL approaches use a *task incremental* setting where data arrives one task (i.e., set of classes to be identified) at a time and the model can utilize task identity during both training and testing [77, 91, 103]. Specifically, a common practice in this setting is to assign a separate output layer (head) for each task; then the model just needs to classify labels within a task, which is known as multi-head evaluation [18]. However, this setting requires additional supervisory signals at test time — namely the task identity — to select the corresponding head, which obviates its use when the task label is unavailable.

Recently, methods have started addressing the more realistic and challenging settings, known as Online Class Incremental (OCI) and Online Domain Incremental (ODI), where a model needs to learn from an online stream of data, with each sample being seen only once. Also, the incoming data either include new classes (class incremental) or data nonstationarity (domain incremental). In contrast to the task incremental setting, these settings adopt the single-head evaluation: the model needs to classify all labels without task-IDs [18]. These settings are based on the practical CL desiderata proposed recently [87, 26, 37] and have received much attention in the past year [6, 8, 85].

Over the past few years, a broad range of methods and tricks have been introduced to address the OCI and ODI settings, but many have not been fairly and systematically compared under a variety of settings. To better understand the relative advantages of different approaches and the settings where they work best, this chapter aims to do the following:

- We fairly compare state-of-the-art methods in OCI and determine which works best at different

memory and data settings. We observe earlier proposed iCaRL [130] remains competitive when the memory buffer is small; GDumb [125] is a strong baseline that outperforms many recently proposed methods in medium-size datasets, while MIR [6] performs the best in a larger-scale dataset. Also, we experimentally and theoretically confirm that a key cause of CF is due to the recency learning bias towards new classes in the last fully connected layer owing to the imbalance between previous data and new data.

- We determine if the best OCI methods are also competitive in the ODI setting. We note that GDumb performs quite poorly in ODI, whereas MIR — already competitive in OCI — is still strongly competitive in ODI. Overall, these results allow us to conclude that MIR is a strong and versatile online CL method across a wide variety of settings.
- We evaluate the performance of 7 simple but effective “tricks” to assess their relative impacts. We find that all tricks are beneficial and when augmented with the “review trick” [104] and a nearest class mean (NCM) classifier [130], MIR produces performance levels that bring online CL much closer to its ultimate goal of matching offline training.

2.2 Related Work

With the surge in the popularity of CL, there are multiple reviews and surveys covering the advances of CL. The first group of surveys are not empirical. [121] discusses the biological perspective of CL and summarizes how various approaches alleviate catastrophic forgetting. [87] formalizes the CL problem and outlines the existing benchmarks, metrics, approaches and evaluation methods with the emphasis on robotics applications. They also recommend some desiderata and guidelines for future CL research. [122] emphasizes the importance of online CL and discusses recent advances in this setting. Although these three surveys descriptively review the recent development of CL and provide practical guidelines, they do not perform any empirical comparison between methods.

In contrast, the second group of surveys on CL are empirical. For example, [58, 150] evaluate multiple CL methods on three CL scenarios: task incremental, class incremental and domain incremental. [37] empirically analyzes and criticizes some common experimental settings, including the multi-head evaluation [18] with an exclusive output layer for each task and the use of permuted-type datasets (e.g., permuted MNIST). However, the analysis in these three works is limited to small datasets such as MNIST and Fashion-MNIST. Another two empirical studies on the performance of CL include [124, 73], but only a small number of CL methods are compared. The first extensive comparative CL survey with empirical analysis is presented in [26] which focuses on the task incremental setting with the multi-head evaluation, whereas our work addresses more practical and realistic settings, namely Online Class Incremental (OCI) and Online Domain Incremental (ODI).

2.3 Online Class/Domain Incremental Learning

2.3.1 Problem Definition

We consider the supervised image classification problem with an online (potentially infinite) non-i.i.d stream of data, following the recent CL literature [6, 8, 122, 87]. Formally, we define a data stream of

| Scenario | $P(X_{i-1}) \neq P(X_t)$ | $P(Y_{i-1}) \neq P(Y_i)$ | $\{Y_{i-1}\} \neq \{Y_i\}$ | Task-ID | Online |
|--------------------|--------------------------|--------------------------|----------------------------|--------------|----------|
| Task Incremental | ✓ | ✓ | ✓ | Train & Test | No |
| Class Incremental | ✓ | ✓ | | No | Optional |
| Domain Incremental | ✓ | | | No | Optional |

Table 2.1: Three continual learning scenarios based on the difference between D_{i-1} and D_i , following [58]. $P(X)$ is the input data distribution; $P(Y)$ is the target label distribution; $\{Y_{i-1}\} \neq \{Y_i\}$ denotes that output space are from a disjoint space which is separated by task-ID.

unknown distributions $\mathcal{D} = \{D_1, \dots, D_N\}$ over $X \times Y$, where X and Y are input and output random variables respectively, and a neural network classifier parameterized by θ , $f : X \mapsto \mathbb{R}^C$ where C is the number of classes observed so far as in [87]. At time t , a CL algorithm A^{CL} receives a mini-batch of samples (x_t^i, y_t^i) from the current distribution D_i , and the algorithm only sees this mini-batch once.

An algorithm A^{CL} is defined with the following signature:

$$A_t^{CL} : \langle f_{t-1}, (x_t, y_t), M_{t-1} \rangle \rightarrow \langle f_t, M_t \rangle \quad (2.1)$$

Where:

- f_t is the classifier at time step t .
- (x_t^i, y_t^i) is a mini-batch received at time t from D_i which contains $\{(x_{tj}^i, y_{tj}^i) \mid j \in [1, \dots, b]\}$ where b is the mini-batch size.
- M_t is an external memory which can be used to store a subset of the training samples or other useful data (e.g., the classifier from previous time step as in LwF [91]).

Note that we assume, for simplicity, a locally i.i.d stream of data where each task distribution D_i is stationary as in [103, 19]; however, this framework can also accommodate the setting in which samples are drawn non-i.i.d from D_i as in [41, 50], where concept drift may occur within D_i .

The goal of A^{CL} is to train the classifier f to continually learn new samples from the data stream without interfering with the performance of previously observed samples. Note that unless the current samples are stored in M_t , A^{CL} will not have access to these sample in the future. Formally, at time step τ , A^{CL} tries to minimize the loss incurred by all the previously seen samples with only access to the current mini-batch and data from $M_{\tau-1}$:

$$\min_{\theta} \sum_{t=1}^{\tau} \mathbb{E}_{(x_t, y_t)} [\ell(f_{\tau}(x_t; \theta), y_t)] \quad (2.2)$$

Recently, [58, 150] have categorized the CL problem into three scenarios based on the difference between D_{i-1} and D_i . Table 2.1 summarizes the differences between the three scenarios, i.e., task incremental, class incremental and domain incremental. For task incremental, the output spaces are separated by task-IDs and are disjoint between D_{i-1} and D_i . We denote this setting as $\{Y_{i-1}\} \neq \{Y_i\}$, which in turn leads to $P(Y_{i-1}) \neq P(Y_i)$. In this setting, task-IDs are available during both train and test times. For class incremental, mutually exclusive sets of classes comprise each data distribution D_i , meaning that there is no duplicated class among different task distributions. Thus $P(Y_{i-1}) \neq P(Y_i)$,

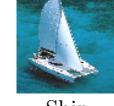
| Task | Task Incremental | | Class Incremental | | Domain Incremental | |
|---------------|------------------|---|-------------------|---|--------------------|---|
| D_{i-1} | x: |  | x: |  | x: |  |
| | y: | Bird | y: | Bird | y: | Bird |
| task-ID(test) | i-1 | | Unknown | | Unknown | |
| D_i | x: |  | x: |  | x: |  |
| | y: | Ship | y: | Guitar | y: | Dog |
| task-ID(test) | i | | Unknown | | Unknown | |

Table 2.2: Examples of the three CL scenarios. (x , y , task-ID) represents (input images, target label and task identity). The main distinction between task incremental and class incremental is the availability of task-ID. The main difference between class incremental and domain incremental is that, in class incremental, a new task contains completely new classes, whereas domain incremental, a new task consists of new instances with nonstationarity (e.g., noise) of all the seen classes.

but the output space is the same for all distributions since this setting adopts the single-head evaluation where the model needs to classify all labels without a task-ID. Domain incremental represents the setting where input distributions are different, while the output spaces and distribution are the same. Note that task IDs are not available for both class and domain incremental. Table 2.2 shows examples of these three scenarios. Following this categorization, the settings we focus in this work are known as Online Class Incremental (OCI) and Online Domain Incremental (ODI).

2.3.2 Metrics

Besides measuring the final accuracy across tasks, it is also critical to assess how fast a model learns, how much the model forgets and how well the model transfers knowledge from one task to another. To this end, we use five standard metrics in the CL literature to measure performance: (1) the average accuracy for overall performance [19]; (2) the average forgetting to measure how much of the acquired knowledge the model has forgotten [18]; (3) the forward transfer and (4) the backward transfer to assess the ability for knowledge transfer [87, 103]; (5) the total running time, including training and testing times.

Formally, we define $a_{i,j}$ as the accuracy evaluated on the held-out test set of task j after training the network from task 1 through to i , and we assume there are T tasks in total.

Average Accuracy can be defined as Eq. (2.3). When $i = T$, A_T represents the average accuracy by the end of training with the whole data sequence (see example in Table 2.3).

$$\text{Average Accuracy}(A_i) = \frac{1}{i} \sum_{j=1}^i a_{i,j} \quad (2.3)$$

Average Forgetting at task i is defined as Eq. (2.4). $f_{i,j}$ represents how much the model has forgot about task j after being trained on task i . Specifically, $\max_{l \in \{1, \dots, k-1\}} (a_{l,j})$ denotes the best test accuracy the model has ever achieved on task j before learning task k , and $a_{k,j}$ is the test accuracy on task j after learning task k .

| a | te_1 | te_2 | ... | te_{T-1} | te_T |
|------------|-------------|-------------|-----|---------------|-------------|
| tr_1 | $a_{1,1}$ | $a_{1,2}$ | ... | $a_{1,T-1}$ | $a_{1,T}$ |
| tr_2 | $a_{2,1}$ | $a_{2,2}$ | ... | $a_{2,T-1}$ | $a_{2,T}$ |
| ... | ... | ... | ... | ... | ... |
| tr_{T-1} | $a_{T-1,1}$ | $a_{T-1,2}$ | ... | $a_{T-1,T-1}$ | $a_{T-1,T}$ |
| tr_T | $a_{T,1}$ | $a_{T,2}$ | ... | $a_{T,T-1}$ | $a_{T,T}$ |

Table 2.3: Accuracy matrix following the notations in [87]. tr_i and te_i denote training and test set of task i . A_T is the average of accuracies in the box. BWT^+ is the average of accuracies in purple and FWT is the average of accuracies in green.

$$\text{Average Forgetting}(F_i) = \frac{1}{i-1} \sum_{j=1}^{i-1} f_{i,j}$$

where $f_{k,j} = \max_{l \in \{1, \dots, k-1\}} (a_{l,j}) - a_{k,j}, \forall j < k$ (2.4)

Positive Backward Transfer(BWT^+) measures the positive influence of learning a new task on preceding tasks' performance (see example in Table 2.3).

$$BWT^+ = \max\left(\frac{\sum_{i=2}^T \sum_{j=1}^{i-1} (a_{i,j} - a_{j,j})}{\frac{T(T-1)}{2}}, 0\right) \quad (2.5)$$

Forward Transfer(FWT) measures the positive influence of learning a task on future tasks' performance (see example in Table 2.3).

$$FWT = \frac{\sum_{i=1}^{j-1} \sum_{j=1}^T a_{i,j}}{\frac{T(T-1)}{2}} \quad (2.6)$$

2.4 Hyperparameter Tuning for Online Continual Learning

In practice, most CL methods have to rely on well-selected hyperparameters to effectively balance the trade-off between stability and plasticity. Hyperparameter tuning, however, is already a challenging task in learning conventional deep neural network models, which becomes even more complicated in the online CL setting. Meanwhile, a large volume of CL works still tune hyperparameters in an offline manner by sweeping over the whole data sequence and selecting the best hyperparameter set with grid-search on a validation set. After that, metrics are reported on the test set with the selected set of hyperparameters. This tuning protocol violates the online CL setting where a classifier can only make a single pass over the data, which implies that the reported results in the CL literature may be too ideal and cannot be reproduced in real online CL applications.

Recently, several hyperparameter tuning protocols that are useful for CL settings have been proposed. [124] introduces a tuning protocol for two tasks (D_1 and D_2). Firstly, they determine the best combination of model hyperparameters using D_1 . Then, they tune the learning rate to be used for learning D_2 such that the test accuracy on D_2 is maximized. [26] proposes another protocol that dynamically determines the stability-plasticity trade-off. When the model receives a new task, the hyperparameters are set to ensure minimal forgetting of previous tasks. If a predefined threshold for the current task's performance is not met, the hyperparameters are adjusted until achieving the threshold. While the

aforementioned protocols assume the data of a new task is available all at once, [19] presents a protocol targeting the online CL. Specifically, a data stream is divided into two sub-streams — D^{CV} , the stream for cross-validation and D^{EV} , the stream for final training and evaluation. Multiple passes over D^{CV} are allowed for tuning, but a CL algorithm can only perform a single pass over D^{EV} for training. The metrics are reported on the test sets of D^{EV} . Since the setting of our interest in this work is the online CL, we adopt the protocol from [19] for our experiments. We summarize the protocol in Algorithm 1.

Algorithm 1: Hyperparameter Tuning Protocol

```

Input : Hyperparameter set  $\mathcal{P}$ 
Require:  $D^{CV}$  data stream for tuning,  $D^{EV}$  data stream for learning & testing
Require:  $f$  classifier,  $A^{CL}$  CL algorithm
1 for  $p \in \mathcal{P}$  do                                 $\triangleright$  Multiple passes over  $D^{CV}$  for tuning
2   for  $i \in \{1, \dots, T^{CV}\}$  do           $\triangleright$  Single pass over  $D^{CV}$  with  $p$ 
3     for  $B_n \sim D_i^{CV}$  do
4        $\quad A^{CL}(f, B_n, p)$ 
5     Evaluate( $f, D_{test}^{CV}$ )            $\triangleright$  Store performance on test set of  $D^{CV}$ 
6 Best hyperparameters,  $\mathbf{p}^* \leftarrow$  based on Average Accuracy of  $D_{test}^{CV}$ , see Eq.(2.3)
7 for  $i \in \{1, \dots, T^{EV}\}$  do           $\triangleright$  Learning over  $D^{EV}$ 
8   for  $B_n \sim D_i^{EV}$  do
9      $\quad A^{CL}(f, B_n, \mathbf{p}^*)$            $\triangleright$  Single pass over  $D^{EV}$  with  $\mathbf{p}^*$ 
10    Evaluate( $f, D_{test}^{EV}$ )
11 Report performance on  $D_{test}^{EV}$ 

```

2.5 Overview of Continual Learning Techniques

A broad range of methods have been introduced recently to address the CL problem, but due to the plethora of settings in CL, the assumptions that each method makes are very different. Some methods have a better ability to generalize to different CL settings because they require less supervisory signals during both training and inference time. For example, ER [20] is proposed in the task incremental setting, but it can easily be used in all other CL settings since it does not need any additional supervisory signals. However, there is no systematic summary of what supervisory signals are required for each method, even though keeping track of them is critical for fair comparison and to understand the generalization and limitations of each method. Moreover, CL methods typically can be taxonomized into three major categories based on the techniques they use: regularization-based, memory-based and parameter-isolation-based [26, 121], and an increasing number of recent works simultaneously apply multiple techniques to tackle the CL problem. In this section, we comprehensively summarize recently proposed methods based on techniques they use and supervisory signals required at training and inference time (see Table 2.4).

Supervisory Signals The most important supervisory signal is task-ID. When task-ID is available, a training or testing sample is given as (x, y, t) instead of (x, y) where t is the task-ID. For the task incremental setting, task-ID is available at both training and inference time. In terms of the class incremental setting, task-ID is not available at both inference time but can be inferred at training time as each task has disjoint class labels, while in domain incremental setting, task-ID is not available at training and inference time. Other supervisory signals include a natural language description of the task or a matrix specifying the attribute values of the objects to be recognized in the task [19].

Moreover, a method is online-able if it does not need to revisit samples it has processed before, which requires the model to learn efficiently from one pass of the data. For example, the herding-based memory update strategy proposed in iCaRL [130] needs all the samples from a class to select a representative set, and therefore, methods using this strategy are not online-able.

Techniques Regularization techniques impose constraints on the update of network parameters to mitigate catastrophic forgetting. This is done by either incorporating additional penalty terms into the loss function [84, 5, 166, 133] or modifying the gradient of parameters during optimization [103, 19, 54].

Knowledge Distillation (KD) [55] is an effective way for knowledge transfer between networks. It has been widely adopted in CL methods [17, 91, 157, 128], and it is often considered as one of the regularization techniques. Due to the prevalence of KD in CL methods, we list it as a separate technique in Table 2.4. One shortcoming of regularization-based techniques including KD is that when the data stream is long, it is difficult to strike a balance between the regularization and the current learning.

Memory-based techniques store a subset of samples from previous tasks for either replay while training on new task [20, 6, 8] or regularization purpose [117, 103, 149]. These methods become infeasible when storing raw samples is not possible due to privacy or storage concerns.

Instead of saving the raw samples, an alternative is **Generative Replay** which trains a deep generative model such as GAN [44] to generate pseudo-data that mimic past data for replay [144, 158, 151]. The main disadvantages of generative replay are that it takes long time to train such models, and that it is not a viable option for more complex datasets given the current state of deep generative models [86, 6].

Parameter-isolation (PI)-based techniques bypass interference by allocating different parameters to each task. PI can be subdivided into Fixed Architecture (FA) that only activates relevant parameters for each task without modifying the architecture [106, 39, 142], and Dynamic Architecture (DA) that adds new parameters for new tasks while keeping old parameters unchanged [139, 163, 7]. Most previous works require the task-ID at inference time, and a few recent methods have been introduced to predict without task-ID [85, 126].

For a more detailed discussion of these techniques, we refer the reader to the recent CL surveys [26, 121, 87].

2.6 Compared Methods

2.6.1 Regularization-based methods

Elastic Weight Consolidation (EWC) EWC [77] incorporates a quadratic penalty to regularize the update of model parameters that were important to past tasks. The importance of parameters is approximated by the diagonal of the Fisher Information Matrix F . Assuming a model sees two tasks A and B in sequence, the loss function of EWC is:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_j \frac{\lambda}{2} F_j (\theta_j - \theta_{A,j}^*)^2 \quad (2.7)$$

where $\mathcal{L}_B(\theta)$ is the loss for task B, $\theta_{A,j}^*$ is the optimal value of j^{th} parameter after learning task A and λ controls the regularization strength. There are three major limitations of EWC: (1) It requires storing the Fisher Information Matrix for each task, which makes it impractical for a long sequence of

| Methods | Settings | | | Techniques | | | | | |
|---------------------------------|-----------------|------------------|-------------|------------|-----|----|--------|--------|------------|
| | t-ID free(test) | t-ID free(train) | Online-able | Reg | Mem | KD | PI(FA) | PI(DA) | Generative |
| MIR[6], GSS[8], ER[20] | | | | | | | | | |
| GDumb[125], DER[13], MER[131] | | | | | | ✓ | | | |
| CBRS[25], GMED[66], PRS[75] | | | | | | | | | |
| La-MAML[46], MEFA[61], CBO[12] | | | | | | | | | |
| MERLIN[67] | | | | | | | ✓ | ✓ | |
| CN-DPM[85], TreeCNN[136] | | | | | | | | ✓ | |
| A-GEM[19], GEM[103], VCL[117] | ✓ | ✓ | ✓ | | ✓ | ✓ | | | |
| WA[169], BIC[157], LUCIR[57] | | | | | ✓ | ✓ | | | |
| IL2M[10], ILO[51] | | | | | | | | | |
| LwF-MC[130], LwM[29], DMC[168] | | | | | | ✓ | | | |
| SRM[132], AQM[14] | | | | | ✓ | | | | ✓ |
| EWC++[18] | | | | | ✓ | | | | |
| AR1[107] | | | | | ✓ | | ✓ | | |
| EEIL[17], iCaRL[130], MCIL[100] | | | | | ✓ | ✓ | | | |
| SDC[164] | ✓ | ✓ | x | ✓ | | | | | |
| DGR[144] | | | | | | | | | ✓ |
| DGM[119] | | | | | | ✓ | ✓ | ✓ | |
| ICGAN[159], RtF[151] | | | | | ✓ | | | | ✓ |
| iTAML[127], CCG[2] | ✓ | x | ✓ | ✓ | ✓ | ✓ | | | |

Table 2.4: Summary of recently proposed CL methods based on supervisory signals required and techniques they use. t-ID free(test/train) means task-ID is not required at test/train time.

tasks or models with millions of parameters. (2) It needs an extra pass over each task at the end of training, leading to its infeasibility for the online CL setting. (3) Assuming the Fisher to be diagonal may not be accurate enough in practice. Several variants of EWC are proposed lately to address these limitations [141, 18, 97]. As we use the online CL setting, we compare EWC++, an efficient and online version of EWC that keeps a single Fisher Information Matrix calculated by moving average. Specifically, given F^{t-1} at $t-1$, the Fisher Information Matrix at t is updated as:

$$F^t = \alpha F_{tmp}^t + (1 - \alpha) F^{t-1} \quad (2.8)$$

where F_{tmp}^t is the Fisher Information Matrix calculated with the current batch of data and $\alpha \in [0, 1]$ is a hyperparameter controlling the strength of favouring the current F^t .

Learning without Forgetting (LwF) LwF [91] utilizes knowledge distillation [55] to preserve knowledge from past tasks in the multi-head setting [18]. In LwF, the teacher model is the model after learning the last task, and the student model is the model trained with the current task. Concretely, when the model receives a new task (X_n, Y_n) , LwF computes Y_o , the output of old tasks for the new data X_n . During training, LwF optimizes the following loss:

$$\mathcal{L}(\theta) = \left(\lambda_o \mathcal{L}_{KD} \left(Y_o, \hat{Y}_o \right) + \mathcal{L}_{CE} \left(Y_n, \hat{Y}_n \right) + \mathcal{R}(\theta) \right) \quad (2.9)$$

where \hat{Y}_o and \hat{Y}_n are the predicted values of the old task and new task using the same X_n . \mathcal{L}_{KD} is the knowledge distillation loss incorporated to impose output stability of old tasks with new data and \mathcal{L}_{CE} is the cross-entropy loss for the new task. \mathcal{R} is a regularization term, and λ_o is a hyperparameter controlling the strength of favouring the old tasks over the new task. A known shortcoming of LwF is its

heavy reliance on the relatedness between the new and old tasks. Thus, LwF may not perform well when the distributions of the new and old tasks are different [91, 130, 7]. To apply LwF in the single-head setting where all tasks share the same output head, a variant of LwF (LwF.MC) is proposed in [130], and we evaluate LwF.MC in this work.

2.6.2 Memory-based Methods

A generic online memory-based method is presented in Algorithm 2. For every incoming mini-batch, the algorithm retrieves another mini-batch from a memory buffer, updates the model using both the incoming and memory mini-batches and then updates the memory buffer with the incoming mini-batch. What differentiate various memory-based methods are the memory retrieval strategy (line 3) [6, 105], model update (line 4) [19, 103] and the memory update strategy (line 5) [25, 75, 8].

Algorithm 2: Generic online Memory-based method

```

Input : Batch size  $b$ , Learning rate  $\alpha$ 
Initialize: Memory  $\mathcal{M} \leftarrow \{\} * M$ ; Parameters  $\theta$ ; Counter  $n \leftarrow 0$ 
1 for  $t \in \{1, \dots, T\}$  do
2   for  $B_n \sim D_t$  do
3      $B_{\mathcal{M}} \leftarrow \text{MemoryRetrieval}(B_n, \mathcal{M})$ 
4      $\theta \leftarrow \text{ModelUpdate}(B_n \cup B_{\mathcal{M}}, \theta, \alpha)$ 
5      $\mathcal{M} \leftarrow \text{MemoryUpdate}(B_n, \mathcal{M})$ 
6    $n \leftarrow n + b$ 
7 return  $\theta$ 

```

Averaged GEM (A-GEM) A-GEM [19] is a more efficient version of GEM [103]. Both methods prevent forgetting by constraining the parameter update with the samples in the memory buffer. At every training step, GEM ensures that the loss of the memory samples for each individual preceding task does not increase, while A-GEM ensures that the average loss for all past tasks does not increase. Specifically, let g be the gradient computed with the incoming mini-batch and g_{ref} be the gradient computed with the same size mini-batch randomly selected from the memory buffer. In A-GEM, if $g^T g_{ref} \geq 0$, g is used for gradient update but when $g^T g_{ref} < 0$, g is projected such that $g^T g_{ref} = 0$. The gradient after projection is:

$$\tilde{g} = g - \frac{g^T g_{ref}}{g_{ref}^T g_{ref}} g_{ref} \quad (2.10)$$

As we can see, A-GEM focuses on *ModelUpdate* in Algorithm 2, and we apply reservoir sampling [154] in *MemoryUpdate* and random sampling in *MemoryRetrieval* for A-GEM.

Incremental Classifier and Representation Learning (iCaRL) iCaRL [130] is a replay-based method that decouples the representation learning and classification. For representation learning, the training set is constructed by mixing all the samples in the memory buffer and the current task samples. The loss function includes a classification loss to encourage the model to predict the correct labels for new classes and a KD loss to prompt the model to reproduce the outputs from the previous model for old classes. Note that the training set is imbalanced since the number of new-class samples in the current

task is larger than that of the old-class samples in the memory buffer. iCaRL applies the binary cross entropy (BCE) for each class to handle the imbalance, but BCE may not be effective in addressing the relationship between classes. For the classifier, iCaRL uses a nearest-class-mean classifier [110] with the memory buffer to predict labels for test images. Moreover, it proposes a *MemoryUpdate* method based on the distance in the latent feature space with the inspiration from [155]. For each class, it looks for a subset of samples whose mean of latent features are closest (in the Euclidean distance) to the mean of all the samples in this class. However, this method requires all samples from every class, and therefore it cannot be applied in the online setting. As such, we modify iCaRL to use reservoir sampling [154], which has been shown effective for *MemoryUpdate* [20].

Experience Replay (ER) ER refers to a simple but effective replay-based method that has been discussed in [20, 47]. It applies reservoir sampling [154] in *MemoryUpdate* and random sampling in *MemoryRetrieval*. Reservoir sampling ensures every streaming data point has the same probability, mem_{sz}/n , to be stored in the memory buffer, where mem_{sz} is the size of the buffer and n is the number of data points observed up to now. We summarize the detail in Algorithm 3. For *ModelUpdate*, ER simply trains the model with the incoming and memory mini-batches together using the cross-entropy loss. Despite its simplicity, recent research has shown that ER outperforms many specifically designed CL approaches with and without a memory buffer [20].

Algorithm 3: Reservoir sampling

```

1 procedure MemoryUpdate ( $\text{mem}_{sz}, t, n, B$ )
3    $j \leftarrow 0$ 
5   for  $(x, y)$  in  $B$  do
7      $M \leftarrow |\mathcal{M}|$                                 ▷ Number of samples currently stored in the memory
9     if  $M < \text{mem}_{sz}$  then
11        $\mathcal{M}.\text{append} (x, y, t)$ 
12     else
14        $i = \text{randint}(0, n + j)$ 
16       if  $i < \text{mem}_{sz}$  then
18          $\mathcal{M}[i] \leftarrow (x, y, t)$                   ▷ Overwrite memory slot
20        $j \leftarrow j + 1$ 
21   return  $\mathcal{M}$ 

```

Maximally Interfered Retrieval (MIR) MIR [6] is a recently proposed replay-based method aiming to improve the *MemoryRetrieval* strategy. MIR chooses replay samples according to the loss increases given the estimated parameter update based on the incoming mini-batch. Concretely, when receiving a mini-batch B_n , MIR performs a virtual parameter update $\theta^v \leftarrow \text{SGD}(B_n, \theta)$. Then it retrieves the top-k samples from the memory buffer with the criterion $s(x) = l(f_{\theta^v}(x), y) - l(f_\theta(x), y)$, where $x \in M$ and M is the memory buffer. Intuitively, MIR selects memory samples that are maximally interfered (the largest loss increases) by the parameter update with the incoming mini-batch. MIR applies reservoir sampling in *MemoryUpdate* and replays the selected memory samples with new samples in *ModelUpdate*.

Gradient based Sample Selection (GSS) GSS [8] is another replay-based method that focuses on the *MemoryUpdate* strategy¹. Specifically, it tries to diversify the gradient directions of the samples in the memory buffer. To this end, GSS maintains a score for each sample in the buffer, and the score is calculated by the maximal cosine similarity in the gradient space between the sample and a random subset from the buffer. When a new sample arrives and the memory buffer is full, a randomly selected subset is used as the candidate set for replacement. The score of a sample in the candidate set is compared to the score of the new sample, and the sample with a lower score is more likely to be stored in the memory buffer. Algorithm 4 shows the main steps of this update method. Same as ER, GSS uses random sampling in *MemoryRetrieval*.

Algorithm 4: GSS-Greedy

```

2 Input :  $n, M$ 
4 Initialize:  $\mathcal{M}, \mathcal{C}$ 
6 Receive:  $(x, y)$ 
8 Update:  $(x, y, \mathcal{M})$ 
10  $X, Y \leftarrow \text{RandomSubset}(\mathcal{M}, n)$ 
12  $g \leftarrow \nabla \ell_\theta(x, y); G \leftarrow \nabla_\theta \ell(X, Y)$ 
14  $c = \max_i \left( \frac{\langle g, G_i \rangle}{\|g\| \|G_i\|} \right) + 1$  ▷ make the score positive
16 if  $\text{len}(\mathcal{M}) >= M$  then
18   if  $c < 1$  then ▷ cosine similarity < 0
20      $i \sim P(i) = \mathcal{C}_i / \sum_j \mathcal{C}_j$ 
22      $r \sim \text{uniform}(0, 1)$ 
24     if  $r < \mathcal{C}_i / (\mathcal{C}_i + c)$  then
26        $\mathcal{M}_i \leftarrow (x, y); \mathcal{C}_i \leftarrow c$ 
28     end if
30   end if
32 else
34    $\mathcal{M} \leftarrow \mathcal{M} \cup \{(x, y)\}; \mathcal{C} \cup \{c\}$ 
36 end if

```

Greedy Sampler and Dumb Learner (GDumb) GDumb [125] is not specifically designed for CL problems but shows very competitive performance. Specifically, it greedily updates the memory buffer from the data stream with the constraint to keep a balanced class distribution (Algorithm 5). At inference, it trains a model from scratch using the balanced memory buffer only.

2.6.3 Parameter-isolation-based Methods

Continual Neural Dirichlet Process Mixture (CN-DPM) CN-DPM [85] is one of the first dynamic architecture methods that does not require a task-ID. The intuition behind this method is that if we train a new model for a new task and leave the existing models intact, we can retain the knowledge of the past tasks. Specifically, CN-DPM is comprised of a group of experts, where each expert is responsible for a subset of the data and the group is expanded based on the Dirichlet Process Mixture [38] with

¹[8] proposes two gradient-based methods, and we select the more efficient one with better performance, dubbed GSS-Greedy

Algorithm 5: Greedy Balancing Sampler

```

2 Init: counter  $C_0 = \{\}$ ,  $\mathcal{D}_0 = \{\}$  with capacity  $k$ . Online samples arrive from  $t = 1$ 
4 function SAMPLE  $(x_t, y_t, \mathcal{D}_{t-1}, \mathcal{Y}_{t-1})$                                  $\triangleright$  Input: New sample and past state
6    $k_c = \frac{k}{|\mathcal{Y}_{t-1}|}$ 
8   if  $y_t \notin \mathcal{Y}_{t-1}$  or  $C_{t-1}[y_t] < k_c$  then
10    if  $\sum_i C_i >= k$  then
12       $y_r = \text{argmax}(C_{t-1})$                                  $\triangleright$  If memory is full, replace
14       $(x_i, y_i) = \mathcal{D}_{t-1} \cdot \text{random}(y_r)$                  $\triangleright$  Select largest class, break ties randomly
16       $\mathcal{D}_t = (\mathcal{D}_{t-1} - (x_i, y_i)) \cup (x_t, y_t)$            $\triangleright$  Select random sample from class  $y_r$ 
18       $C_t[y_r] = C_{t-1}[y_r] - 1$ 
20    else
22       $\mathcal{D}_t = \mathcal{D}_{t-1} \cup (x_t, y_t)$                                  $\triangleright$  If memory has space, add
24    end if
26     $\mathcal{Y}_t = \mathcal{Y}_{t-1} \cup y_t$ 
28     $C_t[y_t] = C_{t-1}[y_t] + 1$ 
30  end if
32  return  $\mathcal{D}_t$ 
34 end function

```

| Method | e(n, o) | e(n, n) | e(o, o) | e(o, n) | er(n, o) | er(o, n) |
|--------|---------|---------|---------|---------|----------|----------|
| A-GEM | 0 | 177 | 0 | 9500 | 0% | 100% |
| ER | 37 | 148 | 2269 | 5852 | 20% | 72% |
| MIR | 54 | 113 | 2770 | 5330 | 32% | 66% |

Table 2.5: Error analysis of CIFAR-100 by the end of training with M=5k. e(n, o) & e(n, n) represent the number of test samples from new classes that are misclassified as old classes and new classes, respectively. Same notation rule is applied to e(o, o) & e(o, n).

Sequential Variational Approximation [92]. Each expert consists of a discriminative model (classifier) and a generative model (VAE [76] is used in this work). The goal of CN-DPM is to model the overall conditional distribution as a mixture of task-wise conditional distributions as the following, where K is the number of experts in the current model:

$$p(y | x) = \sum_{k=1}^K \underbrace{p(y | x, z=k)}_{\text{discriminative}} \overbrace{\frac{\overbrace{p(x | z=k)}^{\text{generative}} p(z=k)}{\sum_{k'=1}^K p(x | z=k') p(z=k')}}^{(2.11)}$$

2.7 Tricks for Memory-based Methods in the Online Class Incremental Setting

In class incremental learning, old class samples are generally not available while training on new class samples. Although keeping a portion of old class samples in a memory buffer has been proven effective [20, 130], the *class imbalance* is still a serious problem given a limited buffer size. Moreover, multiple recent works have revealed that class imbalance is one of the most crucial causes of catastrophic forgetting [17, 157, 57]. To alleviate the class imbalance, many simple but effective tricks have been proposed as

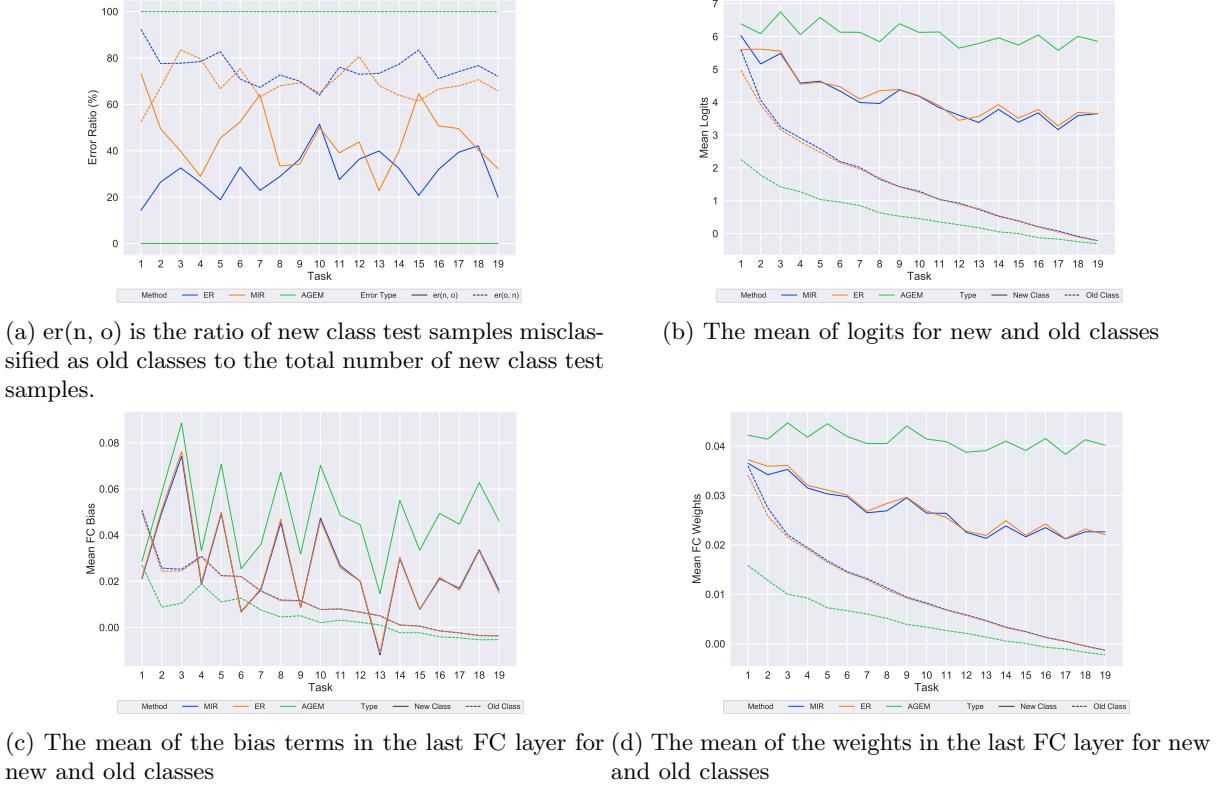


Figure 2.1: Error analysis for three memory-based methods(A-GEM, ER and MIR) with 5k memory buffer on Split CIFAR-100 described in Section 2.8.1.

the building blocks for CL methods by modifying the loss function, post-processing, or using different types of classifiers.

In Section 2.7.1 and 2.7.2, we perform quantitative error analysis and disclose how class imbalance results in catastrophic forgetting. Section 2.7.3 explains the compared tricks in detail.

2.7.1 Error Analysis of Memory-based Methods

We perform quantitative error analysis for three memory-based methods with 5k memory buffer (A-GEM, ER and MIR) on Split CIFAR-100 described in Section 2.8.1. We define $e(n, o)$ and $e(n, n)$ as the number of test samples from new classes that are misclassified as old classes and new classes, respectively. The same notation rule is applied to $e(o, o)$ and $e(o, n)$. Also, $er(n, o)$ denotes the ratio of new class test samples misclassified as old classes to the total number of new class test samples. $er(o, n)$ is similarly defined.

As shown in Table 2.5, all methods have strong bias towards new classes by the end of the training: A-GEM classifies all old class samples as new classes; ER and MIR misclassify 72% and 66% old class samples as new classes, respectively. Moreover, as we can see in Fig. 2.1a, $er(o, n)$ is higher than $er(n, o)$ most of the times along the training process for all three methods. These phenomena are not specific to the OCI setting, and [4, 169] also found similar results in the offline class incremental learning. Additionally, we easily find that ER and MIR are always better than A-GEM in terms of $er(o, n)$. This is because ER and MIR use the memory samples more directly, namely replaying them with the new

class samples. The indirect use of memory samples in A-GEM is less effective in the class incremental setting.

2.7.2 Biased Fully Connected Layer

To better understand how class imbalance affects the learning performance, we define the following notations. The convolutional neural network (CNN) classification model can be split into a feature extractor $\phi(\cdot) : \mathbf{x} \mapsto \mathbb{R}^d$ where d is the dimension of the feature vector of image \mathbf{x} , and a fully-connected (FC) layer with Softmax output. The output of the FC layer is obtained with:

$$\text{logits}(\mathbf{x}) = \mathbf{W}^T \phi(\mathbf{x}) \quad (2.12)$$

$$\mathbf{o}(\mathbf{x}) = \text{Softmax}(\text{logits}(\mathbf{x})) \quad (2.13)$$

where $\mathbf{W} \in \mathbb{R}^{d \times (|C_{old}| + |C_{new}|)}$, and C_{old} and C_{new} are the sets of old and new classes respectively, with $|\cdot|$ denoting the number of classes in each set. \mathbf{W}_i represents the weight vector for class i . For notational brevity, \mathbf{W} also contains the bias terms.

We start by analyzing the mean of logits for new and old classes. As shown in Fig. 2.1b, the mean of logits for new classes is always much higher than that for old classes, which explains the high $er(o, n)$ for all three methods. As we can see from Eq. (2.12), both feature extractor $\phi(\mathbf{x})$ and FC layer \mathbf{W} may potentially contribute to the logit bias. However, previous works have found that even a small memory buffer (implying high class imbalance) can greatly alleviate catastrophic forgetting in the multi-head setting where the model can utilize the task-id to select the corresponding FC layer for each task [20, 19]. This suggests that the feature extractor is not heavily affected by the class imbalance, and therefore we hypothesize that the FC layer is biased.

To validate the hypothesis, we plot the means of bias terms and weights in the FC layer for new and old classes in Fig. 2.1c and 2.1d. As we can see, the means of weights for the new classes are much higher than those for the old classes, and the means of bias terms for the new classes are also higher than those for the old classes most of the times. Since the biased weights and bias terms in \mathbf{W} have direct impacts on the output logits, the model has a higher chance of predicting a sample as new classes. [4, 157, 169] have also verified the same hypothesis but with different validation methods.

A recent work reveals how does class imbalance result in a biased \mathbf{W} [4]. We denote $s_i = \mathbf{W}_i^T \phi(\mathbf{x})$ as the logit of sample \mathbf{x} for class i . The gradient of the cross-entropy loss L_{CE} with respect to \mathbf{W}_i for the Softmax output is:

$$\frac{\partial L_{CE}}{\partial \mathbf{W}_i} = \left(\frac{e^{s_i}}{\sum_{j \in C_{old} + C_{new}} e^{s_j}} - \mathbb{1}_{\{i=y\}} \right) \phi(\mathbf{x}) \quad (2.14)$$

where y is the ground-truth class and $\mathbb{1}_{\{i=y\}}$ is the indicator for $i = y$. Since ReLU is often used as the activation function for the embedding networks [52], $\phi(\mathbf{x})$ is always positive. Therefore, the gradient is always positive for $i \neq y$. If i belongs to old classes, $i \neq y$ will hold most of the time as the new class samples significantly outnumber the old class samples during training. Thus, the logit for class i will keep being penalized during the gradient descent. As a result, the logits for the old classes are prone to be smaller than those for the new classes, and the model is consequently biased towards new classes.

Other than the bias related to the Softmax classifier mentioned above, another problem induced by class imbalance is under-representation of the minority [33, 161], where minority classes do not show

a discernible pattern in the latent feature space [75]. The under-representation introduces additional difficulty for other classifiers apart from the Softmax classifier, such as nearest-class-mean [110] and cosine-similarity-based classifier [43].

2.7.3 Compared Tricks

To mitigate the strong bias towards new classes due to class imbalance, multiple methods have been proposed lately [83, 69, 169, 10, 34]. For example, LUCIR [57] proposes three tricks: cosine normalization to balance class magnitudes, a margin ranking loss for inter-class separation, and a less-forgetting constraint to preserve the orientation of features. BiC [157] trains an additional linear layer to remove bias with a separate validation set. We compare seven simple (effortless integration without additional resources) but effective (decent improvement) tricks in this work.

Labels Trick (LB) [167] proposes to consider only the outputs for the classes in the current mini-batch when calculating the cross-entropy loss, in contrast to the common practice of considering outputs for all the classes. To achieve this, the outputs that do not correspond to the classes of the current mini-batch are masked out when calculating the loss.

Although the author did not demonstrate the motivation of this trick, we can easily find the rationale based on the analysis in Section 2.7.2. Masking out all the outputs that don't match the classes in the current mini-batch is equivalent to changing the loss function to:

$$\mathcal{L}_{CE}(x_i, y_i) = -\log \left(\frac{e^{s_{y_i}}}{\sum_{j \in C_{cur}} e^{s_j}} \right) \quad (2.15)$$

where C_{cur} denotes the classes in the current mini-batch. We can see that $\frac{\partial \mathcal{L}_{CE}}{\partial s_j} = 0$ for $j \notin C_{cur}$, and therefore training with the current mini-batch will not overly penalize the logits for classes that are not in the mini-batch.

Knowledge Distillation with Classification Loss (KDC) KD [55] is an effective way for knowledge transfer between networks. Multiple recent works have proposed different ways to combine the KD loss with the classification loss [130, 17, 62]. In this part, we compare the methods from [157]. Specifically, the loss function is given as:

$$\mathcal{L}(\mathbf{x}, y) = \lambda \mathcal{L}_{CE}(\mathbf{x}, y) + (1 - \lambda) \mathcal{L}_{KD}(\mathbf{x}) \quad (2.16)$$

where λ is set to $\frac{|C_{new}|}{|C_{old}|+|C_{new}|}$. Note that (\mathbf{x}, y) is from both new class data from the current task and old class data from the memory buffer.

As shown in Table 2.9, however, this method does not perform well in our experiment setting, especially with a large memory buffer. We identify λ as the key issue. We find that the accuracy for new class samples becomes almost zero around the end of training because λ is very small. In other words, \mathcal{L}_{KD} dominates the loss, and the model cannot learn any new knowledge. Hence, we suggest setting λ to $\sqrt{\frac{|C_{new}|}{|C_{old}|+|C_{new}|}}$. We denote the trick with this modification as KDC*.

Multiple Iterations (MI) Most of the previous works only perform a single gradient update on the incoming mini-batch in the online setup. [6] suggests performing multiple gradient updates to maximally

| Dataset | #Task | #Train/task | #Test/task | #Class | Image Size | Setting |
|------------------------|-------|-------------|------------|--------|------------|---------|
| Split MiniImageNet | 20 | 2500 | 500 | 100 | 3x84x84 | OCI |
| Split CIFAR-100 | 20 | 2500 | 500 | 100 | 3x32x32 | OCI |
| CORe50-NC | 9 | 12000~24000 | 4500~9000 | 50 | 3x128x128 | OCI |
| NS-MiniImageNet | 10 | 5000 | 1000 | 100 | 3x84x84 | ODI |
| CORe50-NI ² | 8 | 15000 | 44972 | 50 | 3x128x128 | ODI |

Table 2.6: Summary of dataset statistics

utilize the current mini-batch. Particularly for replay methods, additional updates with different replay samples can improve performance. We run 5 iterations per incoming mini-batch and retrieve different memory samples for each iteration in this work.

Nearest Class Mean (NCM) Classifier To tackle the biased FC layer, one can replace the FC layer and Softmax classifier with another type of classifier. Nearest Class Mean classifier (NCM) [110] is a popular option in CL [130, 164]. To make prediction for a sample \mathbf{x} , NCM computes a prototype vector for each class and assigns the class label with the most similar prototype:

$$\mu_y = \frac{1}{|M_y|} \sum_{\mathbf{x}_m \in M_y} \phi(\mathbf{x}_m) \quad (2.17)$$

$$y^* = \operatorname{argmin}_{y=1,\dots,t} \|\phi(\mathbf{x}) - \mu_y\| \quad (2.18)$$

In the class incremental setting, the true prototype vector for each class cannot be computed due to the unavailability of the training data for previous tasks. Instead, the prototype vectors can be approximated using the data in the memory buffer. In Eq. (2.17), M_y denotes the memory samples of class y .

Separated Softmax (SS) Since training the whole FC layer with one Softmax output layer results in bias as explained in Section 2.7.2, SS [4] employs two Softmax output layers: one for new classes and another one for old classes. The loss function can be calculated as below:

$$\begin{aligned} & \mathcal{L}(\mathbf{x}_i, y_i) \\ &= -\log \left(\frac{e^{s_{y_i}}}{\sum_{j \in C_{old}} e^{s_j}} \right) \cdot \mathbb{1}\{y_i \in C_{old}\} - \log \left(\frac{e^{s_{y_i}}}{\sum_{j \in C_{new}} e^{s_j}} \right) \cdot \mathbb{1}\{y_i \in C_{new}\} \end{aligned} \quad (2.19)$$

Depending on whether $y_i \in C_{new}$ or C_{old} , the corresponding Softmax is used to compute the cross-entropy loss. We can find that $\frac{\partial \mathcal{L}}{\partial s_j} = 0$ for $j \in C_{old}$ and $y_i \in C_{new}$. Thus, training with new class samples will not overly penalize the logits for the old classes.

Review Trick (RV) To alleviate the class imbalance, [17] proposes an additional fine-tuning step with a small learning rate, which uses a balanced subset from the memory buffer and the training set of the current task. A temporary distillation loss for new classes is applied to avoid forgetting the new classes during the fine-tuning phase mentioned above. A similar yet simplified version, dubbed Review Trick, is applied in the winning solution in the continual learning challenge at CVPR2020 [104]. At the end of learning the current task, the review trick fine-tunes the model with all the samples in the memory buffer using only the cross-entropy loss. In this work, we compare the review trick from [104] with a learning rate 10 times smaller than the training learning rate.

2.8 Experiments

Section 2.8.1 explains the general setting for all experiments. Then, we focus on the OCI setting in Section 2.8.2 and 2.8.3: we evaluate all the compared methods and baselines in Section 2.8.2 and investigate the effectiveness of the seven tricks in Section 2.8.3. In Section 2.8.4, we assess the compared methods in the ODI setting to investigate their abilities to generalize, and Section 2.8.5 provides general comments on the surveyed methods and tricks.

2.8.1 Experiment Settings

Datasets

We evaluate the nine methods summarized in Section 2.6 and additional two baselines on three class incremental datasets. We also propose a new domain incremental dataset based on Mini-ImageNet and examine the compared methods in the ODI setting to see how well the methods can generalize to this setting. The summary of dataset statistics is provided in Table 2.6.

Class Incremental Datasets

- **Split CIFAR-100** is constructed by splitting the CIFAR-100 dataset [79] into 20 tasks with disjoint classes, and each task has 5 classes. There are 2,500 $3 \times 32 \times 32$ images for training and 500 images for testing in each task.
- **Split MiniImageNet** splits MiniImageNet dataset [153], a subset of ImageNet [28] with 100 classes, into 20 disjoint tasks as in [20]. Each task contains 5 classes, and every class consists of 500 $3 \times 84 \times 84$ images for training and 100 images for testing.
- **CORe50-NC** [101] is a benchmark designed for class incremental learning with 9 tasks and 50 classes: 10 classes in the first task and 5 classes in the subsequent 8 tasks. Each class has around 2,398 $3 \times 128 \times 128$ training images and 900 testing images.

Domain Incremental Datasets

- **NonStationary-MiniImageNet** (NS-MiniImageNet) The most popular domain incremental datasets are still based on MNIST [81], such as Rotation MNIST [103] and Permutation MNIST [77]. To evaluate the domain incremental setting in a more practical scenario, we propose NS-MiniImageNet with three nonstationary types: noise, blur and occlusion. The number of tasks and the strength of each nonstationary type are adjustable in this dataset. In this survey, we use 10 tasks for each type, and each task comprises 5,000 $3 \times 84 \times 84$ training images and 1,000 testing images. As shown in Table 2.7, the nonstationary strength increases over time, and to ensure a smooth distribution shift, the strength always increases by the same constant. More details about the nonstationary strengths used in the experiments can be found in Appendix A.
- **CORe50-NI** [101] is a practical benchmark designed for assessing the domain incremental learning with 8 tasks, where each task has around 15,000 training images of 50 classes with different types of nonstationarity including illumination, background, occlusion, pose and scale. There is one single test set for all tasks, which contains 44,972 images.

²CORe50-NI uses one test set(44972 images) for all tasks

| Type | Task 1 | ... | Task 5 | ... | Task 10 |
|-----------|---|-----|---|-----|---|
| Noise |  | ... |  | ... |  |
| Blur |  | ... |  | ... |  |
| Occlusion |  | ... |  | ... |  |

Table 2.7: Example images of different nonstationary types in NS-MiniImageNet. The nonstationary strength increases over time, and to ensure a smooth distribution shift, the strength always increases by the same constant.

Task Order and Task Composition Since the task order and task composition may impact the performance [101], we take the average over multiple runs for each experiment with different task orders and composition to reliably assess the robustness of the methods. For CORe50-NC and CORe50-NI, we follow the number of runs (i.e., 10), task order and composition provided by the authors. For Split CIFAR-100 and Split MiniImagenet, we average over 15 runs, and the class composition in each task is randomly selected for each run.

Baselines

We compare the methods we discussed in Section 2.6 with two baselines:

- **Finetune** greedily updates the model with the incoming mini-batch without considering the previous task performance. The model suffers from catastrophic forgetting and is regarded as the lower-bound.
- **Offline** trains a model using all the samples in a dataset in an offline manner. The baseline is trained for multiple epochs within each of which mini-batches are sampled i.i.d from differently shuffled dataset. We train the model for 70 epochs with the mini-batch size of 128.

Other settings

Models Similar to [20, 103], we use the reduced ResNet18 [53] as the base model for all datasets and methods. The network is trained via the cross-entropy loss with a stochastic gradient descent optimizer and a mini-batch size of 10. The size of the mini-batch retrieved from the memory buffer is also set to 10, irrespective of the size of the memory buffer as in [20]. Note that with techniques such as transfer learning (e.g., using a pre-trained model from ImageNet), data augmentation and deeper network architectures, it is possible to achieve much higher performance in this setting [102]. However, since those techniques are orthogonal to our investigation and deviate from the simpler experimental settings of other papers we cite and compare, we do not use them in our experiments.

| Method | Split CIFAR-100 | | | Split Mini-ImageNet | | | CORe50-NC | | |
|-------------|-------------------|-------------------|-------------------|---------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| Buffer Size | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k |
| Finetune | 3.7 ± 0.3 | | | 3.4 ± 0.2 | | | 7.7 ± 1.0 | | |
| Offline | 49.7 ± 2.6 | | | 51.9 ± 0.5 | | | 51.7 ± 1.8 | | |
| EWC++ | 3.7 ± 0.4 | | | 3.5 ± 0.4 | | | 8.3 ± 0.3 | | |
| LwF | 7.2 ± 0.4 | | | 7.6 ± 0.7 | | | 7.1 ± 1.9 | | |
| ER | 7.6 ± 0.5 | 17.0 ± 1.9 | 18.4 ± 1.4 | 6.4 ± 0.9 | 14.5 ± 2.1 | 15.9 ± 2.0 | 23.5 ± 2.4 | 27.5 ± 3.5 | 28.2 ± 3.3 |
| MIR | 7.6 ± 0.5 | 18.2 ± 0.8 | 19.3 ± 0.7 | 6.4 ± 0.9 | 16.5 ± 2.1 | 21.0 ± 1.1 | 27.0 ± 1.6 | 32.9 ± 1.7 | 34.5 ± 1.5 |
| GSS | 7.7 ± 0.5 | 11.3 ± 0.9 | 13.4 ± 0.6 | 5.9 ± 0.7 | 11.2 ± 0.9 | 13.5 ± 0.8 | 19.6 ± 3.0 | 22.2 ± 4.4 | 21.1 ± 3.5 |
| iCaRL | 16.7 ± 0.8 | 19.2 ± 1.1 | 18.8 ± 0.9 | 14.7 ± 0.4 | 17.5 ± 0.6 | 17.4 ± 1.5 | 22.1 ± 1.4 | 25.1 ± 1.6 | 22.9 ± 3.1 |
| A-GEM | 3.7 ± 0.4 | 3.6 ± 0.2 | 3.8 ± 0.2 | 3.4 ± 0.2 | 3.7 ± 0.3 | 3.3 ± 0.3 | 8.7 ± 0.6 | 9.0 ± 0.5 | 8.9 ± 0.6 |
| CN-DPM | 14.0 ± 1.7 | - | - | 9.4 ± 1.2 | - | - | 7.6 ± 0.4 | - | - |
| GDumb | 10.4 ± 1.1 | 22.1 ± 0.9 | 28.8 ± 0.9 | 8.8 ± 0.4 | 21.1 ± 1.7 | 31.0 ± 1.4 | 15.1 ± 1.2 | 28.1 ± 1.4 | 32.6 ± 1.7 |

Table 2.8: Average accuracy (end of training) for the OCI setting of Split CIFAR-100, Split Mini-ImageNet and CORe50-NC. Replay-based methods and a strong baseline GDumb show competitive performance across three datasets.

Other Details We evaluate the performance with 5 metrics we described in Section 2.3.2: Average Accuracy, Average Forgetting, Forward Transfer, Backward Transfer and Run Time. We use the hyperparameter tuning protocol described in Section 2.4 and give each method similar tuning budget. The details of the implementation including hyperparameter selection can be found in Appendix A.2.

2.8.2 Performance Comparison in the Online Class Incremental Setting

Regularization-based Methods

As shown in Table 2.8, EWC++ has almost the same performance as Finetune in the OCI setting. We find that the gradient explosion of the regularization terms is the root cause. Specifically, λ in EWC++ controls the regularization strength, and we need a larger λ to avoid forgetting. However, when λ increases to a certain value, the gradient explosion occurs. If we take θ_j in Eq. (2.7) as an example, the regularization term for θ_j has the gradient $\lambda F_j(\theta_j - \theta^*)$. Some model weights change significantly when it receives data with new classes, and therefore the gradients for those weights are prone to explode with a large λ . The Huber regularization proposed lately could be a possible remedy [95]. Surprisingly, we also observe that LwF, a method relying on KD, has similar performance as replay-based methods with a small memory buffer(1k) such as ER, MIR and GSS in Split CIFAR-100 and even outperforms them in Split Mini-ImageNet. In the larger and more realistic CORe50-NC, however, both EWC++ and LwF fail. This also confirms the results of three recent studies, where [88] shows the shortcomings of regularization-based approaches in the class incremental setting, [78] theoretically explains why regularization-based methods underperform memory-based methods and [11] empirically demonstrates that KD is more useful in small-scale datasets.

Memory-based Methods

Firstly, A-GEM does not work in this setting as it has almost the same performance as Finetune, implying that the indirect use of the memory samples is less efficient than the direct relay in the OCI setting. Secondly, given a small memory buffer in Split CIFAR-100 and Mini-ImageNet, iCaRL—proposed in 2017—shows the best performance. On the other hand, other replay-based methods such as ER, MIR and GSS do not work well because simply replaying with a small memory buffer yields severe class imbalance. When equipped with a larger memory buffer (5k and 10k), GDumb—a simple baseline that

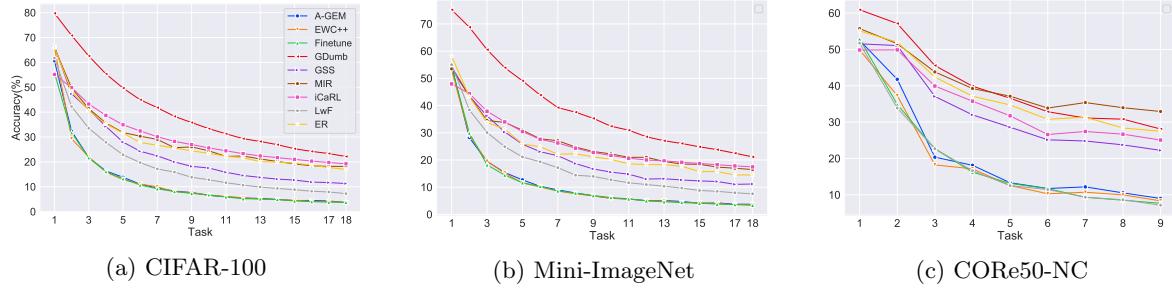


Figure 2.2: The average accuracy measured by the end of each task for the OCI setting with a 5k memory buffer. More detailed results for different memory buffer sizes are shown in Appendix B.1.

trains with the memory buffer only—outperforms other methods by a large margin. Additionally, as shown in Fig. 2.2a and Fig. 2.2b, GDumb dominates the average accuracy not only at the end of training but also at any other evaluation points along the data stream. This raises concerns about the progress in the OCI setting in the literature. Next, in the larger CORe50-NC dataset, GDumb is less effective since it only relies on the memory and the memory is smaller in a larger dataset in proportion. MIR is a robust and strong method as it exhibits remarkable performance across different memory sizes. Also, even though GSS is claimed to be an enhanced version of ER, ER consistently surpasses GSS across different memory sizes and datasets, which is also confirmed by other studies [85, 13].

Parameter-isolation-based Methods

As one of the first dynamic-architecture methods without using a task-ID, CN-DPM shows competitive results in Split CIFAR-100 and Mini-ImageNet but fails in CORe5-NC. The key reason is that CN-DPM is very sensitive to hyperparameters, and when applying CN-DPM in a new dataset, a good performance cannot be guaranteed given a limited tuning budget.

Performance on Other Metrics

We show the performance of all five metrics in Fig. 2.3. Generally speaking, we find that a high average accuracy comes with low forgetting, but methods using KD such as iCaRL and LwF, and dynamic-architecture methods such as CN-DPM have lower forgetting. The reason for the low forgetting of these methods is intransigence, the model’s inability to learn new knowledge [18]. For iCaRL and LwF, KD imposes a strong regularization, which may lead to a lower accuracy on new tasks. For CN-DPM, the inaccurate expert selector is the cause of the intransigence [85]. Furthermore, most methods have similar running time except for CN-DPM, GDumb and GSS. CN-DPM requires a significantly longer training time as it needs to train multiple experts, and each expert contains a generative model (VAE [76]) and a classifier(10-layer ResNet [53]). GDumb has the second longest running time as it requires training the model from scratch with the memory at every evaluation point. Lastly, we notice none of the methods show any forward and backward transfer, which is expected since a model tends to classify all the test samples as the current task labels due to the strong bias in the last FC layer.

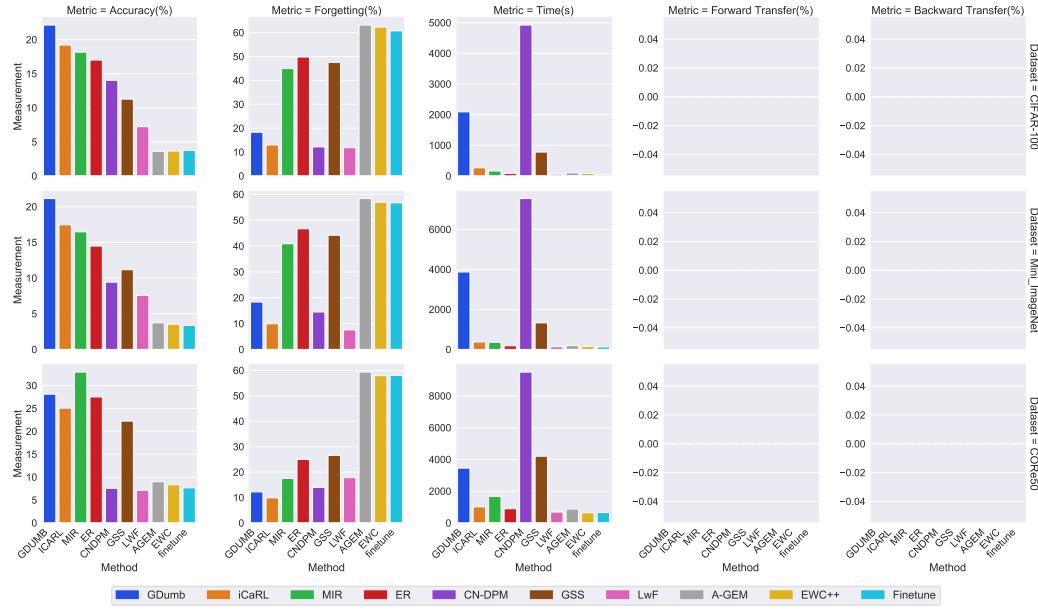


Figure 2.3: Average Accuracy, Forgetting, Running Time, Forward Transfer and Backward Transfer for the OCI setting with a 5k memory buffer. Each column represents a metric and each row represents a dataset. In this setting, none of the methods show any forward or backward transfer.

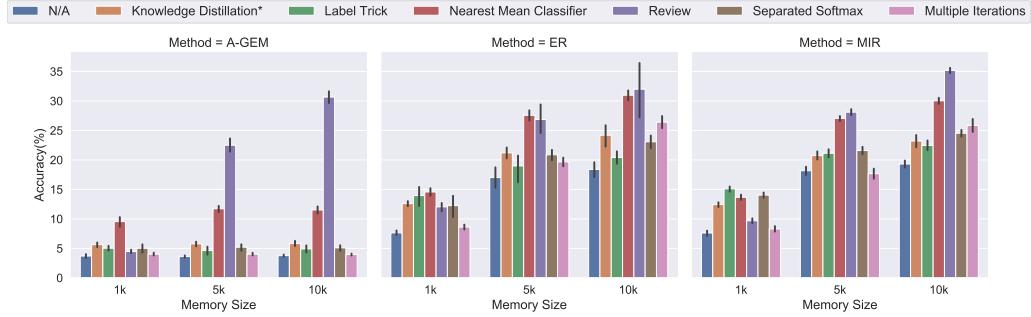


Figure 2.4: Comparison of various tricks for the OCI setting on Split CIFAR-100. We report average accuracy (end of training) for memory buffer with size 1k, 5k and 10k. N/A denotes the performance of the base methods (A-GEM, ER, MIR) without any trick.

| Finetune Offline | 3.7 ± 0.3 49.7 ± 2.6 | | | | | | | | |
|---------------------|---------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| Trick | A-GEM | | | ER | | | MIR | | |
| Buffer Size | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k |
| N/A | 3.7 ± 0.4 | 3.6 ± 0.2 | 3.8 ± 0.2 | 7.6 ± 0.5 | 17.0 ± 1.9 | 18.4 ± 1.4 | 7.6 ± 0.5 | 18.2 ± 0.8 | 19.3 ± 0.7 |
| LB | 5.0 ± 0.5 | 4.6 ± 0.8 | 4.9 ± 0.7 | 14.0 ± 2.0 | 19.0 ± 2.6 | 20.4 ± 1.2 | 15.1 ± 0.6 | 21.1 ± 0.8 | 22.5 ± 0.9 |
| KDC | 8.3 ± 0.7 | 8.8 ± 0.7 | 7.7 ± 1.1 | 11.7 ± 0.8 | 10.9 ± 1.9 | 11.9 ± 2.2 | 12.0 ± 0.5 | 12.3 ± 0.7 | 11.8 ± 0.6 |
| KDC* | 5.6 ± 0.5 | 5.8 ± 0.5 | 5.8 ± 0.5 | 12.6 ± 0.5 | 21.2 ± 1.1 | 24.2 ± 1.9 | 12.4 ± 0.5 | 20.7 ± 0.8 | 23.2 ± 1.2 |
| MI | 4.0 ± 0.3 | 4.0 ± 0.3 | 4.0 ± 0.2 | 8.6 ± 0.5 | 19.7 ± 0.9 | 26.4 ± 1.2 | 8.5 ± 0.4 | 17.7 ± 1.0 | 25.9 ± 1.2 |
| SS | 5.0 ± 0.8 | 5.2 ± 0.6 | 5.1 ± 0.5 | 12.3 ± 2.1 | 20.9 ± 1.0 | 23.1 ± 1.2 | 14.0 ± 0.5 | 21.6 ± 0.7 | 24.5 ± 0.7 |
| NCM | 9.5 ± 0.9 | 11.7 ± 0.6 | 11.5 ± 0.7 | 14.6 ± 0.7 | 27.6 ± 1.0 | 31.0 ± 1.0 | 13.7 ± 0.5 | 27.0 ± 0.5 | 30.0 ± 0.6 |
| RV | 4.5 ± 0.4 | 22.5 ± 1.3 | 30.7 ± 1.2 | 12.0 ± 0.8 | 26.9 ± 2.8 | 32.0 ± 5.3 | 9.7 ± 0.5 | 28.1 ± 0.6 | 35.2 ± 0.5 |
| Best OCI | 16.7 ± 0.8 | 22.1 ± 0.9 | 28.8 ± 0.9 | 16.7 ± 0.8 | 22.1 ± 0.9 | 28.8 ± 0.9 | 16.7 ± 0.8 | 22.1 ± 0.9 | 28.8 ± 0.9 |

Table 2.9: Performance of compared tricks for the OCI setting on Split CIFAR-100. We report average accuracy (end of training) for memory buffer with size 1k, 5k and 10k. Best OCI refers to the best performance achieved by the compared methods in Table 2.8.

| Trick | Running Time(s) | | |
|-------|-----------------|------|-------|
| | M=1k | M=5k | M=10k |
| N/A | 83 | 82 | 84 |
| LB | 87 | 88 | 89 |
| KDC | 105 | 106 | 106 |
| KDC* | 105 | 105 | 107 |
| MI | 328 | 324 | 325 |
| SS | 89 | 90 | 90 |
| NCM | 126 | 282 | 450 |
| RV | 98 | 159 | 230 |

Table 2.10: Running time of different tricks applying to ER with different memory sizes on Split CIFAR-100.

2.8.3 Effectiveness of Tricks

We evaluate Label trick (LB), Knowledge Distillation and Classification (KDC), Multiple Iterations (MI), Separated Softmax(SS), Nearest Class Mean (NCM) classifier, Review trick(RV) on three memory-based methods: A-GEM, ER and MIR. The results are shown in Table 2.9 and Fig. 2.4.

Firstly, although all tricks enhance the basic A-GEM, only RV can bring A-GEM closer to ER and MIR, which reiterates that the direct replay of the memory samples is more effective than the gradient projection approach in A-GEM.

For replay-based ER and MIR, LB and NCM are the most effective when $M=1k$ and can improve the accuracy by around 100% ($7.6\% \rightarrow 14.5\%$ on average). KDC, KDC* and SS have similar performance improvement effect and can boost the accuracy by around 64% ($7.6\% \rightarrow 12.4\%$ on average). With a larger memory size, NCM remains very effective, and RV becomes much more helpful. When $M=10k$, RV boosts ER’s performance by 74% ($18.4\% \rightarrow 32.0\%$) and improve MIR’s performance by 82% (from $19.3\% \rightarrow 35.2\%$). Also, KDC fails with a larger memory due to over-regularization of the KD term, and the modified KDC* has much better performance. Compared with other tricks, MI and RV are more sensitive to the memory size since these tricks highly depend on the memory. Note that when equipped with NCM or RV, both ER and MIR can outperform the best performance achieved by the compared methods (Table 2.8) when $M=5k$ or $10k$.

As shown in Table 2.10, the running times of LB, KDC, KDC*, MI and SS do not depend on the memory size and have a limited increase compared to the baseline. Since NCM needs to calculate the means of all the classes in the memory and RV requires additional training of the whole memory before

| Method | Mini-ImageNet-Noise | | | Mini-ImageNet-Occlusion | | | Mini-ImageNet-Blur | | | CORe50-NI | | |
|-------------|---------------------|-------------------|-------------------|-------------------------|-------------------|-------------------|--------------------|------------------|-------------------|-------------------|-------------------|-------------------|
| Finetune | 11.1 ± 1.0 | | | 13.8 ± 1.6 | | | 2.4 ± 0.2 | | | 14.0 ± 2.8 | | |
| Offline | 37.3 ± 0.8 | | | 38.6 ± 4.7 | | | 11.9 ± 1.0 | | | 51.7 ± 1.8 | | |
| EWC | 12.5 ± 0.8 | | | 14.8 ± 1.1 | | | 2.6 ± 0.2 | | | 11.6 ± 1.5 | | |
| LwF | 9.2 ± 0.9 | | | 12.8 ± 0.8 | | | 3.4 ± 0.4 | | | 11.1 ± 1.1 | | |
| Buffer Size | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k |
| ER | 19.4 ± 1.3 | 21.6 ± 1.1 | 24.3 ± 1.2 | 19.2 ± 1.5 | 23.4 ± 1.4 | 23.7 ± 1.1 | 5.3 ± 0.6 | 8.6 ± 0.8 | 9.4 ± 0.7 | 24.1 ± 4.2 | 28.3 ± 3.5 | 30.0 ± 2.8 |
| MIR | 18.1 ± 1.1 | 22.5 ± 1.4 | 24.4 ± 0.9 | 17.6 ± 0.7 | 22.0 ± 1.1 | 23.8 ± 1.2 | 5.5 ± 0.5 | 8.1 ± 0.6 | 9.6 ± 1.0 | 26.5 ± 1.0 | 34.0 ± 1.0 | 33.3 ± 1.7 |
| GSS | 18.9 ± 0.8 | 21.4 ± 0.9 | 23.2 ± 1.1 | 17.7 ± 0.8 | 21.0 ± 2.2 | 23.2 ± 1.4 | 5.2 ± 0.5 | 7.6 ± 0.6 | 8.0 ± 0.6 | 25.5 ± 2.1 | 27.2 ± 2.0 | 25.3 ± 2.1 |
| A-GEM | 14.0 ± 1.3 | 14.6 ± 0.7 | 14.2 ± 1.4 | 16.4 ± 0.7 | 13.9 ± 2.6 | 14.4 ± 2.0 | 4.4 ± 0.4 | 4.4 ± 0.4 | 4.3 ± 0.5 | 12.4 ± 1.1 | 13.8 ± 1.2 | 15.0 ± 2.2 |
| CN-DPM | 4.6 ± 0.5 | - | - | 3.9 ± 0.8 | - | - | 2.2 ± 0.2 | - | - | 9.6 ± 3.9 | - | - |
| GDumb | 5.4 ± 1.0 | 12.5 ± 0.7 | 15.2 ± 0.5 | 5.4 ± 0.4 | 14.2 ± 0.6 | 20.2 ± 0.4 | 3.3 ± 0.2 | 7.5 ± 0.2 | 10.0 ± 0.2 | 9.6 ± 1.5 | 11.2 ± 2.0 | 11.5 ± 1.7 |

Table 2.11: The Average Accuracy (end of training) for the ODI setting of Mini-ImageNet with three nonstationary types (Noise, Occlusion, Blur) and CORe50-NI.

evaluation, their running times increase as the memory size grows.

To sum up, all of the tricks are beneficial to the base methods (A-GEM, ER, MIR); NCM is a useful and robust trick across all memory sizes, while LB and RV are more advantageous in smaller and larger memory, respectively. The running times of NCM and RV go up with the increase in the memory size, and other tricks only add a fixed overhead to the running time. We also get similar results in Split Mini-ImageNet, as shown in Appendix B.

2.8.4 Performance Comparison in the Online Domain Incremental Setting

Since most of the surveyed methods are only evaluated in the class incremental setting in the original papers, we evaluate them in the ODI setting to investigate their robustness and ability to generalize to other CL settings. We assess the methods with CORe50-NI—a dataset designed for ODI—and the proposed NS-MiniImageNet dataset consisting of three nonstationary types: noise, occlusion, and blur (see Table 2.7). The average accuracy at the end of training is summarized in Table 2.11.

Generally speaking, all replay-based methods (ER, MIR, GSS) show comparable performance across three memory sizes and outperform all other methods. GDumb, the strong baseline that dominates the OCI setting in most cases, is no longer as competitive as the replay-based methods and fails completely in CORe50-NI. One of the reasons is that class imbalance, the key cause of forgetting in the OCI setting, does not exist in ODI since the class labels are the same for all tasks. Moreover, in the ODI setting, samples in the data stream change gradually and smoothly with different nonstationary strengths (NS-MiniImageNet) or nonstationary types (CORe50-NI). Learning new samples sequentially with the replay samples(replay-based) may be more effective for the model to adapt to the gradually changing nonstationarity than learning only the samples in the buffer (GDumb). Additionally, the greedy memory update strategy (see Algorithm 5 in Appendix) in GDumb is not suitable for the ODI setting as the buffer will comprise mostly of samples in the latest tasks due to the greedy update, and GDumb will have very limited access to samples in the earlier tasks. Using reservoir sampling as the update strategy may alleviate this shortcoming since reservoir sampling ensures every data point has the same probability to be stored in the memory.

CN-DPM has terrible performance in this setting because it is very sensitive to hyperparameters, and the method cannot find the hyperparameter set that works in this setting within the same tuning budget as other methods.

Regarding methods without a memory buffer, the KD-based LwF underperforms EWC and Finetune, implying KD may not be useful in the ODI setting.

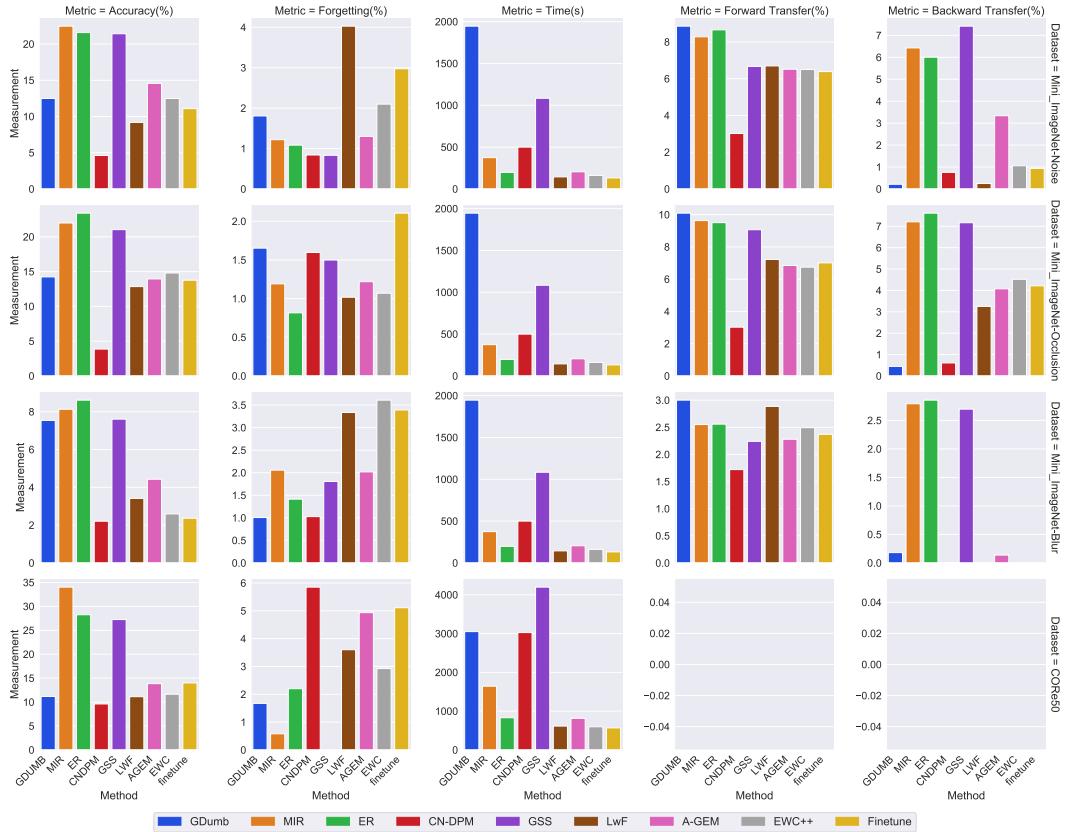


Figure 2.5: Average Accuracy, Forgetting, Running Time, Forward Transfer and Backward Transfer for the ODI setting with a 5k memory buffer. Each column represents a metric and each row represents a dataset. Forward and backward transfer are not applicable in CORe50 since it uses one test set for all tasks.

Another interesting observation is that all methods, including Offline, show unsatisfactory results in the blur scenario. The pivotal cause may be due to the backbone model we use (ResNet18 [53]) since a recent study points out that Gaussian blur can easily degrade the performance of ResNet [137].

In terms of other metrics, as shown in Fig. 2.5, we find that all methods show positive forward transfer, and replay-based methods show much better backward transfer than others. The first reason is that tasks in the ODI setting share more cross-task resemblances than the OCI setting; secondly, the bias towards the current task due to class imbalance does not happen since new tasks contain the same class labels as old tasks. Thus, the model is able to perform zero-shot learning (forward transfer) and improve the preceding tasks (backward transfer).

In summary, replay-based methods exhibit more robust and surpassing performance in the ODI setting. Considering the running time and the performance in the larger scale dataset, MIR stands out as a versatile and competitive method in this setting.

2.8.5 Overall Comments for Methods and Tricks

We summarize the key findings and give comments on each method and trick based on our findings in Table 2.12 and Table 2.13.

2.9 Emerging Directions in Online Continual Learning

In this section, we discuss some emerging directions in online CL that have attracted interest and are expected to gain more attention in the future.

Raw-Data-Free Methods In some applications, storing raw images is not feasible due to privacy and security concerns, and this calls for CL methods that maintain reasonable performance without storing raw data. Regularization [91, 77] is one of the directions but [88] shows that this approach has theoretical limitations in the class incremental setting and cannot be used alone to reach decent performance. We also have empirically confirmed their claims in this work. Generative replay [156, 144] is another direction but it is not viable for more complex datasets as the current deep generative models still cannot generate satisfactory images for such datasets [6, 86].

Feature replay is a promising direction where latent features of the old samples at a given layer (feature extraction layer) are relayed instead of raw data [61, 48, 98, 123]. Since the model changes along the training process, to keep the latent features valid, [123] proposes to slow-down—in the limit case, freeze—the learning of all the layers before the feature extraction layer, while [61] proposes a feature adaptation method to map previous features to their correct values as the model is updated. Another way is to generate latent features with a deep generative model [98].

There are other lately proposed approaches which do not require storing the raw data. For example, SDC [164] leverages embedding networks [24] and the nearest class mean classifier [110]. The approach proposes a method to estimate the drift of features during learning the current task and compensate for the drift in the absence of previous samples. DMC [168] trains a separate model for new tasks and combines the new and old models using publicly available unlabeled data via a double distillation training objective. DSLDA [49] freezes the feature extractor and uses deep Streaming Linear Discriminant Analysis [120] to train the output layer incrementally.

With the increasing data privacy and security concerns, the raw-data-free methods are expected to attract more research endeavour in the coming years.

Meta Learning Meta-learning is an emerging learning paradigm where a neural network evolves from multiple related learning episodes and generalizes the learned knowledge to unseen tasks [56]. Since meta-learning builds up a potential framework to advance CL, a lot of meta-learning based CL methods have been proposed recently, and most of them support the online setting. MER [131] combines experience replay with optimization based meta-learning to maximize transfer and minimize interference based on future gradients. OML [63] is a meta-objective that uses interference as a training signal to learn a representation that accelerates future learning and avoid catastrophic interference. More recently, iTAML [127] proposes to learn a task-agnostic model that automatically predicts the task and quickly adapts to the predicted task with meta-update. La-MAML [46] proposes an efficient gradient-based meta-learning that incorporates per-parameter learning rates for online CL. MERLIN [67] proposes an online CL method based on consolidation in a meta-space, namely, the latent space that generates model weights for solving downstream tasks. In [15], authors propose Continual-MAML, an online extension of MAML [40], that can cope the new CL scenario they propose. We believe meta-learning based online CL methods will continue to be popular with recent advances in meta-learning.

CL in Other Areas Although image classification and reinforcement learning are the main focuses for most CL works, CL has drawn more and more attention in other areas. Object detection has been another emerging topic in CL, and multiple works have been proposed lately to tackle this problem. Most methods leverage KD [55] to alleviate CF, and the main differences between the methods are the base object detector and distillation parts in the network [145, 89, 21, 99]. More recently, a meta-learning based approach is proposed to reshape model gradients for better information share across incremental tasks [68]. A replay-based method is introduced to address streaming object detection by replaying compressed representation in a fixed memory buffer [3].

Beyond computer vision, CL with sequential data and recurrent neural network (RNN) has gained attention over the past few years. Recent works have confirmed that RNNs, including LSTMs, are also immensely affected by CF [147, 140, 9]. In [147], the authors unify GEM [103] and Net2Net [22] to tackle forgetting in RNN. More recently, [36] shows that weight-importance based CL in RNNs are limited and that the hypernetwork-based approaches are more effective in alleviating forgetting. Meanwhile, [35] proposes a learning rule to preserve network dynamics within subspaces for previous tasks and encourage interfering dynamics to explore orthogonal subspaces when learning new tasks. Moreover, multiple works are proposed to address general CL language learning [148, 90] and specific language tasks, such as dialogue systems [112, 108, 94], image captioning [27], sentiment classification [72] and sentence representation learning [96].

Recommender systems have also started to adopt CL [113, 165, 114, 160]. ADER [114] is proposed to handle CF in session-based recommendation using the adaptive distillation loss and replay with heading [155] technique. GraphSAIL [160] is introduced for Graph Neural Networks based recommender systems to preserve a user’s long-term preference during incremental model updates using local structure distillation, global structure distillation and self-embedding distillation.

Several works also address the deployment of CL in practice. [93] introduces on-the-job learning, which requires a deployed model to discover new tasks, collect training data continuously and incre-

mentally learn new tasks without interrupting the application. The author also uses chat-bots and self-driving cars as the examples to highlight the necessity of on-the-job learning. [31] presents a reference architecture for self-maintaining intelligent systems that can adapt to shifting data distributions, cope with outliers, retrain when necessary, and learn new tasks incrementally. [82] discusses the clinical application of CL from three perspectives: diagnosis, prediction and treatment decisions. [80] addresses a practical scenario where a high-capacity server interacts with a large group of resource-limited edge devices and proposes a Dual User-Adaptation framework which disentangles user-adaptation into model personalization on the server and local data regularization on the user device.

2.10 Conclusion

To better understand the relative advantages of recently proposed CL approaches and the settings where they work best, this chapter performed extensive experiments with nine methods and seven tricks in the online class incremental (OCI) and online domain incremental (ODI) settings.

Regarding the performance in the OCI setting (see Table 2.8, Fig. 2.2 and 2.3), we conclude:

- For memory-free methods, LwF is effective in CIFAR100 and Mini-ImageNet, showing similar performance as replay-based methods with a small memory buffer. However, all memory-free methods fail in the larger CORe50-NC.
- When the memory buffer is small, iCaRL shows the best performance (by large margins) in CIFAR100 and Mini-ImageNet, followed by CN-DPM.
- With a larger memory buffer, GDumb—a simple baseline—outperforms methods designed specifically for the CL problem in CIFAR100 and Mini-ImageNet at the expense of much longer training times.
- In the larger and more realistic CORe50-NC dataset, MIR consistently surpasses all the other methods across different memory sizes.
- We experimentally and theoretically confirm that a key cause of CF is the bias towards new classes in the last fully connected layer due to the imbalance between previous data and new data [157, 169, 4].
- None of the methods show any positive forward and backward transfer due to the bias mentioned above.

The conclusions from our experiments for the OCI tricks (see Table 2.9, Fig. 2.4) are as follows:

- When the memory size is small, LB and NCM are the most effective, showing around 64% relative improvement.
- With a larger memory buffer, NCM remains effective, and RV becomes more helpful, showing around 80% relative improvement.
- When equipped with NCM or RV, both ER and MIR can outperform the best performance of the compared methods without tricks.

- The running times of NCM and RV increase with the growth in memory size, but other tricks only add a fixed overhead to the running time.

For the ODI setting (see Table 2.11, Fig. 2.5), we conclude:

- Generally speaking, all replay-based methods (ER, MIR, GSS) show comparable performance across three memory sizes and outperform all other methods.
- GDumb, the strong baseline that dominates the OCI setting in most cases, is no longer effective, possibly due to its memory update strategy.
- Other OCI methods that we compared cannot generalize to the ODI setting.

We provide detailed comments for compared methods and tricks in Table 2.12 and Table 2.13.

In summary, we can conclude that for the OCI setting, iCaRL, which combines replay, knowledge distillation and nearest class mean classifier, should be used when the memory buffer is very small. When a larger memory buffer is available, MIR with the review trick or the nearest class mean classifier is the best option. In terms of the ODI setting, MIR is the strong and versatile method across different datasets and buffer sizes. With best methods and latest tricks, online CL (with a very small mini-batch) is now approaching the performance levels that are much closer to its ultimate goal of matching offline training.

To simplify the experiment setting, most of the CL literature, including the work in this chapter, do not apply common image classification techniques such as transfer learning (e.g., using a pre-trained model), data augmentation and deeper network architectures to boost the performance. However, since these techniques have been the crucial components for deep-learning-based image applications, we have to consider them when evaluating if the current CL methods are ready to face real-world applications.

In the next chapter, we will discuss the practicality of CL methods and introduce a simple but effective approach that won the first CL competition for computer vision at the Conference on Computer Vision and Pattern Recognition, CVPR2020.

| Method | Comments |
|----------------------------------|--|
| Regularization-based | |
| EWC++ | <ul style="list-style-type: none"> Ineffective in both OCI and ODI settings Suffers from gradient explosion |
| LwF | <ul style="list-style-type: none"> Effective on small scale datasets in OCI (achieves similar performance as replay-based methods with small memory) Ineffective in ODI setting |
| Memory-based | |
| ER | <ul style="list-style-type: none"> Efficient training time over other memory-based methods Better than GSS in most cases but worse than MIR, especially with a large memory buffer |
| MIR | <ul style="list-style-type: none"> A versatile and competitive method in both OCI and ODI settings Works better on a large scale dataset and a large memory buffer |
| GSS | <ul style="list-style-type: none"> Inefficient training time Worse than other memory-based methods in most cases |
| iCaRL | <ul style="list-style-type: none"> Best performance (with large margins) with a small memory buffer on small scale datasets |
| A-GEM | <ul style="list-style-type: none"> Ineffective in both OCI and ODI settings Best performance with a large memory buffer on small scale datasets in OCI setting |
| GDumb | <ul style="list-style-type: none"> Ineffective in ODI mostly due to its memory update strategy Inefficient training time due to training from scratch at every inference point |
| Parameter-isolation-based | |
| CN-DPM | <ul style="list-style-type: none"> Effective when memory size is small Sensitive to hyperparameters and when testing on a new dataset, it may not find a working hyperparameter set given the same tuning budget as others Longest training time among compared methods |

Table 2.12: Overall comments for compared methods

| Trick | Comments |
|-------|---|
| LB | <ul style="list-style-type: none"> • Effective when memory buffer is small • Fixed and limited training time overhead |
| KDC | <ul style="list-style-type: none"> • Fails because of over-regularization of knowledge distillation loss |
| KDC* | <ul style="list-style-type: none"> • Provides moderate improvement with fixed and acceptable training time overhead |
| MI | <ul style="list-style-type: none"> • Better improvement with a larger memory buffer • Training time increases with more iterations |
| SS | <ul style="list-style-type: none"> • Similar improvement as KDC* but with less training time overhead |
| NCM | <ul style="list-style-type: none"> • Provides very strong improvement across different memory sizes. • Baselines equipped with it outperform state-of-the-art methods when the memory buffer is large • Inference time increases with the growth of the memory size |
| RV | <ul style="list-style-type: none"> • Presents very competitive improvement, especially with a larger memory buffer • Baselines equipped with it outperform state-of-the-art methods • Training time increases with the growth of the memory size but it is more efficient than NCM |

Table 2.13: Overall comments for compared tricks

Chapter 3

Continual Learning for Real-World Applications

Given the insights from the survey we presented in the previous chapter, we provide a simple but effective approach that combines memory replay, CL tricks summarized in the survey and common image classification techniques, including transfer learning, data augmentation and deeper network architectures to prepare CL for real-world applications. The proposed method shows remarkable performance in different metrics, including accuracy, RAM usage and run time, etc., in various realistic settings and demonstrates the practicality of CL algorithms.

3.1 Introduction

The primary evaluation of CL in the last few years has been centred around accuracy-related metrics. However, this may lead to a biased conclusion without accounting for the scalability of these techniques over an increasing number of tasks and more complex settings [30]. When considering more metrics, including accuracy, total run time, RAM usage, which method works the best remains a open question. Another important question is whether CL methods that have mostly been proved on artificial benchmarks such as MNIST [81] or CIFAR [79] can generalize to more realistic CL datasets and settings (e.g. much longer data sequence).

The *1st Continual Learning in Computer Vision Challenge*, organized within the *CLVision* workshop at CVPR 2020, is one of the first attempts to address these questions. In particular, the main objectives of the competition were (1) invite the research community to scale up continual learning approaches to natural images and possibly on video benchmarks; (2) invite the community to work on solutions that can generalize over multiple continual learning protocols and settings (e.g. with or without a “task” supervised signal) (3) provide the first opportunity for a comprehensive evaluation on a shared hardware platform for a fair comparison.

Our approach to this challenge is based on a replay-based method called Experience Replay(ER) that we discussed in Section 2.5, which has been shown effective and efficient in various CL settings in Section 2.8. For every incoming mini-batch, the traditional ER concatenates the incoming mini-batch with another mini-batch of samples retrieved from the memory buffer. Then, it simply takes an SGD step with the combined mini-batch, followed by an update of the memory. Since we need to perform



Figure 3.1: Example images of the 50 objects in *CORe50*, the main video dataset used in the challenge. Each column denotes one of the 10 categories [101]

the retrieval and update steps for every minibatch, these approaches will not be efficient when we have thousands of mini-batches. Since data arrive one batch(much larger than a mini-batch) at a time in the challenge, we concatenate the memory examples at the batch level instead of at the minibatch level and then perform the model update with the concatenated batch. Additionally, we add a review step before the final testing to remind the model of the knowledge it has learned. Also, we find that it is vital to use standard image classification techniques such as transfer learning (e.g., using a pre-trained model), data augmentation and deeper network architectures to obtain good accuracy performance.

The final solution based on the approach introduced above ranked first in all three competition tracks and won the competition.

3.2 CLVision Continual Learning Competition

The challenge is based on the CORe50 dataset [101] with three different scenarios and five metrics. This section summarizes the framework of the challenge. For more details, we refer the reader to the challenge website [1].

3.2.1 Dataset

CORe50 [101] was specifically designed as an object recognition video benchmark for continual learning. It consists of 164,866 128×128 images of 50 domestic objects belonging to 10 categories (see Figure 3.1); for each object the dataset includes 11 video sessions (~ 300 frames recorded with a Kinect 2 at 20 fps) characterized by relevant variations in terms of lighting, background, pose and occlusions. Classification on CORe50 can be performed at Object level (50 classes) or at Category level (10 classes). The former, being a more challenging task, was the configuration chosen for this competition. The egocentric vision of hand-held objects allows for the emulation of a scenario where a robot has to incrementally learn to recognize objects while manipulating them. Objects are presented to the robot by a human operator who can also provide the labels, thus enabling a supervised classification.

3.2.2 Scenarios

Based on the CORe50 dataset, the challenge included three different scenarios based on the different settings considered:

1. New Instances (NI): In this setting, 8 training batches of the same 50 classes are encountered over time. Each training batch is composed of different images collected in different environmental conditions. The differences of background and lighting between two batches vary. Some batches are very contrasting, while some batches are quite similar. We also remark that in the NI scenario, no task label is given during both training and testing time.
2. Multi-Task New Classes (MT-NC)¹: In this setting, the 50 different classes are split into 9 different tasks: 10 classes in the first batch and 5 classes in the other 8. In this case, the task label will be provided during training and test.
3. New Instances and Classes (NIC): this protocol is composed of 391 training batches containing 300 images of a single class. No task label will be provided, and each batch may contain images of a class seen before as well as a completely new class.
4. All together (ALL): All the settings presented above.

3.2.3 Evaluation Metrics

In the last few years, the main evaluation focus in continual learning has always been centred around accuracy-related forgetting metrics. However, as argued by [30], this may lead to a biased conclusion without accounting for the real scalability of such techniques over an increasing number of tasks/batches and more complex settings. For this reason, in the competition, each solution was evaluated across a number of metrics:

1. *Final accuracy on the test set*²: computed only at the end of the training procedure.
2. *Average accuracy overtime on the validation set*: computed at every batch/task.
3. *Total training/test time*: total running time from start to end of the main function (in minutes).
4. *RAM usage*: total memory occupation of the process and its eventual sub-processes. It is computed at every epoch (in MB).
5. *Disk usage*: only of additional data produced during training (like replay patterns) and additionally stored parameters. It is computed at every epoch (in MB).

The final aggregation metric (CL_{score}) is the weighted average of the 1-5 metrics (0.3, 0.1, 0.15, 0.125, 0.125 respectively).

¹Multi-Task-NC constitutes a simplified variation of the originally proposed New Classes (NC) protocol [101] (where the task label is not provided during train and test).

²Accuracy in CORe50 is computed on a fixed test set. The rationale behind this choice is explained in [101]

3.3 Batch-level Experience Replay with Review

3.3.1 General Approach

As we mentioned in Section 3.2.2, the data distributions of two batches are different and therefore, naively updating the neural network with a new batch will result in catastrophic forgetting. To mitigate this problem, a common approach in CL is the memory-based method described in Section 2.5, which uses a memory buffer to store a subset of the data from past batches to tackle forgetting. The data from the memory buffer can be used to either constrain the optimization of parameters such that the loss on past batches can never increase [103, 19] or conduct experience replay [20, 6]. In this competition, we choose the experience replay approach as it is more efficient and computationally cheaper than the constraint optimization approach.

Compared with the simplest baseline model that fine-tunes the parameters based on the new task without any techniques to prevent forgetting, conventional experience replay approaches store a subset of the samples from the past batches in a memory buffer \mathcal{M} of limited size mem_sz . During the training of the current batch, it concatenates the incoming mini-batch with another mini-batch of samples retrieved from the memory buffer. Then, it simply takes an SGD step with the combined mini-batch, followed by an update of the memory [20, 6].

Since we need to perform the retrieval and update steps for every minibatch, these approaches will not be efficient when we have thousands of mini-batches. Since data arrive one batch(much larger than a mini-batch) at a time in the challenge, we concatenate the memory examples at the batch level instead of at the minibatch level and then perform the model update with the concatenated batch. Additionally, we add a review step before the final testing to remind the model of the knowledge it has learned.

The overall training procedure is presented in Algorithm 6. For every batch of data except the first batch, we do a batch level experience replay. Concretely, for every epoch, we draw another batch of data $D_{\mathcal{M}}$ randomly from the episodic memory with size $replay_sz$, concatenate it with the current batch and conduct the gradient descent parameters update. We note that $D_{\mathcal{M}}$ is different for every epoch. When we finish the last epoch of the current batch, we will randomly select $\frac{mem_sz}{n}$ examples from the current batch where n is the total number of batches in the whole scenario. After training all the data batches, we will do a final review step where we draw a batch of size D_R from memory and conduct the gradient update again. To prevent overfitting, this step's learning rate needs to be lower than the learning rate used for processing new batches.

3.3.2 Architecture and Training Details

For all three scenarios, we use the DenseNet-161 model [60] pre-trained on ImageNet [28]. DenseNet-161 is the largest model in the DenseNet group with a size around 100MB. As shown in Figure 3.2, the DenseNet-161 model consists of 4 dense blocks and we freeze all the layers before the third blocks. This ensures the pre-trained model can extract the basic features from the image and shorten the training time as well. The network is trained via cross-entropy loss and stochastic gradient descent with the mini-batch size equal to 32. The detailed hyper-parameters of the optimizer for each scenario are listed in the Appendix in [104].

As we mentioned in 3.2.2, the critical difference between batches are background and lighting, and most of the target objects are in the center of the images. Therefore, we center-crop the image with a

Algorithm 6: Batch-level Experience Replay with Review

Input : mem_sz memory buffer size, $replay_sz$ replay mini-batch size, $review_sz$ review data size
Input : lr_replay replay learning rate, lr_review review learning rate
Require: D data stream, θ model parameters

```

1  $\mathcal{M} \leftarrow \{\} * mem\_sz$                                  $\triangleright$  Allocate memory of size  $mem\_sz$ 
2 for  $t \in \{1, \dots, T\}$  do
3   for epochs do
4     if  $t > 1$  then
5        $D_{\mathcal{M}} \xrightarrow{replay\_sz} \mathcal{M}$                  $\triangleright$  Sample a batch of data with size  $replay\_sz$  from  $\mathcal{M}$ 
6        $D_{train} = D_{\mathcal{M}} \cup D_t$                        $\triangleright$  Concatenate the current data batch and the memory batch
7     else
8        $D_{train} = D_t$ 
9      $\theta \leftarrow SGD(D_{train}, \theta, lr\_replay)$          $\triangleright$  One pass minibatch gradient descent over  $D_{train}$ 
10     $\mathcal{M} \leftarrow UpdateMemory(D_t, \mathcal{M}, mem\_sz)$        $\triangleright$  Update memory
11   $D_R \xrightarrow{review\_sz} \mathcal{M}$                           $\triangleright$  Sample a batch of data with size  $review\_size$  from  $\mathcal{M}$ 
12   $\theta \leftarrow SGD(D_R, \theta, lr\_review, batch\_sz)$        $\triangleright$  One pass minibatch gradient descent over  $D_R$  return  $\theta$ 

```

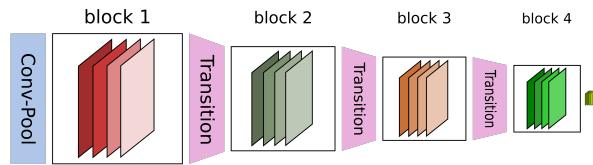


Figure 3.2: DenseNet-161 Architecture [146]

(100, 100) window to mitigate the background's effect to some extent and make the target object occupy more pixels in the image. As the size of the input image is (128, 128, 3) and we do not train any layers before the third dense block, we resize the cropped image to (224, 224, 3) to ensure no size discrepancy between the input of the pre-trained model and the training image. Noted that this preprocessing step is applied to both training and testing images. Figure 3.3 shows an example of the center-cropped image.

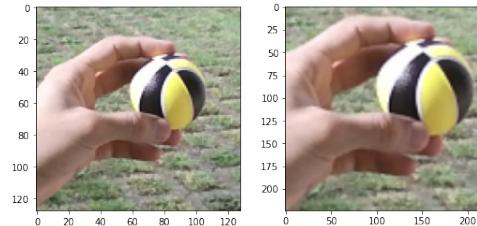


Figure 3.3: Example of pre-processing (e.g., center-crop)

Moreover, to get better generalization, we leverage data augmentation techniques, including pixel-level and spatial-level transformations. Specifically, we use five spatial-level transformations, including HorizontalFlip, RandomRotate90, ElasticTransform, GridDistortion, and OpticalDistortion. For pixel-level transformations, we use RandomContrast, RandomGamma, and RandomBrightness. The details of the data preprocessing steps used in the final submission are shown in Table 3.1.

| Step | Augmentation | Probability |
|--------|---|-------------|
| Step 1 | CenterCrop(100, 100) | $p = 1.0$ |
| Step 2 | One of $\begin{cases} \text{HorizontalFlip}(p = 1) \\ \text{RandomRotate90}(p = 1) \end{cases}$ | $p = 0.5$ |
| Step 3 | One of $\begin{cases} \text{RandomContrast}(0.4) \\ \text{RandomGamma}(20, 180) \\ \text{RandomBrightness}(0.4) \end{cases}$ | $p = 0.5$ |
| Step 4 | One of $\begin{cases} \text{ElasticTransform}(\alpha = 120, \sigma = 6, \alpha_{\text{affine}} = 3.6) \\ \text{GridDistortion}() \\ \text{OpticalDistortion}(\text{distort_limit} = 2, \text{shift_limit} = 0.5) \end{cases}$ | $p = 0.3$ |
| Step 5 | Resize(224, 224) | $p = 1.0$ |
| Step 6 | Normalize $\left(\begin{array}{l} \text{mean} = [0.485, 0.456, 0.406], \\ \text{std} = [0.229, 0.224, 0.225] \end{array} \right)$ | $p = 1.0$ |

Table 3.1: The details of the data preprocessing steps.

3.3.3 Final Solutions

In both NI and NIC scenarios, we use the same approach introduced above, but because the number of batches is different in these two scenarios, we use different mem_sz and replay_sz . The detailed hyperparameters of the models for each scenario are listed in Appendix in [104].

Since the task label is provided in the MT-NC scenario, the task difficulty is much smaller than the other two scenarios. As [131] proposed recently, we can treat the CL problem as solving Transfer-Interference Trade-off, where we want to maximize transfer and minimize interference. In this scenario, we found that interference outweighs transfer when we share the same model across all the batches. Thus we decided to assign a fresh pre-trained model for each batch to prevent interference. Moreover, since we do not need any extra steps to avoid forgetting, we will have a shorter training time as well. Nevertheless, the drawback of this method is that it prevents positive transfer due to a lack of weight sharing.

3.4 Experiments

3.4.1 Multi-Task New Classes (MT-NC)

The Baseline method in MT-NC shares all the layers before the last fully-connected layer between all the batches and each batch has its own fully connected layer. Ind_model represents the independent model approach mentioned in 3.3.3 that a fresh pre-trained model is assigned to each batch of data. Ind_model_prepoc is the Ind_mdoel plus the preprocessing steps mentioned in Section 3.3.2. DenseNet161_tune_all means tuning the model without freezing any of the layers and DenseNet161_freeze freezes all the layers before the third block as mentioned in Section 3.3.2.

As we can see in Table 3.2, the baseline method experiences huge forgetting. By assigning an independent model to each batch and freezing the first half of the model, the performance improves significantly and the processing step helps the model generalize better.

| Method | Architecture | avg_val_acc | final_val_acc |
|-------------------|----------------------|-------------|---------------|
| Baseline | DenseNet161_tune_all | 14.0% | 12.8% |
| Ind_model | DenseNet161_freeze | 54.7% | 98.6% |
| Ind_model_preproc | DenseNet161_freeze | 55.1% | 99.3% |

Table 3.2: Multi-Task-NC Performance Comparison

3.4.2 New Instances (NI)

In the NI scenario, all batches share the same model and the baseline is fine-tuning the model based on new incoming batches without any steps to prevent forgetting. As we can see in Table 3.3, the Batch-level Experience Replay (BER) improves the final validation accuracy by around 7.7%. The final review step increases the final validation accuracy by 1.1% but it does not help the average validation accuracy since the final review is done by the end of the training. The data preprocessing and augmentation yield much better generalization and consequently enhance both metrics.

| Method | Architecture | avg_val_acc | final_val_acc |
|--------------------|----------------------|-------------|---------------|
| Baseline | DenseNet161_tune_all | 71.1% | 81.1% |
| BER | DenseNet161_tune_all | 77.1% | 88.7% |
| BER_review | DenseNet161_tune_all | 77.0% | 89.8% |
| BER_review_preproc | DenseNet161_freeze | 90.1% | 96.7% |

Table 3.3: NI Performance Comparison

3.4.3 New Instances and Classes (NIC)

The baseline of NIC is the same as NI. However, since in NIC, every batch contains only one class, the data distribution difference between two batches is much greater than the NI scenario, which explains the extremely poor result of the baseline. We use the same algorithm, Batch-level Experience Replay with Review, to tackle this scenario as well. As we can see in Table 3.4, the BER_review_preproc method gets the highest values in both metrics, similar to the result of NI,. The average validation accuracy of NIC is much lower than NI. This is because we capture validation accuracy by the end of each batch and at the beginning of the training, the validation accuracy is very low as the model has not seen enough data yet.

| Method | Architecture | avg_val_acc | final_val_acc |
|--------------------|----------------------|-------------|---------------|
| Baseline | DenseNet161_tune_all | 0.02% | 0.02% |
| BER_review | DenseNet161_tune_all | 55.3% | 90.1% |
| BER_review_preproc | DenseNet161_freeze | 59.4% | 96.0% |

Table 3.4: NIC Performance Comparison

3.4.4 Official Competition Result

We participated in the competition tracks for all three scenarios. Our team *UT_LG* won all three tracks as well as the ALL track that averages the results across the three tracks. The results of this competition are reported in Table 3.5. For detailed competition results, we refer the readers to [102].

| TEAM NAME | TEST ACC (%) | VAL ACC _{avg} (%) | TIME (M) | RAM _{avg} (MB) | RAM _{max} (MB) | DISK _{avg} (MB) | DISK _{max} (MB) | CL _{score} |
|------------|--------------|----------------------------|----------|-------------------------|-------------------------|--------------------------|--------------------------|---------------------|
| UT_LG | 0.92 | 0.68 | 68.67 | 10643.25 | 11624.87 | 0 | 0 | 0.694 |
| JODELET | 0.88 | 0.64 | 6.59 | 15758.62 | 18169.32 | 0 | 0 | 0.680 |
| AR1 | 0.80 | 0.58 | 20.46 | 8040.47 | 10092.72 | 0 | 0 | 0.663 |
| Yc14600 | 0.91 | 0.65 | 64.88 | 16425.64 | 19800.48 | 0 | 0 | 0.653 |
| ICT_VIPL | 0.95 | 0.68 | 76.73 | 2459.31 | 2459.68 | 392.187 | 562.5 | 0.617 |
| SOONY | 0.88 | 0.63 | 120.33 | 14533.97 | 15763.60 | 0 | 0 | 0.612 |
| REHEARSAL | 0.75 | 0.52 | 22.87 | 19056.77 | 23174.11 | 0 | 0 | 0.570 |
| JIMIB | 0.91 | 0.74 | 242.12 | 17995.61 | 23765.51 | 0 | 0 | 0.542 |
| NOOBMASTER | 0.76 | 0.53 | 147.59 | 24714.06 | 30266.62 | 0 | 0 | 0.464 |
| NAÏVE | 0.23 | 0.24 | 5.16 | 15763.46 | 18158.02 | 0 | 0 | 0.327 |
| Avg | 0.80 | 0.59 | 77.54 | 14539.12 | 17327.49 | 39.22 | 56.25 | 0.58 |

Table 3.5: ALL track results for the top-10 finalists of the competition. Our team (UT_LG) obtained the best CL_{Score}.

3.5 Conclusion

In this chapter, we provide a simple but effective approach that combines memory replay and commonly used techniques in image classification including, transfer learning, data augmentation and deeper network architectures. With an average testing accuracy of $\sim 92\%$, and a running time of ~ 68 minutes, our team's relatively simple approach suggests CL for practical image classification applications to be feasible in the real-world setting, even with a large number of small non-i.i.d. batches.

However, the memory sample retrieval strategy used in our winning solution is still simple random sampling, and it may not be optimal in some scenarios. In the next chapter, we will discuss an open but crucial question for replay-based methods, *how to update and retrieve memory samples when new data arrives?*

Chapter 4

Adversarial Shapley value Experience Replay

In the last two chapters, we have demonstrated the effectiveness of the replay-based methods in the OCI setting. However, the best method for selecting which buffered images to replay is still an open question. In this chapter, we contribute a novel Adversarial Shapley value scoring method that scores memory data samples according to their ability to preserve latent decision boundaries for previously observed classes (to maintain learning stability and avoid forgetting) while interfering with latent decision boundaries of current classes being learned (to encourage plasticity and optimal learning of new class boundaries). Through extensive experiments on three commonly used benchmarks in the CL literature, we demonstrate that ASER provides competitive or improved performance compared to state-of-the-art replay-based methods, especially when the memory buffer size is small.

This work is a collaboration with Jihwan Jeong, Dongsub Shim and Scott Sanner from the University of Toronto, Hyunwoo Kim and Jongseong Jang from the LG AI Research. It has been published at the AAAI Conference on Artificial Intelligence (AAAI 2021) as "Online Class-Incremental Continual Learning with Adversarial Shapley Value" [105].

4.1 Introduction

In this chapter, we focus on the online class incremental (OCI) setting that we defined and discussed in Section 2.3, where a model needs to learn new classes continually from an online data stream (each sample is seen only once).

As we mentioned in Section 2.5, most CL methods can be taxonomized into three major categories: regularization-based, parameter isolation, and memory-based methods [121, 26]. Regularization-based methods incorporate an additional penalty term into the loss function to penalize the update of critical model parameters [77, 166, 5, 133]. Other regularization-based methods imposed knowledge distillation techniques to penalize the feature drift on previous tasks [91, 157, 128]. Parameter isolation methods assign per-task parameters to bypass interference by expanding the network and masking parameters to prevent forgetting [106, 85, 163]. Memory-based methods deploy a memory buffer to store a subset of data from previous tasks. The samples from the buffer can be either used to constrain the parameter updates such that the loss on previous tasks cannot increase [19, 103], or simply for replay to prevent

forgetting [130, 20].

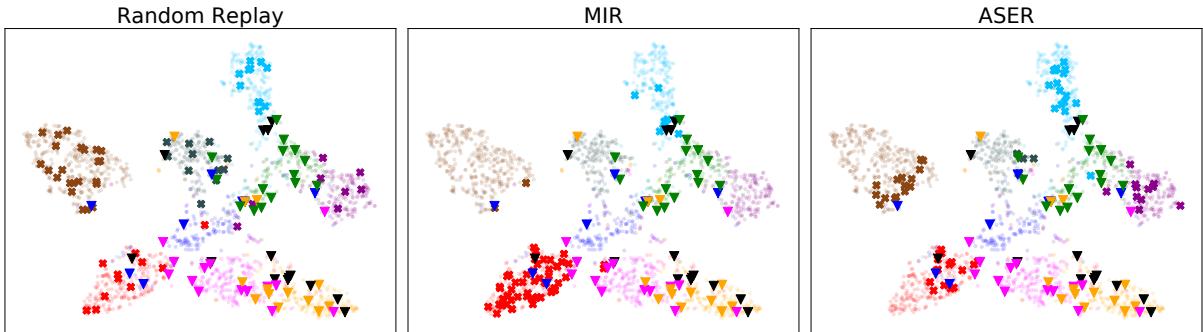


Figure 4.1: 2D t-SNE [152] visualization of CIFAR-100 data embeddings and their class labels (different colors) showing current task samples (triangle), memory samples (pale circle) and retrieved memory samples for rehearsal (bold x). For each point, we obtain the latent embedding from reduced ResNet18 [19]. Note that Random Replay [20] distributes its retrieved samples non-strategically, MIR disproportionately selects seemingly redundant samples in a single – apparently most interfered – class (red), whereas ASER strategically retrieves memory samples that are representative of different classes in memory but also adversarially located near class boundaries and current task samples.

As we can see in Section 2.8, regularization methods only protect the model’s ability to classify within a task and thus they do not work well in the OCI setting where the ability to discriminate among classes from different tasks is crucial [88]. Also, most parameter isolation methods require task identity during inference, which violates our setting. Therefore in this work, we consider the replay approach which has shown to be successful and efficient for the OCI setting [8, 6]. Since the memory buffer is the only place to store data from previous tasks, a key question for replay-based methods is *how to update and retrieve memory samples when new data arrives?* For example, [20] proposed a simple but strong baseline that randomly updates and retrieves samples, while the highly effective Maximally Interfered Retrieval (MIR) method [6] chooses replay samples whose loss most increases after a current task update. However, if we visualize the latent space of retrieved memory samples chosen by each method in Fig. 4.1, we see that the methods mentioned above fail to strategically select samples that both preserve existing memory-based class boundaries while protecting against current task samples that interfere with these boundaries (detailed discussion in caption of Fig. 4.1). Moreover, as we can see in Section 2.8.2 when memory buffer is small, replay-based methods, such as ER and MIR, only show similar performance as the memory-free LwF [91]. This implies more careful analysis of the memory retrieval and update steps is necessary for the small memory setting.

We address the deficiencies observed above by proposing a novel replay-based method called Adversarial Shapley value Experience Replay (ASER). ASER is inspired by the Shapley value (SV) [143] used in cooperative game theory to fairly distribute total gains to all players — in our CL setting, we use the SV to determine the contribution of memory samples to learning performance [42, 64, 65]. We also introduce an adversarial perspective of SV for CL memory retrieval that aims to score memory samples according to their preservation of decision boundaries for “friendly” samples in the memory buffer (to maintain learning stability and avoid forgetting) and their interference with “opponent” samples from the current task that disrupt existing memory-based class boundaries (to encourage plasticity and optimal learning). Through extensive experiments on three commonly used benchmarks in the CL literature, we demonstrate that ASER provides competitive or improved performance compared to state-of-the-art

replay-based methods, especially when the memory buffer size is small.

4.2 Online Class Incremental Learning

4.2.1 Problem Definition

In this work, we focus on the online class incremental (OCI) setting that we defined and discussed in Section 2.3. In the OCI setting, a neural network classifier $f : \mathbb{R}^d \mapsto \mathbb{R}^C$ parameterized by θ needs to learn new classes continually from an online data stream (each sample is seen only once). It will receive input batches B_n^t of size b from task t , which consists of classes that the classifier has never seen before in task $1:t$. Moreover, we adopt the single-head evaluation setup [18] where the classifier has no access to task identity during inference and hence must choose among all labels. Our goal is to train the classifier f to continually learn new classes from the data stream without forgetting. For a more formal definition of the OCI setting, please refer to Section 2.3.

4.2.2 Experience Replay Methods

As we can see in Section 2.8.2, the replay-based methods have shown to be effective and efficient in the OCI settings. Hence, we focus on the replay-based approach in this work.

As we mentioned in Section 2.5, most of the replay-based methods follow the generic algorithm we summarized (see Algorithm 2). What differentiates various replay-based methods are the memory retrieval strategy (line 3) [6, 105], and the memory update strategy (line 5) [25, 75, 8]. For example, Experience Replay (ER) applies reservoir sampling [154] in *MemoryUpdate* and random sampling in *MemoryRetrieval*. Despite its simplicity, recent research has shown that naive ER outperforms many specifically designed CL approaches with and without a memory buffer [20]. Maximally-interfered Retrieval (MIR) [6] aims to improve the *MemoryRetrieval* strategy by choosing replay samples according to loss increases given the estimated parameters update based on the newly arrived data. However, samples with significant loss increases tend to be similar in the latent space, which may lead to redundancy in the retrieved data, as shown in Fig. 4.1. Like ER, MIR uses reservoir sampling for the *MemoryUpdate*. Different from MIR, Gradient-based Sample Selection (GSS) [8] pays attention to the *MemoryUpdate* strategy [8]. Specifically, it tries to diversify the gradients of the samples in the memory buffer. Like ER, GSS uses random sampling in *MemoryRetrieval*.

4.3 Efficient Computation of Shapley Value via KNN Classifier

When we return to Fig. 4.1 and analyze the latent embeddings of memory samples, we observe the natural clustering effect of classes in the embedding space, which has been well-observed previously in the deep learning literature [115, 32]. On account of this, we observe that some samples may indeed be more important than others in terms of preserving what the neural network has learned. For example, data from one class that are near the boundary with data from another class in some sense act as sentinels to guard the decision boundary between classes. This suggests the following question: *how can we value data in the embedded space in terms of their contribution to accurate classification?*

Given that the embedding plot of Fig. 4.1 suggests that a new data point is likely to take the classification of its nearest neighbors in the embedding space, we could rephrase this question as asking

how much each data point in memory contributes to correct classification from the perspective of a K-Nearest Neighbors (KNN) classifier. Fortunately, the existing research literature already provides both a precise and efficient answer to this question viewed through the lens of Shapley data valuation for KNN classifiers [64, 42, 65]. Before we cover this solution, we first pause to recap the purpose of Shapley values.

4.3.1 Shapley Value (SV) for Machine Learning

The SV [143, 135] was originally proposed in cooperative game theory to decide the share of total gains for each player in a coalition. The SV has a set of mathematical properties that make it appealing to many applications: *group rationality*, *fairness*, and *additivity*. Conversely, it can be shown that the SV is the *only* allocation scheme that satisfies these three properties.

In the context of machine learning, the SV has been used to estimate the individual contribution of data points to the performance of a trained model *in the context of all other data* [42, 65]. Formally, let N denote the number of data points and $I = \{1, \dots, N\}$ be the associated index set. Then, each datum is interpreted as a player of a cooperative game with the goal of maximizing test-time performance. Let $v(S)$ define a utility function of the ML model over a subset $S \subset I$ on which the model is trained. Then, the SV of a data point of index i with the utility $v(S)$ is the following:

$$s(i) = \frac{1}{N} \sum_{S \subseteq I \setminus \{i\}} \frac{1}{\binom{N-1}{|S|}} [v(S \cup \{i\}) - v(S)] \quad (4.1)$$

Intuitively, when we consider every possible subset of data points, $s(i)$ measures the average marginal improvement of utility given by the sample i . By setting the utility as test accuracy in ML classification tasks, the SV can discover how much of the test accuracy is attributed to a training instance.

4.3.2 Efficient KNN Shapley Value Computation

Specific to our requirements for data valuation in this paper, recent work has developed an efficient method for SV computation in a KNN classification framework [64]. This is a critical innovation since the direct powerset-based computation of the SV requires $O(2^N)$ evaluations for general, bounded utility functions. Furthermore, each evaluation involves training an ML model with a given subset of data (S). This is prohibitive in most modern deep learning applications, not to mention online CL with neural networks. As shown in [64] and summarized below, the exact KNN-SV can be computed in $O(N \log N)$.

Let $(\mathbf{x}_j^{\text{ev}}, y_j^{\text{ev}})$ denote an *evaluation point* and $D_c = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_c}$ a *candidate set*, where y_j^{ev} and y_i are labels. We compute the KNN-SVs of all examples in D_c w.r.t. the evaluation point with the utility function (4.2). The KNN utility function over a subset $S \subset D_c$ measures the likelihood of correct classifications:

$$v_{j,\text{KNN}}(S) = \frac{1}{K} \sum_{k=1}^{\min(K, |S|)} \mathbb{1}[y_{\alpha_k(S)} = y_j^{\text{ev}}] \quad (4.2)$$

where $\alpha_k(S)$ is the index of the k th closest sample (from \mathbf{x}_j^{ev}) in S based on some distance metric. Each sample i is assigned a KNN-SV — $s_j(i)$ — that represents the average marginal contribution of the instance to the utility. Due to the additivity of SV, we obtain the KNN-SV of a candidate sample w.r.t.

the evaluation set ($D_e = \{(\mathbf{x}_j^{\text{ev}}, y_j^{\text{ev}})\}_{j=1}^{N_e}$) by taking the average: $s_{\text{avg}}(i) = 1/N_e \sum_{j=1}^{N_e} s_j(i)$.

(4.3) and (4.4) show how to recursively compute the exact KNN-SVs of samples in D_c w.r.t. $(\mathbf{x}_j^{\text{ev}}, y_j^{\text{ev}}) \in D_e$ starting from $\mathbf{x}_{\alpha_{N_c}}$ (the farthest point from \mathbf{x}_j^{ev}) [64]:

$$s_j(\alpha_{N_c}) = \frac{\mathbb{1}[y_{\alpha_{N_c}} = y_j^{\text{ev}}]}{N_c} \quad (4.3)$$

$$s_j(\alpha_m) = s_j(\alpha_{m+1}) + \frac{\mathbb{1}[y_{\alpha_m} = y_j^{\text{ev}}] - \mathbb{1}[y_{\alpha_{m+1}} = y_j^{\text{ev}}]}{K} \frac{\min(K, m)}{m} \quad (4.4)$$

Here, $s_j(\alpha_m)$ is the KNN-SV of the m th closest candidate sample from \mathbf{x}_j^{ev} . Note that the dependency on the utility v is suppressed as v_{KNN} is always used. We refer readers to [64] for detailed derivation of these results.

4.4 Adversarial Shapley Value Experience Replay (ASER)

We have now affirmatively answered how to value data in the embedded space in terms of its contribution to accurate classification by leveraging the efficient KNN-SV computation. Equipped with this powerful global data valuation algorithm, we now present our novel ER method dubbed Adversarial Shapley value ER (ASER) that leverages the SV for both *MemoryRetrieval* and *MemoryUpdate*.

4.4.1 ASER Memory Retrieval

A key insight with our ASER approach for *MemoryRetrieval* is that we need to balance the competing needs at the crux of CL, i.e., we need to retrieve memory samples for replay that prevent forgetting while also finding samples that maximally interfere with the incoming batch B_n to ensure plasticity in learning. This leads us not only to leverage a cooperative notion of the SV (where higher SV is better) as it relates to \mathcal{M} but also an adversarial notion of the SV as it relates to B_n (where lower – and, in fact, negative – SVs indicate interference). In addition ASER also adopts a cooperative SV approach to the *MemoryUpdate* process.

Formally, we can view a neural network classifier (f) as two separate parts: a feature extractor ($f_{\text{ext}} : \mathbb{R}^d \mapsto \mathbb{R}^h$) and a fully connected neural classifier ($f_{\text{cl}} : \mathbb{R}^h \mapsto \mathbb{R}^C$), where h is the dimensionality of the latent space \mathcal{X}^l . We implicitly define a KNN classifier and use the Euclidean distance in \mathcal{X}^l . Then, by (4.3)-(4.4), we can compute the KNN-SVs of candidate samples w.r.t. evaluation samples.

As previously noted, ER’s performance depends on deciding what to store in memory (i.e., *MemoryUpdate*) and what to replay from memory (i.e., *MemoryRetrieval*).

One key desiderata is that we want samples in \mathcal{M} as well as B_n to be well-separated by f_{ext} in the latent space. To this end, we target two types of samples in \mathcal{M} for retrieval: those near the samples in B_n but have different labels (Type 1); those that are representative of samples in the memory (Type 2). Training with samples in Type 1 encourages the model to learn to differentiate current classes from previously seen classes. Samples in Type 2 help retain latent decision boundaries for previously observed classes.

We ground our intuition as to how samples interfere and cluster with each other in the latent space based on two properties of the KNN-SV. Given a candidate sample $i \in D_c$ and an evaluation set D_e , the

KNN-SV of the point i w.r.t. an evaluation point $j \in D_e$, i.e. $s_j(i)$, satisfies the following (see Appendix of [105]¹ for proof):

- **Property 1.** $s_j(i) > 0$ if and only if $y_i = y_j^{\text{ev}}$. Also, $s_j(i) = 0$ only when $S = \{i' | y_{i'} = y_j^{\text{ev}}, \forall i' \in \{i+1, \dots, N_c\}\} = \emptyset$.
- **Property 2.** $|s_j(m)|$ is a non-increasing function of m for m such that $y_m = y_j^{\text{ev}}$. Similarly, $|s_j(n)|$ is a non-increasing function of n for n such that $y_n \neq y_j^{\text{ev}}$. And for $m \geq K$, $|s_j(m)| - |s_j(m')| > 0$ holds for $m < m'$, where m' is the smallest index with $\mathbb{1}(y_m = y_j^{\text{ev}}) = \mathbb{1}(y_{m'} = y_j^{\text{ev}})$, if there exists $l \in (m, m')$ such that $\mathbb{1}(y_l = y_j^{\text{ev}}) \neq \mathbb{1}(y_m = y_j^{\text{ev}})$. In other words, as i gets closer to the evaluation point j , $|s_j(i)|$ cannot decrease for points with the same $\mathbb{1}(y_i = y_j^{\text{ev}})$, and for $i \geq K$, it can only increase when there exist more than one differently labeled points.

The first property states that a candidate sample i has a *positive KNN-SV* if it has the same label as the evaluation point being considered (*cooperative*); the sample will have a *negative KNN-SV* if its label is different than the evaluation point (*adversarial*). By combining both properties, we note:

If $s_j(i)$ is large, the candidate i is close to the evaluation point j in the latent space (\mathcal{X}^l) and has the same label ($y_i = y_j^{\text{ev}}$). On the other hand, if $s_j(i)$ is a negative value of large magnitude, then i is close to j , yet has a different label ($y_i \neq y_j^{\text{ev}}$). Thus, we conjecture that a good data candidate i has *high positive SV* for memory \mathcal{M} and *negative SV with large magnitude* for the current input task B_n .

When we consider the whole evaluation set, we take the mean $s_{D_e}(i) = 1/|D_e| \cdot \sum_{j \in D_e} s_j(i)$, and the above analysis still holds in average. Therefore, by examining the KNN-SVs of candidate samples, we can get a sense of how they are distributed with respect to the evaluation set in \mathcal{X}^l . Then, we define the **adversarial SV (ASV)** that encodes the Type 1 & 2 criteria

$$\mathbf{ASV}(i) = \max_{j \in S_{\text{sub}}} s_j(i) - \min_{k \in B_n} s_k(i), \quad (4.5)$$

as well as a “softer” mean variation \mathbf{ASV}_μ

$$\mathbf{ASV}_\mu(i) = \frac{1}{|S_{\text{sub}}|} \sum_{j \in S_{\text{sub}}} s_j(i) - \frac{1}{b} \sum_{k \in B_n} s_k(i), \quad (4.6)$$

where $i \in \mathcal{M} \setminus S_{\text{sub}}$ and S_{sub} is constructed by subsampling some number of examples from \mathcal{M} such that it is balanced in terms of the number of examples from each class. This prevents us from omitting any latent decision boundaries of classes in the memory. Note that S_{sub} is used as the evaluation set in the first term, whereas the input batch B_n forms the evaluation set in the latter term. The candidate set is $\bar{\mathcal{M}} = \mathcal{M} \setminus S_{\text{sub}}$, and we retrieve samples of size $b_{\mathcal{M}}$ from the set that have the highest ASVs (Algorithm 7). We denote our ER method using the score \mathbf{ASV} (4.5) as ASER, while ASER_μ uses \mathbf{ASV}_μ (4.6) instead. For computational efficiency, we randomly subsample N_c candidates from $\bar{\mathcal{M}}$.

Note that both ASER methods do not greedily retrieve samples with the smallest distances to either S_{sub} or B_n . This is because for a single evaluation point j , $s_j(\alpha_m) = s_j(\alpha_{m+1})$ when $y_{\alpha_m} = y_{\alpha_{m+1}}$. So, a few points can have the same score even if some of them are farther from the evaluation point. This is

¹ Please find the appendix in our extended version of [105] on arXiv. Link: <https://arxiv.org/abs/2009.00093>

Algorithm 7: ASER *MemoryRetrieval*

```

Input : Memory batch size  $b_{\mathcal{M}}$ 
        Input batch  $B_n$ ; Candidate size  $N_c$ ;
        Subsample size  $N_{\text{sub}}$ ;
        Feature extractor  $f_{\text{ext}}$ 

1  $S_{\text{sub}} \xrightarrow{N_{\text{sub}}} \mathcal{M}$                                 // get evaluation set
2  $D_c \xrightarrow{N_c} \mathcal{M} \setminus S_{\text{sub}}$           // get candidate set
   /* get latent embeddings */
3  $L_{B_n}, L_{S_{\text{sub}}}, L_{D_c} \leftarrow f_{\text{ext}}(B_n), f_{\text{ext}}(S_{\text{sub}}), f_{\text{ext}}(D_c)$ 
4 for  $i \in D_c$  do
5   for  $j \in S_{\text{sub}}$  do
6      $s_j(i) \leftarrow \text{KNN-SV}(L_{D_c}, L_{S_{\text{sub}}})$  as per (4.3), (4.4)
7   for  $k \in B_n$  do
8      $s_k(i) \leftarrow \text{KNN-SV}(L_{D_c}, L_{B_n})$  as per (4.3), (4.4)
9    $score(i) \leftarrow \text{ASV}(i)$  as per (4.5) or (4.6)
10  $B_{\mathcal{M}} \leftarrow b_{\mathcal{M}}$  samples with largest  $score(\cdot)$ 
11 return  $B_{\mathcal{M}}$ 

```

in contrast to a pure distance-based score where the closest point gets the highest score. In Appendix¹ B of [105], we show that our method outperforms pure distance-based methods, proving the effectiveness of the global way in which the SV scores candidate data based on the KNN perspective.

We summarize our method in Algorithm 7, and compare it with other state-of-the-art ER methods on multiple challenging CL benchmarks in Section 4.5.

4.4.2 KNN Shapley Value Memory Update

For *MemoryUpdate*, we find that samples with high KNN-SV promote clustering effect in the latent space. Therefore, they are useful to store in the memory, which aligns with the original meaning of the SV. More concretely, we subsample $S_{\text{sub}} \sim \mathcal{M}$ and compute $\frac{1}{|S_{\text{sub}}|} \sum_{j \in S_{\text{sub}}} s_j(i)$ for $i \in \bar{\mathcal{M}} \cup B_n$. Then, we replace samples in $\bar{\mathcal{M}}$ having smaller average KNN-SVs than samples in B_n with the input batch samples.

We use KNN-SV *MemoryUpdate* for ASER throughout experiments in Section 4.5, while the ablation analysis of different variations with random *MemoryUpdate* or random *MemoryRetrieval* (both random retrieval and update reduces to ER) is presented in Appendix¹ C of [105]. Note that ASER with KNN-SV *MemoryUpdate* performs competitively or better than the variations, underscoring the importance of SV-based methods for both *MemoryUpdate* and *MemoryRetrieval*.

4.5 Experiments

To test the efficacy of ASER and its variant ASER $_{\mu}$, we evaluate their performance by comparing them with several state-of-the-art CL baselines. We begin by reviewing the benchmark datasets, baselines we compared against and our experiment setting. We then report and analyze the result to validate our approach.

4.5.1 Experiment Settings

Datasets

Split CIFAR-10 splits the CIFAR-10 dataset [79] into 5 different tasks with non-overlapping classes and 2 classes in each task, similarly as in [6].

Split CIFAR-100 is constructed by splitting the CIFAR-100 dataset [79] into 10 disjoint tasks, and each task has 10 classes.

Split miniImagenet consists of splitting the miniImageNet dataset [153] into 10 disjoint tasks, where each task contains 10 classes

The detail of datasets, including the general information of each dataset, class composition and the number of samples in training, validation and test sets of each task is presented in Appendix¹ D of [105].

Baselines

We compare our proposed ASER against several state-of-the-art continual learning algorithms:

- **AGEM** [19]: Averaged Gradient Episodic Memory, a memory-based method that utilizes the samples in the memory buffer to constrain the parameter updates.
- **ASER & ASER_μ**: Our proposed methods. ASER scores samples in the memory with **ASV** in (4.5). ASER_μ uses the mean variation **ASV_μ** in (4.6).
- **ER** [20]: Experience replay, a recent and successful rehearsal method with random sampling in *MemoryRetrieval* and reservoir sampling in *MemoryUpdate*.
- **EWC** [77]: Elastic Weight Consolidation, a prior-focused method that limits the update of parameters that were important to the past tasks, as measured by the Fisher information matrix.
- **GSS** [8]: Gradient-Based Sample Selection, a *MemoryUpdate* method that diversifies the gradients of the samples in the replay memory.
- **MIR** [6]: Maximally Interfered Retrieval, a *MemoryRetrieval* method that retrieves memory samples that suffer from an increase in loss given the estimated parameters update based on the current task.
- **iid-online & iid-offline**: iid-online trains the model with a single-pass through the same set of data, but each mini-batch is sampled iid from the training set. iid-offline trains the model over multiple epochs on the dataset with iid sampled mini-batch. We use 5 epochs for iid-offline in all the experiments as in [6, 8].
- **fine-tune**: As an important baseline in previous work [6, 8, 85], it simply trains the model in the order the data is presented without any specific method for forgetting avoidance.

Other settings

Single-head Evaluation Most of the previous work in CL applied multi-head evaluation [18] where a distinct output head is assigned for each task and the model utilizes the task identity to choose the corresponding output head during test time. But in many realistic scenarios, task identity is not available during test time, so the model should be able to classify labels from different tasks. As in [6, 8], we adopt

the single-head evaluation setup where the model has one output head for all tasks and is required to classify all labels. Note that the setting we use – online and single-head evaluation – is more challenging than many other reported CL settings.

Model We use a reduced ResNet18, similar to [20, 103], as the base model for all datasets, and the network is trained via cross-entropy loss with SGD optimizer and mini-batch size of 10. The size of the mini-batch retrieved from memory is also set to 10 irrespective of the size of the memory. More details of the experiment can be found in Appendix¹E of [105].

4.5.2 Comparative Performance Evaluation

| Method | M=1k | M=2k | M=5k | M=1k | M=2k | M=5k | M=0.2k | M=0.5k | M=1k |
|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|------------|------------|------------|------------|
| iid online | 14.7 ± 0.6 | 14.7 ± 0.6 | 14.7 ± 0.6 | 20.5 ± 0.4 | 20.5 ± 0.4 | 20.5 ± 0.4 | 62.9 ± 1.5 | 62.9 ± 1.5 | 62.9 ± 1.5 |
| iid offline | 42.4 ± 0.4 | 42.4 ± 0.4 | 42.4 ± 0.4 | 47.4 ± 0.3 | 47.4 ± 0.3 | 47.4 ± 0.3 | 79.7 ± 0.4 | 79.7 ± 0.4 | 79.7 ± 0.4 |
| AGEM | 7.0 ± 0.4 | 7.1 ± 0.5 | 6.9 ± 0.7 | 9.5 ± 0.4 | 9.3 ± 0.4 | 9.7 ± 0.3 | 22.7 ± 1.8 | 22.7 ± 1.9 | 22.6 ± 0.7 |
| ER | 8.7 ± 0.4 | 11.8 ± 0.9 | 16.5 ± 0.9 | 11.2 ± 0.4 | 14.6 ± 0.4 | 20.1 ± 0.8 | 26.4 ± 1.0 | 32.2 ± 1.4 | 38.4 ± 1.7 |
| EWC | 3.1 ± 0.3 | 3.1 ± 0.3 | 3.1 ± 0.3 | 4.8 ± 0.2 | 4.8 ± 0.2 | 4.8 ± 0.2 | 17.9 ± 0.3 | 17.9 ± 0.3 | 17.9 ± 0.3 |
| fine-tune | 4.3 ± 0.2 | 4.3 ± 0.2 | 4.3 ± 0.2 | 5.9 ± 0.2 | 5.9 ± 0.2 | 5.9 ± 0.2 | 17.9 ± 0.4 | 17.9 ± 0.4 | 17.9 ± 0.4 |
| GSS | 7.5 ± 0.5 | 10.7 ± 0.8 | 12.5 ± 0.4 | 9.3 ± 0.2 | 10.9 ± 0.3 | 15.9 ± 0.4 | 26.9 ± 1.2 | 30.7 ± 1.2 | 40.1 ± 1.4 |
| MIR | 8.1 ± 0.3 | 11.2 ± 0.7 | 15.9 ± 1.6 | 11.2 ± 0.3 | 14.1 ± 0.2 | 21.2 ± 0.6 | 28.3 ± 1.6 | 35.6 ± 1.2 | 42.4 ± 1.5 |
| ASER | 11.7 ± 0.7 | 14.4 ± 0.4 | 18.2 ± 0.7 | 12.3 ± 0.4 | 14.7 ± 0.7 | 20.0 ± 0.6 | 27.8 ± 1.0 | 36.2 ± 1.1 | 43.1 ± 1.2 |
| ASER _μ | 12.2 ± 0.8 | 14.8 ± 1.1 | 18.2 ± 1.1 | 14.0 ± 0.4 | 17.2 ± 0.5 | 21.7 ± 0.5 | 26.4 ± 1.5 | 36.3 ± 1.2 | 43.5 ± 1.4 |

(a) Mini-ImageNet

(b) CIFAR-100

(c) CIFAR-10

Table 4.1: Average Accuracy (higher is better), M is the memory buffer size. All numbers are the average of 15 runs. ASER_μ has better performance when M is small and dataset is more complex.²

| Method | M=1k | M=2k | M=5k | M=1k | M=2k | M=5k | M=0.2k | M=0.5k | M=1k |
|-------------------|------------|-------------------|-------------------|------------|-------------------|------------|------------|-------------------|------------|
| AGEM | 29.3 ± 0.9 | 30.0 ± 0.9 | 29.9 ± 0.8 | 40.4 ± 0.7 | 39.7 ± 0.8 | 39.8 ± 1.0 | 36.1 ± 3.8 | 43.2 ± 4.2 | 48.1 ± 3.0 |
| ER | 29.7 ± 1.3 | 29.2 ± 0.9 | 26.6 ± 1.1 | 45.0 ± 0.5 | 40.5 ± 0.8 | 34.5 ± 0.8 | 72.8 ± 1.7 | 63.1 ± 2.4 | 55.8 ± 2.6 |
| EWC | 28.1 ± 0.8 | 28.1 ± 0.8 | 28.1 ± 0.8 | 39.1 ± 1.2 | 39.1 ± 1.2 | 39.1 ± 1.2 | 81.5 ± 1.4 | 81.5 ± 1.4 | 81.5 ± 1.4 |
| fine-tune | 35.6 ± 0.9 | 35.6 ± 0.9 | 35.6 ± 0.9 | 50.4 ± 1.0 | 50.4 ± 1.0 | 50.4 ± 1.0 | 81.7 ± 0.7 | 81.7 ± 0.7 | 81.7 ± 0.7 |
| GSS | 29.6 ± 1.2 | 27.4 ± 1.1 | 29.9 ± 1.2 | 46.9 ± 0.7 | 42.3 ± 0.8 | 39.2 ± 0.9 | 75.5 ± 1.5 | 65.9 ± 1.6 | 54.9 ± 2.0 |
| MIR | 29.7 ± 1.0 | 27.2 ± 1.1 | 26.2 ± 1.4 | 45.5 ± 0.8 | 40.4 ± 0.6 | 31.4 ± 0.6 | 67.0 ± 2.6 | 68.9 ± 1.7 | 47.7 ± 2.9 |
| ASER | 30.1 ± 1.3 | 24.7 ± 1.0 | 20.9 ± 1.2 | 50.1 ± 0.6 | 45.9 ± 0.9 | 36.7 ± 0.8 | 71.1 ± 1.8 | 59.1 ± 1.5 | 50.4 ± 1.5 |
| ASER _μ | 28.0 ± 1.3 | 22.2 ± 1.6 | 17.2 ± 1.4 | 45.0 ± 0.7 | 38.6 ± 0.6 | 30.3 ± 0.5 | 72.4 ± 1.9 | 58.8 ± 1.4 | 47.9 ± 1.6 |

(a) Mini-ImageNet

(b) CIFAR-100

(c) CIFAR-10

Table 4.2: Average Forgetting (lower is better). Memory buffer size is M. All numbers are the average of 15 runs.

Table 4.1 and Table 4.2 show the average accuracy and average forgetting by the end of the data stream for Mini-ImageNet, CIFAR-100 and CIFAR-10. Based on the performance of iid-online and iid-offline, we verify that Mini-ImageNet and CIFAR-100 are more complex than CIFAR-10, even though three datasets have the same number of samples. Overall, ASER and ASER_μ show competitive or

²The discrepancy of CIFAR-10 result for MIR between the original paper and this work is discussed in Appendix¹ F of [105]

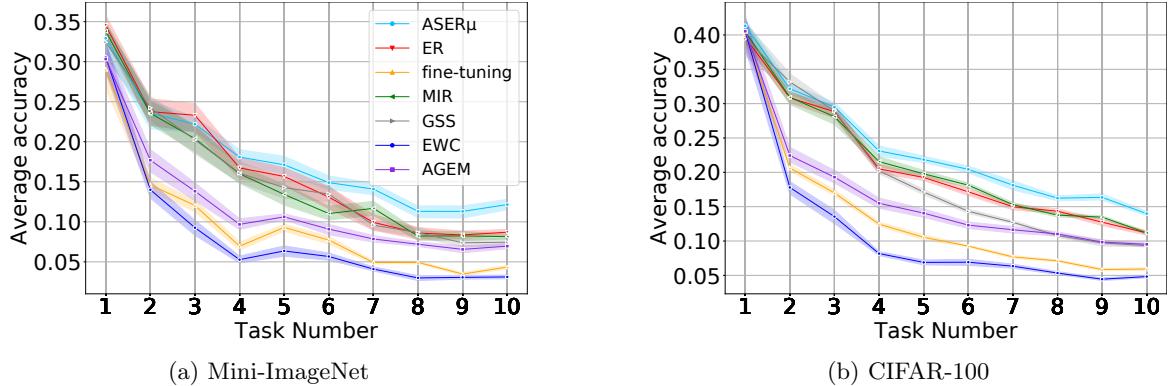


Figure 4.2: Average accuracy on observed tasks when $M=1k$. The shaded region represents the 95% confidence interval. ASER $_{\mu}$ outperforms other baselines especially when the model sees more classes (each task contains new classes).

improved performance in three standard CL datasets. Especially, we observe that ASER $_{\mu}$ outperforms all the state-of-the-art baselines by significant margins in a more difficult setting where memory size is small and dataset is complex. Since the difficulty of the three datasets is different, comparing the absolute accuracy improvement may not be fair. Therefore, percentage improvement³ is more appropriate here. Taking Mini-ImageNet as an example, ASER $_{\mu}$ improves the strongest baseline by 40.2% ($M=1k$), 25.4% ($M=2k$) and 10.3% ($M=5k$) in terms of percentage improvement. Moreover, as we can see in Fig. 4.2, ASER $_{\mu}$ is consistently better than other baselines in both datasets. We also note that ASER $_{\mu}$ generally performs better than ASER. This is because if we use the ASV criterion as in (4.5), it has a higher chance that the value is affected by an outlier point in the *evaluation* set. So the **ASV** $_{\mu}$ in (4.6) gives a more stable and accurate value in complicated datasets.

Another interesting observation is that ER has very competitive performances. Especially in more complex datasets, it surpasses GSS and performs similarly as MIR, which proves it to be a simple but powerful CL baseline. In addition, we find that for complex datasets, when memory size is larger than 5000 (10% of the training data), most of the replay-based methods (except for GSS) outperform the iid-online, a baseline that trains the model with a one-pass through the data but with iid-sampled mini-batch from the whole dataset. This means that storing a small number of training samples is crucial for combating forgetting as well as the learning of the current task in the OCI setting.

We also verify claims from previous work [88, 37, 6]. EWC, a regularization-based method, not only is surpassed by all memory-based methods but also underperforms the fine-tuning baseline. Additionally, AGEM, a method that uses memory samples to constrain parameter updates, delivers worse performance compared with reply-based methods (ER, MIR, and GSS), especially when memory size increases.

Overall, by evaluating on three standard CL datasets and comparing to the state-of-the-art CL methods, we have shown the effectiveness of ASER and its variant ASER _{μ} in overcoming catastrophic forgetting, especially in more complex datasets and memory size is relatively small.

³Percentage improvement is the ratio between absolute improvement and baseline performance. For example, in Mini-ImageNet(M=1k), ASER_μ improves MIR by $\frac{12.2 - 8.7}{8.7} = 40.2\%$

4.6 Conclusion

In this chapter, we proposed a novel ASER method that scores memory data samples according to their ability to preserve latent decision boundaries for previously observed classes while interfering with latent decision boundaries of current classes being learned. Overall, in the OCI setting, we observed that ASER and its ASER_μ variant provide competitive or improved performance on a variety of datasets compared to state-of-the-art ER-based continual learning methods. We also remark that this work paves the way for a number of interesting research directions building on this work.

Although our SV-based method has greatly improved the memory retrieval and update strategies, we may be able to do better than simply concatenating retrieved samples with the incoming batch. Hence, future work could focus on more sophisticated methods to utilize the retrieved samples. It would also be interesting to investigate alternate CL-specific utility function variations for SV.

Chapter 5

Conclusion

In this thesis, we investigated three important questions in online CL:

- Over the past few years, a large range of methods and tricks have been introduced to address the online CL problem, but there is limited consensus in the literature on experimental setups and datasets. Although most papers show that their methods surpass others in one specific setting, the open question is: *what are the relative advantages of these approaches and the settings where they work best?*
- The primary evaluation of CL in the last few years has been centred around accuracy-related metrics. However, this may lead to a biased conclusion without accounting for the scalability of these techniques over an increasing number of tasks and more complex settings [30]. *When considering more metrics, including accuracy, total run time, RAM usage, and more realistic CL datasets and settings, which method works the best?*
- Methods with a memory buffer have shown to be successful and efficient for the online class incremental setting [8, 6]. Since the memory buffer is the only place to store data from previous tasks, a vital but open question for these methods is *how to retrieve memory samples and update the memory buffer when new data arrives?*

In the next sections, we summarize the main contributions of this thesis and discuss a few possible future research directions.

5.1 Summary of Contributions

This thesis aims to study the online CL in image classification that requires a neural network to learn continually from an online stream of non-i.i.d data and accumulate the acquired knowledge without forgetting. Specifically, we try to address the three crucial problems we mentioned in the previous section, and the contributions of this thesis can be summarized as follows:

- (1) In Chapter 2, we provided a practical comparative survey of the state-of-the-art methods and recently proposed tricks. In summary, we concluded that for the OCI setting, iCaRL, which combines replay, knowledge distillation and nearest class mean classifier, should be used when the

memory buffer is very small. When a larger memory buffer is available, MIR with the review trick or the nearest class mean classifier is the best option. In terms of the ODI setting, MIR is the strong and versatile method across different datasets and buffer sizes. With best methods and latest tricks, online CL (with a very small mini-batch) is now approaching the performance levels that are much closer to its ultimate goal of matching offline training.

- (2) To better prepare the CL methods for real-world applications in image classification, in Chapter 3, we introduce a simple but effective approach that combines memory replay and commonly used techniques in image classification including, transfer learning, data augmentation and deeper network architectures. The proposed method shows remarkable performance in different metrics, including accuracy, RAM usage and run time, etc., in various realistic settings and demonstrates the practicality of CL algorithms. Our solution based on this method won the CLVISION Continual Learning challenge at Computer Vision and Pattern Recognition conference, CVPR 2020.
- (3) In Chapter 4, we propose a novel memory-based method called Adversarial Shapley value Experience Replay (ASER) that leverages Shapley value (SV) to determine the contribution of memory samples to learning performance. We introduce an adversarial perspective of SV for CL memory retrieval that scores memory data samples according to their ability to preserve latent decision boundaries for previously observed classes (to maintain learning stability and avoid forgetting) while interfering with latent decision boundaries of current classes being learned (to encourage plasticity and optimal learning of new class boundaries). Through extensive experiments on three commonly used benchmarks in the CL literature, we demonstrate that ASER provides competitive or improved performance compared to state-of-the-art memory-based methods in the OCI setting, especially when the memory buffer size is small.

5.2 Future Directions

In this section, we highlight some of the relevant directions for future research.

- **More effective way to utilize retrieved samples:** In Chapter 4, although our proposed ASER has greatly improved the memory retrieval and update strategies, we may be able to do better than simply concatenating retrieved samples with the incoming batch and perform SGD update. Hence, future work could focus on more sophisticated methods to utilize the retrieved samples. A possible direction is to leverage meta-learning techniques, such as Reptile [118], but meta-learning techniques often have a longer running time than the simple reply.
- **Alternate CL-specific utility function variations for SV:** In ASER, the utility function is set as the test accuracy, and the SV can measure how much of the test accuracy is attributed to a training instance. It would also be interesting to investigate alternate CL-specific utility function variations for SV. Moreover, in the context of regression tasks, accuracy is not a good utility anymore. How to choose the utility function for regression tasks and if ASER can generalize to regression tasks are open questions for future work.
- **More efficient memory storage:** Most memory-based methods, including ASER, store the raw images in the memory buffer. By storing more compact representations of images [123], a fixed-size memory buffer can save more information. It would be interesting to investigate if ASER can work

well with the recently proposed online continual compression technique [14, 48] and boost the CL performance by storing more information in the memory buffer.

- **Leverage data imbalance techniques:** As shown in Section 2.7.2, given the limited size of the memory buffer, the imbalance between previous and new data is a crucial cause of catastrophic forgetting. Since data imbalance has been widely studied in the computer vision area, future work can consider leveraging techniques [33, 70, 59] for data imbalance to combat forgetting.
- **Supervised contrastive continual learning:** As shown in Section 2.8.3, Nearest Class Mean (NCM) classifier is a very competitive substitute for the Softmax classifier. However, the NCM classifier requires well-separated class embeddings, namely embeddings from the same class should be closer than embeddings from different classes. The prevalent cross-entropy loss may not be effective for obtaining the well-separated class embeddings. At the same time, the recently proposed supervised contrastive loss (SupCon) [74] is designed precisely for the purpose mentioned above. A promising direction is to train the representation using the SubCon loss and classify with the NCM classifier.

In conclusion, this thesis has made three meaningful contributions for online CL in image classification, including comprehensively reviewing and evaluating recent approaches, addressing the challenges for bringing online CL into more realistic applications, and proposing a novel memory buffer management method that shows the state-of-the-art performance when memory buffer size is small. We hope that the presented work encourages further investigation of online CL and further moves this emerging field along the path of practical impact.

Bibliography

- [1] CVPR 2020 CLVision challenge on "Continual Learning for Computer Vision, 14th June 2020. <https://sites.google.com/view/clvision2020/challenge>.
- [2] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940, 2020.
- [3] Manoj Acharya, Tyler L Hayes, and Christopher Kanan. Rodeo: Replay for online object detection. *arXiv preprint arXiv:2008.06439*, 2020.
- [4] Hongjoon Ahn and Taesup Moon. A simple class decision balancing for incremental learning, 2020.
- [5] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.
- [6] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems 32*, pages 11849–11860. 2019.
- [7] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375, 2017.
- [8] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems 32*, pages 11816–11825. 2019.
- [9] Gaurav Arora, Afshin Rahimi, and Timothy Baldwin. Does an lstm forget more than a cnn? an empirical study of catastrophic forgetting in nlp. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 77–86, 2019.
- [10] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 583–592, 2019.
- [11] Eden Belouadah and Adrian Popescu. Scail: Classifier weights scaling for class incremental learning. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1266–1275, 2020.

- [12] Zalán Borsos, Mojmír Mutný, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *arXiv preprint arXiv:2006.03875*, 2020.
- [13] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *arXiv preprint arXiv:2004.07211*, 2020.
- [14] Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. Online learned continual compression with adaptive quantization modules. *arXiv*, pages arXiv–1911, 2019.
- [15] Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexande Lacoste, David Vazquez, et al. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *arXiv preprint arXiv:2003.05856*, 2020.
- [16] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.
- [17] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [18] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.
- [19] Arslan Chaudhry, MarcAurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019.
- [20] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc’Aurelio Ranzato. On tiny episodic memories in continual learning, 2019.
- [21] Li Chen, Chunyan Yu, and Lvcai Chen. A new knowledge distillation for incremental object detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019.
- [22] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.
- [23] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.
- [24] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, volume 1, pages 539–546. IEEE, 2005.
- [25] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. *Proceedings of Machine Learning Research*, 2020.

- [26] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.
- [27] Riccardo Del Chiaro, Bartłomiej Twardowski, Andrew Bagdanov, and Joost van de Weijer. Ratt: Recurrent attention to transient tasks for continual image captioning. *Advances in Neural Information Processing Systems*, 33, 2020.
- [28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [29] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5138–5146, 2019.
- [30] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don’t forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.
- [31] Tom Diethe, Tom Borchert, Eno Thereska, Borja Balle, and Neil Lawrence. Continual learning in practice. *arXiv preprint arXiv:1903.05202*, 2019.
- [32] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML’14, page I–647–I–655. JMLR.org, 2014.
- [33] Qi Dong, Shaogang Gong, and Xiatian Zhu. Imbalanced deep learning by minority class incremental rectification. *IEEE transactions on pattern analysis and machine intelligence*, 41(6):1367–1381, 2018.
- [34] Arthur Douillard, Matthieu Cord, Charles Ollion, and Thomas Robert. Podnet: Pooled outputs distillation for small-tasks incremental learning. Springer.
- [35] Lea Duncker, Laura Driscoll, Krishna V Shenoy, Maneesh Sahani, and David Sussillo. Organizing recurrent network dynamics by task-computation to enable continual learning. *Advances in Neural Information Processing Systems*, 33, 2020.
- [36] Benjamin Ehret, Christian Henning, Maria R Cervera, Alexander Meulemans, Johannes von Oswald, and Benjamin F Grewe. Continual learning in recurrent neural networks with hypernetworks. *arXiv preprint arXiv:2006.12109*, 2020.
- [37] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning, 2018.
- [38] Thomas S Ferguson. Bayesian density estimation by mixtures of normal distributions. In *Recent advances in statistics*, pages 287–302. Elsevier, 1983.
- [39] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

- [40] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
- [41] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. 2016.
- [42] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR, 2019.
- [43] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.
- [44] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [45] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.
- [46] Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-maml: Look-ahead meta learning for continual learning. *arXiv preprint arXiv:2007.13904*, 2020.
- [47] T. L. Hayes, N. D. Cahill, and C. Kanan. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776, 2019.
- [48] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer, 2020.
- [49] Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 220–221, 2020.
- [50] Tyler L Hayes, Ronald Kemker, Nathan D Cahill, and Christopher Kanan. New metrics and experimental paradigms for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2031–2034, 2018.
- [51] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13926–13935, 2020.
- [52] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

- [54] Xu He and Herbert Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on Learning Representations*, 2018.
- [55] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [56] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.
- [57] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019.
- [58] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.
- [59] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5375–5384, 2016.
- [60] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017.
- [61] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. *arXiv preprint arXiv:2004.00713*, 2020.
- [62] Khurram Javed and Faisal Shafait. Revisiting distillation and incremental classifier learning. In *Asian Conference on Computer Vision*, pages 3–17. Springer, 2018.
- [63] Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, pages 1820–1830, 2019.
- [64] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *Proc. VLDB Endow.*, 12(11):1610–1623, July 2019.
- [65] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176. PMLR, 2019.
- [66] Xisen Jin, Junyi Du, and Xiang Ren. Gradient based memory editing for task-free continual learning. *arXiv preprint arXiv:2006.15294*, 2020.
- [67] KJ Joseph and Vineeth N Balasubramanian. Meta-consolidation for continual learning. *arXiv preprint arXiv:2010.00352*, 2020.
- [68] KJ Joseph, Jathushan Rajasegaran, Salman Khan, Fahad Shahbaz Khan, Vineeth Balasubramanian, and Ling Shao. Incremental object detection via meta-learning. *arXiv preprint arXiv:2003.08798*, 2020.

- [69] Heechul Jung, Jeongwoo Ju, and Junmo Kim. Less-forgetful learning for domain expansion in deep neural networks. In *AAAI-18: Thirty-Second AAAI Conference on Artificial Intelligence*. the Association for the Advancement of Artificial Intelligence, 2018.
- [70] Bingyi Kang, Saiming Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.
- [71] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. volume 80 of *Proceedings of Machine Learning Research*, pages 2497–2506, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.
- [72] Zixuan Ke, Bing Liu, Hao Wang, and Lei Shu. Continual learning with knowledge transfer for sentiment classification. In *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2020.
- [73] Ronald Kemker, Angelina Abitino, Marc McClure, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI*, 2018.
- [74] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.
- [75] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. *arXiv preprint arXiv:2009.03632*, 2020.
- [76] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [77] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017.
- [78] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is np-hard. *arXiv preprint arXiv:2006.05188*, 2020.
- [79] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, April 2009.
- [80] Matthias De Lange, Xu Jia, Sarah Parisot, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Unsupervised model personalization while preserving privacy and scalability: An open problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14463–14472, 2020.
- [81] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [82] Cecilia S Lee and Aaron Y Lee. Clinical applications of continual learning machine learning. *The Lancet Digital Health*, 2(6):e279–e281, 2020.

- [83] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 312–321, 2019.
- [84] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, pages 4652–4662, 2017.
- [85] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*, 2020.
- [86] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.
- [87] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020.
- [88] Timothée Lesort, Andrei Stoian, and David Filliat. Regularization shortcomings for continual learning. *arXiv preprint arXiv:1912.03049*, 2019.
- [89] Dawei Li, Serafettin Tasci, Shalini Ghosh, Jingwen Zhu, Junting Zhang, and Larry Heck. Rilod: near real-time incremental learning for object detection at the edge. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 113–126, 2019.
- [90] Yuanpeng Li, Liang Zhao, Kenneth Church, and Mohamed Elhoseiny. Compositional language continual learning. In *International Conference on Learning Representations*, 2019.
- [91] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *ECCV*, pages 614–629. Springer, 2016.
- [92] Dahua Lin. Online learning of nonparametric mixture models via sequential variational approximation. In *Advances in Neural Information Processing Systems*, pages 395–403, 2013.
- [93] Bing Liu. Learning on the job: Online lifelong and continual learning. In *AAAI*, pages 13544–13549, 2020.
- [94] Bing Liu and Sahisnu Mazumder. Lifelong learning dialogue systems: Chatbots that self-learn on the job. *arXiv preprint arXiv:2009.10750*, 2020.
- [95] L. Liu, Z. Kuang, Y. Chen, J. H. Xue, W. Yang, and W. Zhang. Incdet: In defense of elastic weight consolidation for incremental object detection. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2020.
- [96] Tianlin Liu, Lyle Ungar, and João Sedoc. Continual learning for sentence representations using conceptors. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3274–3279, 2019.

- [97] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268. IEEE, 2018.
- [98] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 226–227, 2020.
- [99] Xialei Liu, Hao Yang, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Multi-task incremental learning for object detection.
- [100] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12245–12254, 2020.
- [101] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26, 2017.
- [102] Vincenzo Lomonaco, Lorenzo Pellegrini, Pau Rodriguez, Massimo Caccia, Qi She, Yu Chen, Quentin Jodelet, Ruiping Wang, Zheda Mai, David Vazquez, et al. Cvpr 2020 continual learning in computer vision competition: Approaches, results, current challenges and future directions. *arXiv preprint arXiv:2009.09929*, 2020.
- [103] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30*, pages 6467–6476. 2017.
- [104] Zheda Mai, Hyunwoo Kim, Jihwan Jeong, and Scott Sanner. Batch-level experience replay with review for continual learning. *arXiv preprint arXiv:2007.05683*, 2020.
- [105] Zheda Mai, Dongsub Shim, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Adversarial shapley value experience replay for task-free continual learning. *arXiv preprint arXiv:2009.00093*, 2020.
- [106] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.
- [107] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.
- [108] Sahisnu Mazumder, Bing Liu, Shuai Wang, and Nianzu Ma. Lifelong and interactive learning of factual knowledge in dialogues. *arXiv preprint arXiv:1907.13295*, 2019.
- [109] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

- [110] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2624–2637, 2013.
- [111] Martial Mermilliod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504, 2013.
- [112] Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. Continual learning for natural language generation in task-oriented dialog systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3461–3474, 2020.
- [113] Fei Mi and Boi Faltings. Memory augmented neural model for incremental session-based recommendation. *arXiv preprint arXiv:2005.01573*, 2020.
- [114] Fei Mi, Xiaoyu Lin, and Boi Faltings. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *Fourteenth ACM Conference on Recommender Systems*, pages 408–413, 2020.
- [115] Renqiang Min, David A. Stanley, Zineng Yuan, Anthony Bonner, and Zhaolei Zhang. A deep non-linear feature mapping for large-margin knn classification. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, ICDM ’09*, page 357–366. IEEE Computer Society, 2009.
- [116] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL. In *International Conference on Learning Representations*, 2019.
- [117] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.
- [118] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.
- [119] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11321–11329, 2019.
- [120] Shaoning Pang, Seiichi Ozawa, and Nikola Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE transactions on Systems, Man, and Cybernetics, part B (Cybernetics)*, 35(5):905–914, 2005.
- [121] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54 – 71, 2019.
- [122] German I Parisi and Vincenzo Lomonaco. Online continual learning on sequences. In *Recent Trends in Learning From Data*, pages 197–221. Springer, 2020.
- [123] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. *arXiv preprint arXiv:1912.01100*, 2019.

- [124] B. Pfülb and A. Gepperth. A comprehensive, application-oriented study of catastrophic forgetting in DNNs. In *International Conference on Learning Representations*, 2019.
- [125] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pages 524–540. Springer, 2020.
- [126] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 12669–12679. Curran Associates, Inc., 2019.
- [127] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13588–13597, 2020.
- [128] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1320–1328, 2017.
- [129] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*, pages 7647–7657, 2019.
- [130] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.
- [131] Matthew Reimer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019.
- [132] Matthew Reimer, Tim Klinger, Djallel Bouneffouf, and Michele Franceschini. Scalable recollections for continual lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1352–1359, 2019.
- [133] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, pages 3738–3748, 2018.
- [134] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems 32*, pages 350–360. 2019.
- [135] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.
- [136] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. Tree-cnn: a hierarchical deep convolutional neural network for incremental learning. *Neural Networks*, 121:148–160, 2020.

- [137] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya, and Umapada Pal. Effects of degradations on deep neural network architectures. *arXiv preprint arXiv:1807.10108*, 2018.
- [138] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
- [139] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.
- [140] Monika Schak and Alexander Gepperth. A study on catastrophic forgetting in deep lstm networks. In *International Conference on Artificial Neural Networks*, pages 714–728. Springer, 2019.
- [141] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, 2018.
- [142] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557, 2018.
- [143] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.
- [144] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.
- [145] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3400–3409, 2017.
- [146] Thalles Silva. Densely Connected Convolutional Networks in Tensorflow, December 2017. <https://sthalles.github.io/densely-connected-conv-nets/>.
- [147] Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. Toward training recurrent neural networks for lifelong learning. *Neural computation*, 32(1):1–35, 2020.
- [148] Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*, 2019.
- [149] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *European Conference on Computer Vision*, pages 254–270. Springer, 2020.
- [150] Gido M. van de Ven and A. Tolias. Three scenarios for continual learning. *ArXiv*, abs/1904.07734, 2019.
- [151] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

- [152] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
- [153] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.
- [154] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.
- [155] Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128, 2009.
- [156] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. In *Advances in Neural Information Processing Systems*, pages 5962–5972, 2018.
- [157] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.
- [158] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, and Yun Fu. Incremental classifier learning with generative adversarial networks. *arXiv preprint arXiv:1802.00853*, 2018.
- [159] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Zhengyou Zhang, and Yun Fu. Incremental classifier learning with generative adversarial networks. *arXiv preprint arXiv:1802.00853*, 2018.
- [160] Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. Graphsail: Graph structure aware incremental learning for recommender systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2861–2868, 2020.
- [161] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for deep face recognition with under-represented data. *arXiv preprint arXiv:1803.09014*, 2018.
- [162] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.
- [163] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.
- [164] Lu Yu, Bartłomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6982–6991, 2020.
- [165] Fajie Yuan, Guoxiao Zhang, Alexandros Karatzoglou, Xiangnan He, Joemon Jose, Beibei Kong, and Yudong Li. One person, one model, one world: Learning continual user representation without forgetting. *arXiv preprint arXiv:2009.13724*, 2020.

- [166] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70:3987, 2017.
- [167] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes, 2018.
- [168] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasçi, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1131–1140, 2020.
- [169] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217, 2020.

Part I

Appendix

| Dataset | #Task | #Train/task | #Test/task | #Class | Image Size | Setting |
|--------------------|-------|-------------|------------|--------|------------|---------|
| Split MiniImageNet | 20 | 2500 | 500 | 100 | 3x84x84 | OCI |
| Split CIFAR-100 | 20 | 2500 | 500 | 100 | 3x32x32 | OCI |
| CORes50-NC | 9 | 12000~24000 | 4500~9000 | 50 | 3x128x128 | OCI |
| NS-MiniImageNet | 10 | 5000 | 1000 | 100 | 3x84x84 | ODI |
| CORes50-NI | 8 | 15000 | 44972 | 50 | 3x128x128 | ODI |

Table A.1: Summary of dataset statistics

Appendix A Survey Experiment Details

Appendix A.1 Dataset Detail

The summary of dataset statistics is provided in Table A.1.

The strength of each nonstationary type used in the experiments are summarized below.

- Noise: [0.0, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6]
- Occlusion: [0.0, 0.07, 0.13, 0.2, 0.27, 0.33, 0.4, 0.47, 0.53, 0.6]
- Blur: [0.0, 0.28, 0.56, 0.83, 1.11, 1.39, 1.67, 1.94, 2.22, 2.5]

Appendix A.2 Implementation Details

This section describes the implementation details of each method, including the hyperparameter grid considered for each dataset (see Table A.2). As we described in Section 2.4 of the main paper, the first D^{CV} tasks are used for hyperparameter tuning to satisfy the requirement that the model does not see the data of a task more than once, and D^{CV} is set to 2 in this work.

- EWC++: We set the α in Eq. (2.8) to 0.9 as suggested in the original paper. We tune three hyperparameters in EWC++, learning rate (LR), weight decay(WD) and λ in Eq. (2.7).
- LwF: We set the temperature factor $T = 2$ as the original paper and other CL papers. The coefficient λ for \mathcal{L}_{KD} is set to $\frac{|C_{new}|}{|C_{old}|+|C_{new}|}$ following the idea from [157] and the coefficient for \mathcal{L}_{CE} is set to $1 - \lambda$.
- ER: The reservoir sampling used in *MemoryUpdate* follows Algorithm 3. For *MemoryRetrieval*, we randomly select samples with mini-batch size of 10 irrespective of the size of the memory buffer.
- MIR: To reduce the computational cost, MIR selects C random samples from the memory buffer as the candidate set to perform the criterion search. We tune LR, WD as well as C .
- GSS: For every incoming sample, GSS computes the cosine similarity of the new sample gradient to n gradient vectors of samples randomly drawn from the memory buffer (see Algorithm 4). Other than LR and WD, we also tune n .
- iCaRL: We replace the herding-based [155] memory update method with reservoir sampling to accommodate the online setting. We use random sampling for *MemoryRetrieval* and tune LR and WD.
- A-GEM: We use reservoir sampling for *MemoryUpdate* and random sampling for *MemoryRetrieval* and tune LR and WD.

| Method | CIFAR-100 | Mini-ImageNet | CORe50-NC | NS-MiniImageNet | CORe50-NI |
|--------|--|---|---|---|---|
| EWC++ | | LR: [0.0001, 0.001, 0.01, 0.1] WD: [0.0001, 0.001], λ : [0, 100, 1000] | | | |
| LwF | | LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1] WD: [0.0001, 0.001, 0.01, 0.1] | | | |
| ER | | LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1] WD: [0.0001, 0.001, 0.01, 0.1] | | | |
| MIR | | LR: [0.0001, 0.001, 0.01, 0.1] WD: [0.0001, 0.001], C : [25, 50, 100] | | | |
| GSS | | LR: [0.0001, 0.001, 0.01, 0.1] WD: [0.0001, 0.001], n : [10, 20, 50] | | | |
| iCaRL | | LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1] WD: [0.0001, 0.001, 0.01, 0.1] | | | |
| A-GEM | | LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1] WD: [0.0001, 0.001, 0.01, 0.1] | | | |
| CN-DPM | LR: [0.0001, 0.001, 0.01, 0.1] cc : [0.001, 0.01, 0.1] α : [-100, -300, -500] | [0.001, 0.005, 0.01] [0.001, 0.0015, 0.002] [-1200, -1000, -800] | [0.001, 0.01] [0.0005, 0.001, 0.002] [-1200, -1000, -800, -300] | [0.001, 0.005, 0.01] [0.0005, 0.001, 0.002] [-15000, -5000, -500] | [0.001, 0.01] [0.0005, 0.001, 0.002] [-1200, -1000, -800, -300] |
| GDumb | | | LR: .001, 0.01, 0.1], WD:[0.0001, 0.000001] | | |

Table A.2: Hyperparameter grid for the compared methods.

- CN-DPM: CN-DPM is much more sensitive to hyperparameters than others. We need to use different hyperparameter grids for different scenarios and datasets. Other than LR, we tune α , the concentration parameter controlling how sensitive the model is to new data and classifier_chill cc , the parameter used to adjust the VAE loss to have a similar scale as the classifier loss.
- GDumb: We use batch size of 16 and 30 epochs for all memory sizes. We clip gradient norm with max norm 10.0 and tune LR and WD.

Appendix B Additional Experiments and Results

Appendix B.1 More Results for OCI Setting

Fig B.1, B.2 and B.3 show the average accuracy measured by the end of each task on Split CIFAR-100, Mini-ImageNet and CORe50-NC with three different memory buffer sizes (1k, 5k, 10k).



Figure B.1: The average accuracy measured by the end of each task for the OCI setting on Split CIFAR-100 with three memory sizes.

Appendix B.2 OCI Tricks on Split Mini-ImageNet

We evaluate the tricks described in Section 2.7.3 on Split Mini-ImageNet. As shown in Table B.1 and Fig. B.4, we find similar results as in Split CIFAR-100 that all tricks are beneficial. LB and KDC* are

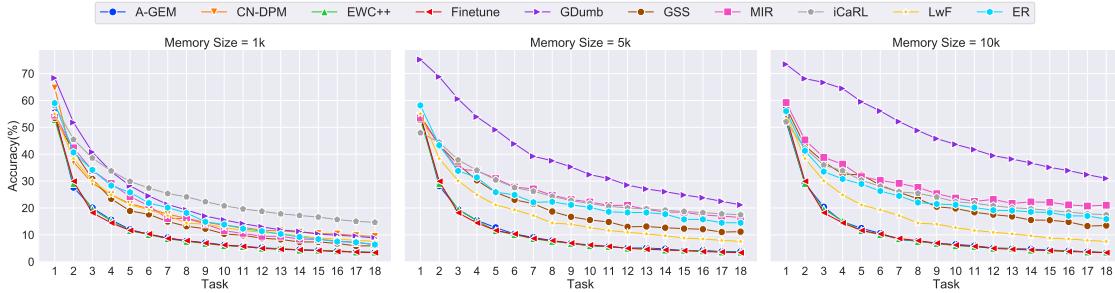


Figure B.2: The average accuracy measured by the end of each task for the OCI setting on Split Mini-ImageNet with three memory sizes.

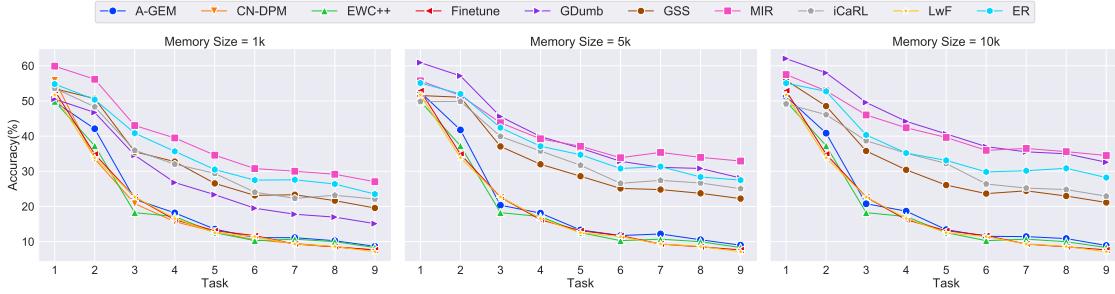


Figure B.3: The average accuracy measured by the end of each task for the OCI setting on CORe50-NC with three memory sizes.

| Finetune Offline | 3.4 ± 0.2 51.9 ± 0.5 | | | | | | | | |
|---------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| Method | A-GEM | | | ER | | | MIR | | |
| Buffer Size | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k | M=1k | M=5k | M=10k |
| NA | 3.4 ± 0.2 | 3.7 ± 0.3 | 3.3 ± 0.3 | 6.4 ± 0.9 | 14.5 ± 2.1 | 15.9 ± 2.0 | 6.4 ± 0.9 | 16.5 ± 2.1 | 21.0 ± 1.1 |
| LB | 5.8 ± 0.8 | 5.8 ± 0.5 | 5.4 ± 0.9 | 14.4 ± 2.1 | 19.3 ± 2.3 | 22.1 ± 1.1 | 17.1 ± 0.9 | 21.7 ± 0.7 | 23.0 ± 0.8 |
| KDC | 8.0 ± 1.1 | 7.5 ± 1.5 | 8.2 ± 1.7 | 12.3 ± 2.5 | 15.4 ± 0.4 | 14.6 ± 2.1 | 14.3 ± 0.5 | 15.8 ± 0.4 | 15.5 ± 0.5 |
| KDC* | 5.6 ± 0.4 | 5.5 ± 0.5 | 5.4 ± 0.4 | 16.4 ± 0.8 | 20.3 ± 2.5 | 23.0 ± 3.1 | 16.4 ± 0.6 | 25.1 ± 0.8 | 26.1 ± 0.9 |
| MI | 3.5 ± 0.2 | 3.7 ± 0.2 | 3.6 ± 0.3 | 6.4 ± 0.6 | 16.3 ± 1.3 | 24.1 ± 1.3 | 6.6 ± 0.6 | 15.2 ± 1.1 | 22.0 ± 1.9 |
| SS | 5.7 ± 0.9 | 6.2 ± 0.8 | 5.7 ± 0.8 | 12.5 ± 1.9 | 20.5 ± 2.1 | 24.1 ± 1.1 | 14.2 ± 1.0 | 21.9 ± 0.8 | 24.7 ± 0.9 |
| RV | 4.1 ± 0.2 | 19.9 ± 3.7 | 25.5 ± 4.7 | 11.4 ± 0.6 | 32.1 ± 0.8 | 36.3 ± 1.5 | 9.1 ± 0.5 | 29.9 ± 0.7 | 37.3 ± 0.5 |
| NCM | 10.2 ± 0.4 | 11.7 ± 1.5 | 13.0 ± 0.5 | 14.2 ± 0.7 | 26.7 ± 0.7 | 28.2 ± 0.6 | 13.6 ± 0.6 | 26.4 ± 0.7 | 28.6 ± 0.4 |
| Best OCI | 14.7 ± 0.4 | 21.1 ± 1.7 | 31.0 ± 1.4 | 14.7 ± 0.4 | 21.1 ± 1.7 | 31.0 ± 1.4 | 14.7 ± 0.4 | 21.1 ± 1.7 | 31.0 ± 1.4 |

Table B.1: Performance of compared tricks for the OCI setting on Split Mini-ImageNet. We report average accuracy (end of training) for memory buffer with size 1k, 5k and 10k. Best OCI refers to the best performance from the compared methods in Table 2.8.

most useful when the memory buffer is small, and NCM and RV are more effective when the memory buffer is large. One main difference is that NCM is not as effective as in CIFAR-100 with a 10k memory buffer as base methods with NCM cannot outperform the best OCI performance.

Appendix B.3 More Results for ODI Setting

Fig. B.5, B.6 and B.7 show the average accuracy measured by the end of each task on Mini-ImageNet-Noise, Mini-ImageNet-Occlusion and CORe50-NI with three different memory buffer sizes (1k, 5k, 10k).

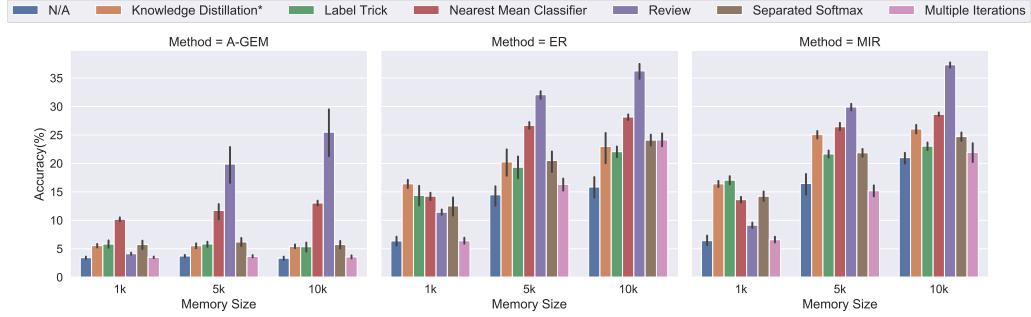


Figure B.4: Comparison of various tricks for the OCI setting on Split Mini-ImageNet. We report average accuracy (end of training) for memory buffer with size 1k, 5k and 10k.

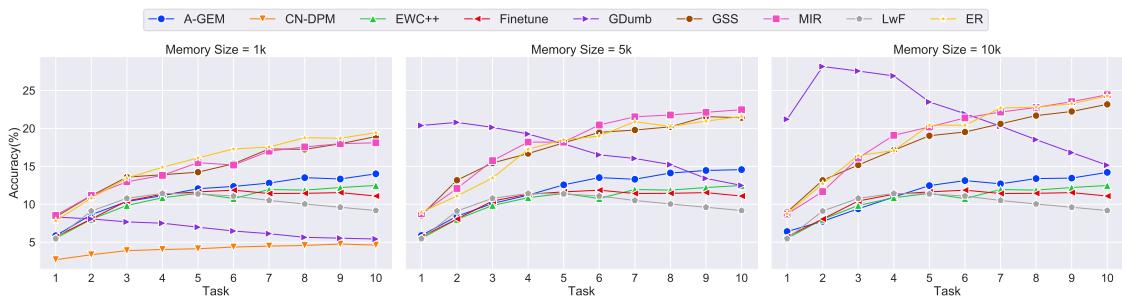


Figure B.5: The average accuracy measured by the end of each task for the ODI setting on Mini-ImageNet-Noise with three memory sizes.

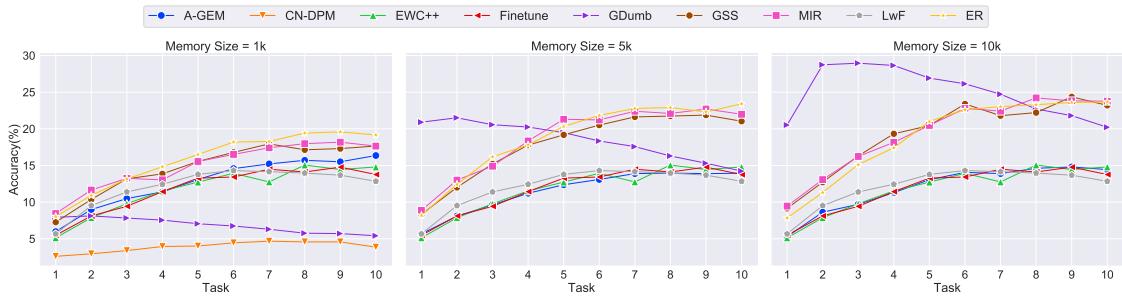


Figure B.6: The average accuracy measured by the end of each task for the ODI setting on Mini-ImageNet-Occlusion with three memory sizes.

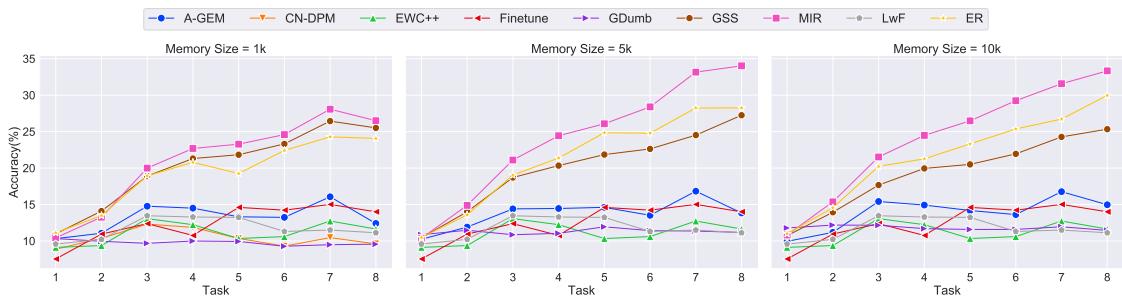


Figure B.7: The average accuracy measured by the end of each task for the ODI setting on CORe50-NI with three memory sizes.