École des Ponts ParisTech

cea list

THÈSE DE DOCTORAT
de l'École des Ponts ParisTech

Class-Incremental Learning in an Exemplar-Free Setup

École doctorale MSTIC N°532

Informatique

Thèse préparée au CEA Tech, équipe LASTI et au LIGM - UMR 8049, équipe IMAGINE

Thèse soutenue le 5 décembre 2023, par
Grégoire PETIT

Composition du jury:

Françoise PRETEUX Institut Mines-Télécom	*Présidente & Examinatrice*
Ioannis KANELLOS IMT Atlantique	*Rapporteur*
Joost VAN DE WEIJER Universitat Autónoma de Barcelona	*Rapporteur*
Tyler HAYES NAVER LABS Europe	*Examinatrice*
David PICARD Ecole des Ponts ParisTech	*Directeur de thèse*
Bertrand DELEZOIDE CEA LIST	*Directeur de thèse*
Adrian POPESCU CEA LIST	*Encadrant de thèse*

Abstract

This thesis explores the challenges and strategies associated with Class-Incremental Learning (CIL), a branch of machine learning that focuses on the efficient integration of new classes into existing models. The work addresses key difficulties, including catastrophic forgetting and the trade-off between stability and plasticity, especially in scenarios where past class data is not storable. A number of innovative methods are proposed, ranging from strategic management of the model parameter distribution to the creation of a pseudo-feature generator for improved stability-plasticity balance. The study also provides a thorough experimental analysis of various factors influencing CIL performance, such as initial training strategies, neural architecture selection, and CIL algorithm choice. The research culminates in the provision of practical guidelines for effective incremental learning to facilitate the real-world application of CIL. In summary, the thesis contributes significant insights and methods to the field of machine learning, fostering a better understanding and use of Class-Incremental Learning.

Cette thèse explore les défis et les stratégies associés à l'apprentissage incrémental de classe (CIL), une branche de l'apprentissage automatique qui se concentre sur l'intégration efficace de nouvelles classes dans les modèles existants. Le travail aborde les principales difficultés, y compris l'oubli catastrophique et le compromis entre la stabilité et la plasticité, en particulier dans les scénarios où les données sur les classes passées ne sont pas stockables. Un certain nombre de méthodes innovantes sont proposées, allant de la gestion stratégique de la distribution des paramètres du modèle à la création d'un générateur de pseudo-caractéristiques pour un meilleur équilibre entre stabilité et plasticité. L'étude fournit également une analyse expérimentale approfondie des divers facteurs influençant les performances du CIL, tels que les stratégies d'entrainements initiaux, la sélection de l'architecture neuronale et le choix de l'algorithme du CIL. La recherche aboutit à la fourniture de lignes directrices pratiques pour un apprentissage incrémental efficace afin de faciliter l'application du CIL dans le monde réel. En résumé, cette thèse apporte significativement des idées et des méthodes au domaine de l'apprentissage automatique, en favorisant une meilleure compréhension et une meilleure utilisation de l'apprentissage incrémental par classe.

Acknowledgement

I don't really know how to start this section. I've been thinking about it for a while, and I still don't know how to express my gratitude to everyone who helped me along this journey. I've been fortunate to have the support of many people, and I am deeply grateful for their contributions.

I offer my sincerest gratitude to my advisor, *Dr. Adrian Popescu*, for his unwavering support, invaluable insights, and constructive feedback, all of which have been pivotal in shaping this thesis. Your dedication to my academic growth has been a source of inspiration and motivation. You taught me to be a researcher, to think critically, and to persevere in the face of challenges. I am also thankful for the freedom you gave me to explore my ideas and the trust you placed in me to deliver. I am deeply appreciative of your mentorship and friendship. I look forward to our continued collaboration and friendship. I am thankful to my directors, *Dr. Bertrand Delezoide* and *Dr. David Picard*, for their mentorship and support, creating an environment that fostered my learning and development. Your trust in my abilities and the freedom you gave to my research has greatly contributed to my confidence and to build the researcher I am today.

I extend my gratitude to the members of the Jury, whose expertise and discernment greatly enhanced the quality of my defense. *Prof. Francoise Prêteux*, as the jury president, your thought-provoking questions provided new perspectives on my work, for which I am immensely thankful. A profound acknowledgment goes to *Prof. Ioannis Kanellos*, one of the reviewers. *Prof. Kanellos*, your role in my academic journey transcends that of a reviewer; you have been an inspirational mentor since my days at IMT Atlantique. Your encouragement and faith in my capabilities catalyzed my pursuit of this PhD, profoundly influencing my academic path. I am also deeply appreciative of *Dr. Joost van de Weijer*'s meticulous evaluations and insightful feedback during the review. Your dedication and engagement in this process have been both encouraging and enjoyable. Additionally, I thank *Dr. Tyler Hayes* for her perceptive inquiries and unique perspectives during my defense. I eagerly anticipate future scholarly exchanges at upcoming conferences, where we can continue to learn from each other's work in a stimulating and enjoyable manner.

My heartfelt thanks go to my family, my unwavering pillars of support. Your strength and resilience have empowered me to chase my dreams, even in the toughest times.

To my parents, *Eric* and *Stéphanie*, your constant faith and support have been a guiding beacon in my journey. Your love, wisdom, and devotion are the bedrock of my academic pursuits. Your teachings, like a resounding bell, constantly remind me of our roots and values. Thank you for instilling in me the ethos of hard work, perseverance, and an unquenchable desire for knowledge... Without ever forgetting the importance of love. Your contributions to my life are immeasurable, and I am eternally grateful.

To my brothers, *Marc* and *Pierre-Yves*, my lifelong companions and friends, your influence and support have been priceless. You've been with me through every challenge and success, providing a listening ear, wise advice, or a laugh to lighten the mood. Our shared experiences and mutual respect have added immense value to my life, offering solace and normalcy in the midst of academic demands. Your journeys, challenges, and achievements continually remind me of our family's resilience and core values. Thank you for accompanying me on this path.

I extend my deepest appreciation to my cherished friends, *Alban*, *Alice*, *Bastien*, *Gavin*, *Hugo*, *Jessi*, *Joseph*, *Kamil*, *Louis*, *Mathilde*, *Micha*, *Nathan*, *Noha*, *Nolwenn*, *Pauline*, *Pierre*, *Raphaël*, *Raymond*, *Sébastien*, *Tobias*, and *Tom*, who have been a source of constant encouragement, joy, and companionship. Your support during our collective struggles and victories has made this path not just tolerable but enjoyable. Thank you for enduring my frequent academic ramblings, the laughter at almost every occasion, late-night calls, all the little touches, and for sharing dreams and aspirations. You've turned the strenuous process of earning a PhD into a profound and humane adventure. I am indebted to you for your steadfast support and friendship. Your presence has brightened my journey and reassured me of a group of loyal friends for life's challenges. I eagerly anticipate celebrating our future achievements together.

Finally, I extend my deep appreciation to my colleagues at the CEA Tech and the IMAGINE Lab of ENPC. I am grateful for the stimulating discussions, the shared moments of joy and frustration, and the sense of scientific community we've friendly built together. Also, I thank everyone who has directly or indirectly contributed to this work and supported me throughout this journey.

You all embody the idea that pursuing a PhD is a communal effort, not a solitary pursuit.

Contents

0.1 Definitions 1
0.2 Acronyms 4
0.3 Notations 4

1 Introduction **7**
1.1 Motivation 7
1.2 Background 9
1.2.1 What mean to learn? 9
1.2.2 Datasets 10
1.2.3 Data preprocessing 11
1.2.4 Risks 12
1.2.5 Learning approaches 14
1.2.6 Deep Learning 15
1.3 Challenges in Class-Incremental Learning 18
1.4 Metrics 20
1.4.1 Initial Accuracy 21
1.4.2 Final Accuracy 21
1.4.3 Mean Incremental Accuracy 21
1.4.4 Mean Forgetting 22
1.5 Training Procedures 22
1.5.1 Classical Training 22
1.5.2 Class Incremental Training 24
1.6 Examplar-Free Class-Incremental Learning 24
1.7 Contributions overview 27
1.7.1 PlaStIL: Plastic and Stable Memory-Free Class-Incremental Learning 28
1.7.2 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning 28
1.7.3 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning 29

2 State of the Art in Exemplar-Free Class Incremental Learning **31**
2.1 Model-Growth based Incremental Learning 33
2.2 Fixed-Representation-Based Incremental Learning 39
2.3 Finetuning-Based Class-Incremental Learning 42
2.3.1 Basic Concepts of Finetuning-Based Class-Incremental Learning . 42

2.3.2 Major Developments and Methods in Finetuning-Based Class-Incremental Learning . . . 43
2.3.3 Advanced Applications and Emerging Trends in Finetuning-Based Class-Incremental Learning . . . 44
2.4 Pre-training Techniques for CIL . . . 45
2.4.1 Self-Supervised Learning and Its Role in Pre-Training . . . 46
2.4.2 Pre-training Applications in Class Incremental Learning . . . 46
2.5 Exemplar-Based Class-Incremental Learning . . . 46
2.6 Our EFCIL contributions . . . 49

3 PlaStIL: Plastic and Stable Exemplar-Free Class-Incremental Learning 51
3.1 Introduction . . . 51
3.2 Related Work . . . 52
3.3 Proposed Method . . . 55
3.3.1 Problem Formalization . . . 55
3.3.2 PlaStIL Description . . . 56
3.4 Experiments . . . 58
3.4.1 Datasets . . . 58
3.4.2 State-of-the-art methods . . . 59
3.4.3 Implementation . . . 60
3.4.4 Main results . . . 61
3.4.5 Method analysis . . . 64
3.5 Conclusion . . . 67

4 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning 69
4.1 Introduction . . . 69
4.2 Related Work . . . 71
4.3 Proposed Method . . . 73
4.3.1 Generation of pseudo-features . . . 74
4.3.2 Selection of pseudo-features . . . 75
4.3.3 Linear classification layer training . . . 75
4.4 Evaluation . . . 76
4.4.1 Results . . . 79
4.4.2 Method analysis . . . 82
4.5 Conclusion . . . 85

5 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning 87
5.1 Introduction . . . 88
5.2 Background . . . 89
5.2.1 Pre-training methods . . . 89
5.2.2 Class-Incremental Learning (CIL) . . . 90

5.3 Problem statement 91
5.3.1 EFCIL process 92
5.3.2 Training strategies for the initial model 92
5.4 Experimental setting 93
5.4.1 Initial training strategies 93
5.4.2 Target datasets 93
5.4.3 Incremental learning 94
5.4.4 Metrics 94
5.5 Analysis of results 95
5.5.1 Modeling causal effects 95
5.5.2 Metrics and confounding Factors 97
5.5.3 Linear Regression 98
5.5.4 Factors influencing incremental performance 99
5.5.5 Comparison of initial training strategies 101
5.5.6 Further analysis of initial training strategies 102
5.6 Discussion 105
5.7 Conclusion 106

6 Conclusion 107
6.1 General conclusion 107
6.2 Contributions 109
6.2.1 PlaStIL: Plastic and Stable Memory-Free Class-Incremental Learning 109
6.2.2 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning 110
6.2.3 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning 111
6.3 Future work and perspective 111

Bibliography 115

A List of publications 137

B Datasets details and implementation details 139
B.1 Datasets details 139
B.1.1 PlaStIL: Plastic and Stable Exemplar-Free Class-Incremental Learning 139
B.1.2 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning 139
B.1.3 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning 140
B.2 Implementation details 140
B.2.1 PlaStIL: Plastic and Stable Exemplar-Free Class-Incremental Learning 141
B.2.2 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning 142

B.2.3 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning 143
B.3 Influence of factors on accuracy 144

C Résumé en français **149**
C.1 Introduction 149
C.2 État de l'art 150
C.3 PlaStIL : Apprentissage par classe sans mémoire, plastique et stable 151
C.4 FeTrIL : Translation des caractéristiques pour l'apprentissage incrémental par classe sans exemple 152
C.5 Analyse des stratégies d'initialisation pour l'apprentissage incrémental de classes sans exemple 153

Lexicon

0.1 Definitions

- A **class** in machine learning denotes a set of data that has inherent similarities to each other and distinct differences when compared to instances in other classes.

- A **state** in learning systems denotes a particular configuration of classes that the system is to learn. It is often constituted as a set of classes.

- An **incremental state** refers to a state that is different from the initial or original state, often indicating the incorporation of new classes or data into the learning system.

- **Catastrophic Forgetting** describes the unfavorable scenario in which a learning model, in the process of learning new states, inadvertently erases information about previously learned states, thus hindering its performance on older tasks.

- A **Backbone** refers to the basic model, typically a deep neural network architecture, that the incremental learning system uses to learn and discriminate between new classes.

- A **Loss Function** quantifies the discrepancy between the model's predictions and the actual results during the training phase and serves as a guide for the optimal adjustment of the model's parameters.

- **Contrastive Learning** is a self-supervised learning approach that learns to discriminate between classes by contrasting the representations of positive (similar) and negative (dissimilar) pairs of samples, allowing the system to build robust class representations without the need for additional exemplars from previous classes.

- **Knowledge Distillation** refers to the process of transferring knowledge from an existing, typically larger and more complex model (the "teacher") to a newer, often simpler model (the "student") learning a new class. This technique serves to combat catastrophic forgetting by encouraging the student model to retain insights about past classes while learning new ones.

- **Backpropagation** is an efficient algorithm used to compute the gradient of the loss function with respect to the model's parameters, which is then used to update the parameters during the training phase.

- **Gradient Descent** is an optimization strategy used to iteratively refine the parameters of the model during the training phase, using the gradient of the loss function with respect to those parameters.

- An **Optimizer** is a method or tool that strategically changes the model's parameters or hyperparameters during the training phase in order to minimize the loss function.

- **Weight Decay** is a regularization technique used to prevent overfitting during the model training process by adding a penalty to the size of the model parameters.

- The term **Momentum** in machine learning refers to a hyperparameter that determines the extent to which past gradients influence the current step taken by the optimizer during the model's training phase.

- The term **Learning Rate** refers to a critical hyperparameter that determines the size of the step taken by the optimizer during the model's training process.

- A **Learning Rate Scheduler** is a function or policy that systematically modulates the learning rate during the training process, typically reducing it over time to improve convergence.

- An **Epoch** is a complete pass or iteration over the entire training data set.

- The **Chain Rule** is a fundamental mathematical rule used to compute the derivative of composite functions and plays a critical role in the backpropagation algorithm.

- A **Batch** is defined as a smaller subset of the training data set that is used in an iteration of model training. The amount of data within the batch, known as the **batch size**, affects the speed and accuracy of the learning process.

- **Overfitting** describes a common problem in machine learning where a model overly adapts to the training data, leading to lower performance when encountering unseen data due to a lack of generalization.

- **Underfitting** is a scenario where the simplicity of the model fails to capture the complexity of the data, resulting in suboptimal performance on both training and unseen data.

- **Regularization** refers to a technique used to prevent overfitting. It does this by integrating a penalty term into the loss function that limits the complexity of the model.

- An **Activation Function** is a mathematical function used within a neural network. Its purpose is to transform the input signal into a usable output signal for the next layer, introducing nonlinear properties into the network's decision function and thus enabling the learning of complex patterns.

- **Dropout** is a specific regularization technique used in neural networks where a portion of the neurons within a layer are randomly ignored during the training process. This helps prevent overfitting.

- A **Convolutional Neural Network (CNN)** is a specialized type of deep learning model. CNNs excel at processing grid-like data, such as images, by preserving the spatial relationships between features. They use convolutional layers with filter windows called "kernels" to scan over the data.

- A **Transformer** is a specialized type of deep learning model. Characterized by its use of self-attention mechanisms, a transformer model assigns varying levels of importance, or attention, to different elements within the input sequence. This ability empowers the model to comprehend the context and dependencies between elements, regardless of their distance from each other in the sequence.

- **Transfer Learning** is a machine learning approach where a pre-trained model is utilized as a starting point for a related task. This method leverages the knowledge acquired from solving a previous problem to tackle a similar, often smaller, problem.

- **Fine-tuning** refers to the process of making minor adjustments to the parameters of an already trained model when it's applied to a related task.

- **Data Augmentation** describes the technique of enlarging the size of a training dataset by creating modified versions of existing instances, including transformations like rotating, scaling, or cropping in the context of image data.

- **Feature Extraction** is the procedure of converting raw data into a collection of features or a feature vector, which can be effectively processed by a machine learning model. These features are designed to encapsulate the essential attributes of the data.

- **Latent Space** refers to the compressed, abstract representation of data within the hidden layers of a machine learning model. It is a lower-dimensional space in which high-dimensional inputs are encoded in a way that captures their essential features.

0.2 Acronyms

During the course of this thesis, the following acronyms are used:

- **AI** Artificial Intelligence
- **ML** Machine Learning
- **DL** Deep Learning
- **CL** Continual Learning
- **IL** Incremental Learning
- **CIL** Class Incremental Learning
- **DIL** Domain Incremental Learning
- **TIL** Task Incremental Learning
- **EBCIL** Exemplar Based Class Incremental Learning
- **EFCIL** Exemplar Free Class Incremental Learning
- **CNN** Convolutional Neural Network

0.3 Notations

During the course of this thesis, the following notations are used:

$\mathcal{M}$	the full model;
$\mathcal{F}$	the feature extractor, i.e. the full model without the classification layer;
$\mathcal{B}$	the model base which includes the initial layers;

$\mathcal{T}$	the model top which includes the subsequent layers up to the classification one;
$\mathcal{W}$	the classification layer which provides class predictions.
$\mathcal{D}$	the training dataset;
C_i	the i-th class in the dataset;
$\mathcal{D}_k$	the training dataset for the k-th incremental state;
w_i	the weight of the classification layer associated to the i-th class;
T	the number of incremental states in a evenly divided incremental learning scenario;
K	the number of incremental states in an incremental learning scenario where half of the classes are learned in the first state and the other half in the subsequent states;
N	the number of classes in the dataset;
$\overline{Acc}$	the incremental accuracy;

Introduction 1

Ma mémoire est plus fidèle
qui sait si bien oublier.
Elle a sans doute un peu brouillé
les lignes, défait les contours,
estompé les décors qui restent imprécis. . .
Mais au souvenir réussi
elle a laissé son goût d'amour.

— **Paul Géraldy**
Toi et Moi

1.1 Motivation

Embarking on a journey through the corridors of mind, one of my earliest experiences comes to life: the image of my parents patiently introducing me to the world of reading, writing, numbers, and the diverse realm of animals, plants, country flags, and even car brands. Those formative moments, marked by the excitement of acquiring new knowledge and the pride of recognizing distinct patterns, remain etched in my mind like intricate brushstrokes on the canvas of my childhood. Little did I know that those innocent encounters with learning would lay the foundation for a lifelong quest to understand the intricate workings of advanced learning algorithms, neural networks, and the compelling orchestration of data analysis.

As I grew older, I became increasingly fascinated by the power of learning and the ability to acquire new knowledge and skills. I was intrigued by the idea of learning from experience and the ability to adapt to new situations, acquire new skills and knowledge, and continuously improve my performance. I noticed that with each new language I tackled, the learning process seemed to become somewhat less daunting. Similarly, when I took up a new instrument, I found the challenge to be a bit more manageable. I was able to acquire new knowledge more rapidly and effectively due to my prior acquisition of general knowledge foundations upon which subsequent learning in those specific domains could build.

Therefore, learning from experience is a critical aspect of human intelligence, enabling us to adapt to new situations, acquire new skills and knowledge, and continuously improve our performance. This ability is desired in artificial intelligence (AI) systems that aim to replicate and enhance human cognitive abilities. One of the most ambitious goals in Machine Learning (ML) research is to enable machines to learn continuously from a stream of data, similar to how humans learn throughout their lives. This capability, called Continual Learning (CL), allows machines to acquire new knowledge incrementally while preserving what was learned in the past.

However, Continual Learning poses a strong challenge because traditional Machine Learning (ML) algorithms are not designed to handle the complexity and variability of real-world data streams [Fre99]. One of the primary obstacles is the phenomenon of catastrophic forgetting, a situation in which a model tends to lose previously learned information when introduced to new data [MC89]. This problem curbs the adaptability of Machine Learning systems to novel tasks, as they risk losing previously acquired knowledge.

Therefore, it is crucial to develop Machine Learning algorithms that are able to learn continuously, retain their knowledge, and adapt to new situations without losing previously learned information. This thesis explores the fundamental concepts, challenges, and developments in incremental learning, with a particular focus on overcoming the problem of plasticity/stability.

Continual learning takes several forms, including Task-Incremental Learning, Domain-Incremental Learning, and Class-Incremental Learning (CIL). Among these, Class-Incremental Learning is particularly challenging due to the nature of the problem. In Class-Incremental Learning, the learning system must incrementally learn a sequence of new classes, where each class is learned once and never revisited. This constraint means that the system cannot retrieve data from previously learned classes, making it highly vulnerable to catastrophic forgetting.

The problem of catastrophic forgetting in Class-Incremental Learning is exacerbated by the likelihood that the model's representations of previously learned classes will be overwritten or distorted when new classes are introduced. As a result, the model's ability to recognize previously learned classes can be severely compromised by the plasticity of the network, leading to degraded performance on earlier tasks. To address this problem, various techniques such as repetition, regularization, generative replay, fixed model approach, and model-growth approach have been proposed to mitigate catastrophic forgetting in Class-Incremental Learning.

The plasticity/stability dilemma, often considered a fundamental issue in Continual Learning, overshadows even the challenge of catastrophic forgetting alone. Plasticity refers to the ability of an AI system to adapt to new information by adjusting its internal models, while stability involves preserving and protecting the knowledge that the system has already acquired. Essentially, there is a delicate balance between learning new information (plasticity) and retaining old information (stability). This balance is arguably more critical than only preventing catastrophic forgetting because it also involves the integration of new information.

For example, consider a Machine Learning system that learns to recognize different breeds of dogs. If the system relies too much on plasticity, it may efficiently learn to recognize new breeds that are introduced to it, but at the same time, it may forget how to recognize the breeds that it has learned about before: this is catastrophic forgetting. On the other hand, if the system is biased too much toward stability, it may retain the ability to recognize the breeds it has previously learned but struggle to learn new breeds.

This dilemma also arises when comparing traditional Machine Learning models with their Continual Learning counterparts. Traditional models that emphasize stability can remember their training data quite well but often fail to incorporate new data or adapt to shifts in data distribution. In contrast, a model that adopts a Continual Learning approach and maintains a balance between plasticity and stability can effectively handle real-world situations where the data is not stationary but evolves continuously over time. Therefore, solving the plasticity/stability dilemma is essential for developing truly adaptive and resilient AI systems.

This thesis provides an overview of these techniques and discusses their effectiveness in real-world scenarios.

1.2 Background

Now that we have introduced the motivation for this thesis, we will provide some background information on the topics covered in this thesis. This section provides a brief overview of the fundamental concepts and terminology.

1.2.1 What mean to learn?

From the Oxford dictionary, the definition of *to learn* is:

to gain knowledge or skill by studying, from experience, from being taught, etc.

Subsequently, the concept of *learning* can be delineated as the dynamic acquisition of knowledge or skills through dedicated study, immersive experience, or guided instruction. Nevertheless, within the domain of Machine Learning, this definition takes on a nuanced perspective. Through discerning patterns and interconnections within the dataset, the Machine Learning model extrapolates these insights to formulate predictions or decisions.

This process is akin to the human learning process, where we learn from experience and apply our knowledge to new situations.

To do so, the first need for a Machine Learning model to learn is data.

1.2.2 Datasets

Machine Learning techniques are used to extract information from data and apply it to new data that was not in the original data set. This is called data-driven processing. Data-driven processing is necessary when there is no specific knowledge about how to process the data, such as how to recognize text characters. In this case, Machine Learning can be used to learn a process from a large amount of data (here, text scan images and their transcription into strings of characters).

Here is another example of a problem that can be solved with Machine Learning: Spam filtering. There is no known algorithm that can filter spam perfectly. However, Machine Learning can be used to learn a process from a large amount of data (email messages labeled as spam or non-spam) that can accurately filter spam emails with high precision and recall.

To be able to classify those emails as spam or non-spam, we could think of programming something with domain knowledge. For example, we could program a rule that says that if the mail contains the words *Awesome* and *Deal*, then it is spam. However, this rule would not be very effective because it would not be able to detect spam emails that do not contain the words *Awesome* and *Deal*. Domain knowledge refers to the specialized information and expertise about a specific subject or field that is used to solve problems in that field.

The key difference between Machine Learning and domain knowledge is that Machine Learning can be used to solve problems for which there is no known algorithm. This is because Machine Learning can learn from data, even in cases where the data doesn't explicitly provide rules to solve the problem. Domain knowledge, on the other hand, is useful for problems for which there is a known solution that can be expressed in an algorithm.

To learn from data, we first need to define the data that we will use. This data is called a dataset. A dataset is a collection of data that is used for Machine Learning. Datasets can be as simple as a single file or as complex as a database with millions of records. The size and complexity of the dataset will depend on the complexity of the Machine Learning model that we want to train. In other words, a dataset is a collection of data that is used to train a machine-learning model. The dataset should be representative of the data that the model will be used to process in the real world. For example, if we want to train a model to classify spam emails, the dataset should include a representative sample of spam and non-spam emails.

While datasets serve as the foundation for training Machine Learning models, the concept of dynamic datasets introduces an element of adaptability and evolution into the realm of data-driven processing. Dynamic datasets embrace the idea that Machine Learning models can be continuously refined and improved as new data becomes available, even in cases where the problem does not have a predefined algorithmic solution.

1.2.3 Data preprocessing

Because raw data is often noisy and unstructured, it is necessary to preprocess the data before using it for Machine Learning. Data preprocessing is the process of preparing raw data for Machine Learning. Imagine you have to deal with the spam email classification problem. You have a dataset of emails labeled as spam or non-spam. The dataset contains a lot of noise and unstructured data. For example, some emails are missing the subject line, some emails have multiple recipients, and some emails have attachments. To make the data more structured and easier to process, you need to preprocess it. This involves cleaning the data, formatting it correctly, and creating new features.

Data preprocessing is the place for introducing specific knowledge. This includes tasks such as:

- **Cleaning the data:** Removing errors, outliers, and missing values. This can be done by removing rows or columns with missing values, replacing missing values with the mean or median, or imputing missing values with a value from a similar row.

- **Formatting the data:** Standardizing data types and ranges. This can be done by converting categorical variables to numerical variables, converting numerical variables to categorical variables, or converting numerical variables to a standard range.

- **Feature engineering:** Create new features from existing ones. This can be done by combining multiple features into one, creating new features based on existing ones, or creating new features based on domain knowledge.

- **Dimensionality reduction:** Reduce the number of features to improve performance. This can be done by removing features that are not predictive of the target variable, removing features that are highly correlated with each other, or combining several features into a single feature.

- **Data partitioning:** Separating data into training, validation, and test sets. This can be done by randomly splitting the data into training, validation, and test sets, or by using a stratified split to ensure that each set contains a representative sample of the data.

Data preprocessing is an important step in Machine Learning because it can improve the accuracy and performance of Machine Learning models. By cleaning and properly formatting the data, we can ensure that the model is trained on data that is accurate and consistent. Feature engineering can help us create new features that are more predictive of the target variable. Dimensionality reduction can help us improve model performance by reducing the number of features the model needs to learn. Splitting the data into training, validation, and test sets allows us to evaluate the performance of the model on unseen data.

Here are some of the benefits of data preprocessing in Machine Learning:

- **Improved accuracy:** Data preprocessing can help improve the accuracy of Machine Learning models by removing errors, outliers, and missing values from the data. This can help ensure that the model is trained on data that is accurate and consistent.

- **Increase performance:** Data preprocessing can also help increase the performance of Machine Learning models by reducing the number of features the model needs to learn. This can be done through dimensionality reduction, which can help improve the speed and accuracy of the model.

- **Understand the data better:** Data preprocessing can also help us better understand the data we are working with. This can be done by cleaning the data, formatting it correctly, and creating new features. This can help us identify patterns in the data and make better decisions about how to train the machine-learning model.

Overall, data preprocessing is an important step in Machine Learning that can help improve the accuracy, performance, and understanding of Machine Learning models.

1.2.4 Risks

Once the data is preprocessed, we can use it to train a Machine Learning model. To do so, the model must be able to learn from the data. This means that the model must be able to know when it is making a mistake and correct itself. This is called risk minimization. Risk can be interpreted as the cost of making a mistake. In most cases, however, it is impossible to know the true risk. Therefore, we need to define a proxy for risk, called empirical risk. Empirical risk is calculated by averaging the loss over a sample of data. The key to defining empirical risk is related to the law of large numbers. The law of large numbers states that as the number of samples increases, the empirical risk converges to the actual risk.

Empirical risk is used to train Machine Learning models. Empirical Risk Minimization (ERM) aims to find a model that minimizes empirical risk. This means that the model will make as few mistakes as possible on the training data. However, it is important to remember that empirical risk is only an estimate of generalization error: the model can still make mistakes on unseen data.

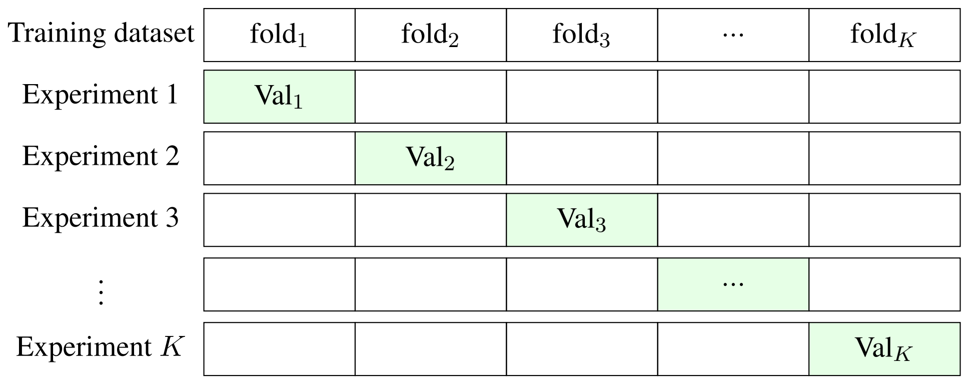


Fig. 1.1.: Illustration of the cross-validation procedure, with K folds.

To reduce the risk of making mistakes on unseen data, we can use a technique called cross-validation, illustrated in Figure 1.1. In K-fold cross-validation, the training set is K times divided into two parts: a pseudo-training set and a validation set. The model is trained on the pseudo-training set and then evaluated on the validation set. The cross-validation error is the error rate on the validation set. The goal of cross-validation is to find a model that minimizes the cross-validation error. This means that the model will make as few errors as possible on unseen data.

Cross-validation is particularly usefull because it allows us to estimate the true risk of a Machine Learning model. Indeed, since the process is repeated K times, and each time the model is trained on a different pseudo-training set and evaluated on a different validation set, we can get a better estimate of the true risk of the model. This, at the end, helps us choose a model that is more likely to make accurate predictions on unseen data.

Figure 1.1 illustrates the principle of cross-validation with K folds of the training set $\mathcal{D}$. K experiments are performed. In the experiment $k \in [\![1, K]\!]$, the model is trained on the pseudo-training set $\mathcal{D} \backslash \text{fold}_k$ and evaluated on the validation set fold_k. The cross-validation error is the average error rate over the K experiments. Note that the folds are not necessarily chosen adjacent, but rather randomly seeded and then chosen.

We can evaluate two main things using cross-validation: the mean error rate and the variance of the error rate. The mean error rate is the average error rate over the K experiments. The variance of the error rate is the variance of the error rate over the K experiments. The goal of cross-validation is to find a model that minimizes the mean error rate and the variance of the error rate. This means that the model will make as few errors as possible on unseen data and that the model will make consistent errors on unseen data.

By using cross-validation, we can get a better estimate of the true risk of a Machine Learning model. This helps us choose a model that is more likely to make accurate predictions on unseen data.

1.2.5 Learning approaches

We will now introduce the different learning approaches that are used in Machine Learning. These approaches are used to solve different types of problems. Depending on the availability of the labels, we can define 4 main learning approaches, from the most to the least supervised: supervised learning, weakly supervised learning, semi-supervised learning, and unsupervised learning.

- **Supervised learning:** in this approach, the model is trained on a set of labeled data. The data consists of input features and output labels. The goal of supervised learning is to learn a function that maps input features to output labels. This function can then be used to make predictions on unseen data. We distinguish two types of supervised learning: classification and regression. In regression, the output labels are continuous values. In classification, the output labels are discrete values.

- **Weakly-supervised learning:** in this approach, the model is trained on a set of weakly-labeled data. The data consists of input features and weakly-labeled output labels. The goal of weakly-supervised learning is to learn a function that maps input features to weakly-labeled output labels, i.e. labels that do not necessarily match the actual output labels (if any). This function can then be used to make predictions on unseen data.

- **Semi-supervised learning:** in this approach, the model is trained on a set of labeled and unlabeled data. The data consists of input features and output labels. Some of the data is labeled, and some of the data is unlabeled. The goal of semi-supervised learning is to learn a function that maps input features to output labels. This function can then be used to make predictions on unseen data.

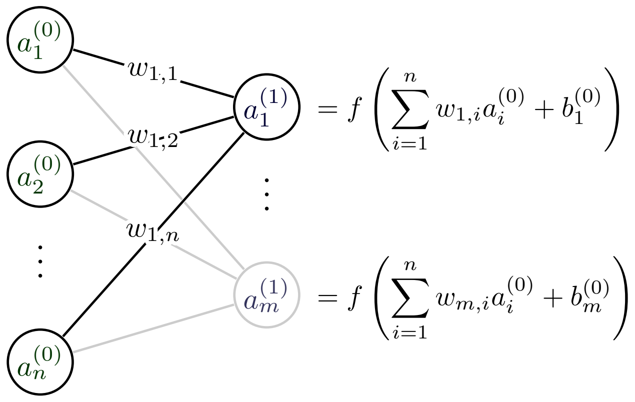
- **Unsupervised learning:** in this approach, the model is trained on a set of unlabeled data. The data consists of input features only. The goals of unsupervised learning are multiple: generalized feature extractor training, clustering, dimensionality reduction, density estimation, and anomaly detection.

1.2.6 Deep Learning

To learn hierarchical features from raw data, we are introducing Deep learning, a branch of Machine Learning that uses deep neural networks to learn from data. Deep neural networks are inspired by the structure of the human brain. They consist of multiple layers of neurons that are connected to each other. Each neuron applies a mathematical transformation to its input (usually a weighted sum followed by an activation function). The output of the model represents predicted probabilities for each possible class. Which makes it well-suited for tasks involving unstructured data such as images, text, and audio.

Activation and neurons

To introduce non-linearity into the network, an activation function f is applied to the output of each neuron. This allows the network to learn more complex patterns in the data. Without activation functions, deep neural networks would only be able to learn linear relationships between the input and output data. This would severely limit their ability to learn complex patterns, such as those found in images, text, and audio. Figure 1.2 illustrates how a neuron works.



$a_1^{(0)}$
$w_{1,1}$
$w_{1,2}$
$a_2^{(0)}$
$w_{1,n}$
$a_n^{(0)}$
$a_1^{(1)}$
$= f\left(\sum_{i=1}^{n} w_{1,i}a_i^{(0)} + b_1^{(0)}\right)$
$a_m^{(1)}$
$= f\left(\sum_{i=1}^{n} w_{m,i}a_i^{(0)} + b_m^{(0)}\right)$

Fig. 1.2.: Illustration of the working principle of a neuron.

The activation of this layer is then calculated as follows:

$$\begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_m^{(1)} \end{pmatrix} = f\left[\begin{pmatrix} w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ w_{2,0} & w_{2,1} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,0} & w_{m,1} & \dots & w_{m,n} \end{pmatrix}\begin{pmatrix} a_1^{(0)} \\ a_2^{(0)} \\ \vdots \\ a_n^{(0)} \end{pmatrix} + \begin{pmatrix} b_1^{(0)} \\ b_2^{(0)} \\ \vdots \\ b_m^{(0)} \end{pmatrix}\right] \tag{1.1}$$

In Equation 1.1, $a_i^{(l)}$ denotes the activation of the neuron i in the layer l, $w_{i,j}$ denotes the weight of the connection between the neuron i in the layer l and the neuron j in the layer $l-1$, $b_i^{(l)}$ denotes the bias of the neuron i in the layer l, and f denotes the activation function.

The activation function is applied to the weighted sum of the inputs and the bias. The output of the activation function is the activation of the neuron. The activation of the neuron is then passed to the next layer of the network:

Multi-layer Perceptron

A fundamental neural architecture is the multi-layer perceptron (MLP), which operates as a feedforward neural network consisting of multiple layers of neurons. Each layer maintains full connectivity with the next layer, with the input layer being the foremost, the output layer being the final, and the intermediate layers being referred to as hidden layers. The number of neurons in the input layer reflects the input features, while the number of neurons in the output layer reflects the output labels. The number of hidden layer neurons depends on the previous layer. Figure 1.3 illustrates the structure of an MLP.

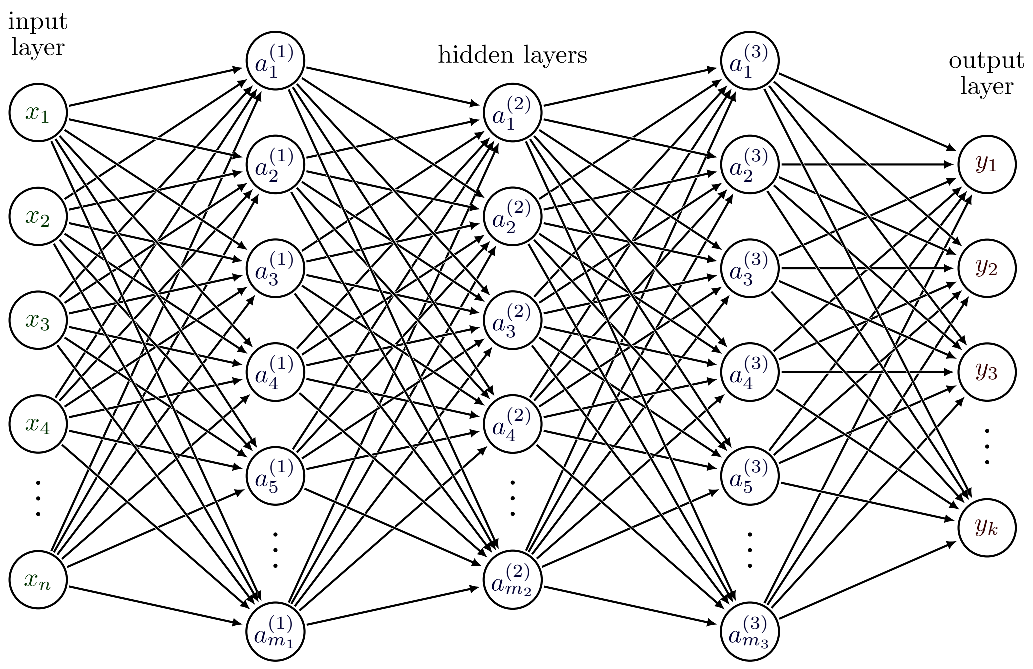


Fig. 1.3.: Illustration of the working principle of an MLP.

In Figure 1.3, the input layer consists of n neurons, the hidden layers consist of m_1, m_2, and m_3 neurons, and the output layer consists of k neurons.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are used to extract richer insights from data, especially when dealing with complex inputs such as images. CNNs are a category of deep neural networks tailored for tasks such as image classification and object detection in computer vision. Inspired by the architecture of the human visual cortex, CNNs consist of a series of convolutional and pooling layers. The convolutional layers apply a series of filters to the input image to extract features, illustrated in Figure 1.4. The pooling layers, illustrated in Figure 1.5, reduce the dimensionality of the feature maps by downsampling them. The output of the final pooling layer is fed into a fully connected layer, which performs the final classification.

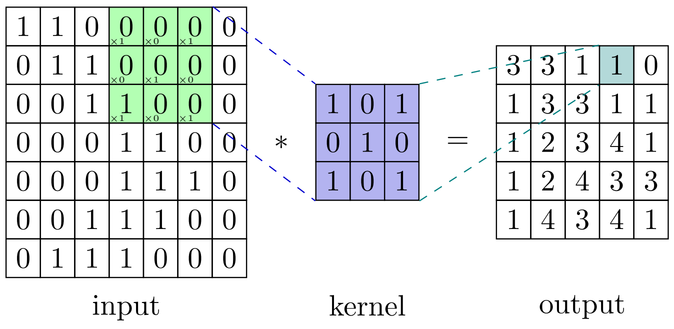


Fig. 1.4.: Illustration of the working principle of a convolutional layer.

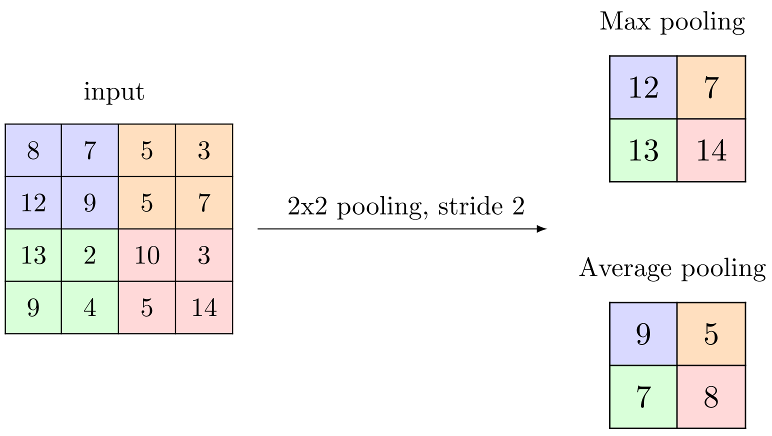


Fig. 1.5.: Illustration of the working principle of two common pooling layers.

Transformers

The Transformer emerges as a distinctive neural network architecture and a current alternative to the commonly used CNNs.

Central to Transformer's innovation is its attention mechanism, a key feature that allows the model to assign importance to different parts of an input sequence during prediction. The attention mechanism allows the model to capture relationships between different patches, enabling it to recognize spatial hierarchies and dependencies in the image.

One of the most current transformer architectures is ViT, introduced by [Dos+21], described in Figure 1.6. ViT is a transformer-based architecture that uses a sequence of an image patches and their position embedding as input. The image patches are flattened and fed into a transformer encoder, which outputs a sequence of feature vectors. The feature vectors are then fed into an MLP.

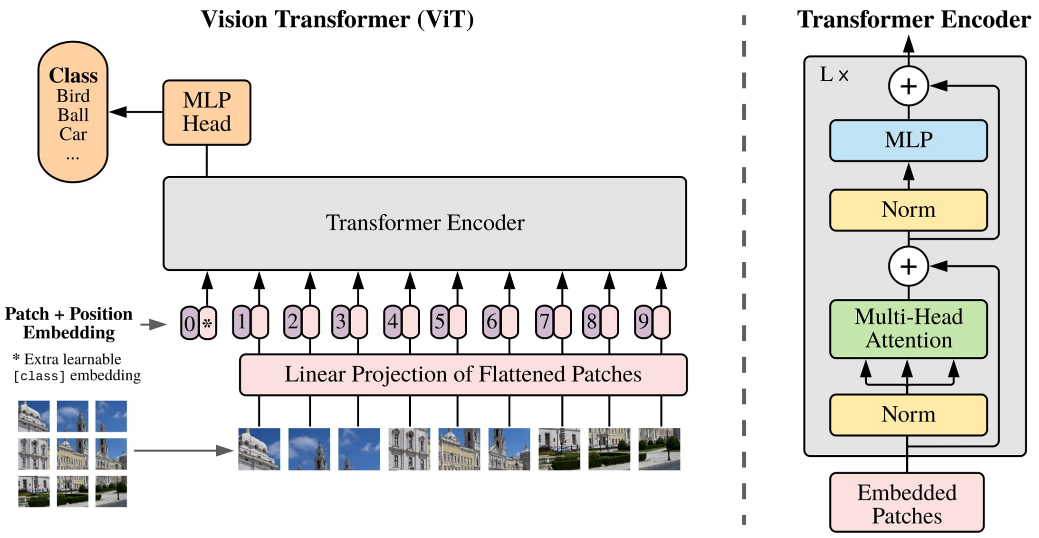


Fig. 1.6.: Illustration of the ViT architecture. Source [Dos+21].

As Machine Learning continues to evolve and expand its reach into diverse domains, these foundational principles remain essential. They serve as a compass to help researchers, data scientists, and engineers navigate the complexities of real-world data and develop models that enhance our understanding, automate decision-making, and drive innovation in fields ranging from healthcare and finance to natural language processing and computer vision. In the following sections, we take a closer look at the challenges of Machine Learning, with a particular focus on Class-Incremental Learning. We will also discuss the specific challenges of Class-Incremental Learning and how they can be addressed.

1.3 Challenges in Class-Incremental Learning

As introduced in Section 1.1, the ability to learn continuously is a critical aspect of human intelligence, enabling us to adapt to new situations, acquire new skills and knowledge, and continuously improve our performance. This ability is desired in artificial intelligence (AI) systems that aim to replicate and enhance human cognitive abilities. One of the most ambitious goals in Machine Learning (ML) research is to enable machines to learn continuously from a stream of data, similar to how humans learn throughout their lives. This capability, called Continual Learning (CL), allows machines to acquire new knowledge incrementally while preserving what was learned in the past.

Though, there are many challenges associated with Continual Learning:

Memory requirements are critical in CIL because the model must continuously learn a large number of classes, each with a limited number of examples. It's infeasible to store all images of previously learned classes, as memory requirements would grow linearly with the number of classes, leading to computational and storage problems. Therefore, the model should be designed to minimize the impact on memory usage when learning new classes.

Computational requirements are also important in CIL. The model should learn new classes quickly and efficiently without compromising the quality of the learned representations. As the number of classes increases, the training time for each new class escalates, leading to computational constraints. Techniques used to mitigate feature drift, such as distillation, can further increase the computational cost of training. Therefore, efficient learning algorithms are needed to learn new classes with limited computational resources.

The **number of states** the model must learn in CIL can significantly affect its performance. Learning a large number of states exacerbates the effect of catastrophic forgetting, as the model must retain more information without revisiting past examples. As the number of classes increases, the model's memory and computational requirements also increase, making it more difficult to learn new classes effectively.

The **variability of data stream structures** plays a critical role in the model's ability to adapt and learn new classes efficiently. Aspects such as data type, update frequency, and the size of incremental states can all influence the learning process. The variability in data stream structures requires tailored Class-Incremental Learning approaches that can handle different data types, adapt to different update frequencies, and effectively use the information available in incremental states.

Scenario variability can have a significant impact on model performance in CIL, particularly with respect to mitigating catastrophic forgetting. Different scenarios dictate how many classes make up the initial state, and thus the richness of the feature space from which the model learns.

Addressing these challenges is critical to developing more intelligent and versatile artificial intelligence systems. Class-Incremental Learning aims to emulate human learning capabilities, allowing machines to continuously acquire new knowledge without losing previously learned information. By addressing issues such as memory requirements, computational costs, and variability in scenarios and data stream structures, with respect to plasticity/stability, we can make Machine Learning models more adaptive and efficient.

In addition, Class-Incremental Learning has significant implications for practical applications. In real-world scenarios, the emergence of new classes or concepts is common. For example, in computer vision, models must learn and recognize new object categories without losing the ability to recognize previously learned categories. Addressing the challenges of Class-Incremental Learning enables the deployment of Machine Learning systems that can continuously adapt to new tasks, domains, or classes, thereby improving performance and capabilities in various application domains.

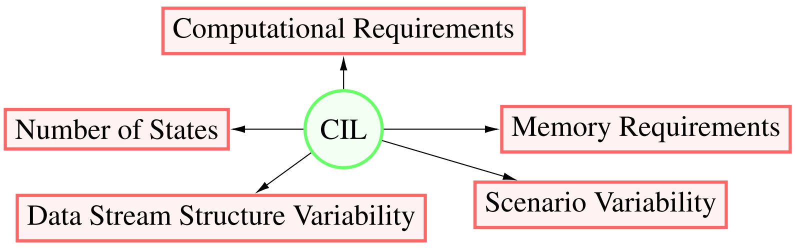


Fig. 1.7.: Outlines of the five major challenges that are directly linked to Class-Incremental Learning.

Overcoming the challenges of Class-Incremental Learning also promotes resource efficiency. By reducing memory requirements and computational costs, we can design models that require fewer resources to train and deploy, extending their accessibility and applicability to a wider range of devices and environments. This is particularly important for deploying Machine Learning in resource-constrained environments like edge devices or embedded systems.

In addition to promoting resource efficiency, Class-Incremental Learning offers promising advantages for addressing environmental issues, especially in the context of energy conservation and sustainable computing practices. The reduced memory and computational requirements of Class-Incremental Learning methods make them well-suited for use on energy-efficient devices and green computing environments.

Finally, overcoming these challenges advances the broader field of Machine Learning and artificial intelligence. Developing robust techniques for dealing with Class-Incremental Learning improves our understanding of Continual Learning, memory management, and adaptation in Machine Learning models. This knowledge can also contribute to the development of more sophisticated learning algorithms, improved training methods, and better knowledge retention and transfer strategies in AI systems.

1.4 Metrics

In order to evaluate the performance of a model, we need to define a notion of performance. This notion of performance is called a metric. There are many different metrics that can be used to evaluate the performance of a model. The most common metric is accuracy. The accuracy of a model on a dataset $\mathcal{D}_{test}$ is defined in Equation 1.2 as the proportion of correct predictions made by the model on the dataset $\mathcal{D}_{test}$:

$$acc(\mathcal{M}, \mathcal{D}_{test}) = \frac{1}{|\mathcal{D}_{test}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{test}} \mathbb{1}_{\mathcal{M}(\mathbf{x})=y} \tag{1.2}$$

where $\mathcal{M}$ denotes the model, $\mathcal{D}_{test}$ denotes the test dataset, $\mathbf{x}$ denotes an input sample, y denotes the corresponding output label, and $\mathbb{1}_{\mathcal{M}(\mathbf{x})=y}$ denotes the indicator function that is equal to 1 if $\mathcal{M}(\mathbf{x}) = y$ and 0 otherwise.

Several metrics are commonly used to evaluate the performance of a model in the context of CIL. In this section, we will introduce the most common metrics used in the literature and explain how they are calculated. We will also discuss the advantages and disadvantages of each metric and how they can be used to evaluate the performance of a model in the context of CIL. Here, the focus is on the model's performance over a K-step incremental learning procedure [Zhu+21b; Zhu+22; Zhu+21a; JLM21].

1.4.1 Initial Accuracy

To provide a more balanced statistical perspective in Section 5.5, we account for the initial accuracy. This metric corresponds to the performance of the first model on the initial data subset $\mathcal{D}_1$ and is designated as Acc_1, i.e., $Acc_1 = acc(\mathcal{M}_1, \mathcal{D}_1)$. This metric is not properly incremental, as it does not account for the performance of the model on the incremental data subsets $\mathcal{D}_2, \mathcal{D}_3, \ldots, \mathcal{D}_K$. However, the following metrics are heavily influenced by the initial accuracy (this will be studied in Chapter 5), and thus it is important to consider it in the analysis.

1.4.2 Final Accuracy

The final accuracy of the incremental learning process, noted as Acc_K, reflects the performance of the last model on the comprehensive dataset $\mathcal{D}$, i.e., $Acc_K = acc(\mathcal{M}_K, \mathcal{D})$.

1.4.3 Mean Incremental Accuracy

The average incremental accuracy, denoted here as $\overline{Acc}$, offers a commonly utilized evaluation technique in the realm of CIL. The calculation of $\overline{Acc}$ is as follows:

$$\overline{Acc} = \frac{1}{K-1} \sum_{k=2}^{K} acc(\mathcal{M}_k, \bigcup_{i=1}^{k} \mathcal{D}_i) \tag{1.3}$$

In the above formula, $acc(\mathcal{M}, \mathcal{D})$ denotes the accuracy of the model $\mathcal{M}$ on the dataset $\mathcal{D}$. This metric does not factor in the accuracy of the initial model.

In some cases [Pet+23a], the mean incremental accuracy includes the first, non-incremental, state. It is then defined as follows:

$$\overline{Acc} = \frac{1}{K}\sum_{k=1}^{K} acc(\mathcal{M}_k, \bigcup_{i=1}^{k} \mathcal{D}_i) \tag{1.4}$$

1.4.4 Mean Forgetting

The metric of average forgetting, represented as F, can be computed as:

$$F = b \times f(\mathcal{D}_1) + \frac{1-b}{K-1}\sum_{k=2}^{K} f(\mathcal{D}_k) \tag{1.5}$$

where b denotes the proportion of the initial data subset $\mathcal{D}_1$ in the dataset $\mathcal{D}$, i.e., $b = |\mathcal{D}_1|/|\mathcal{D}|$. Here, $f(\mathcal{D}_k) = \max_{k' \in [\![k,K]\!]}(acc(\mathcal{M}_{k'}, \mathcal{D}_k) - acc(\mathcal{M}_K, \mathcal{D}_k))$ represents the difference between the highest performance attained on the data subset $\mathcal{D}_k$ during the CIL process and the final performance of the model on the same data subset [Mir+22].

In the $\overline{Acc}$ metric, more weight is attributed to earlier classes since the model is evaluated on all classes encountered up to each step. This implies that a high $\overline{Acc}$ does not necessarily assure superior performance in more recent classes, especially when a substantial number of classes are initially learned. The forgetting metric serves as a complement to accuracy, emphasizing model stability. A lower value for F signifies that the model's performance for a particular class remains relatively stable throughout the incremental process.

1.5 Training Procedures

Now that we set our objectives, we will delve into the details of our training procedures, emphasizing the aspects that distinguish the usual, non-continual, approach from the continual one.

1.5.1 Classical Training

The classical model training process typically consists of the following steps:

1. **Model Initialization:** The model parameters, including weights and biases, are typically initialized randomly or with predetermined values. These parameters determine the initial behavior of the model.

2. **Forward Propagation:** The model receives input data and processes them through its layers (made up of neurons) to generate predictions. Each neuron applies a mathematical transformation to its input (usually a weighted sum followed by an activation function). The output of the model represents predicted probabilities for each possible class.

3. **Loss Calculation:** After generating the predicted probabilities, the model calculates the loss by comparing these predictions to the true labels or target values. The goal is to minimize this loss by encouraging the model to assign high probabilities to correct classes and lower probabilities to incorrect ones.

4. **Backpropagation:** Once the loss is calculated, the model uses backpropagation to update its parameters. Backpropagation calculates the loss gradients associated with each model parameter. These gradients provide the direction and magnitude of parameter updates needed to minimize the loss. The chain rule helps to efficiently backpropagate the gradients through the layers of the model.

5. **Parameter Update:** The gradients obtained by backpropagation are used to update the model parameters. An optimization algorithm such as Stochastic Gradient Descent (SGD) or its variants is used. The update rule adjusts the parameters to reduce the loss. The learning rate, which controls the step size of the parameter update, is crucial for the convergence and stability of the training process.

6. **Iteration:** Steps 2-5 are repeated iteratively, either for a specified number of epochs or until a convergence criterion is met. Each iteration processes a mini-batch of training data, a subset of the total training set. This mini-batch training allows for computationally efficient parameter updates.

7. **Validation and Early Stopping:** During training, the performance of the model is periodically evaluated against a validation set. Validation metrics, such as accuracy or loss, help monitor the model's generalization to unseen data and potential overfitting. If performance begins to decline on the validation set, early termination can be used to avoid unnecessary iterations.

8. **Model Evaluation:** Once the training process is complete, the model's performance is evaluated on a separate test set. This evaluation provides insight into the model's generalization ability and predictive accuracy for real-world scenarios.

1.5.2 Class Incremental Training

Class-incremental training is similar to classical training but requires adaptation to accommodate the incremental nature of the learning process. The main differences between the two procedures are:

1. **Model Initialization:** Similar to classical training, but in Class-Incremental Learning, the model is usually initialized with the final classifier of the size of the number of classes in the first task.
2. **Forward Propagation:** The forward propagation step is identical to classical training, but in Class-Incremental Learning, only the data of the classes of the current task are fed into the model.
3. **Loss Calculation:** Since the model is only exposed to images from the classes of the current task, the loss is calculated only for those classes, making it agnostic to previous states. To address this, several methods are proposed in the chapter 2.
4. **Backpropagation:** This step is identical to classical training.
5. **Parameter Update:** Also identical to classical training. However, in Class-Incremental Learning, the gradients are computed only for the classes of the current task. Hyperparameters must be tuned to avoid erasing prior knowledge.
6. **Iteration:** This step is the same as in classical training.
7. **Validation and Early Stopping:** The procedures for validation and early termination are the same as in classical training, but the validation set must consist of the classes of the current task.
8. **Model Evaluation:** This step is identical to the classical training. However, the test set must include all classes from each task.

Figure 1.8 illustrates the process of training, and highlights the main differences we face in Class-Incremental Learning.

1.6 Examplar-Free Class-Incremental Learning

As introduced in subsection 1.3, one of the main challenges of incremental learning is mitigating the phenomenon of catastrophic forgetting. This problem arises when an agent

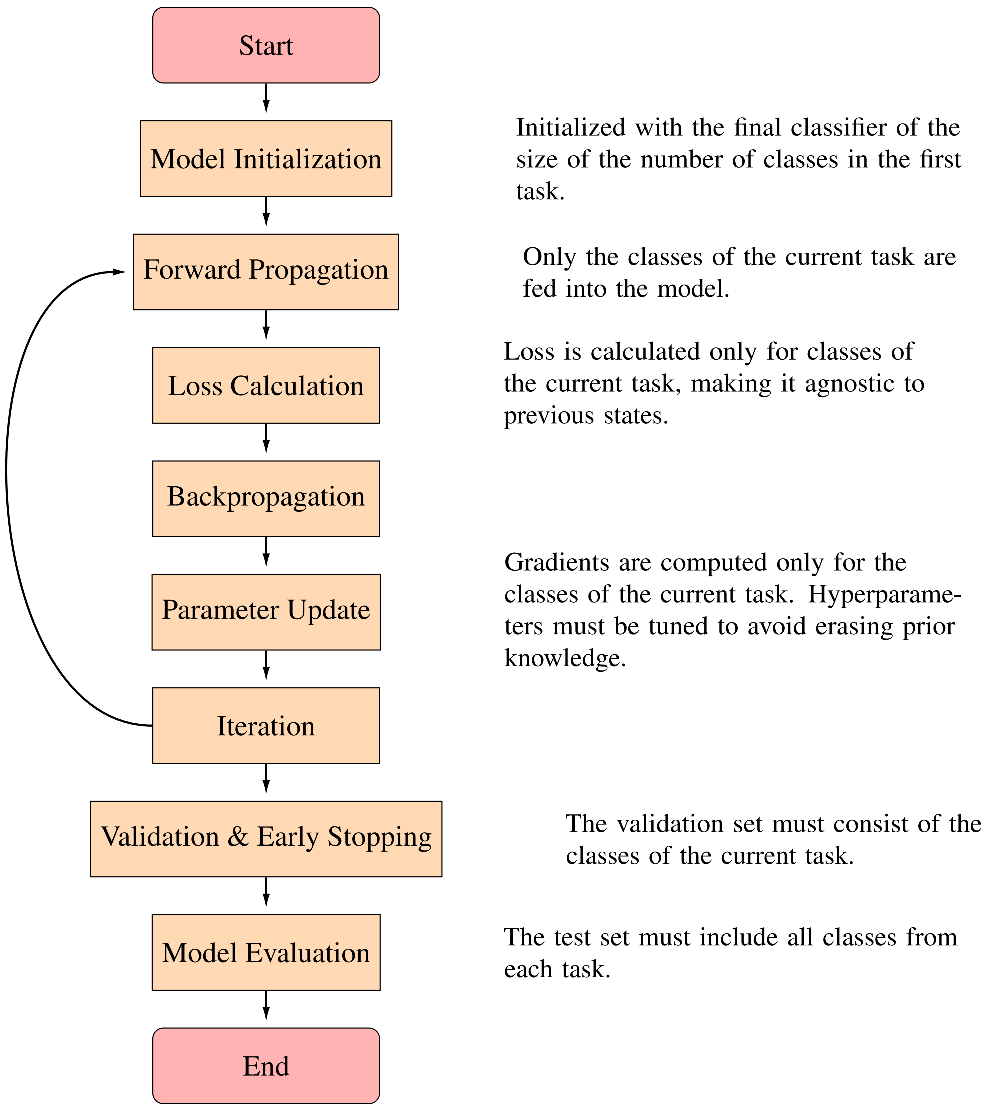

Start
Model Initialization
Initialized with the final classifier of the size of the number of classes in the first task.
Forward Propagation
Only the classes of the current task are fed into the model.
Loss Calculation
Loss is calculated only for classes of the current task, making it agnostic to previous states.
Backpropagation
Parameter Update
Gradients are computed only for the classes of the current task. Hyperparameters must be tuned to avoid erasing prior knowledge.
Iteration
Validation & Early Stopping
The validation set must consist of the classes of the current task.
Model Evaluation
The test set must include all classes from each task.
End

Fig. 1.8.: Flowchart of the Classical Training Procedure for Machine Learning Models and description of the specificities of the CIL training.

loses previously acquired knowledge as it assimilates new information [MC89; Fre99; Kem+18]. A critical goal of incremental learning is to strike a balance between plasticity, which allows the agent to adapt to new information, and stability, which facilitates the retention of previously acquired knowledge [MBB13; Cha+18].

To combat catastrophic forgetting, detailed in Section 2.5, many incremental learning methods resort to using a buffer that stores past data samples for replay during training [Mas+21; BPK21]. By replaying these past samples, the agent can better preserve its previously acquired knowledge, making the incremental learning process more akin to unbalanced learning [BPK21]. However, the assumption that past samples are always available limits the applicability of incremental learning. Here are the main reasons that justify this paradigm:

- **Resource constraints:** Storing and managing a large buffer of past data samples can be computationally expensive and memory intensive. In resource-constrained environments or on devices with limited storage capacity, it may not be feasible to retain and access all past samples during the incremental learning process.

- **Privacy and security:** Certain applications, especially those involving sensitive data such as healthcare or finance, may have strict privacy and security regulations that prevent the storage and replay of past samples. Storing and managing historical data can raise concerns about data breaches or unauthorized access.

- **Data variability:** As the incremental learning process continues, the model may encounter a wide range of data types, distributions, and concepts. Some past examples may no longer be relevant or representative of the current task, reducing the effectiveness of knowledge retention through replay.

- **Real-time data streams:** In some real-world scenarios, data is streamed in real-time, and the agent must learn from the most recent information without the luxury of replaying past samples. For example, in online learning or robotics, the agent must adapt to the latest data it receives without dwelling on historical data.

To address this limitation, there has been a burgeoning research effort dedicated to Examplar-Free Class-Incremental Learning (EFCIL) [BPK21; Mas+21]. In this approach, an agent learns to classify new data incrementally without relying on examples of previously seen classes. This setup presents a particularly challenging plasticity/stability dilemma, as the agent must learn from new data while avoiding catastrophic forgetting of previously acquired knowledge. EFCIL is an active area of research, with recent studies showing promising results [BPK21; Mas+21].

In the field of EFCIL, three main types of approaches can be distinguished:

- **Model-Growth Methods:** These methods expand the model's capacity to accommodate new classes. This approach allows the model to learn new classes without too much interference with previously learned classes. However, this approach is not scalable, as the model's size increases with each new task.

- **Finetuning-based approaches:** Knowledge distillation methods are often used in EFCIL. In these approaches, the model is trained to emulate its own previous results on the new data, in addition to learning the new tasks. The purpose of this method is to ensure that the model does not forget its previously learned decision boundaries. It usually does favor plasticity.

- **Fixed Model Methods:** These approaches maintain a fixed part of the model, such as the feature extractor, while allowing other parts to be updated. The fixed part of the model encapsulates knowledge from previous tasks and is not affected by new data, mitigating catastrophic forgetting, and favoring stability.

Each of these approaches provides unique solutions to the challenges of EFCIL, addressing the trade-off between learning new classes and retaining knowledge of old ones. These methods continue to be an active area of research in the Machine Learning community.

In real-world scenarios, the challenges of EFCIL often arise in applications where resource constraints and the inability to store large amounts of historical data are paramount. For example, in Internet of Things (IoT) environments, resource-constrained devices with limited memory and processing power must continuously adapt to new sensor data streams while retaining knowledge of previously encountered classes. Similarly, microchips embedded in various devices and systems, such as medical instruments or autonomous vehicles, must learn and adapt to new tasks without the luxury of storing large amounts of historical data. These cases underscore the critical need for efficient and effective EFCIL solutions to ensure the continuous evolution and adaptability of intelligent systems in resource-constrained, data-rich environments.

1.7 Contributions overview

This section explains the main contributions of this thesis, which primarily focuses on addressing the plasticity/stability dilemma in Class-Incremental Learning (CIL). This thesis presents two novel methodologies - Plastic and Stable Memory-Free Class-Incremental Learning (PlaStIL)[1] and Feature Translation for Exemplar-Free Class-Incremental Learn-

[1] **Grégoire Petit**, Adrian Popescu, Eden Belouadah, David Picard, and Bertrand Delezoide. „PlaStIL: Plastic and Stable Memory-Free Class-Incremental Learning". In: *Proceedings of The 2nd Conference on Lifelong Learning Agents*. Ed. by Sarath Chandar, Razvan Pascanu, and Doina Precup. Proceedings of Machine Learning Research. PMLR, 2023

ing (FeTrIL)[2]. It also provides a comprehensive analysis of initial training strategies for exemplar-free Class-Incremental Learning[3].

1.7.1 PlaStIL: Plastic and Stable Memory-Free Class-Incremental Learning

In Chapter 3 we present an approach to the plasticity-stability dilemma, introduced in Section 1.6, which requires a balance between plasticity (the ability to learn new data) and stability (the ability to retain previously learned knowledge). Traditionally, the compromise between plasticity and stability is achieved by using a memory buffer, detailed in Section 2.5 or by storing two deep models, detailed in Section 2.3. The latter method involves integrating new classes through fine-tuning, coupled with knowledge distillation from the previous incremental state. However, this approach can be inefficient because it requires the storage and computation of two separate deep models. This work proposes a solution that uses a similar number of parameters as distillation-based methods, detailed in Subsection 2.3, but distributes them differently. This distribution aims to achieve a better balance between plasticity and stability. Specifically, the proposed method freezes the feature extractor after the initial state and embeds several model tops to ensure high plasticity. This is a technique inspired by transfer-based incremental methods. By freezing the feature extractor, the model provides stability by training the classes from the oldest incremental states with this unchanging extractor. The model uses partially fine-tuned models to introduce plasticity to new classes. A specially designed plasticity layer is introduced that can be incorporated into any transfer-based method designed for exemplar-free incremental learning. The method is validated by applying it to two existing transfer-based incremental learning methods and evaluating its performance on three large datasets. The results show that the proposed method outperforms the existing methods in all tested configurations, demonstrating improved performance and effectiveness in class incremental learning.

1.7.2 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning

In Chapter 4 we address mainly the same plasticity-stability dilemma, introduced in 1.6 as PlaStIL does, as tackled in Subsection 1.7.1. Existing methods typically prioritize either stability, by using a fixed feature extractor after the initial incremental state, or plasticity, by successive fine-tuning of the model. In this section, we propose a novel method that

[2]**Grégoire Petit**, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. „FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2023, pp. 3911–3920

[3]**Grégoire Petit**, Michael Soumm, Feillet Eva, Adrian Popescu, David Picard, and Bertrand Delezoide. „An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024

combines the best of both approaches to improve the balance between stability and plasticity. The approach introduces a fixed feature extractor coupled with a pseudo-feature generator, an effective, though simple, component that uses the geometric translation of new class features to create representations of past classes. The key strength of this generator is its simplicity and effectiveness; it doesn't need to store exemplars of past classes, but only the centroid representations of those classes. Using these centroids, the generator creates pseudo-features for past classes, which are combined with actual features of new classes. These pseudo-features and actual features are then fed into a linear classifier that is incrementally trained to discriminate between all classes. The proposed method is computationally efficient because it updates a minimal component of the deep model, the linear classifier, during the incremental process, making it faster than mainstream methods that update the entire deep model. The method is validated through experiments on three challenging datasets in different incremental settings. Compared to ten existing methods, the FeTrIL approach showed superior performance in most cases, demonstrating its effectiveness in exemplar-free Class-Incremental Learning.

1.7.3 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning

In Chapter 5, we delve into the central dilemma in EFCIL: effectively integrating new classes into the model while mitigating the risk of catastrophic forgetting, particularly in scenarios where past data cannot be stored. Existing methods tend to focus on the target dataset of the EFCIL process, but recent trends suggest that models pre-trained in a self-supervised manner on large datasets can also be effective, detailed in Section 2.4.

The study points out that the initial model of the EFCIL process can be built in two ways: using only the first batch of the target dataset, or incorporating pre-trained weights from an auxiliary dataset. This choice of initial training strategy can significantly affect the performance of the incremental learning model, yet this aspect hasn't been thoroughly studied. This section also points out that performance is influenced by factors such as the EFCIL algorithm used, the neural architecture, the nature of the target task, the distribution of classes in the data stream, and the number of examples available for learning. To gain insight into these factors, the work conducted an extensive experimental study and proposed a statistical analysis framework. This framework aims to quantify the relative contribution of each factor to incremental performance. The main finding of the study is that the initial training strategy has a significant impact on the average incremental accuracy. However, the choice of EFCIL algorithm plays a more critical role in preventing forgetting. Based on its findings, Chapter 5 provides in Section 5.6 practical recommendations for choosing the right initial training strategy for different incremental learning use cases. These suggestions are designed to facilitate the practical application and deployment of incremental learning, which is the main interest of this study. By providing a comprehensive analysis of influencing factors and practical recommendations, the study makes a significant contribution to our understanding of class-based incremental learning and its optimization.

2 State of the Art in Exemplar-Free Class Incremental Learning

> *Do not go gentle into that good night*
>
> — **Dylan Thomas**
> (1914-1953)

Continual Learning is a subfield of Machine Learning that aims to enable a model to learn from a stream of data. It can be divided into several subfields, as presented in Figure 2.1:

- **Generic Incremental Learning** aims to learn from a stream of tasks, without any assumption on the domain. Each task is a set of classes, without any necessary coherence between the classes of the same task. *It is not the focus of this thesis.*

- **Domain-Incremental Learning** aims to learn from a stream of tasks, where each task is a different domain. At inference time, the model does not have access to the domain of the instance to classify. *It is not the focus of this thesis.*

- **Task-Incremental Learning** aims to learn from a stream of tasks, where each task is a different domain. At inference time, the model has access to the task id of the instance to classify. *It is not the focus of this thesis.*

- **Class-Incremental Learning** aims to learn from a stream of tasks, without any assumption on the domain. Each task is a set of classes, without any necessary coherence between the classes of the same task. At inference time, the model does not have access to the domain of the instance to classify. *It is the focus of this thesis.* **One-Class Incremental Learning** is a subfield of Class-Incremental Learning that aims to learn from a stream of instances, but with the constraint that the model can only learn one class per state, without any prior on the label space.

To solve Class-Incremental Learning problems, researchers have proposed many approaches. They can be divided into two main categories: *Exemplar-Based* and *Exemplar-Free*. Exemplar-Based methods store a subset of the training data, called exemplars, and use them to mitigate the forgetting of the model. Exemplar-Free methods do not store any exemplar, and use other techniques to manage a good stability-plasticity trade-off.

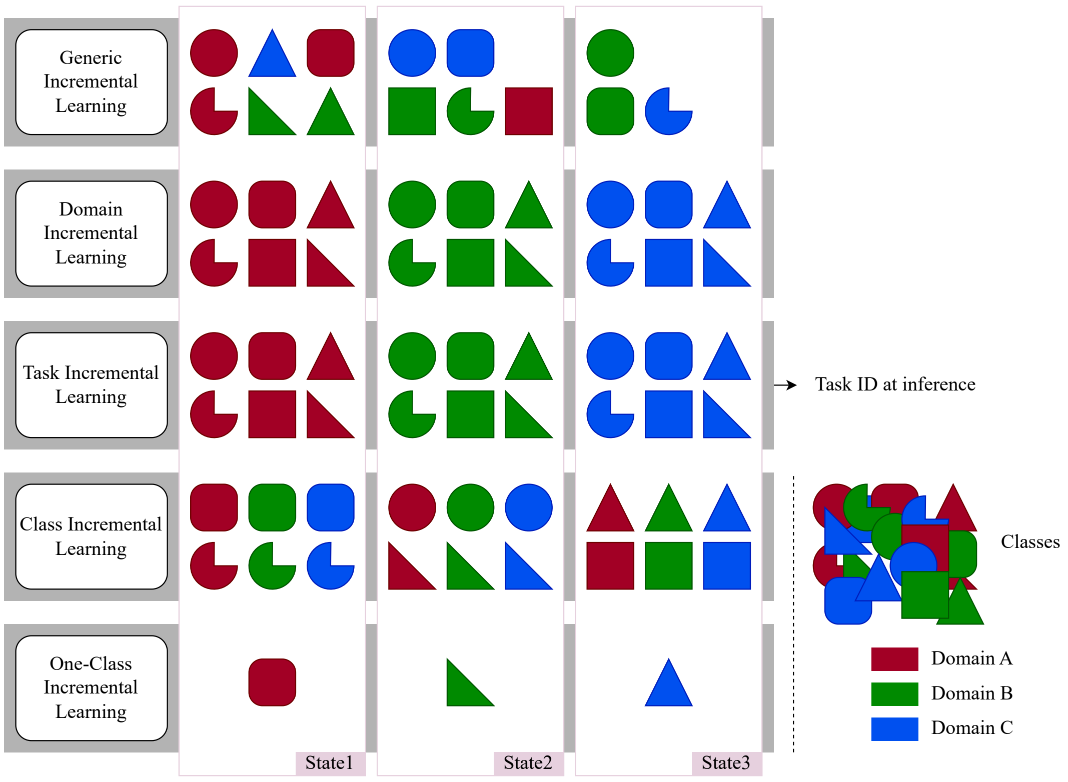

Generic Incremental Learning
Domain Incremental Learning
Task Incremental Learning
Class Incremental Learning
One-Class Incremental Learning
State1
State2
State3
Task ID at inference
Classes
Domain A
Domain B
Domain C

Fig. 2.1.: Comparative illustration showing the unique characteristics of Generic Incremental Learning, Domain Incremental Learning, Task Incremental Learning, and (One-)Class Incremental Learning. Each of these learning approaches is shown with its specific context and learning paradigm, providing a clear distinction in terms of task adaptability, data domain change, class integration, and generic incremental adaptation.

More than the plasticity-stability tradeoff, there is 6 main criteria that can be used to evaluate Continual Learning methods:

- **Complexity:** Incremental learning refers to the ability of a model to incorporate new information or classes without making significant changes to its existing structure. This is important to avoid catastrophic forgetting, where new information overrides or erases previously learned knowledge. A good incremental learning system should be able to adjust its internal representations or model parameters to accommodate new information without significantly disrupting existing knowledge.

- **Memory:** Memory in incremental learning refers to the ability of the system to retain information about past classes. Some incremental learning methods use external memory (a buffer or storage) to store exemplars or samples from previous classes, allowing the model to remember and adapt its knowledge. The model can then use these exemplars to prevent forgetting and improve performance on old classes when new classes are introduced.

- **Accuracy:** The accuracy of an incremental learning system is critical. As new classes are added, the system should strive to achieve a level of performance comparable to what would be achieved if all classes were learned simultaneously using standard methods. Maintaining high accuracy is challenging because of the potential for interference between old and new classes. Effective methods should minimize such interference and ensure that the accuracy of both old and new classes remains satisfactory.

- **Timeliness:** Timeliness refers to the delay required for new data to be integrated into incremental models. Rapid integration is desirable to avoid falling behind evolving data distributions. A good incremental learning system should be able to quickly adapt to new data while maintaining a stable representation of previously learned information.

- **Plasticity-Stability:** Plasticity refers to the model's ability to adapt and learn new information, while stability refers to its ability to retain previously learned information. Finding the right balance between plasticity and stability is critical for incremental learning. A system that's too plastic can quickly adapt to new classes but risks forgetting old ones, while a system that's too stable may resist change and struggle to incorporate new knowledge.

- **Scalability:** Scalability is the ability of incremental learning methods to handle large numbers of classes. As the number of classes increases, the challenges of interference and forgetting become more pronounced. Effective methods should show good performance even when dealing with a large number of classes, without significant degradation in accuracy or efficiency.

Tables 2.1 and 2.2 present the 3 main types of methods and how well they perform in terms of these factors. We can see that although the objective is quite clear, depending on the type of method, no one reaches perfection in terms of these 6 criteria.

We can divide the Exemplar-Free methods into three main categories: *Model-Growth based*, *Finetuning based*, and *Fixed-Model based*. We will now present each of these categories, and how they rank in terms of criteria presented in Section 1.3.

2.1 Model-Growth based Incremental Learning

On the 6 criteria presented above, Model-Growth based methods rank as follows:

	Model-Growth based	Fixed-Representation based	Fine-Tuning based
Complexity	🟥 The model grows by integrating new parameters and connecting them via weights, or by adding small networks to incorporate new knowledge. The key challenge is to optimize this model expansion to improve performance.	🟩 After the initial non-incremental phase, the model remains fixed. In a simple setting, only the parameters needed for new class weights are added. In a more sophisticated setting, additional parameters are introduced to improve the performance of previous classes.	🟩 This category of incremental learning techniques operates with a fixed backbone structure. The number of parameters is minimally affected by adjustments to the classification layer to balance between past and new classes.
Memory	🟧 Model growth allows these methods to be used without the need for exemplar memory. Instead of storing raw data from past classes, memory is dedicated to additional model parameters and weights, providing a more efficient way to retain information about past classes.	🟩 Fixed representations don't update the model during the incremental learning process and then have minimal reliance on past class memory. Class weights once learned during their first encounter, can be used consistently in all future incremental stages.	🟧 The effectiveness of these methods depends heavily on the size of the past memory. However, the goal of EFCIL is at odds with storing past exemplars. By using knowledge distillation or by exploiting statistical properties of past states, memory requirements can be minimized.
Accuracy	🟧 Performance is tied to the amount of model growth allowed. When growth is limited, model growth-based methods underperform compared to fine-tuning-based methods. However, when significant growth is allowed, performance approaches that of traditional learning, although this contradicts the need of EFCIL to keep model complexity relatively constant.	🟧 Fixed representation methods typically offer lower accuracy than fine-tuning approaches due to limited initial data. Yet, with a robust pseudo-feature generator, they can achieve superior results. Performance benefits from training the initial model on a large dataset, contingent upon dataset availability. Over time, fixed representation methods exhibit greater stability and resistance to forgetting compared to fine-tuning and model growth strategies.	🟧 Recent methods have significantly improved performance over previous work. These gains are achieved by advanced knowledge distillation definitions, framing incremental learning as an unbalanced learning problem, or a combination of both. The performance gap with classical machine learning cannot be reduced enough because no memory of past classes is allowed in EFCIL.

Tab. 2.1.: Sensibilities of Model-Growth based, Fixed-Representation based and Fine-Tuning based methods over complexity, memory, and accuracy. Green means that the type of method is well suited for the criterion, orange means that it is not well suited, and red means that it is not suited at all.

	Model-Growth based	Fixed-Representation based	Fine-Tuning based
Timeliness	Because retraining is required for each incremental update, the complexity of model growth is typically similar to that of fine-tuning-based methods.	Since training is only required for the classifier weight layer, fast integration of new knowledge is possible.	New classes cannot be classified until retraining has completed their incorporation into the model. For time-critical applications, the training process can be accelerated, but the optimality of the results may be sacrificed.
Plasticity-Stability	Model-Growth-based methods are intentionally designed to handle different visual tasks. The key challenge is to minimize the number of additional parameters needed for each new task.	The plasticity of the model is limited because the representation is fixed in the initial state after learning. Performance can decrease significantly if the incremental tasks change drastically and the initial representation is no longer transferable. Though the stability is highly favored	Model updates allow adaptation to new data flowing into the system. Because no memory is allocated in EFCIL, the plasticity becomes too significant, and this shift is managed by knowledge distillation or imbalance handling.
Scalability	These methods scale well with new classes or tasks, provided that the systems on which they are deployed have sufficient resources to support the associated model growth during the training and inference phases, as well as for storage.	Fixed representation-based methods have limited dependence on bounded memory and can accommodate a large number of classes. This is possible because class weights are learned initially and reused later.	Finetuning based method can scale well with new classes or tasks, provided that the systems on which they are deployed have sufficient resources to support the operations associated with the training.

Tab. 2.2.: Sensibilities of Model-Growth based, Fixed-Representation based and Fine-Tuning based methods over timeliness, plasticity, and scalability. Green means that the type of method is well suited for the criterion, orange means that it is not well suited, and red means that it is not suited at all.

- **Complexity:** Because the model grows with each new task, the complexity of the model increases with the number of tasks. This is not ideal for long-term learning scenarios, where the model can become too large and resource-intensive.

- **Memory:** Model-Growth based methods do not need the use of external memory to store exemplars or samples from previous classes. Instead, they rely on the model's internal structure to retain knowledge of past classes. This can be problematic in long-term learning scenarios, where the model's capacity to retain knowledge is limited.

- **Accuracy:** Model-Growth based method's accuracy highly depends on the model's ability to grow with each new task. If the model is not able to grow enough, the accuracy will generally be low. If the model is able to grow enough, the accuracy can be high.

- **Timeliness:** Retraining the model with each new task is time-consuming, especially because the model's size increases with each new task. This can be problematic in long-term learning scenarios, where the model can become too large and resource-intensive.

- **Plasticity-Stability:** Model-Growth based methods are generally stable and plastic because the model grows with each new task. However, the model's plasticity and stability can be limited by the model's capacity to grow with each new task.

- **Scalability:** Model-Growth based methods are as scalable as the model's ability to grow with each new task. If the model is not able to grow enough, the scalability will be low. If the model is able to grow enough, the scalability can be high.

Incremental learning and network growth have become critical areas of focus in the quest for more flexible and adaptive deep learning models. Various methods have been proposed in the literature to increase the representational capacity of a model by adding complexity or modifying the network architecture.

In the first stages of incremental learning, Neural Gas (NG) networks [MBS93] and their growing variant [Fri94] were noteworthy. These are related to Self-Organizing Maps (SOMs) and have been widely used for incremental learning tasks. PROjection-PREdiction (PROPRE) [GK17], an incremental learner based on NG and SOMs, implements an additional supervised readout layer and a concept drift detection mechanism, thereby improving its adaptability to changing data distributions.

Neural Gas with local Principal Component Analysis (NGPCA) [AAP10] focuses on online incremental learning, especially for robotic platforms performing object manipulation tasks.

Similarly, Dynamic Online Growing Neural Gas (DYNG) [BC13] controls the growth rate of the NG network to accelerate learning for new knowledge while slowing growth for previously learned knowledge.

The TOpology-Preserving knowledge InCrementer (TOPIC) [Tao+20b] adapts neural gas for class incremental learning, with a special focus on visual datasets and few-shot learning. Similarly, a topology-preserving network called TPCIL was introduced in [Tao+20a]. TPCIL fights catastrophic forgetting by modeling the feature space using an Elastic Hebbian Graph and preserving the topology using a topology-preserving loss.

Wang et al [WRH17] proposed a unique method called "Growing a Brain", which increases the representational capacity by modifying the structure of the network, either by widening (adding more neurons to a layer) or deepening (adding more layers) the network. This flexibility allows the model to adapt to increasingly complex data sets or tasks.

Similarly, Progressive Neural Networks (PNNs) [Rus+16] use knowledge from past tasks and retain it by using multiple models during the training process. This design forms a "progressive" structure in which each new task directly benefits from the retained knowledge of previous tasks, thereby reducing catastrophic forgetting.

Roy et al. [RPR20] introduced an adaptive network that operates in a tree-like growth pattern. This network expands based on the reorganization of the feature hierarchy whenever new tasks are introduced, allowing for a dynamic and responsive increase in representational capacity.

Aljundi et al. [ACT17] presented a lifelong learning architecture based on a network of specialized subnetworks (experts) coupled with a gating mechanism. This gating mechanism determines which expert to engage for knowledge transfer based on the given task, thereby maximizing task-specific learning and performance.

Deep Adaptation Networks (DANs) [RT17] also contribute to the Model-Growth based Class-Incremental Learning literature by introducing additional parameters for each new task. This approach significantly expands the architecture as new tasks arrive, which is particularly useful when a model encounters a large number of novel tasks.

An alternative method [RBV18] builds on the idea of shared parameters across multiple neural networks. It uses modular adapters to connect these networks, allowing each to specialize for a particular task, thereby increasing the overall adaptability and efficiency of the model.

Several techniques, including PackNet [ML18], Uncertainty-guided Continual Learning (UCL) [Ahn+19], Continual Learning through Neuronal Plasticity (CLNP) [GKC19], AGS-

CL [Jun+20], and Network Importance Sampling and Pruning Algorithm (NISPA) [Mus], identify the most important neurons or parameters for the current task. They then free the less important parts for use in subsequent tasks. Liberation is achieved by various strategies such as iterative pruning [ML18], activation value analysis [GKC19; Mus; Jun+20], or uncertainty estimation [Ahn+19]. However, these approaches have limitations in handling large numbers of tasks due to performance loss at high compression.

In response, Piggyback [MDL18] extends PackNet by implementing network quantization and suggesting masks for individual weights, allowing a larger number of tasks to be trained with a single base network. In the same vein, Hard Attention to the Task (HAT) [Ser+18], SupSup [Wor+20], MEAT [Xue+22], WSN [Kan+22], and H^2 [JK22] use optimization of a binary mask to designate specific neurons or parameters for each task. The masked regions of old tasks are then frozen, preventing their modification and preserving the learned knowledge.

Another approach, Memory Aware Synapses (MAS) [Alj+18], uses a mechanism to identify the most important weights in the model based on the sensitivity of the output function. This approach was later adapted for use with unlabeled data sets [AKT19], demonstrating its versatility and adaptability.

Methods such as Dynamically Expandable Networks (DEN) [Yoo+18], Co-Participation Geometric (CPG) [Hun+19], and Dual-Memory Generative Memory Networks (DGM) [Ost+19] have introduced a dynamic element into network architectures. These methods expand the network architecture when the capacity is insufficient to learn a new task. This dynamic expansion is intended to alleviate the limitations imposed by fixed network capacity and to provide the system with a flexible learning structure.

The dynamic architecture can be further optimized to increase parameter efficiency and facilitate knowledge transfer. Techniques such as Reinforcement Continual Learning (RCL) [XZ18], Balanced Neural Structure (BNS) [Qin+21], Learn to Grow (LtG) [Li+19], and Bayesian Structure Adaptation (BSA) [KCR21] employ strategies such as reinforcement learning, architecture search, and variational Bayes to optimize the architecture for incremental learning.

Despite the potential benefits of dynamically expanding the network, it is critical to ensure scalability by applying sparsity and parameter reusability constraints, as explained in Subsection 1.3. Since network expansion should occur at a rate slower than task growth, these constraints prevent excessive network growth and promote efficient use of parameters.

In summary, recent advances in network growth and incremental learning have shown promising results in improving the adaptability and performance of deep learning models.

These diverse approaches provide the foundation for further development of sophisticated methods to handle dynamically changing tasks and datasets.

Consequently, we can recap the main advantages and limitations of Model-Growth-based methods as follows:

- **Advantages:**
  - Designed to handle different visual tasks: Model-Growth based methods are intentionally designed to accommodate various tasks, making them versatile in handling incremental learning scenarios with diverse classes.
  - Scalability: These methods can scale well with new classes or tasks, provided that sufficient resources are available for model growth during training and inference.
  - Minimal Catastrophic Forgetting: As the model grows with new classes, it can retain the knowledge of past classes, mitigating catastrophic forgetting to some extent.
- **Limitations:**
  - Memory and Computational Demands: Model-Growth-based methods require substantial memory and computational resources as the model grows with each new class, making them computationally expensive in long-term learning scenarios and not adapted for EFCIL scenarios.
  - Need for Parameter Control: To prevent the model from becoming too large or resource-intensive, careful control of the number of additional parameters for each new task is essential.

2.2 Fixed-Representation-Based Incremental Learning

Another branch of research in incremental learning is fixed representation methods. These methods typically use a deep model trained in the first, non-incremental, state to extract features that are then used for incremental learning. However, once trained, the deep representation is kept fixed and is not updated for each incremental step.

On the 6 criteria presented above, Fixed-Representation based methods rank as follows:

- **Complexity:** Fixed-Representation based methods have really low complexity because the model is not updated during the incremental learning process.
- **Memory:** Fixed-Representation based methods have a low memory dependency.
- **Accuracy:** Fixed-Representation based methods' accuracy highly depends on the quality of the initial representation. If the initial representation is good, the accuracy will generally be high. If the initial representation is not good, the accuracy can be low.
- **Timeliness:** Fixed-Representation based methods are really fast because the model is not updated during the incremental learning process.
- **Plasticity-Stability:** Fixed-Representation based methods are generally stable because the model is not updated during the incremental learning process. However, the model's plasticity can be limited by the quality of the initial representation.
- **Scalability:** Fixed-Representation based methods' scalability heavily depends on the fixed representation.

A basic variant of this approach is briefly described in [Reb+17], where a Nearest Class Mean classifier is directly applied on top of the fixed representation of the model trained on the initial state classes. Despite its simplicity, this approach often yields suboptimal results due to inefficient use of the learned deep representations. Recent studies have proposed the use of a substantial pre-trained model paired with a k-NN classifier, forming a robust baseline for continual learning algorithms [Jan+22; Pel22].

Deep Shallow Incremental Learning (DeeSIL) [BP18] provides a more effective alternative by applying a transfer learning scheme [KSL18; Raz+14]. DeeSIL uses a fixed deep representation to learn the initial set of classes and then applies support vector machines (SVMs) [BGV92] to incrementally learn new classes. This hybrid method ensures that the deep representation is used effectively, and the classifier can be incrementally updated without retraining the entire model.

FearNet [KK18] provides a unique, biologically inspired approach to fixed representation learning. FearNet uses separate networks for long-term and short-term memory to represent past and novel classes. A decision mechanism is implemented to decide which of the two networks should be used for each test example, providing a flexible way to access past and newly learned knowledge.

Deep Streaming Linear Discriminant Analysis (DSLDA) [HK20] is another online approach based on the SLDA algorithm [POK05]. In this method, the network is trained on the initial batch of classes and then frozen. During subsequent training stages, a class-specific running mean vector and a common covariance matrix are continuously updated, allowing incremental learning while preserving the initial deep representation.

REplay using Memory INDexing (REMIND) [Hay+20] presents a methodology inspired by the hippocampal indexing theory. REMIND uses an initial representation that is only partially updated afterward. It uses a vector quantization technique to store compressed intermediate representations of images. These representations are later used for memory consolidation, effectively balancing the demands of storage and retrieval.

In the study by Gallardo et al. [GHK20], the authors investigate the use of a pre-trained, fixed feature extractor in a self-supervised manner. An innovative approach in this area is represented by Learning to Prompt for Continual Learning (L2P) [Wan+22b]. This method focuses on training a concise memory system that efficiently manages both task-invariant and task-specific knowledge.

In summary, fixed representation methods provide valuable insights into how deep learning models can be adapted for incremental learning tasks. While there are challenges to be overcome, particularly in terms of effectively exploiting deep representations and managing memory resources, the methods described above provide promising avenues for future research in this area.

Consequently, we can recap the main advantages and limitations of Fixed-Representation-Based methods as follows:

- **Advantages:**
  - Scalability: Fixed-Representation-based methods can accommodate a large number of classes without linearly increasing memory requirements, making them more scalable in long-term learning scenarios.
  - Stability: Fixed-Representation-based methods are more stable as the representation is fixed and does not change during training of new tasks.
  - Reduced Computational Demands: Since the representation is fixed, the computational overhead associated with model growth is avoided during training and inference.
- **Limitations:**
  - Limited Plasticity: Fixed-Representation-based methods have limited plasticity due to the fixed representation in the initial state after learning. Performance

can decrease significantly if incremental tasks change drastically, and the initial representation is no longer transferable.

  – Update Challenges: Incorporating new knowledge into the fixed representation can be challenging since the representation is not updated during training of new tasks.

2.3 Finetuning-Based Class-Incremental Learning

On the 6 criteria presented above, Finetuning-Based methods ranks as follows:

- **Complexity:** Finetuning-Based methods have a low complexity because the model is increasing slowly during the incremental learning process.
- **Memory:** Finetuning-Based models are better with a big memory because they can store more exemplars, and then use them to mitigate the forgetting of the model.
- **Accuracy:** Finetuning-Based methods' accuracy highly depends on the amount of memory available. If the memory is big enough, the accuracy will generally be high. However, in EFCIL scenarios, the memory is prohibitive, and the accuracy is generally lower than fixed-representation-based methods.
- **Timeliness:** Finetuning-Based methods' timeliness is not that good because the model needs to be retrained with each new task.
- **Plasticity-Stability:** Finetuning-Based methods are generally plastic because the model is updated during the incremental learning process as new tasks arrive. However, the model's stability can be limited by the amount of memory available: if the memory is not big enough, the model will forget the old classes, sacrificing stability for plasticity.
- **Scalability:** Finetuning-Based methods' scalability heavily depends on the amount of memory available.

2.3.1 Basic Concepts of Finetuning-Based Class-Incremental Learning

As explained in the Subsection 1.3, one main challenge in Class-Incremental Learning is the discrepancy between the states' distributions. Therefore, to mitigate the forgetting of the model, one common approach is to craft a loss that penalizes the drift between the latent representations of the old and new states. This type of method is called *Distillation-Based* methods.

Knowledge distillation approaches aim to preserve the network's ability to correctly identify previously learned classes by preventing activation drift [Cha+18; Hou+19]. These methods are primarily designed to preserve the network's ability to identify previously learned classes based on the fact that the distillation loss constrained the latent space of the current state to correctly classify previous classes. Importantly, most knowledge distillation-based methods, including those proposed by Li and Hoiem [LH16] and Dhar et al. [Dha+18], are designed to learn continuously from a stream of data instances sufficient for each new task.

2.3.2 Major Developments and Methods in Finetuning-Based Class-Incremental Learning

Among Distillation-Based methods, Learning without Forgetting (LwF) [LH16] is one of the first techniques that applies a loss function to penalize the differences between the current network's output and the output it had on previous tasks. This was originally proposed for image classification. iCaRL, proposed by Rebuffi et al. [Reb+17], extended LwF by introducing an exemplar set and a trade-off between old and new classes to balance the learning process.

Rannen et al. [Ran+17] extended LwF by introducing an under-complete autoencoder, which projects features to a manifold with fewer dimensions, allowing important features from previous tasks to be retained without increasing the model size. They also pointed out the limitations of LwF in handling different data distributions between different tasks. Belouadah et al. [BPK20] uses a vanilla FT backbone and tackles catastrophic forgetting by reusing the past classifiers learned when these classes were first learned, while Castro et al. [Cas+18] propose a similar approach, but with an end-to-end architecture. Simon et al. [SKH21] conducted knowledge distillation directly on low-dimensional manifolds.

The Learning without Memorizing (LwM) method [Dha+18], applies attention maps from previous tasks to guide the training of new tasks. It aims to ensure that important features for class labels remain consistent across tasks. The method introduces an attention distillation loss. In Bias Correction (BiC) proposed by Wu et al. [Wu+19], a dynamic trade-off term is introduced to account for the varying number of old and new classes, thus adjusting the balance of learning. Fini et al. proposed a two-stage method known as Batch-Level Distillation (BLD) [Fin+20]. Douillard et al. also approached the problem differently with Pooled Outputs Distillation (PODNet) [Dou+20], treating continual learning as a representation learning problem.

Co-transport for Class-Incremental Learning (COIL) [ZYZ21] uses the co-transport technique to exploit semantic relationships between old and new models. Kurmi et al. proposed a distillation loss using prediction uncertainty and self-attention [Kur+21].

2.3.3 Advanced Applications and Emerging Trends in Finetuning-Based Class-Incremental Learning

Knowledge distillation has been further applied to diverse areas such as semantic image segmentation [MZ19; Cer+20; Dou+21; MZ21], object detection [SSA17], class-conditional image generation [Wu+18; Zha+19], and person re-identification [Pu+21]. It has also been used in image and video captioning [Ngu+19], and in methods that use unlabeled data instances with labeled instances and a global distillation method [Lee+19].

Knowledge distillation also addresses challenges such as inconsistent causal effect [Hu+21], data imbalance [Hou+19], bias toward new tasks [Zha+20], concept drift [He+20a], and semantic distribution shift [Cer+20; Dou+21]. It has also been used for knowledge transfer at selected levels using the Expectation-Maximization (EM) method [LBE21] and in the Continual Learning with Forgetting Avoidance and Knowledge Transfer (CAT) approach [KLH20]. Prototype Augmentation and Self-Supervision for Incremental Learning (PASS) [Zhu+21b] uses prototypes of past classes in combination with distillation in order to counter catastrophic forgetting. Self-Supervised Models are Continual Learners [Fin+22] proposes a novel approach that transforms self-supervised loss functions into distillation mechanisms by introducing a prediction network that maps current representations to their past state, thereby significantly improving the quality of learned representations across different CL settings with minimal need for hyperparameter tuning.

Class-Incremental Learning via Dual Augmentation (IL2A) [Zhu+21a] uses class and semantic augmentation to manage representation and classifier biases. Lastly, the MUlti-Classifier (MUC) [Liu+20c] integrates an ensemble of auxiliary classifiers to provide more effective regularization constraints. Self-Sustaining Representation Expansion (SSRE) [Zhu+22] utilizes structure reorganization, main-branch distillation, and a prototype selection mechanism to optimize representation, retain old class features, and enhance discrimination between old and new classes, without the need to store old class samples. Balanced Softmax Cross-Entropy for Incremental Learning with and without Memory (BSIL) [JLM21] addresses catastrophic forgetting and model bias in deep neural networks by employing a balanced softmax cross-entropy that can be integrated with other state-of-the-art class-incremental learning approaches to improve accuracy and potentially reduce computational cost, even in EFCIL.

More recently, novel methods such as R-DFCIL [Gao+22] and MBP [Liu+22] have emerged. These encode structural information from the old model into the new, model relationships, and ensure consistent distance rankings between the old and new models.

Consequently, we can recap the main advantages and limitations of Finetuning-Based methods as follows:

- **Advantages:**
  - Plasticity: Fine-tuning enables the model to adapt to new data flowing into the system, allowing for some level of plasticity.
- **Limitations:**
  - Timeliness: Fine-tuning does not allow fast integration of new knowledge because it requires more than the training for the classifier weight layer, which can be critical in time.
  - Limited Stability: Fine-tuning is not stable because it can lead to catastrophic forgetting of the previous knowledge since no previous knowledge is preserved in EFCIL methods.

2.4 Pre-training Techniques for CIL

Pre-training techniques are a critical component of Machine Learning that uses models initially trained on a source dataset as the basis for training subsequent models on a target dataset [RM19]. This strategy, called transfer learning, typically involves starting the weights of the target model with those of the source model. These weights can either remain unchanged, except for the classification layer (*linear probing*), or be updated with the target data (*fine-tuning*).

Transfer learning has several pragmatic advantages, such as reducing the computational effort required to train a new model on a different dataset, and facilitating accurate model learning in few-shot scenarios due to the ability of pre-trained models to extract complicated features from novel input data [Tan+18]. Some scholars have investigated the pre-training of models to enhance their transferability [Gei+18; Tam+17; KSL18; Abn+21], concluding that a model's ability to generalize is enhanced by the volume, caliber, and diversity of its source training data [Oqu+23].

2.4.1 Self-Supervised Learning and Its Role in Pre-Training

Recently, Self-Supervised Learning (SSL) has attracted attention for its potential to generate diverse, reusable features for subsequent tasks [Dha+21]. SSL allows models to learn from unlabeled data without relying on explicit annotations [JT20]. By exploiting inherent structure or information in the data, surrogate labeling tasks such as predicting missing image patches, image rotations, or colorizations can be formulated. An example of this is MoCov3 [He+20b; CXH21], which uses a contrast loss function to obtain similar representations for two randomly enhanced portions of the same input image. Other SSL methods, such as BYOL [Gri+20] and DINOv2 [Oqu+23], trained on large datasets, have demonstrated their efficiency as feature extractors that can be reused for other tasks.

However, it's important to recognize that while repurposing pre-trained models as frozen feature extractors is relatively straightforward, fine-tuning them in the midst of a domain shift can be challenging [Kum+22], a fact that has significant relevance for CIL since many existing models depend on fine-tuning.

2.4.2 Pre-training Applications in Class Incremental Learning

Recent advancements in Continual Incremental Learning (CIL) leverage pre-trained models as an efficient foundation for the incremental process, as suggested in studies like [Tia+23] and [Wu+22]. Similar approaches that incorporate dynamic prompting are explored in works like [Wan+22b]. The research presented in [Ost+22] utilizes pre-trained models to introduce a low-compute method that includes the replay of past training samples. This concept of employing a pre-trained feature extractor is particularly advantageous when training data is limited, as encountered in few-shot CIL scenarios, as discussed in [Ahm+22]. Energy Self-Normalization [Wan+23a] leverages a pre-trained transformer model, trains classifiers per state, and then merges them by combining a temperature-controlled energy metric, an anchor-based energy self-normalization strategy, and a voting-based inference augmentation strategy. However, the substantial parametric size of pre-trained models, often in the hundreds of millions, is often an obstacle for continual learning applications [HK22]. One solution to alleviate this problem is to use knowledge distillation to derive smaller models from larger ones [HVD15; Tou+21].

2.5 Exemplar-Based Class-Incremental Learning

The Exemplar-Based methods can be found in the previous sections too, but since they are not the focus of this thesis, we emphasized their usage in an Exemplar-Free Class-

Incremental Learning setting. Here are the main methods to use exemplars in Class-Incremental Learning.

Exemplar-based methods are used to mitigate model forgetting by storing a subset of the training data, called "exemplars". These exemplars are then used to retrain the model. There are several methods for selecting these exemplars, the most common being:

- **Random**: This approach randomly selects exemplars from the available training data. This simple but effective method ensures a broad representation of the data, although it may not always focus on the most informative or representative samples.

- **Herding**: Unlike random selection, the herding technique deliberately selects exemplars that tend to preserve the mean sample of each class. This strategy ensures that the selected samples are highly representative of their respective classes, thus capturing the central tendencies of the class distributions.

- **Custom selection**: More complex methods, such as Mnemonics introduced in [Liu+20b], use updatable exemplars that are chosen in an end-to-end fashion. Interestingly, these exemplars are typically located at the boundary of the class data distribution. This positioning helps the model discriminate between classes more effectively, as these boundary cases often provide more distinctive and discriminative information for learning.

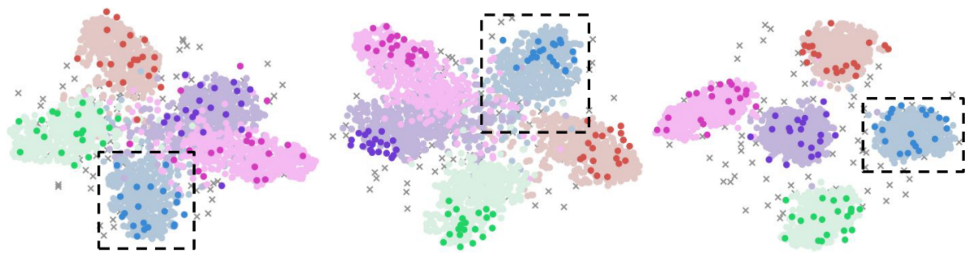

(a) Random selection (b) Herding selection (c) Custom selection

Fig. 2.2.: Comparison of exemplar selection methods. Source [Liu+20b].

On Figure 2.2, we can see the difference between (a)the random selection, (b)the herding selection and (c)the custom (here Mnemonics) selection. The random selection selects random samples from the training set, creating a representative subset of the training set. The herding selection selects samples that are close to the mean of the class, creating a subset of the training set that is representative of the mean of the class. The mnemonics selection selects samples that are at the boundary of the class.

By using these techniques to select exemplars, exemplar-based methods can maintain a reliable and representative summary of past data, facilitating more effective incremental learning. Once the exemplars are selected, the training process is described in Section 1.5.1, which often brings back the training procedure of EBCIL problems to traditional ML problems.

Instead of using the exemplars to directly retrain the model, the Exemplar-Based methods can use them to optimize the model. For instance, they can be used to learn the subset of correlations common to all tasks. For example, A-GEM [Cha+19] replays stored samples, ensuring that the loss on the replayed data is used as an inequality constraint. In other words, the loss on the current data is optimized while adhering to the constraint that the loss on the replayed data cannot increase.

Instead of storing exemplars, some methods use compressed information to perform the incremental learning. For instance, the method proposed in [Wan+22a] utilizes determinantal point processes to establish a well-judged equilibrium between the quality of images and the quantity of samples per class.

The Generative Feature Replay For Class-Incremental Learning (GFR-IL) [Liu+20a] fuses generative replay with distillation, utilizing a generator network to replay learned feature representations. The Deep Generative Replay (DGR) [Shi+17] employs generative models to combine old data with new data. Lastly, the Continual Neural Dirichlet Process Mixture (CN-DPM) [Lee+20] adopts a Bayesian nonparametric approach to dynamically adjust the complexity of the model as new tasks emerge. The End-to-End Incremental Learning (EEIL) [Cas+18] method fuses distillation-based techniques with rehearsal strategies. This method uses a small sample of previous classes to stabilize the learning of new classes.

As a result, it achieves an enhancement in accuracy compared to several pre-existing methods. Additionally, this accuracy boost is attained while ensuring that the computational cost remains closely aligned with that of the herding technique. On a different approach, the method proposed in [Isc+20] retains features from previous classes. This enables the potential to maintain a greater number of exemplar representations within a comparable buffer setup. However, since the method is not based on a fixed representation, the adaptation of features is essential to ensure their compatibility across different incremental states and then requires additional computational resources and an additional network to perform this adaptation. Class prototype creation has been examined in learning contexts other than CIL. An intriguing method, centered around few-shot learning, was introduced in [DL19] and presents a distance-based classifier that utilizes an approximated Mahalanobis distance. In Chapter 4, we will present a method that uses a fixed representation and a pseudo-feature generator to generate features of past classes, thus avoiding the need for additional computational resources and an additional network to perform this adaptation.

2.6 Our EFCIL contributions

As seen in the previous sections, various approaches are attempting to address the challenges of EFCIL. However, existing methods often fall short in providing a comprehensive solution that can adapt to a wide range of real-world scenarios.

In light of these limitations, this thesis aims to make substantial contributions to the field of continual learning. The proposed solutions revolve around achieving an optimal balance between plasticity and stability in the learning process, mitigating the effects of catastrophic forgetting, and efficiently integrating new classes without requiring additional memory buffers or massive computational resources.

The first contribution, PlaStIL[Pet+23a], detailed in Chapter 3, presents a novel approach that distributes model parameters differently to achieve a better balance between plasticity and stability. By introducing a plasticity layer, our method integrates new classes with partially fine-tuned models while preserving past knowledge with a frozen feature extractor. This approach can be easily incorporated into any transfer-based method designed for exemplar-free incremental learning. Experimental results, conducted on three large datasets, validate the performance superiority of our proposed method compared to existing ones.

The second contribution, FeTrIL[Pet+23b], detailed in Chapter 4, approaches the challenge of EFCIL from a unique perspective. Instead of focusing on fine-tuning the model or using a fixed feature extractor, we propose a method that combines both. Our approach introduces a pseudo-feature generator that uses the geometric translation of new class features to create representations of past classes. This method allows for an efficient storage scheme and a significantly faster incremental process. An experimental comparison with ten existing methods shows that our approach outperforms most of them in a variety of scenarios and datasets.

Finally, the third contribution[Pet+24], detailed in Chapter 5, presents a comprehensive experimental study to understand the relative influences of several factors on the performance of EFCIL. These include the initial learning strategy, the choice of the EFCIL algorithm, the neural architecture, the nature of the target task, the class distribution, and the number of examples available for learning. We find that the initial training strategy is the key factor influencing the average incremental accuracy, while the choice of the EFCIL algorithm plays a crucial role in preventing forgetting. Based on these findings, we provide practical recommendations to guide the selection of the appropriate initial training strategy for different incremental learning use cases.

Each contribution provides unique insights and solutions to the various challenges associated with class incremental learning, representing an important step towards more efficient and effective learning from streaming data. The following chapters delve into the specifics of each contribution.

3 PlaStIL: Plastic and Stable Exemplar-Free Class-Incremental Learning

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

— **Robert Frost**
The Road Not Taken (1/4)

3.1 Introduction

Introduced in Chapter 1, Class-incremental learning (CIL) enables the adaptation of artificial agents to dynamic environments in which data occur sequentially. CIL is particularly useful when the training process is performed under memory and/or computational constraints [Mas+21]. However, it is really susceptible to catastrophic forgetting, which refers to the tendency to forget past information when learning new data [Kem+18; MC89]. As seen in Section 2.3 most CIL methods [Dou+20; Hou+19; JS18; Reb+17; Wu+19] use fine-tuning with knowledge distillation [HVD15] from the previous model to preserve past information. Knowledge distillation has been progressively refined [Hou+19; JS18; WGL21; Yu+20; Zho+19] to improve CIL performance. An alternative approach to CIL is inspired by transfer learning [Raz+14]. These methods, described in Section 2.2, use a feature extractor that is frozen after the initial CIL state [BP18; Hay+20; HK20; Reb+17]. They become competitive in exemplar-free CIL, a difficult setting due to a strong effect of catastrophic forgetting [Mas+21]. The main challenge, as explain in Section 1.3, is to find a good plasticity-stability balance because fine-tuning methods favor plasticity, while transfer-based methods only address stability.

In this work, we tackle exemplar-free CIL (EFCIL) by combining the two types of approaches described above. Building on the strong performance of transfer-based methods [BPK21; HK20], we introduce a plasticity component by partially fine-tuning models

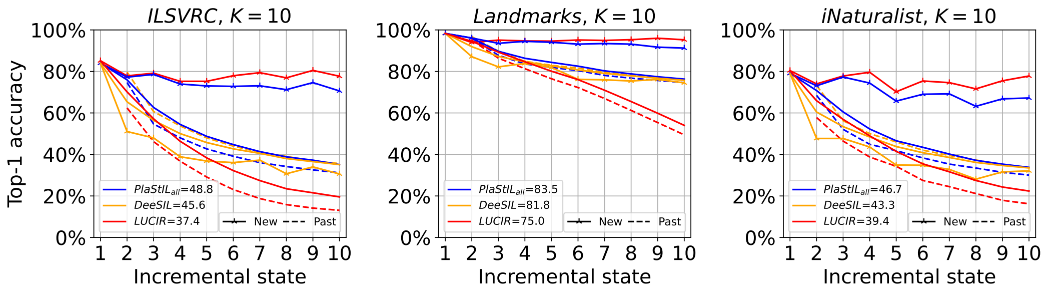


Fig. 3.1.: Accuracy of past and new classes in exemplar-free CIL for three large-scale datasets with $K = 10$ incremental states. LUCIR [Hou+19] uses distillation to preserve past knowledge and favors plasticity. DeeSIL [BP18] transfers features from the initial frozen model to all subsequent states and focuses on stability. PlaStIL offers a better plasticity-stability balance. Note that the proportion of past classes increases as the incremental process advances and so does their weight in global accuracy.

for recent classes. The results from Figure 3.1 show that our method gives a better global accuracy compared to DeeSIL [BP18] and LUCIR [Hou+19], two representative methods focused on stability and plasticity, respectively. Accuracy is presented separately for past and new classes for existing methods to examine the plasticity-stability balance offered by each method. LUCIR has optimal plasticity (best accuracy of new classes), while DeeSIL has optimal stability (best accuracy for past classes). However, the performance of both methods is strongly degraded on the complementary dimensions. Our method is close to LUCIR in terms of plasticity and to DeeSIL in terms of stability. Consequently, it ensures a better balance between these two properties of EFCIL.

PlaStIL is inspired by transfer learning but adds a partial fine-tuning component to boost plasticity. It is applicable to any transfer-based method and we exemplify it with DSLDA [HK20] and DeeSIL [BP18]. We introduce a hybrid classification layer that combines classification weights learned with the initial model for past classes and with the fine-tuned models for recent classes. We evaluate the proposed approach on three datasets which contain 1000 classes each. The number of incremental states is varied to assess the robustness of the tested methods. Results show that performance gains are obtained by adding the proposed plasticity component to transfer-based methods. Equally interesting, important performance improvements are obtained over distillation-based methods, which are the mainstream methods deployed to tackle CIL [Hou+19; JS18; Reb+17; Smi+21; WGL21]. We will open-source the code to facilitate reproducibility.

3.2 Related Work

Incremental learning is interesting when artificial agents need to learn under memory or computational constraints [Mas+21; Par+19; Reb+17]. The main challenge in CIL is to tackle the catastrophic forgetting phenomenon [Kem+18; MC89]. A suitable balance between plasticity and stability of the learned models is sought [MBB13]. Plasticity and stability are needed in order to accommodate new data and preserve previously learned knowledge, respectively [Cha+18]. As noted in a recent survey [Mas+21], a large majority of CIL-related works use a memory buffer that stores samples of past classes in order to improve overall performance. Replaying these samples facilitates the preservation of past knowledge, thus making the incremental learning process akin to imbalanced learning [BPK21]. However, the assumption that past samples are available is strong and limits the applicability of CIL. A growing research effort was devoted to exemplar-free CIL (EFCIL) [BPK21; Mas+21]. The plasticity-stability dilemma is particularly challenging without memory since the effects of catastrophic forgetting are stronger in this case [BPK20; Reb+17; Smi+21; WGL21].

Survey papers such as [Lan+19; Mas+21] analyze different types of continual learning methods which are usable in exemplar-free CIL. Parameter-isolation methods, such as HAT [Ser+18] or PackNet [ML18] were designed for task-incremental learning, a setting in which the task ID is known at inference time. They learn task-specific masks to reduce catastrophic forgetting. However, they are impractical in task-agnostic scenarios since the simultaneous evaluation of all tasks is not possible and specific forward passes are needed for each task [Mas+21]. Regularization-based methods are a popular solution to EFCIL and they fall into two subcategories, namely data-focused or prior-focused approaches. Existing works [Lan+19; Mas+21] showed that data-focused methods outperform prior-focused on in EFCIL scenarios. Consequently, we discuss data-focused methods and use representative examples of them in experiments. According to [BPK21; Mas+21], most methods update learned models in each IL state using fine-tuning for plasticity and different flavors of knowledge distillation [HVD15] for stability. Alternatively, a few works [BP18; Dha+21; Hay+20; HK20] use an initial representation throughout the incremental process. We discuss the merits and limitations of both approaches below and position our contribution with respect to them.

Distillation-based methods are inspired by LwF [LH16], an adaptation of knowledge distillation [HVD15] to an incremental context. The authors of [Dha+18] add an attention mechanism to the distillation loss to preserve information from past classes and obtain an improvement over LwF. Since its initial use for exemplar-based CIL in iCaRL [Reb+17], distillation was refined and complemented with other components to improve the plasticity-stability compromise. LUCIR [Hou+19] applies distillation on feature vectors instead of raw classification scores to preserve the geometry of past classes, and an inter-class separation to maximize the distances between past and new classes. LwM [Dha+18] adds an attention mechanism to the distillation loss to preserve information from base classes. An interesting solution is proposed in [Yu+20], where the feature drift between incremental steps is estimated based on features of samples associated with new classes. However, this method has a large footprint since it needs a large multi-layer perceptron and also stores past features to learn the transformation. A feature transformation method is designed for task-incremental learning and adapted for CIL by predicting the task associated with each test sample [Ver+21]. The authors of [Smi+21] combined feature drift minimization and class separability to improve distillation. The authors of [WGL21] proposed an approach stabilizing the fine-tuned model, adding reciprocal adaptive weights to weigh past and new classes in the loss, and introducing multi-perspective training set augmentation. They reported significant gains in exemplar-free CIL compared to LUCIR [Hou+19] and SDC [Yu+20] using a protocol in which half of the dataset is available initially [Hou+19]. Distillation is widely used but a series of studies question its usefulness [BPK21; Mas+21; PTD20], especially for large-scale datasets. One explanation for the lack of scalability of distillation is that inter-class confusion becomes too strong when the number of past classes is high. Another challenge is that distillation needs to store the previous deep model to preserve past knowledge. The total footprint of these methods is double the footprint of the backbone model used.

A second group of methods learns a deep representation in the initial state and uses it as a feature extractor throughout the CIL process. They are inspired by transfer learning [Tan+18] and favor stability since the initial representation is frozen. Usually, these methods learn shallow external classifiers on top of the initial deep representation. The nearest class mean (NCM) [Men+13] was introduced in [Reb+17], linear SVCs [CV95] were used in [BP18] and extreme value machines [Rud+17] were tested by [Dha+21]. REMIND [Hay+20] uses a vector quantization technique to save compressed image representations which are later reconstructed for memory consolidation by training model tops. The main difference with our proposal is that the past is represented via compressed image representations instead of model tops. DSLDA [HK20] updates continuously a class-specific mean vector and a shared covariance matrix. The predicted label is the one having the closest Gaussian in the feature space defined by these vectors and matrix. These methods are simple and suited for exemplar-free CIL, particularly for large-scale datasets where they outperform distillation-based methods [BPK21; Mas+21]. Equally important, they only use the initial model and thus have a smaller footprint. Their main drawbacks are the genericity of the initial representation and the sensitivity to strong domain variations. Their performance drops if a small number of classes is initially available [BPK21] and if the incremental classes have a large domain shift with classes learned initially [Lan+19]. The robustness of the initial representation can be improved is an assumption is made that a model pre-trained with a large amount of data is available and that its features are transferable to the incremental datasets [Hay+20]. ESN [Wan+23a] is an interesting method that was proposed very recently and makes this assumption, similarly to REMIND and DSLDA. ESN leverages a pre-trained transformer model, trains classifiers per state, and then merges them by combining a temperature-controlled energy metric, an anchor-based energy self-normalization strategy, and a voting-based inference augmentation strategy to ensure impartial and robust EFCIL predictions. Here, we experiment without a large pre-trained model in order to cover cases where there is a large domain drift between the pre-training and the incremental datasets.

3.3 Proposed Method

3.3.1 Problem Formalization

The CIL process is divided into K states, with n classes learned in each state. In EFCIL, no past data can be stored for future use. The predictions associated with observed classes are noted p. We write the structure of a deep model as:

$$\mathcal{M} = \{\mathcal{B}, \mathcal{T}, \mathcal{W}\} \tag{3.1}$$

with: $\mathcal{M}$ - the full model; $\mathcal{B}$ - the model base which includes the initial layers; $\mathcal{T}$ - the model top which includes the subsequent layers up to the classification one; $\mathcal{W}$ - the classification layer which provides class predictions.

Assuming that the CIL process includes K states, the objective is to learn K models in order to incorporate all classes which arrive sequentially. The incremental learning process can be written as:

$$\mathcal{M}_1 \rightarrow \mathcal{M}_2 \rightarrow ... \rightarrow \mathcal{M}_k \rightarrow ... \rightarrow \mathcal{M}_{K-1} \rightarrow \mathcal{M}_K \tag{3.2}$$

Each incremental model needs to integrate newly arrived data, while also preserving past knowledge. Assuming that the current state is $k \geq 2$, the majority of existing CIL methods [Cas+18; Hou+19; Reb+17; Smi+21; WGL21; Wu+19] fine-tunes the entire current model $\mathcal{M}_k$ by distilling knowledge from $\mathcal{M}_{k-1}$. We note w_k^1 to $w_k^{(k-1)\times n}$ the classifier weights of the past states 1 to $k-1$, and $w_k^{(k-1)\times n+1}$ to $w_k^{k\times n}$ the classifier weights of the new state k.

Their classification layer is written as:

$$\mathcal{W}_k^{ft} = \{w_k^1, ..., w_k^n, ..., w_k^{(k-1)\times n}, w_k^{(k-1)\times n+1}, ..., w_k^{k\times n}\} \tag{3.3}$$

They are all trained using $\mathcal{T}_k$, the model top learned in the k^{th} state. The $\mathcal{W}_k^{ft}$ layer is biased toward new classes since it is learned with all samples from the current state, but only with the representation of past classes stored in $\mathcal{M}_{k-1}$ [Hou+19; Reb+17; Wu+19]. This group of methods focuses on CIL plasticity at the expense of stability [BPK21; Mas+21].

Transfer-based methods [BP18; Hay+20; HK20] freeze the feature extractor $\mathcal{F}_1 = \{\mathcal{B}_1, \mathcal{T}_1\}$ after the initial non-incremental state. All the classes observed during the CIL process are learned with $\mathcal{F}_1$ as features extractor. The classification layer can be written as:

$$\mathcal{W}_1^{fix} = \{w_1^1, ..., w_1^n, ..., w_1^{(k-1)\times n}, w_1^{(k-1)\times n+1}, ..., w_1^{k\times n}\} \tag{3.4}$$

All classifier weights from Eq. 3.4 are learned with image features provided by $\mathcal{F}_1$, the feature extractor learned initially, inducing a bias toward initial classes. It is suboptimal for classes learned in states $k \geq 2$ because their samples were not used to train $\mathcal{F}_1$. These methods focus on CIL stability at the expense of plasticity [Mas+21].

3.3.2 PlaStIL Description

PlaStIL is motivated by recent studies which question the role of distillation in CIL, particularly for large-scale datasets [BPK21; Mas+21; PTD20]. Instead of $\mathcal{M}_{k-1}$ needed for distillation, PlaStIL uses two or more model tops $\mathcal{T}$ which have an equivalent number of parameters at most. PlaStIL is inspired by feature transferability works [NSZ20; Yos+14] which show that higher layers of a model, included in $\mathcal{T}$, are the most important for successful transfer learning. Consequently, the initial layers $\mathcal{B}$ are frozen and shared throughout the incremental process. A combination of model tops which includes $\mathcal{T}_1$, the one learned in the first incremental state, and those of the most recent state(s) is used in PlaStIL. Similar to transfer-based CIL methods [BP18; Hay+20; HK20], $\mathcal{T}_1$ ensures stability for classes first encountered in past states for which a dedicated top model is not available. Different from existing methods, model top(s) are available for the most recent state(s), thus improving the overall plasticity of PlaStIL. The number of different model tops which can be stored instead of $\mathcal{M}_{k-1}$ depends on the number of higher layers that are fine-tuned in each incremental state. The larger the number of layers in $\mathcal{T}$, the larger its parametric footprint is and the lower the number of storable model tops will be.

The method is illustrated in Figure 3.2 with a toy example that includes $K = 4$ IL states, with $n = 2$ new classes per state and which assumes that up to three model tops can be stored. Up to the third state, PlaStIL stores a model top per state, and corresponding classifier weights are learned for each model top. In the fourth state, one of the model tops needs to be removed in order to keep the parameters footprint bounded. Consequently, $\mathcal{T}_2$ is removed and the initial model top $\mathcal{T}_1$ is used. Note that $\mathcal{T}_1$ is used to learn classifier weights for all classes when they occur initially. These initial weights are stored for usage in later incremental states in order to cover all past classes for which dedicated model tops cannot be stored. The storage of the initial weights generates a small parameters overhead but its size is small and does not increase over time. If the classifier weights are d-dimensional, the number of supplementary parameters is $n \times d$. Moreover, this overhead can be easily compensated by the choice of the number of parameters in $\mathcal{T}$ and the number of such model tops which are stored. In Figure 3.2, initial classifier weights w_1^3 and w_1^4 are first learned in state 2 but only used in state 4, when $\mathcal{T}_2$ is no longer available. In state 4, $\mathcal{T}_1$ is used for classes that first occurred in states 1 and 2 (classifiers weights w_1^1 to w_1^4). $\mathcal{T}_3$ is reused along with its classifier weights w_3^5 and w_3^6, learned for the classes which were learned in state 3.

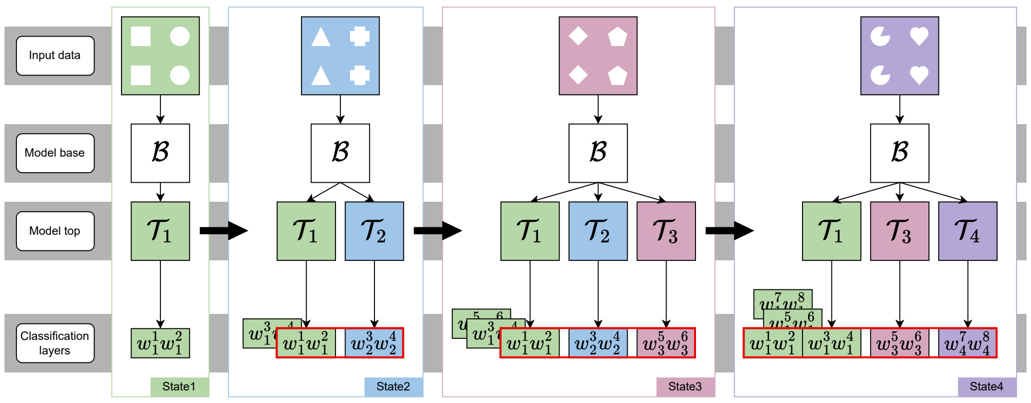


Fig. 3.2.: PlaStIL overview using a toy example with $K = 4$ CIL states and $n = 2$ new classes learned per state. The global memory footprint is equivalent to that of distillation-based methods, but this memory is used differently. We assume that a model base and at most three model tops can be used. A base $\mathcal{B}$, is learned initially and then frozen, as is $\mathcal{T}_1$ which is needed to ensure stability. Initial classifier weights are trained using $\mathcal{T}_1$ in each state and reserved for future use. Classifier weights that are actually used in each state are highlighted in red. In state 4, the recent model tops ($\mathcal{T}_4$ and $\mathcal{T}_3$) are included to ensure plasticity. Classifier weights w_4^7 and w_4^8, associated with the new classes from state 4 are learned with features provided by $\mathcal{T}_4$. w_3^5 and w_3^6 were learned with $\mathcal{T}_3$ features in state 3, when they were new. w_1^1 to w_1^4 were learned with $\mathcal{T}_1$ features, in states 1 and 3. w_1^5 to w_1^8 are reserved for future use. $\mathcal{T}_2$ is discarded to keep the total memory footprint of PlaStIL bounded. *Best viewed in color.*

Finally, classifier weights of new classes w_4^7 and w_4^8 are learned with $\mathcal{T}_4$. This results in a hybrid classification weights layer which is defined as:

$$\mathcal{W}_k^{hyb} = \{w_1^1, ..., w_1^{j\times n}, ..., w_{j+1}^{j\times n+1}, ..., w_{j+1}^{(j+1)\times n}, ..., w_k^{(k-1)\times n+1}, ..., w_k^{k\times n}\} \tag{3.5}$$

where we assume that $k - j + 1$ models can be stored, with $2 \leq j \leq k$; the blocks of classes learned with features from different model tops are color coded.

In Equation 3.5, classifier weights of the first j incremental states are learned with the features provided by the initial model top $\mathcal{T}_1$. Those of the most recent states ($j + 1$ to k) are learned with features provided by model tops $\mathcal{T}_{j+1}$ to $\mathcal{T}_k$. An advantage of the layer from Equation 3.5 is that it ensures a good balance between stability, via $\mathcal{T}_1$ and plasticity, via $\mathcal{T}_{j+1}$ to $\mathcal{T}_k$. The number of storable model tops varies inversely with the number of layers that they include. We report results with three top depths in Section 3.4. A choice between internal and external classifiers has to be made for the implementation of this classification layer. Experiments from [BPK21] indicate that external classifiers are easier to optimize when transferring features from $\mathcal{M}_1$ to subsequent incremental states.

PlaStIL is primarily intended for a CIL scenario under the assumption that the parametric budget should not increase over time. If memory is allowed to grow over time, the method could store a larger number of model tops. The model top creation and removal policy could be adapted depending on the continual learning scenario being explored. For instance, if classes were grouped semantically, as it is the case in task-incremental learning, it might be better to create model tops for states which include classes that are most dissimilar from those of the initial model. Another interesting scenario assumes that past classes can be revisited, with new samples of them arriving later in the incremental process. In this case, model tops could be created for the current state if the amount of samples for revisited classes is smaller than those associated with existing tops. Such a top creation policy would be based on the assumption that the less revisiting there is, the more likely a new top would be due to the fact that the current state includes more novelty. In practice, the total number of stored model tops will depend on the total budget available on the device. When the memory budget is reached, one of the selection strategies listed above can be applied depending on the characteristics of the continual learning process. In Subsection 3.4.5, we show that the creation of model tops for the most recent states is a good solution for class incremental learning. While interesting, the adaptation of PlaStIL to other continual learning scenarios is out of the immediate scope and is thus left for future work.

3.4 Experiments

We evaluate PlaStIL with three large-scale datasets designed for different visual tasks. We compare it to a representative set of EFCIL methods. We vary the total number of states K using $K \in \{5, 10, 20\}$ because the length of the CIL process has strong effects on EFCIL performance [Mas+21]. The evaluation metric is the top-1 accuracy averaged over all incremental states. Following a common practice in CIL [Cas+18; Hou+19; Wu+19], the performance on the initial state is excluded because it is not incremental.

3.4.1 Datasets

Two recent comparative studies [BPK21; Mas+21] show that the size of the evaluation datasets has a strong influence on performance in exemplar-free CIL. We thus select large-scale datasets which provide a more realistic scenario for evaluation compared to medium-scale ones which are still used [Smi+21; WGL21]. We run experiments with:

- ILSVRC [Rus+15] - the well-known subset of ImageNet [Den+09] built for the eponymous competition and also used in CIL [Cas+18; Hou+19; Reb+17; Wu+19]. The training and testing sets are composed of 1,231,167 and 50,000 images, respectively.

- Landmarks - a subset of a landmarks recognition dataset [Noh+17] which includes a total of over 30000 classes. We select the 1000 classes having the largest number of images. The training and testing sets are composed of 374,367 and 20,000 images, respectively.

- iNaturalist - a subset of the dataset used for the iNaturalist challenge [Van+18]. The full version includes 10000 fine-grained classes for natural species. We sample 1000 classes from different super-categories to obtain a diversified subset. The training and testing sets are composed of 300,000 and 10,000 images, respectively.

More details about the datasets are provided in the appendix of this thesis in B.1

3.4.2 State-of-the-art methods

We compare PlaStIL with the following existing methods:

- LwF [Reb+17] - is a CIL version of the initial method from [LH16]. It tackles forgetting using a distillation loss.

- SIW [BPK20] - uses a vanilla *FT* backbone and tackles catastrophic forgetting by reusing the past classifiers learned when these classes were first learned.

- LUCIR [Hou+19] - adapts distillation to feature vectors instead of raw scores to preserve the geometry of past classes and also pushes for inter-class separation. Note that while initially proposed for CIL with memory, this method showed strong performance in EFCIL too [BPK21].

- SPB-M [WGL21] - is a recent method that focuses on balancing plasticity and stability in exemplar-free CIL. We report results with the multi-perspective variant, which has the best overall performance in [WGL21]. SPB-M provides very competitive performance compared to LUCIR when half of the dataset is initially available.

- PASS [Zhu+21b] - uses prototypes of past classes in combination with distillation in order to counter catastrophic forgetting.

- REMIND [Hay+20] - encodes knowledge about past classes by storing compressed image representations. It is compared with our method by allocating the amount of storage used for model tops in our method to compressed representations of past samples.

- DSLDA [HK20] - is based on Gaussian functions defined in the features space by specific mean class vectors and a covariance matrix that is shared among all classes. This method is interesting since its classification layer provides an efficient inter-class separability mechanism.

- DeeSIL [BP18] - freezes the initial model uses linear SVCs [CV95] for the final layer, which is trained independently for each state.

The first four methods fine-tune models incrementally. REMIND trains model tops using a compressed replay buffer and is very relevant for comparison here. DSLDA and DeeSIL are transfer-based, and PlaStIL can be applied to them. They were implemented using their original optimal parameters. Whenever the original experimental settings were different from the ones used here, the correct functioning of the baselines was carefully checked. The obtained accuracy was coherent with the results reported in the original papers and/or in comparative studies such as [BPK21; Mas+21] in all cases. See details about the reproduced results in the appendix.

We experiment with three versions of PlaStIL designed to ensure that its parameters footprint is equivalent to (or lower than) that of distillation-based methods. We assume that the incremental process is in the k^{th} state and test:

- PlaStIL$_1$ - fine-tunes model tops $\mathcal{T}$ limited to the last convolutional layer of ResNet-18, which includes approximately 21.45% of the model parameters. Consequently, we can fit $\mathcal{T}_1$, $\mathcal{T}_{k-3}$, $\mathcal{T}_{k-2}$, $\mathcal{T}_{k-1}$ and $\mathcal{T}_k$ in memory.

- PlaStIL$_2$ - fine-tunes $\mathcal{T}$ which includes the last two convolutional layers of ResNet-18, which includes approximately 42.9% of the model parameters. We can fit $\mathcal{T}_1$, $\mathcal{T}_{k-1}$ and $\mathcal{T}_k$ in the allowed parameters memory.

- PlaStIL$_{all}$ - trains all the layers of the current model in k^{th} state and we can only use $\mathcal{T}_1$ and $\mathcal{T}_k$ in each IL state.

PlaStIL variant test different variants of the compromise between the number of model tops and their depth. PlaStIL$_1$ fine-tunes only the last convolutional layer of model tops and maximizes the number of such storable models. PlaStIL$_{all}$ provides optimal transfer since all layers are trained with new data but can accommodate only the current model. PlaStIL$_2$ provides a compromise between top depth and the number of storable models.

We also provide results for: (1) vanilla fine-tuning (FT) - a baseline that does not counter catastrophic forgetting at all, and (2) $Joint$ - an upper bound that consists of standard training in which all data are available at once.

3.4.3 Implementation

A ResNet-18 [He+16] architecture was used as a backbone in all experiments. All methods were run with the published optimal parameters and minor adaptation of the codes to unify data loaders: FT [BPK20], LwF [Reb+17], LUCIR [Hou+19], SIW [Smi+21], DSLDA [HK20] and DeeSIL [BP18]. SPB-M[WGL21] has no public implementation and we reimplemented the method. We verified its correctness by comparing the accuracy obtained with our implementation (60.1) to the original one (59.7) for ILSVRC split tested by the authors [WGL21]. All methods were implemented in PyTorch [Pas+19], except for LwF which uses the Tensorflow [Mar+15] implementation of LwF from [Reb+17] because it provides better performance compared to later implementations [JS18; Wu+19]. The training procedure from [Hay+20] was used to obtain initial models for all transfer-based methods. These initial models were trained for 90 epochs, with a learning rate of 0.1, a batch size of 128, and a weight decay of 10^{-4}. We used stochastic gradient descent for optimization and divided the learning rate by 10 every 30 epochs. Detailed parameters are presented in the appendix B.2.1 of this thesis.

3.4.4 Main results

The results presented in Table 3.1 show that all PlaStIL variants improve over the transfer-based methods to which they are added for all tested datasets and CIL configurations. The gains are generally higher for $K = 5$ states but remain consistent for the $K = \{10, 20\}$. For instance, PlaStIL$_1$ gains 6.2, 6.4 and 4.4 points for ILSVRC split into 5, 10 and 20 states, respectively. The best overall performance is obtained with PlaStIL$_1$, followed by PlaStIL$_2$ and PlaStIL$_{all}$ applied on top of DeeSIL. A combination of recent model tops which fine-tune only the last convolutional layer is best here. Our method applied to DeeSIL provides best performance for 5 and 10 incremental states. Gains are equally interesting for DSLDA, particularly for $K = 20$. This baseline has better performance than DeeSIL for all three datasets when $K = 20$. The application of PlaStIL on top of DSLDA leads to slightly better performance compared to the version built on top of DeeSIL for ILSVRC (42.5 vs. 41.9) and iNaturalist (37.4 vs 36.4). The better behavior of DSLDA for longer incremental sequences is explainable since this method features a global inter-class separability component. In contrast, DeeSIL only separates classes within each state and its discriminative power is reduced when each state includes a low number of classes.

Distillation-based methods have lower performance compared to transfer-based methods for the tested large-scale datasets. This result is coherent with previous findings regarding scalability problems of distillation [BPK21; Mas+21; PTD20]. The difference between PlaStIL applied to DeeSIL and DSLDA and distillation-based methods is very consequent. It is in the double-digit range compared to LUCIR, the best distillation-based method, for five configurations out of nine tested. This difference reaches a maximum value of 27.6 top-1 accuracy points for Landmarks with $K = 20$ states and a minimum of 2.7 points for the same dataset with $K = 5$ states. SPB-M is a recent method that compares very favorably with LUCIR when half of the dataset is allowed in the initial state [WGL21]. However, its behavior is globally similar to that of LUCIR, with performance gains for

CIL Method	**mem on disk**	*ILSVRC*			*Landmarks*			*iNaturalist*		
		K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
FT (*lower bound*)	44.59MB	26.6	18.3	12.2	31.3	21.0	13.4	25.6	17.5	11.4
LwF [Reb+17]	44.59MB	24.0	21.1	17.4	36.9	34.7	28.0	23.9	21.5	16.3
SIW [BPK20]	44.59MB	38.3	35.2	26.8	66.4	55.7	41.4	38.6	30.9	17.2
LUCIR [Hou+19]	89.18MB	50.4	37.4	24.4	89.5	75.0	50.5	54.9	39.4	24.8
SPB-M [WGL21]	89.18MB	38.9	37.3	30.4	81.6	70.4	57.1	46.7	39.6	29.8
PASS [Zhu+21b]	89.18MB	39.4	35.9	29.8	65.0	55.1	42.3	48.0	40.9	31.8
REMIND [Hay+20]	89.18MB	52.2	44.8	35.9	83.3	77.5	72.2	50.6	39.4	31.3
DSLDA [HK20]	45.59MB	51.3	45.4	39.2	82.7	78.5	74.5	49.7	42.1	34.8
w/ PlaStIL$_1$	93.44MB	56.8	50.1	42.2	86.8	82.1	76.1	53.8	45.6	36.6
w/ PlaStIL$_2$	87.52MB	58.3	50.6	**42.5**	87.8	82.1	76.0	56.1	46.2	36.9
w/ PlaStIL$_{all}$	91.18MB	57.7	49.8	41.9	86.9	81.3	75.5	56.2	46.3	**37.4**
DeeSIL [BP18]	44.59MB	52.4	45.4	37.5	87.4	80.8	73.8	52.7	43.5	33.9
w/ PlaStIL$_1$	88.44MB	58.6	**51.8**	41.9	92.1	**86.4**	**78.1**	56.8	**47.5**	36.4
w/ PlaStIL$_2$	84.52MB	**59.2**	50.2	39.7	**92.2**	85.1	76.7	57.3	46.9	36.0
w/ PlaStIL$_{all}$	89.18MB	57.7	48.6	39.0	90.7	83.4	75.3	**58.2**	47.1	35.6
$Joint$ (*upper bound*)			73.0			97.4			75.6	

Tab. 3.1.: Average top-1 accuracy with three numbers of states K per dataset. PlaStIL is applied on top of DeeSIL and DSLDA. **Best results - in bold**, second best - underlined.

$K = 20$ states and losses for $K = 5$. This happens because SPB-M is more dependent on the representativeness of the initial model compared to LUCIR since it features a strong stability component. LwF and SIW, the two other methods which update models in each incremental state have even lower performance than LUCIR and SPB-M. The comparison with REMIND is also favorable in all tested configurations. This indicates that, given an identical memory budget, the use of model tops is more effective than the use of a replay buffer made of compressed samples of the past. All EFCIL methods need a supplementary budget to ensure the plasticity-stability balance. The takeaway from the comparison of PlaStIL with mainstream methods is that this budget is much better spent on partially fine-tuned model tops than on storing the previous model needed for distillation.

The results per dataset show that ILSVRC and iNaturalist are harder to solve compared to Landmarks. The gains obtained by PlaStIL are smaller for Landmarks since the progress margin is reduced. The performance gap between the evaluated methods and $Joint$ is large. The fact that the gap widens with the number of incremental states has specific explanations for the two types of methods. Past knowledge becomes harder to preserve when the number of fine-tuning rounds increases in LUCIR, SPB-M, SIW, and LwF. For transfer-based methods, past knowledge is encoded using a weaker feature extractor. PlaStIL reduces the gap with $Joint$, but the results reported here show that exemplar-free class-incremental learning remains a very challenging problem.

We propose a more detailed presentation of the incremental accuracy in Figure 3.3. These results confirm the large gap between the proposed methods and distillation-based ones. SPB-M has lower performance at the start of the process, but then becomes better than LUCIR, because of the representativeness of the initial model, as previously explained.

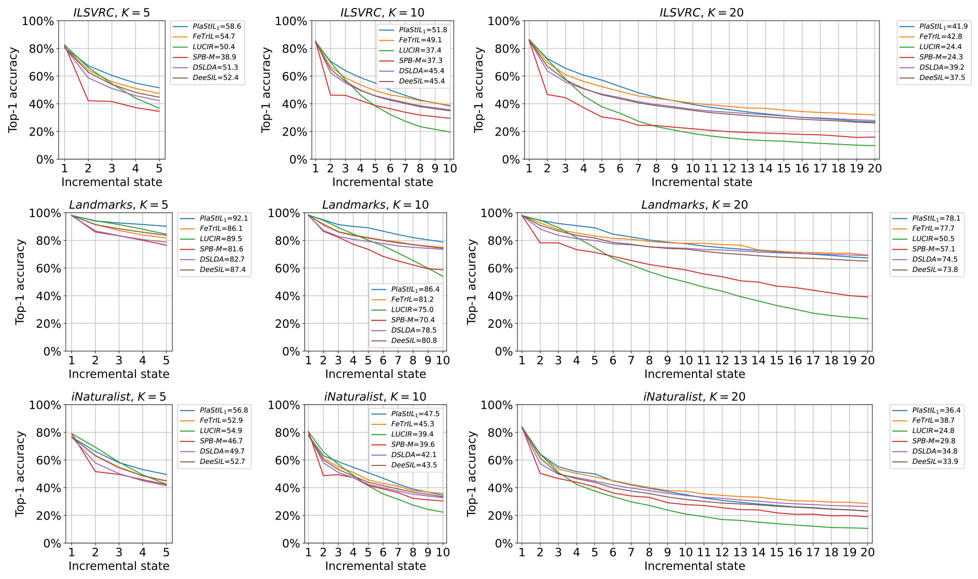


Fig. 3.3.: Incremental accuracy across all states for $K \in \{5, 10, 20\}$. Plots are presented for the best methods from Table 3.1.

Init. dataset	Landmarks	iNaturalist	ILSVRC	iNaturalist	Landmarks	ILSVRC
IL dataset	ILSVRC		Landmarks		iNaturalist	
DSLDA [HK20]	19.3	29.2	67.8	60.1	16.1	35.2
DeeSIL [BP18]	21.6	29.6	70.5	61.9	16.2	36.1
DeeSIL w/ PlaStIL$_1$	**28.1**	**33.8**	**76.8**	**68.3**	**21.4**	**39.2**
Joint	72.98		97.41		75.60	

Tab. 3.2.: Average top-1 incremental accuracy in a dataset transfer learning configuration. Results are given for transferring between all pairs of initial and target datasets. All experiments are run with $K = 10$ states. **Best results are in bold**.

The results from Table 3.1 show that transfer-based methods work well when features are transferred within the same dataset.

In Table 3.2, we examine the behavior of PlaStIL in a transfer scenario that involves a domain gap. The transfer is done between all pairs of initial and incremental datasets. The results show that PlaStIL$_1$ is more resilient to transfer compared to the transfer-based baselines. The best results are obtained with ILSVRC as initial models and iNaturalist and Landmarks as incremental datasets. This is intuitive since ILSVRC contains more diversified concepts and produces more generic features. It also has more samples per class compared to the other two datasets and its feature extractor is more robust. The accuracy obtained with a transfer from ILSVRC is comparable with that of best distillation-based methods from Table 3.1. The results with iNaturalist as the initial dataset are lower, but still interesting. Performance is lower when the domain shift between the initial and the target datasets is more important and if the initial model is learned on domain-specific data. This is, for instance, the case when Landmarks is used to train the initial model since this dataset only includes geographic landmarks. Note that transfer would be more efficient if

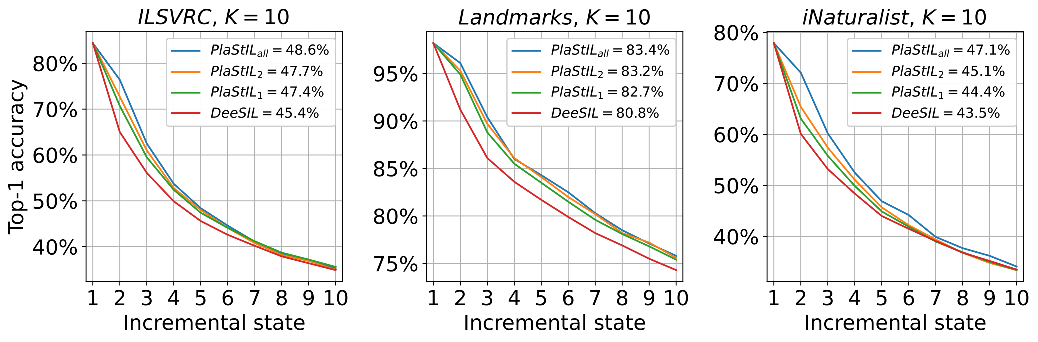


Fig. 3.4.: Top-1 incremental accuracy of three versions of PlaStIL applied to DeeSIL when using a single model top with variable fine-tuning depth. DeeSIL is a limit case in which the whole feature extractor is frozen. *Best viewed in color.*

larger initial datasets were used to reinforce the initial representation, as proposed in [BP18; HK20]. However, for fairness, our focus is on transfer experiments that use the same number of initial classes as in Table 3.1.

3.4.5 Method analysis

We conduct an analysis of PlaStIL in terms of model footprint and number of operations needed to infer test image predictions. These experiments are run using PlaStIL applied on top of DeeSIL since this variant has the best overall results. They are important in order to highlight the merits and limitations of the proposed method.

Model footprint. Incremental learning algorithms are particularly useful in memory-constrained environments. Their model footprint is thus an important characteristic. Distillation-based methods require the storage of models $\mathcal{M}_k$ and $\mathcal{M}_{k-1}$ to preserve past knowledge. Transfer-based methods only need $\mathcal{M}_1$, but they optimize stability at the expense of plasticity. The three versions of PlaStIL, whose performance is presented in Table 3.1, store $\mathcal{B}_1$, the initial model base, and a variable number of model tops. Each of the four recent model tops used by PlaStIL$_1$ fine-tunes the last convolutional layer of ResNet-18 [He+16], which accounts for 21.45% of the total number of the model's parameters. The parametric footprint of PlaStIL$_1$ is thus lower than that of distillation-based methods. PlaStIL$_2$ has the same footprint as PlaStIL$_1$ since it stores two tops which account for 42.9% of ResNet-18 parameters each.

As an ablation of PlaStIL, we also present results with a single model top, regardless of its fine-tuning depth in Figure 3.4. Naturally, PlaStIL$_{all}$ obtains the best results in this configuration, but interesting gains are still obtained with model tops which fine-tune one or two final convolutional layers in PlaStIL$_1$ and PlaStIL$_2$, respectively.

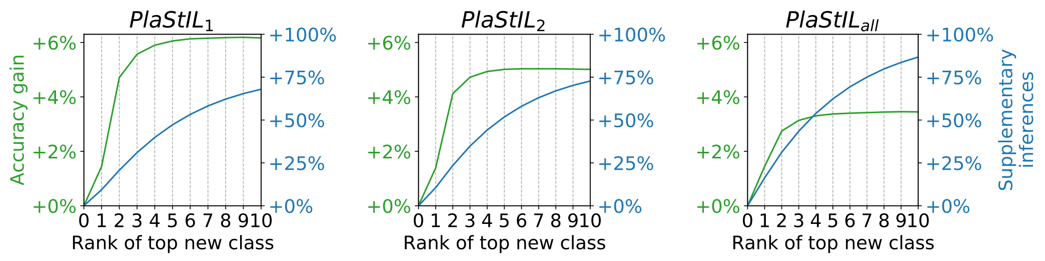


Fig. 3.5.: Top-1 accuracy gains obtained with the three variants of PlaStIL applied to DeeSIL with different thresholds for the rank of the top new class among the predictions generated with Equation 3.4. Results are shown for ILSVRC with $K = 10$ states. The corresponding percentage of supplementary inferences needed for each threshold is also plotted. Interesting gains are obtained starting with a recent class predicted in the second position, which requires approximately 25% of supplementary inferences for PlaStIL$_1$. *Best viewed in color.*

Inference complexity. The classification layer defined in Equation 3.5 is fed with features from the initial and the updated model top(s), for past and recent classes, respectively. By default, PlaStIL inferences requires an extraction of features for all used model tops. This supplementary inferences cost varies for PlaStIL variants due to the different number of parameters in the model top(s) that they use. This supplementary computational cost can be reduced if predictions are first computed using the initial model only, as defined in Equation 3.4. Then, subsequent model tops are used only if one of their classes is strongly activated for the test image. A top is used for inferences only if at least one of its associated classes is ranked among the top classes of the of the classification layer from Equation 3.4. The closer to 1 this rank threshold is, the smaller the added computational cost will be since fewer tops are likely to be used for inference. However, a restriction to small ranks might also discard useful model tops and thus reduce the positive effect of PlaStIL. Evaluation is done with top ranks of the new class in $\mathcal{W}_k^{fix}$ between 1 and 10 to examine the trade-off between inference complexity and performance. The obtained accuracy, as well as the added inference cost, for the three variants of PlaStIL applied over DeeSIL are presented in Figure 3.5. The results show that performance gains relative to DeeSIL rise sharply. The best balance between performance gains and added costs is obtained for PlaStIL$_1$. This is explained by the fact that $\mathcal{T}$ has the lowest number of parameters for this variant. Their activation can be done in a finer manner, resulting in a reduced overall inference cost. Interesting PlaStIL gains are obtained starting with a recent class being ranked second position by DeeSIL. The accuracy curve becomes practically flat if a recent class is ranked beyond the third position by the baseline. The results presented in Figure 3.5 provide further support to the fact that PlaStIL$_1$ is the most appropriate choice as a plasticity layer added on top of DeeSIL.

Choice of model tops. The main experiments used model tops created for the most recent incremental states. Other top creation and removal policies are possible, and we compare the one proposed here with an oracle that performs an optimal selection of tops. The oracle selects model tops associated with different states so as to maximize the average incremental

CIL Method	*ILSVRC*			*Landmarks*			*iNaturalist*		
	K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
DeeSIL	52.4	45.4	37.5	87.4	80.8	73.8	52.7	43.5	33.9
w/ PlaStIL$_1$	58.6	51.8	41.9	92.1	86.4	78.1	56.8	47.5	36.4
w/ PlaStIL$_1$+oracle	58.6	51.8	42.1	92.1	86.4	78.7	56.8	47.8	36.8
w/ PlaStIL$_2$	59.2	50.2	39.7	92.2	85.1	76.7	57.3	46.9	36.0
w/ PlaStIL$_2$+oracle	59.3	50.3	40.1	92.2	85.2	77.3	57.5	47.1	36.5
w/ PlaStIL$_{all}$	57.7	48.6	39.0	90.7	83.4	75.3	58.2	47.1	35.6
w/ PlaStIL$_{all}$+oracle	57.9	48.7	39.3	90.7	83.7	75.8	58.3	47.5	36.0

Tab. 3.3.: Comparison of PlaStIL on top of DeeSIL [BP18] with a version that knowing the final composition of the different states will only fine-tune the relevant states (PlaStIL +oracle). The average gains are of +0.2% with PlaStIL +oracle.

CIL Method	mem on disk	*ILSVRC*			*Landmarks*			*iNaturalist*		
		K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
REMIND	89.18MB	52.2	44.8	35.9	83.3	77.5	72.2	50.6	39.4	31.3
REMIND	133.78MB	52.3	44.9	36.2	83.4	79.3	75.8	50.9	43.8	35.0
REMIND	222.96MB	52.3	44.4	$\underline{39.7}$	84.2	81.0	$\underline{78.0}$	53.7	46.5	**37.8**
DeeSIL	44.59MB	52.4	45.4	37.5	87.4	80.8	73.8	52.7	43.5	33.9
w/ PlaStIL$_1$	88.44MB	$\underline{58.6}$	**51.8**	**41.9**	$\underline{92.1}$	**86.4**	**78.1**	56.8	**47.5**	$\underline{36.4}$
w/ PlaStIL$_2$	84.52MB	**59.2**	$\underline{50.2}$	$\underline{39.7}$	**92.2**	$\underline{85.1}$	76.7	$\underline{57.3}$	46.9	36.0
w/ PlaStIL$_{all}$	89.18MB	57.7	48.6	39.0	90.7	83.4	75.3	**58.2**	$\underline{47.1}$	35.6

Tab. 3.4.: Average top-1 accuracy with three numbers of states K per dataset, comparison of PlaStIL$_1$ on top of DeeSIL [BP18] with REMIND [Hay+20] with different budgets for the storage of their compressed vectors (1, 2 and 4 times the size of a ResNet18 on disk). **Best results - in bold**, $\underline{\text{second best - underlined}}$.

accuracy of the CIL process by aggregating the accuracy of different incremental states computed on the test set. The results from Table 3.3 indicate that the creation of tops for the most recent states provides a performance level that is close to the optimal one achievable by the oracle. This is explained by the fact that the evaluated CIL scenarios use a random assignment of classes to states. Other selection strategies might be more appropriate if the assumptions made about the order of arrival of classes or data were different, as discussed in Subsection 3.3.2.

Analysis of the number of model tops. The results of Figure 3.6 demonstrate that as the number of tops increases, the accuracy of the model improves. However, it is important to note that this improvement in accuracy comes at the cost of increased memory usage on disk. This trade-off between accuracy and resource usage is a crucial consideration for optimizing model performance in practice in an exemplar-free setting.

Comparison with another type of memory usage. The results from Table 3.4 show that PlaStIL strategy of storing model tops suits better the experiments than storing compressed representation vectors, even for a larger memory on disk.

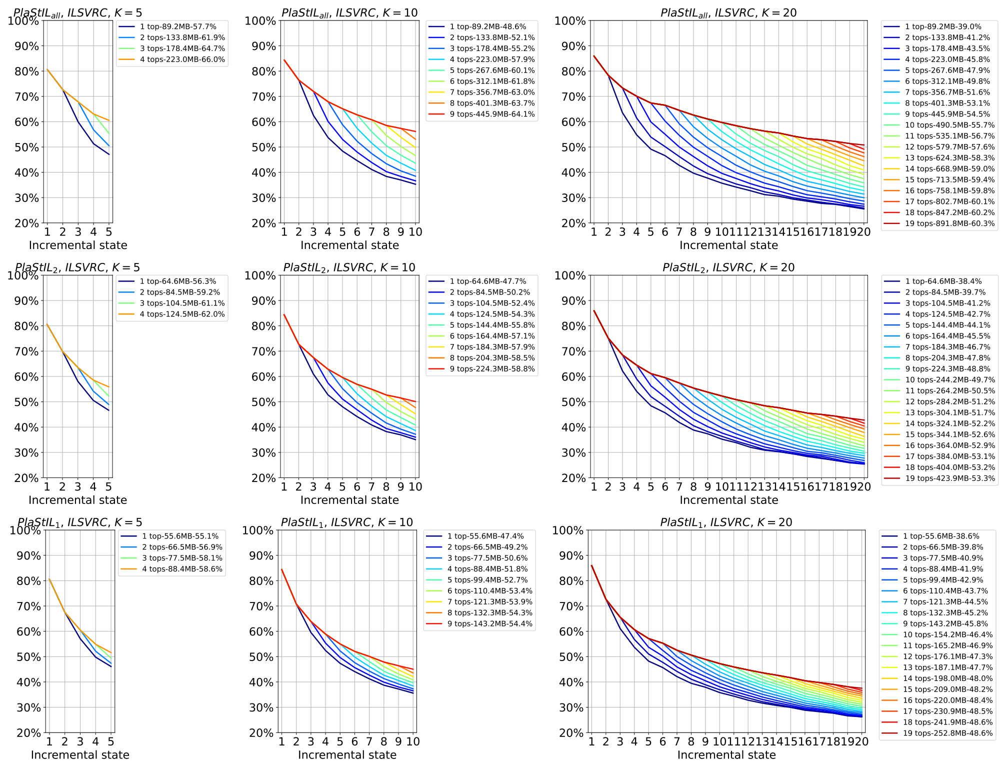


Fig. 3.6.: Effect of varying the number of additional tops on incremental accuracy, on ILSVRC with $K \in \{5, 10, 20\}$. As the number of tops increases, the accuracy of the model improves. However, this comes at the cost of increased memory usage on disk. *Best viewed in color.*

3.5 Conclusion

We proposed a new method that adds a plasticity layer to transfer-based methods in exemplar-free CIL. Plasticity is improved by training dedicated model tops which fine-tune a variable number of deep model layers for one or more recent incremental states. The predictions of the different model tops used by our method are integrated into a hybrid classification layer. Model tops improve accuracy compared to the transfer-based method to which they are added to in order improve plasticity. These improvements are obtained by introducing supplementary parameters so as to have a total footprint that remains lower than that of distillation-based methods. The comparison of these methods with PlaStIL is clearly favorable to the latter. The takeaway is that the parameters allocated to the previous model in distillation-based approaches are better spent on partially fine-tuned model tops. This finding is aligned with those reported in recent comparative studies which question the usefulness of distillation in large-scale CIL with or without memory [BPK21; Mas+21; PTD20]. We believe that future studies should consider transfer-based methods to assess progress in exemplar-free CIL in a fair and comprehensive manner. We plan to optimize the stability-plasticity compromise for EFCIL to further improve the encouraging results reported here. First, we will try to devise a classifier that predicts the most probable initial state for each test sample, following the proposal made in [Ver+21]. If successful, this prediction would reduce the classification space to individual states and remove the need for a hybrid classification layer. Second, we will investigate the use of pre-trained representations learned with larger amounts of supervised or semi-supervised data, as proposed in [Dha+21; Hay+20; Wan+23a]. Such models can be seamlessly integrated into the PlaStIL pipeline.

4 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

— **Robert Frost**
The Road Not Taken (2/4)

4.1 Introduction

As detailled in Chapter 1, Deep learning [GBC16] has dramatically improved the quality of automatic visual recognition, both in terms of accuracy and scale. Current models discriminate between thousands of classes with an accuracy often close to that of human recognition, assuming that sufficient training examples are provided. Unlike humans, algorithms reach optimal performance only if trained with all data at once whenever new classes are learned. This is an important limitation because data often occur in sequences [Lan+19] and their storage is costly. Also, iterative retraining to integrate new data is computationally costly and difficult in time- or computation-constrained applications [HK22; Rav+21]. Incremental learning [SF86] was introduced to reduce the memory and computational costs of machine learning algorithms. The main problem faced by class-incremental learning (CIL) methods is catastrophic forgetting [Kem+18; MC89], the tendency of neural nets to underfit past classes when ingesting new data. As explained in Section 2.5, many solutions [Cas+18; Hou+19; Reb+17; Wu+19; Zha+20], based on deep nets, use replay from a bounded memory of the past to reduce forgetting. However, replay-based methods make a strong assumption because past data are often unavailable [Ven+17]. Also, the footprint of the image memory can be problematic for memory-constrained devices [Rav+21]. Exemplar-free class-incremental learning (EFCIL) methods recently gained momentum [Yu+20; Smi+21; Zhu+21a; Zhu+21b]. Most of them use distillation [HVD15] to preserve past knowledge and generally favor plasticity. New classes are well predicted since models are learned

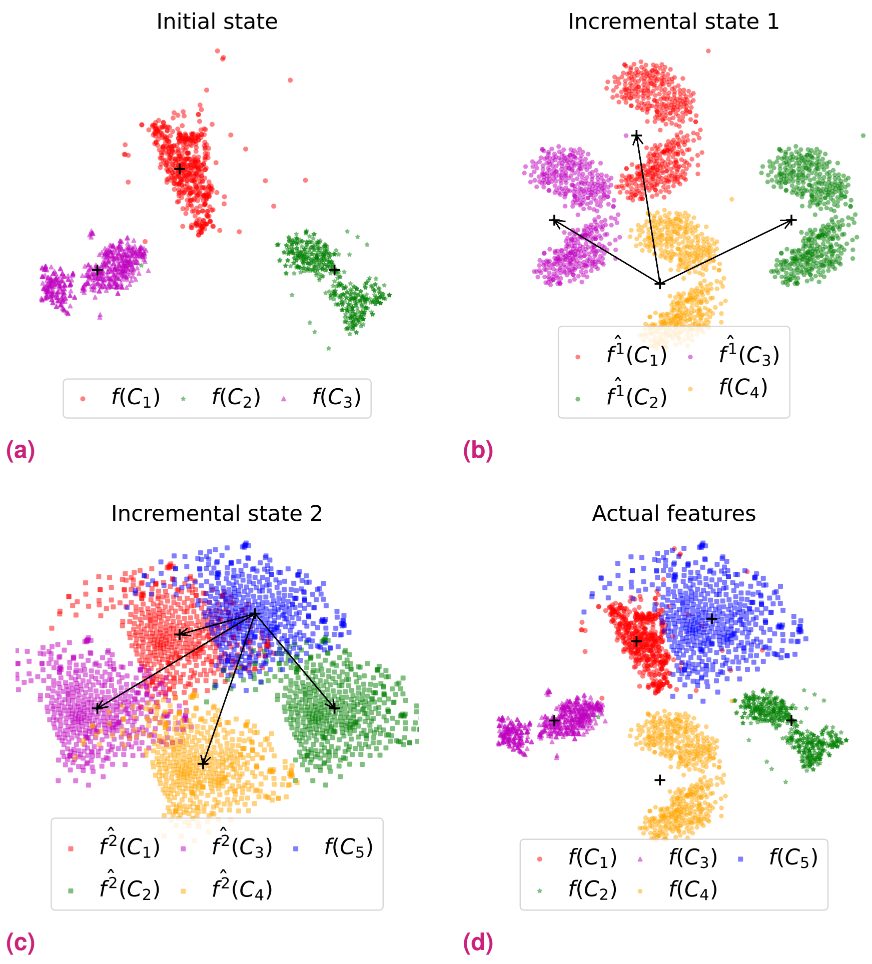


Fig. 4.1.: Illustration of the proposed pseudo-feature generation procedure. This toy example includes an initial state (3 classes) and two IL states (1 new class per state) in subfigures (a), (b) and (c). Subfigure (d) provides the actual features of all classes that would be available for classical learning. The illustration uses a 2D projection of actual features. Pseudo-features of past classes are generated by geometric translation of features of the new class added in each state with the difference between the centroids of the target past class and of the new class. While imperfect, the pseudo-feature generator produces a usable representation of past classes. *Best viewed in color.*

with all new data and only a representation of past data [Mas+21; PTD20; Zhu+22]. A few EFCIL methods [BP18; Dha+21], including PlaStIL [Pet+23a], described in Chapter 3, are inspired by transfer learning [Sha+14; Tan+18]. They learn a feature extractor in the initial state and use it as such later to train new classifiers. In this case, stability is often favored over plasticity since the model is frozen [Mas+21].

We introduce FeTrIL, a new EFCIL method that combines a frozen feature extractor and a pseudo-feature generator to improve incremental performance. New classes are represented by their image features obtained from the feature extractor. Past classes are represented by pseudo-features which are derived from features of new classes by using a geometric translation process. This translation moves features toward a region of the features space which is relevant for past classes. The proposed pseudo-feature generation is adapted for EFCIL since it is simple, fast and only requires the storage of the centroids for past classes. FeTrIL is illustrated with a toy example in Figure 4.1. We run experiments with a standard EFCIL setting [Hou+19; Zhu+21a; Zhu+21b], which consists of a larger initial state, followed by smaller states which include the same number of classes. Results show that the proposed approach has better behavior compared to ten existing methods.

4.2 Related Work

CIL algorithms are needed when data arrive sequentially and/or computational constraints are important [HK22; Lan+19; Mas+21; Par+19]. Their objective is to ensure a good balance between plasticity, i.e. integration of new information, and stability, i.e. preservation of knowledge about past classes [MBB13]. This is challenging because the lack of past data leads to catastrophic forgetting, i.e. the tendency of neural networks to focus on newly learned data at the expense of past knowledge [MC89]. Recent reviews of CIL [BPK21; Mas+21] show that a majority of methods replay samples of past classes to mitigate forgetting [Cas+18; Hou+19; Reb+17; Zha+20]. One advantage here is that the network architecture remains constant throughout the incremental process. However, these methods have two major drawbacks:

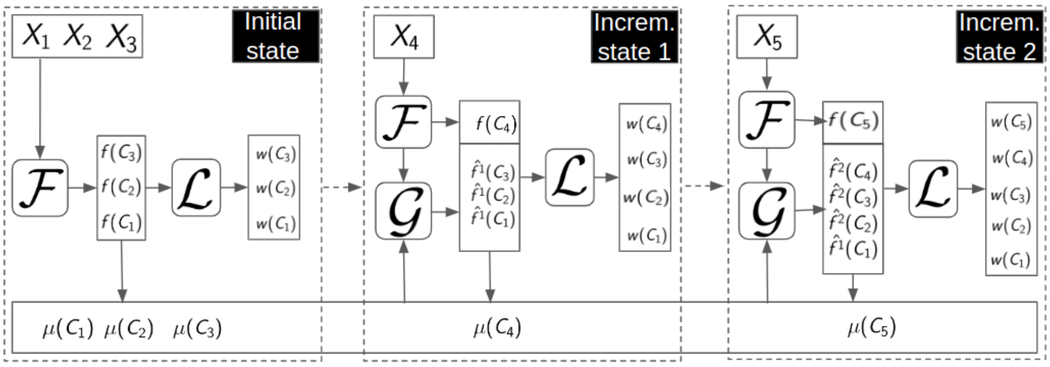
(1) First, the assumption that past samples are available is strong since in many cases past data cannot be stored due, for instance, to privacy restrictions [Ven+17] and

(2) the memory footprint of the stored images is high.

Here, we investigate EFCIL, with a focus on methods that keep the network size constant. This setting is very challenging since it imposes strong constraints on both memory and computational costs. A majority of existing methods use regularization to update the deep model for each incremental step [Mas+21], and adapt distillation [HVD15] to preserve past knowledge by penalizing variations for past classes during model updates. Note that, while some of the distillation-based methods were introduced in an EBCIL setting, many of them are also applicable to EFCIL. This approach to CIL was popularized by iCaRL [Reb+17], itself inspired by learning without forgetting (LwF) [LH16]. Distillation was later refined and complemented with other components to improve the plasticity-stability compromise. LUCIR [Hou+19] applies distillation on features instead of raw classification scores to preserve the geometry of past classes, and an inter-class separation to maximize the distances between past and new classes. The problem was partially addressed by adding specific class separability components in [Dou+20; Hou+19]. Distillation-based methods need to store the current and the preceding model for incremental updates. Their memory footprint is larger compared to methods that do not use distillation [Mas+21].

Another important problem in CIL is the semantic drift between incremental states. Auxiliary classifiers were introduced in [Liu+20c] to reduce the effect of forgetting. ABD [Smi+21] uses image inversion to produce pseudo-samples of past classes. The method is interesting but image inversion is difficult for complex datasets. Another interesting solution is proposed in [Yu+20], where the features drift between incremental steps is estimated from that of new classes. Recent EFCIL approaches [Zhu+21a; Zhu+21b; Zhu+22], described in Section 2.3, use past class prototypes in conjunction with distillation to improve performance. Prototype augmentation is proposed in PASS [Zhu+21b] to improve the discrimination of classes learned in different incremental states. Feature generation for past classes is introduced in IL2A [Zhu+21a] by leveraging information about the class distribution. This approach is difficult to scale up because a covariance matrix needs to be stored for each class. A prototype selection mechanism is introduced in SSRE [Zhu+22] to better discriminate past from new classes. FeTrIL shares the idea of using class prototypes with [Yu+20; Zhu+21a; Zhu+21b; Zhu+22]. An important difference is that we freeze the model after the initial state, while the other methods deploy more sophisticated mechanisms to integrate prototypes in a knowledge distillation process. Past comparative studies [BPK21; Mas+21] found that, while appealing in theory, distillation-based methods underperform in EFCIL, particularly for large-scale datasets. Second, since the representation space is fixed, a simple geometric translation of actual features of new classes is sufficient to produce usable pseudo-features. In contrast, IL2A [Zhu+21a], the work which is closest to ours, needs to store a covariance matrix per class to obtain optimal performance. Third, the use of a fixed extractor simplifies the training process since only the final linear layer is trained, compared to a fine-tuning of the backbone model required by recent methods which use prototypes and feature generation.

Another line of work, described in Section 2.2, takes inspiration from transfer learning [NSZ20; Sha+14] to tackle EFCIL. A feature extractor is trained in the initial non-incremental state and fixed afterward. Then, an external classification layer is updated in each incremental state to integrate new classes. The nearest class mean (NCM) [Men+13] was used in [Reb+17], linear SVMs [Ped+12] were used in [BP18] and extreme value



X_1 X_2 X_3
Initial state
$\mathcal{F}$
$f(C_3)$ $f(C_2)$ $f(C_1)$
$\mathcal{L}$
$w(C_3)$ $w(C_2)$ $w(C_1)$
$\mu(C_1)$ $\mu(C_2)$ $\mu(C_3)$
X_4
Increm. state 1
$\mathcal{F}$
$\mathcal{G}$
$f(C_4)$
$\hat{f}^1(C_3)$ $\hat{f}^1(C_2)$ $\hat{f}^1(C_1)$
$\mathcal{L}$
$w(C_4)$ $w(C_3)$ $w(C_2)$ $w(C_1)$
$\mu(C_4)$
X_5
Increm. state 2
$\mathcal{F}$
$\mathcal{G}$
$f(C_5)$
$\hat{f}^2(C_4)$ $\hat{f}^2(C_3)$ $\hat{f}^2(C_2)$ $\hat{f}^1(C_1)$
$\mathcal{L}$
$w(C_5)$ $w(C_4)$ $w(C_3)$ $w(C_2)$ $w(C_1)$
$\mu(C_5)$

Fig. 4.2.: FeTrIL overview for a toy example with an initial state (3 classes) and two incremental states (1 class per state). The feature extractor $\mathcal{F}$ is trained in the initial state, using sets of data X_1, X_2, X_3, and then frozen afterward. The generator $\mathcal{G}$ uses features $f(C_n)$ of the new class extracted with $\mathcal{F}$ and prototypes of past classes $\mu(C_p)$ to generate pseudo-features of past classes $\hat{f}^t(C_p)$ in the t^{th} state. Prototypes ($\mu(C_i)$) are the centroids of all classes (past and new). They are learned when classes are first seen and then stored throughout the IL process. A linear classifier $\mathcal{L}$ is used to learn classification weights $w(C_i)$ for all seen classes (past and new).

machines [Rud+17] were tested by [Dha+21]. The advantages of transfer-learning methods are their simplicity, since only the classification layer is updated, and their lower memory requirement since they need a single deep model to function. These methods give competitive performance compared to distillation-based ones in EFCIL, particularly at scale [BPK21]. However, features are not updated, and they are sensitive to large domain shifts between incremental tasks [Lan+19]. Equally, existing transfer-learning-inspired works do not sufficiently address inter-class separability, which is in focus here.

Class prototype creation was studied in other learning settings than CIL. A very interesting method focused on few-shot learning was proposed in [DL19]. A distance-based classifier that uses an approximation of the Mahalanobis distance is proposed. The means and variances of new classes are predicted using two supplementary neural networks. While adapted for few-shot learning, such an approach is not fully adapted in CIL. First, the supplementary neural networks require a large number of supplementary parameters. This is a disadvantage here since CIL methods are needed in computationally-constrained environments. Second, we do not focus on few-shot learning and the means of past classes are well-placed in the representation space.

4.3 Proposed Method

The objective of CIL is to learn a total of N classes which appear sequentially during training. This process includes an initial state (0) and T incremental ones. New classes need to be recognized alongside past classes which were learned in previous states. We focus on the EFCIL setting [Reb+17; Smi+21; Yu+20; Zhu+22], which assumes that no past images can be stored. This scenario is more challenging than EBCIL since catastrophic forgetting needs to be tackled without resorting to replay [Mas+21]. There is no intersection between the classes learned in different incremental states.

The global functioning of FeTrIL is illustrated in Figure 4.2. It uses a feature extractor, a pseudo-feature generator based on geometric translation, and an external classification layer in order to address EFCIL. Inspired by transfer-learning based CIL [BP18; Reb+17], the feature extractor $\mathcal{F}$ is frozen after the initial state. This ensures a stable representation space throughout the entire CIL process. Given that images of past classes cannot be stored in EFCIL, a generator $\mathcal{G}$ is used to produce pseudo-features of past classes ($\hat{f}^t(C_p)$). $\mathcal{G}$ takes features of new classes ($f(C_n)$) and prototypes of past and new classes ($\mu(C_p)$, $\mu(C_n)$) as inputs. A linear classifier L combines features and pseudo-features to jointly train classifiers for all seen classes (past and new). The pseudo-features generation is crucial since it enables class discrimination across all incremental states. The hypotheses made here are that:

(1) while imperfect, the pseudo-features still produce effective representations of past classes, and

(2) using a frozen extractor in combination with a generator in EFCIL is preferable to mainstream distillation-based methods [Yu+20; Zhu+21a; Zhu+21b; Zhu+22].

These hypotheses are tested through the extensive experiments in Section 4.4. We present the main components of FeTrIL in the next subsections.

4.3.1 Generation of pseudo-features

The pseudo-feature generator, illustrated in Figure 4.1, produces effective representations of past classes. Existing approaches which generate past data rely on methods such as generative adversarial networks [He+18], image inversion [Smi+21], or covariance-based past class models [Zhu+21a]. We propose a much simpler alternative which is defined as:

$$\hat{f}^t(c_p) = f(c_n) + \mu(C_p) - \mu(C_n) \tag{4.1}$$

with: C_p - target past class for which pseudo-features are needed; C_n - new class for which images b are available; $f(c_n)$ - features of a sample c_n of class C_n extracted with $\mathcal{F}$; $\mu(C_p), \mu(C_n)$ - mean features of classes C_p and C_n extracted with $\mathcal{F}$; $\hat{f}^t(c_p)$ - pseudo-feature vector of a pseudo-sample c_p of class C_p produced in the t^{th} incremental state.

Eq. 4.1 translates the value of each dimension with the difference between the values of the corresponding dimension of $\mu(C_p)$ and $\mu(C_n)$. It creates a pseudo-feature vector situated in the region of the representation space associated to target class C_p based on actual features of a new class $f(C_n)$. The computational cost of generation is very small since it only involves additions and subtractions. $\mu(C_p)$ is needed to drive the geometric translation toward a region of the representation space which is relevant for C_p. Centroids are computed when classes occur for the first time and then stored. Their reuse is possible because $\mathcal{F}$ is fixed after the initial step and its associated features do not evolve.

4.3.2 Selection of pseudo-features

Eq. 4.1 translates the features for a single sample. If each class is represented by s samples, the generation process needs to be repeated s times. The overview of FeTrIL (Figure 4.2) and of the pseudo-feature generation (Figure 4.1) use a minimal example which adds a single class per IL state. When CIL states include several classes C_n, the s pseudo-features of each class C_p can be obtained using different strategies, depending on how features of new classes are used. We deploy the following strategies:

- FeTrIL^k: s features are transferred from the k^{th} similar new class of each past class C_p. Similarities between the target C_p and the C_n available in the current state is computed using the cosine similarity between the centroids of each pair of classes. Experiments are run with different values of k to assess if a variable class similarity has a significant effect on EFCIL performance. Since translation is based on a single new class, the distribution of pseudo-features will be similar to that of features of C_n, but in the region of the representation space around $\mu(C_p)$.
- FeTrIL^{rand}: s features are randomly selected from all new classes. This strategy assesses whether a more diversified source of features from different C_n produces an effective representation of class C_p.
- FeTrIL^{herd}: s features are selected from any new class based on a herding algorithm [Wel09]. It assumes that sampling should include features which produce a good approximation of the past class. Herding was introduced in EBCIL in order to obtain an accurate approximation of each class by using only a few samples [Reb+17] and its usefulness was later confirmed [BPK21; Hou+19; Wu+19]. It is adapted here to obtain a good approximation of the sample distribution of C_p with s pseudo-features.

The comparison of these different strategies will allow us to determine whether the geometric translation of features is prevalent, or if a particular configuration of the features around the centroid of the target past class is needed.

4.3.3 Linear classification layer training

We assume that the CIL process is in the t^{th} CIL state, which includes P past classes and N new classes. The combination of the feature generator (Subsection 4.3.1) and selection

(Subsection 4.3.2) provides a set $\hat{f}^t(C_p)$ of s pseudo-features for each class C_p. The objective is to train a linear classifier for all $P+N$ seen classes which takes pseudo features of past classes and actual features of new classes as inputs. This linear layer is defined as:

$$\mathcal{W}^t = \{w^t(C_1), ..., w^t(C_P), w^t(C_{P+1}), ..., w^t(C_{P+N})\} \tag{4.2}$$

with: w^t - the weight of known classes in the t^{th} CIL state.

$\mathcal{W}^t$ can be implemented using different classifiers, and we instantiate two versions in Section 4.4: (1) FeTrIL using LinearSVCs [Ped+12] as external classifiers, and (2) FeTrIL$_{fc}$ using a fully-connected layer to enable end-to-end training.

4.4 Evaluation

We evaluate FeTrIL by using a comprehensive EFCIL evaluation scenario [Zhu+21a; Zhu+21b; Zhu+22]. This setting includes four datasets and CIL states of different sizes.

Datasets. We use four public datasets:

(1) CIFAR-100 [Kri09] - 100 classes, 32x32 pixels images, 500 and 100 images/class for training and test;

(2) TinyImageNet [LY15] - 200 leaf clases from ImageNet, 64x64 pixels images, 500 and 50 for training and test;

(3) ImageNet-Subset - 100 classes subset of ImageNet LSVRC dataset [Rus+15], 1300 and 50 for training and test;

(4) ILSVRC - full dataset from [Rus+15].

Incremental setting. We use a classical EFCIL protocol from [Zhu+21a; Zhu+21b; Zhu+22]. The number of classes in the initial state is larger, and the rest of the classes are evenly distributed between incremental states. CIFAR-100 and ImageNet-Subset are tested with:

(1) 50 initial classes and 5 IL states of 10 classes,

(2) 50 initial classes and 10 IL states of 5 classes,

(3) 40 initial classes and 20 states of 3 classes, and

(4) 40 initial classes and 60 states of 1 class.

Compared to [Zhu+21a; Zhu+21b; Zhu+22], configurations (1) and (3) for ImageNet-Subset are added for more consistent evaluation. TinyImageNet is tested with 100 initial classes and the other classes are distributed as follows:

(1) 5 states of 20 classes,

(2) 10 states of 10 classes,

(3) 20 states of 5 classes, and

(4) 100 states of 1 class.

Configuration (4) is interesting since it enables one-class increments, which brings us in the One-Class Incremental scenario, described in Figure 2.1. It cannot be deployed for any of the compared EFCIL methods since they require at least two classes per increment to update models. ILSVRC is tested with 500 initial classes, and the other 500 are split evenly among $T \in \{5, 10, 20\}$ states. This enables a comprehensive comparison of the methods in varied EFCIL configurations. Naturally, task IDs are not available at test time.

Compared methods. We use the following EFCIL methods in evaluation: EWC [Kir+17], LwF-MC [Reb+17], DeeSIL [BP18], LUCIR [Hou+19], MUC [Liu+20c], SDC [Yu+20], PASS [Zhu+21b], ABD [Smi+21], IL2A [Zhu+21a], SSRE [Zhu+22]. As we discussed in Section 4.2, these methods cover a large variety of EFCIL approaches. The inclusion of recent works [Zhu+21a; Zhu+21b; Zhu+22] is important to situate our contribution with respect to current EFCIL trends. While focus is on EFCIL, we follow [Zhu+22] and include a comparisonwith EBCIL methods. We test our method against the AANets approach [LSS21], and against the EBCIL methods to which AANETS was added (LUCIR [Hou+19], Mnemonics [Liu+20b], PODNet [Dou+20]). Whenever available, results of compared methods marked with * are reproduced either from their initial paper or from [Zhu+22] for EFCIL or from [LSS21] for EBCIL. The other results are recomputed using the original configurations of the methods.

Implementation details. Following [Reb+17; Zhu+21a; Zhu+21b; Zhu+22], we use ResNet-18 [He+16] in all experiments. FeTrIL initial training is done uniquely with images of initial classes to ensure comparability with existing methods. The feature extractor is trained in the initial state and then frozen for the remainder of the IL process. We implement supervised training with cross-entropy loss, SGD optimization, and a batch size of 128, for a total of 160 epochs. The initial learning rate is 0.1, and it decays by 0.1 after every 50 epochs. To ensure comparability, classes are assigned to IL states using the same random seed as in the compared methods [Hou+19; Zhu+21b; Zhu+21a; Zhu+22].

CIL Method	CIFAR-100				TinyImageNet				ImageNet-Subset			
	T=5	T=10	T=20	T=60	T=5	T=10	T=20	T=100	T=5	T=10	T=20	T=60
EWC* [Kir+17]	24.5	21.2	15.9	x	18.8	15.8	12.4	x	-	20.4	-	x
LwF-MC* [Reb+17]	45.9	27.4	20.1	x	29.1	23.1	17.4	x	-	31.2	-	x
LUCIR [Hou+19]	51.2	41.1	25.2	x	41.7	28.1	18.9	x	56.8	41.4	28.5	x
MUC* [Liu+20c]	49.4	30.2	21.3	x	32.6	26.6	21.9	x	-	35.1	-	x
SDC* [Yu+20]	56.8	57.0	58.9	x	-	-	-	x	-	61.2	-	x
ABD* [Smi+21]	63.8	62.5	57.4	x	-	-	-	x	-	-	-	x
PASS* [Zhu+21b]	63.5	61.8	58.1	x	49.6	47.3	42.1	x	64.4	61.8	51.3	x
IL2A* [Zhu+21a]	66.0	60.3	57.9	x	47.3	44.7	40.0	x	-	-	-	x
SSRE* [Zhu+22]	65.9	65.0	**61.7**	x	50.4	48.9	48.2	x	-	67.7	-	x
DeeSIL [BP18]	60.0	50.6	38.1	x	49.8	43.9	34.1	x				
DSLDA [HK20]	64.0	63.8	60.8	**60.5**	53.1	52.9	**52.8**	**52.6**	71.3	**71.2**	**71.0**	**70.8**
FeTrIL^1	**66.3**	**65.2**	61.5	59.8	**54.8**	**53.1**	52.2	50.2	**72.2**	**71.2**	67.1	65.4

CIL Method	ImageNet		
	T=5	T=10	T=20
LUCIR [Hou+19]	47.4	37.2	26.6
DeeSIL [BP18]	61.9	54.6	45.8
DSLDA [HK20]	64.0	63.8	63.6
FeTrIL^1	**66.1**	**65.0**	**63.8**

Tab. 4.1.: Average top-1 incremental accuracy in EFCIL with different numbers of incremental steps. FeTrIL^1 results are reported with pseudo-features translated from the most similar new class. "-" cells indicate that results were not available. "x" cells indicate that the configuration is impossible for that method. **Best results - in bold**, second best - underlined.

We provide implementation details for the final layer (Eq. 4.2) introduced in Subsection 4.3.3. The hyperparameters of the classification layers were optimized on a pool of 50 classes selected randomly from ImageNet, but disjoint from ILSVRC or ImageNet-Subset. L2-normalization is applied before the linear layer. The LinearSVC layer included in FeTrIL^1 uses 1.0 and 0.0001 for regularization and the tolerance parameters. The number of samples is higher than the dimensionality of the features, and we solve the primal rather than the dual optimization problem. The classifiers are then trained using a standard one-against-the-rest procedure. In Subsection 4.4.2, we also test a one-vs-many strategy to accelerate incremental updates. The second variant, FeTrIL$^1_{fc}$, uses a fully-connected layer as the final layer and implements an end-to-end training strategy. FeTrIL$^1_{fc}$ is trained for 50 epochs with an initial learning rate of 0.1, 0.1 decay, and 10 epochs patience.

Evaluation metric. The average incremental accuracy, widely used in CIL [Mas+21; Reb+17], is the main evaluation measure. For comparability with [Zhu+21a; Zhu+21b; Zhu+22], it is computed as the average accuracy of all states, including the initial one. We equally provide per-state accuracy curves to have a more detailed view of the accuracy evolution during the CIL process. Following [Zhu+22], we run each configuration of FeTrIL three times and report the averaged results.

CIL Method	CUB200		Flower102	
	$T = 5$	$T = 10$	$T = 5$	$T = 10$
SDC[Yu+20]	70.0	65.8	86.8	80.4
FeTrIL[1]	**71.6**	**71.0**	**90.4**	**89.7**

Tab. 4.2.: Comparison of SDC [Yu+20] with FeTrIL[1] using the evaluation protocol for two supplementary datasets used in [Yu+20]. **Best results in bold.**

4.4.1 Results

Comparison to existing EFCIL methods. The results from Table 4.1 show that FeTrIL[1] outperforms all compared methods in 11 tested configurations out of 12. It is also close to the best in the remaining one. The second best results are obtained with the recent SSRE method [Zhu+22]. FeTrIL[1] and SSRE accuracies are close to each other for CIFAR-100, with relative differences between 0.4 and -0.2. The performance gain brought by FeTrIL is of over 4 and 3 top-1 accuracy points for TinyImageNet and ImageNet-Subset, respectively. PASS [Zhu+21b] and IL2A [Zhu+21a], two other recent EFCIL methods, have lower average performance. We note that EFCIL performance boost was recently reported, with methods such as PASS, IL2A, and SSRE. These methods combine knowledge distillation and sophisticated mechanisms for dealing with the stability-plasticity dilemma. In contrast, our method uses a fixed feature extractor and a lightweight pseudo-feature generator. FeTrIL only optimizes a linear classification layer, while compared recent methods use backpropagation of the entire model, and need much more computational resources and time to perform the IL process. A more in-depth discussion of complexity is proposed in Subsection 4.4.2. The performance of the ILSVRC dataset is also very interesting. Direct comparison to PASS or SSRE is impossible since these methods were not tested at scale. However, we can safely assume that FeTrIL[1] is better given PASS and SSRE accuracy for the simpler ImageNet-Subset. ILSVRC results show that the simple method proposed here is effective for a high range of classes. Interestingly, ILSVRC performance is stabler compared to smaller datasets since the pool of new classes available for pseudo-features generation is larger.

In Table 4.2, we compare FeTrIL to SDC [Yu+20] using the evaluation protocol and datasets from [Yu+20]. Half of the datasets are assigned to the initial state and the rest of the classes are split evenly among the remaining states. Following [Yu+20], the training of the initial FeTrIL model for CUB200 and Flower102 datasets is initialized with a pre-trained ILSVRC model. We do the same here to facilitate comparison with the original paper. The results from Table 4.2 indicate that FeTrIL[1] is clearly better than SDC [Yu+20] in all tested configurations.

CIL Method	ImageNet50	ImageNet100
	$T = 5$	$T = 20$
ABD[Smi+21]	71.5	12.1
FeTrIL[1]	**89.0**	**39.0**

Tab. 4.3.: Comparison of ABD [Smi+21] with FeTrIL using the authors' evaluation protocol. ImageNet50 includes 50 classes and 5 states of 10 classes. ImageNet100 includes 100 classes, with 20 states of 5 classes each. Note that [Smi+21] uses top-5 accuracy for ImageNet50 and top-1 for ImageNet100 and we present the same numbers. **Best results in bold.**

CIL Method	CIFAR-100		ImageNet-Subset	
	$T = 5$	$T = 10$	$T = 5$	$T = 10$
LUCIR [Hou+19]	63.2	61.1	70.8	68.3
+AAnets	66.7	65.3	72.6	69.2
Mnemonics [Liu+20b]	63.3	62.3	72.6	71.4
+AAnets	67.6	65.7	72.9	71.9
PODNet [Dou+20]	64.8	63.2	75.5	74.3
+AAnets	66.3	64.3	77.0	75.6
FeTrIL[1]	66.3	65.2	71.9	70.8

Tab. 4.4.: Comparison of FeTrIL with AANets [LSS21], applied on top of EBCIL baselines which store 20 exemplars of past classes to mitigate catastrophic forgetting.

In Table 4.3, we present results obtained with FeTrIL and Always Be Dreaming (ABD) [Smi+21] a method that combines distillation and image inversion to address EFCIL. The comparison is done for two ILSVRC [Rus+15] subsets which include 50 and 100 classes, respectively. FeTrIL outperforms ABD by a large margin in both configurations. This result is explained by the difficulty of deploying image inversion in an efficient manner for visually complex images, such as those included in ImageNet.

Comparison to EBCIL methods. This comparison is interesting because EFCIL is a much more challenging task than EBCIL [BPK21; Mas+21], and an important performance gap between the two was observed. This is intuitive since the storage of images of past classes in EBCIL mitigates catastrophic forgetting. Following [Hou+19; LSS21], a memory of 20 images per class is allowed for all EBCIL methods tested here. FeTrIL is better than all three base methods to which AANets is applied for CIFAR-100. For ImageNet-Subset, FeTrIL accuracy is better than LUCIR's, slightly behind that of Mnemonics [Liu+20b] and approximately 3.5 points lower than that of PODNet [Dou+20]. The performance of FeTrIL remains close to that of EBCIL methods in a majority of cases even after the introduction of AANets. The results from Table 4.4 indicate that, while still present, the gap between EFCIL and EBCIL methods is narrowing.

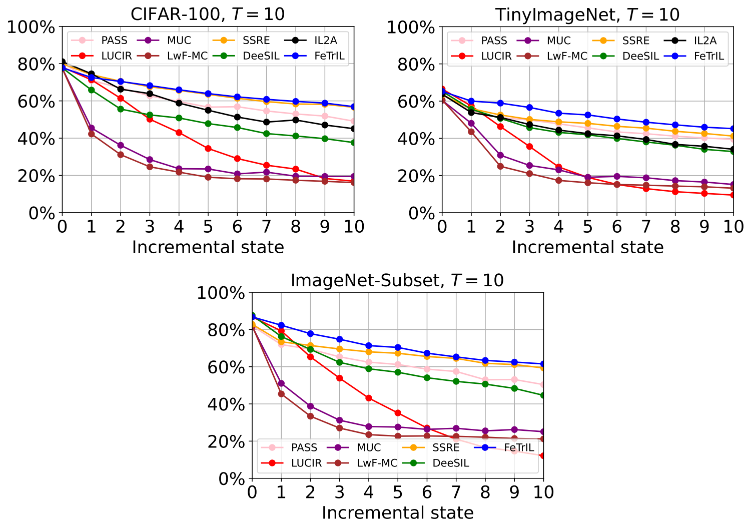


Fig. 4.3.: Evolution of top-1 accuracy for an incremental process with $T = 10$ IL states. *Best viewed in color.*

Comparison to a transfer-learning baseline. DeeSIL [BP18] is a simple application of transfer learning to EFCIL. It has no class separability mechanism across different incremental states since classifiers are learned within each state. The need for global separability, included in FeTrIL, is shown by the comparison of short and long CIL processes. DeeSIL [BP18] performance is good for $T = 5$ because each class is trained against enough other classes, but drops significantly for $T = 20$, when there are few new classes. The important performance gain brought by FeTrIL highlights the importance of class separability.

Behavior for minimal incremental updates. Compared EFCIL methods can only be updated with a minimum of two classes per CIL state since they use discriminative classifiers, which require both positive and negative samples. In practice, it is interesting to enable updates once each new class is available. This is possible with FeTrIL because pseudo-features can all originate from a single new class. Results in the right columns of CIFAR-100, TinyImageNet, and ImageNet-Subset from Table 4.1 show that the accuracy obtained with one class increments is close to that observed for $T = 20$. This highlights the robustness of FeTrIL with respect to frequent updates.

Influence of the final classification layer. FeTrIL^1 compares favorably with FeTrIL$^1_{fc}$. LinearSVC gives better performance than a fully-connected layer, particularly for a large number of incremental steps. However, FeTrIL$^1_{fc}$ is also competitive, outperforming existing methods in most configurations.

Detailed view of accuracy. We illustrate the evolution of accuracy across incremental states in Figure 4.3 to complement the averaged results from Table 4.1. These detailed results confirm the good behavior of the proposed method. The evolution of accuracy for

	CIFAR-100	TinyImageNet	ImageNet-Subset
		$T = 5$	
FeTrIL^1	66.3	54.8	72.2
FeTrIL^5	65.7	53.8	72.2
FeTrIL^{10}	65.1	53.8	71.6
FeTrIL^{herd}	66.2	53.8	72.1
FeTrIL^{rand}	65.1	51.5	70.3

Tab. 4.5.: Average top-1 CIL accuracy obtained with the variants of pseudo-feature selection from Subsection 4.3.2 for $T = 5$. We set $k = \{1, 5, 10\}$ for the similarity rank between the past and new classes to test the effect of class similarities. There are 10 (CIFAR-100 and ImageNet-Subset) and 20 (TinyImageNet) new classes per state from which to select features translation.

FeTrIL and SSRE is very similar for CIFAR-100, FeTrIL method is better throughout the process for TinyImageNet, and also better than SSRE for the first incremental states for ImageNet-Subset. The performance gain with respect to the other compared methods is much more significant for all incremental states.

Generally speaking, no EFCIL method can ensure a class separability comparable to that provided by standard learning with all images of all classes available simultaneously. The objective is to find a good balance between the stability and the plasticity of EFCIL representations. The experiments from Table 4.1 show that, while imperfect, the combination of features and pseudo-features used in FeTrIL^1 provides better performance compared to methods that update the model using variants of knowledge distillation and more complicated class prototypes.

4.4.2 Method analysis

We present an analysis of: (1) the selection strategies, (2) the memory footprint of the methods, (3) the complexity of model updates, and (4) the stability-plasticity balance.

Pseudo-feature selection comparison. FeTrIL can use any past-new classes combination for translation. In Table 4.5, we compare the selection strategies from Subsection 4.3.2. Accuracy varies in a relatively small range for all strategies, indicating that FeTrIL is robust to the way features of new classes are selected, and it can be successfully implemented with any of the strategies. FeTrIL^1 is better than the other selection methods and this motivates its use in the main experiments. Class similarity matters, but results with FeTrIL^{10} remain interesting. FeTrIL^{herd} also has interesting accuracy, but is slightly behind that of FeTrIL^1. The results from Table 4.5 motivate the use of FeTrIL^1 in the main experiments. Overall, the geometric translation toward the centroid of the past class is by far more important than the new classes features sampling policy. This finding is also supported by the results obtained with a single new class per CIL state (Table 4.1).

Memory footprint. A low memory footprint is a desirable property of incremental learning algorithms because they are most useful in memory-constrained applications [Mas+21; Rav+21; Reb+17], and recommended for embedded devices [HK22]. All EFCIL methods need to store a representation of past classes to counter catastrophic forgetting. Naturally, this representation should be as compact as possible. Mainstream methods (such as LwF-MC [LH16], PASS [Zhu+21b], IL2A [Zhu+21b], and SSRE [Zhu+22]) need to the previous and current deep models during CIL updates for distillation. ResNet-18 [He+16], the most frequent CIL backbone, has approximately 11.4M parameters. Consequently, distillation-based methods require around 22.8M parameters. Transfer-based methods, such as DeeSIL [BP18] and FeTrIL, use only the deep model learned in the initial state and frozen afterwards, and only need 11.4M parameters for the model. DeeSIL does not need supplementary parameters during incremental updates. However, this comes at the cost of poor global discrimination of classes, which is reflected in the final performance. FeTrIL stores the class centroids of past classes in order to perform feature translation. Each class needs 512 parameters, which leads to a supplementary 51.2K and 102.4 memory need for 100 and 200 classes, respectively. The class similarities needed for pseudo-feature selection (Subsection 4.3.2) can be computed sequentially and the added memory cost of this step is negligible. PASS [Zhu+21b], IL2A [Zhu+21a] and SSRE [Zhu+22] also requires the storage of a prototype (mean representation) for each past class and their footprint is equivalent to that of FeTrIL. IL2A [Zhu+21a] additionally stores a covariance matrix per past class (512x512 for ResNet-18) for optimal functioning, which is prohibitive.

Complexity of incremental updates. CIL is useful in resource-constrained environments, and the integration of new classes should be fast [HK22; Rav+21]. Distillation-based methods retrain the full backbone model at each update. This is costly because backpropagation complexity depends on the network architecture, the number of samples and the number of epochs [GBC16]. Updates of transfer-based methods are simpler because they update only the final layer. DeeSIL trains linear classifiers using a one-vs-all procedure within each CIL state. The complexity of one training epoch for all classifiers in a CIL state is $O((\frac{n}{T})^2 sd)$ [BB07], with n - total number of classes in the dataset, d - dimensionality of features and s - samples per class. FeTrIL retrains all linear classifiers, past and new, in each CIL state to improve global separability. Its complexity is $O(n^2 sd)$ in the last incremental state, which includes all classes. However, the one-versus-all training can be replaced with a one-versus-many training with negligible loss of accuracy. A sampling of negative features is performed to respect a predefined ratio r between negatives and positives used to train each classifier. This approximation has $O(rnsd)$ complexity. It is interesting since $r < n$, and is more and more useful as n grows during the IL process since r remains constant.

In Figure 4.4, we present results with different r values for CIFAR-100, TinyImageNet and ImageNet-Subset, $T = 10$. Accuracy drops when negative sampling is performed, but it is close to that of one-vs-all training when $r = 25$ and $r = 10$. Performance drops more significantly for $r = 1$, when each linear classifier is learned with an aggressive sampling of

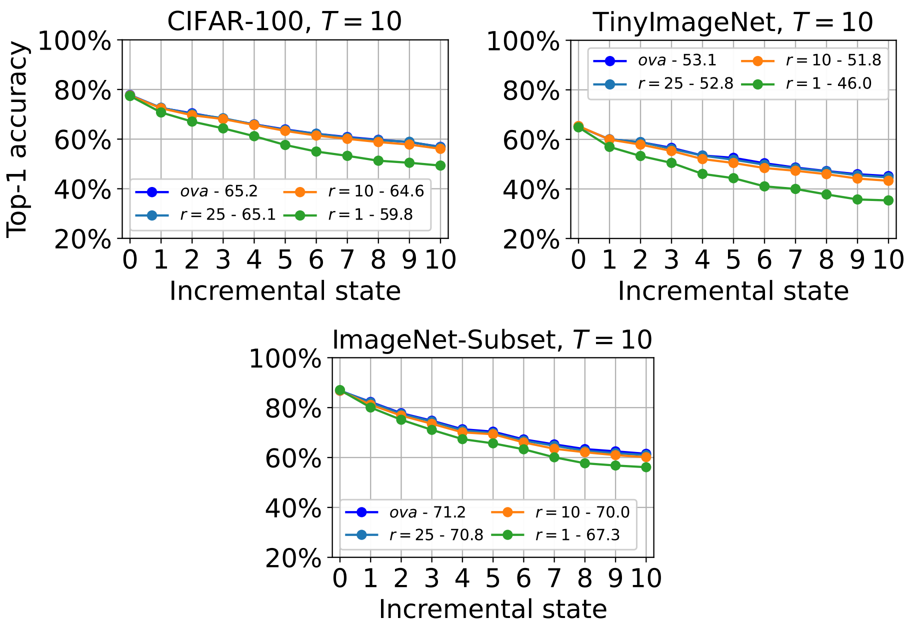


Fig. 4.4.: Top-1 incremental accuracy of FeTrIL[1] for approximate training of the classification layer with different ratios for negative sampling. ova denotes a classical one-vs-all training procedure.

negatives. Globally, Figure 4.4 indicates that FeTrIL increments can be accelerated with little accuracy loss. This highlights the possibility of accelerating the training of FeTrIL with very limited accuracy loss.

We measure the time needed for incremental training of ImageNet-Subset, $T = 10$. The training of the initial model is similar for all models and is thus discarded. FeTrIL training is done on a single thread of an Intel E5-2620v4 CPU, and only takes 1 hour, 4 minutes and 16 seconds. If FeTrIL is run with $r = 10$ ratio between positives and negatives, training time is only 15 minutes and 3 seconds. In comparison, PASS [Zhu+21b] needs 11 hours, 8 minutes and 19 seconds on an NVIDIA V100 GPU, with 4 workers for data loading. While clearly favorable to FeTrIL, the comparison is biased in favor of PASS since this method uses an entire GPU, in comparison to a single CPU thread for FeTrIL. Further speed gains are possible for our method by using a GPU implementation of the linear layer. Our method would run much faster with a GPU implementation of the linear layer. Note that the running time of the other methods, such as LUCIR [Hou+19] and SSRE [Zhu+22], which perform backpropagation is similar to that of PASS [Zhu+21b].

Stability-plasticity balance. CIL should ideally ensure a similar accuracy level for past and new classes [Mas+21; Zhu+22]. Figure 4.5 shows that the two methods have complementary behavior, which results from the way deep backbones are used. SSRE is biased toward new classes since the model is fine-tuned in each incremental state. FeTrIL favors past classes because the deep model is learned with the initial classes (a subset of past classes) and then frozen. The accuracy gap between past and new classes is smaller for FeTrIL compared to

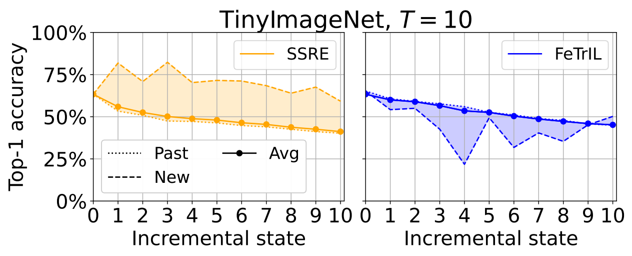


Fig. 4.5.: Top-1 incremental accuracy per state for past and new classes for TinyImageNet, with $T = 10$ incremental states for FeTrIL[1] and SSRE, the best compared method. An ideal method would provide high accuracy, but also similar performance for past and new classes. The accuracy of past and new classes is globally closer for FeTrIL[1] , which indicates that our method provides a better stability-plasticity balance than SSRE. Overall accuracy is better for FeTrIL[1] in Figure 4.3 because the contribution of new classes in each state diminishes during the CIL process.

SSRE, except for state 4. There, low performance on new classes is probably explained by a strong domain shift compared to the initial state. Globally, the proposed method improves the stability-plasticity balance.

4.5 Conclusion

We introduce FeTrIL, a new method that addresses exemplar-free class-incremental learning. The proposed combination of a frozen feature extractor and of a pseudo-feature generator improves results compared to recent EFCIL methods. The generation of pseudo-features is simple since it consists in a geometric translation, yet effective. Our proposal is advantageous from memory and speed perspectives compared to mainstream methods [Hou+19; Reb+17; Smi+21; Ver+21; Yu+20; Zhu+21a; Zhu+21b; Zhu+22]. This is particularly important for edge devices [HK22; Rav+21], whose storage and computation capacities are limited. FeTrIL performance is also close to that of exemplar-based methods, which need to store samples of past classes to mitigate catastrophic forgetting. While a gap between exemplar-based and exemplar-free setting subsists, it becomes significantly narrower. The results reported here resonate with past works which show that simple methods can be highly effective in CIL [BPK21; Mas+21; PTD20]. They question the usefulness of the knowledge distillation component, used by a majority of existing methods. The FeTrIL code will be made public to enable reproducibility.

The main limitations of the proposed method motivate our future work. First, FeTrIL uses a frozen feature extractor learned on the initial state and tends to favor past classes over new ones. We will investigate ways to combine the pseudo-feature generation mechanism and fine-tuning to further improve global performance, as well as the stability-plasticity balance. Second, FeTrIL produces usable pseudo-features, but past class representations would be better if the pseudo-features would be more similar to the original features of past classes. We will study methods that generate more refined features, for instance by using the distribution of the initial features. Last but not least, the tested selection strategies are all effective. However, they could be further improved by filtering out outliers based on the localization of pseudo-features in the representation space.

5 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

— **Robert Frost**
The Road Not Taken (3/4)

Initial training strategy					**CIL Algorithms**					
					BSIL [JLM21]		DSLDA [HK20]		FeTrIL [Pet+23b]	
Arch	Method	FT	Ext	Sup	$\mu_{\overline{Acc}}$	W	$\mu_{\overline{Acc}}$	W	$\mu_{\overline{Acc}}$	W
RN50	CE	✓	×	SL	44.9	0	53.7	4	51.0	0
RN50	CE	×	✓	SL	39.9	0	61.4	0	60.6	0
RN50	CE	✓	✓	SL	62.9	1	65.3	0	68.4	1
RN50	BYOL	✓	×	SSL	11.2	0	42.2	0	34.4	0
RN50	BYOL	×	✓	SSL	35.3	0	63.3	0	62.0	0
RN50	BYOL	✓	✓	SSL	60.2	0	70.0	2	70.2	0
RN50	MoCoV3	✓	×	SSL	14.9	0	49.6	0	41.1	0
RN50	MoCoV3	×	✓	SSL	36.3	0	67.9	1	65.3	0
RN50	MoCoV3	✓	✓	SSL	64.7	2	71.8	2	72.0	0
ViT-S	DeiT	×	✓	SL	35.0	0	58.7	0	56.3	0
ViT-S	DeiT	✓	✓	SL	11.2	0	37.4	0	27.4	0
ViT-S	DINOv2	×	✓	SSL	70.4	4	75.7	9	72.4	6
ViT-S	DINOv2	✓	✓	SSL	24.0	0	45.9	0	39.2	0

Tab. 5.1.: Performance of three EFCIL algorithms with different training strategies for the initial model averaged over 16 target datasets and two EFCIL scenarios. BSIL [JLM21] is a recent EFCIL algorithm that is representative of fine-tuning-based CIL works. DSLDA [HK20] and FetrIL [Pet+23b] adapt linear probing [Kum+22] for EFCIL. We present the averaged incremental accuracy ($\mu_{\overline{Acc}}$) and the number of cases (W) in which a combination of algorithm and initial training strategy performs best for a combination of target dataset and EFCIL scenario (see Sec.5.4). Initial training strategies are defined by: Arch- deep architecture used (ResNet50 (RN50) [He+16] or vision transformer (ViT-S) [Dos+21]); Method - initial training method; FT - fine-tuning on initial classes of the target dataset; Ext- use of an external dataset, such as ILSVRC [Rus+15]; Sup - type of supervision for the initial model: self-supervised (SSL) or supervised (SL).

5.1 Introduction

Real-world applications of Machine Learning (ML) often involve training models from data streams characterized by distributional changes and limited access to past data [HK22; VT19]. This scenario presents a challenge for standard ML algorithms, as explained in Section 1.5, as they assume that all training data is available at once. Continual learning addresses this challenge by building models designed to incorporate new data while preserving previous knowledge [Rin97]. Class-incremental learning (CIL) is a type of continual learning that handles the case where the data stream is made up of batches of classes. As explained in Section 1.6, it is particularly challenging in the exemplar-free case (EFCIL), i.e. when storing examples of previous classes is impossible due to memory or confidentiality constraints [HK20; Zhu+22]. CIL algorithms must find a balance between knowledge retention, i.e. stability, and adaptation to new information, i.e. plasticity [Mas+21; WGL21; MBB13]. Many existing EFCIL methods [JLM21; LH16; Reb+17; Zhu+21a; Zhu+21b; Zhu+22] update the model at each incremental step using supervised fine-tuning combined with a distillation loss, and thus tend to favor plasticity over stability. Another line of work [HK20; Pet+23b] freezes the initial model and only updates the classifier. This approach has recently gained interest [Jan+22; Pel22; Wan+22b] due to the availability of models pre-trained on large external datasets, often through self-supervision [He+20b; Oqu+23]. While pre-trained models provide diverse and generic features, there are limits to their transferability [Abn+21], and these limits have not been studied in depth in the context of EFCIL.

We propose a comprehensive analysis framework to disentangle the factors that influence EFCIL performance. Focus is put on the strategies to obtain the initial model of the incremental process. We consider the type of neural architecture, the training method, the depth of fine-tuning, the availability of external data, and the supervision mode for obtaining this initial model. The initial training strategies are compared using three EFCIL algorithms, representative of the state of the art, on 16 target datasets, under 2 challenging CIL scenarios. The obtained results are summarized in Table 5.1. The main findings are that: (1) pre-training with external data improves accuracy, (2) self-supervision in the initial step boosts incremental learning, particularly when the pre-trained model is fine-tuned on the initial classes, and (3) EFCIL algorithms based on transfer learning have better performance than their fine-tuning-based counterparts. However, the distribution of best performance, presented in Table 5.1, shows that no combination of an EFCIL algorithm and an initial training strategy is best in all cases. This echoes the results of previous studies such as [BPK21; Fei+23]. Therefore, it is interesting to understand the contribution of the different factors influencing EFCIL performance. To this aim, we analyze these strategies in depth in Section 5.5, and use this analysis to formulate EFCIL-related recommendations in Section 5.6. The insights brought by the proposed analysis could benefit both continual learning researchers and practitioners. The proposed framework can improve the evaluation and analysis of EFCIL methods. Continual learning practitioners can use the results of this study to better design their incremental learning systems.

5.2 Background

5.2.1 Pre-training methods

Transfer learning involves using a model trained on a source dataset as a starting point for training another model on a target dataset [RM19]. In the case of transfer learning, the weights of the target model are generally initialized with the weights of the source model. These weights can remain fixed, except for the classification layer (*linear probing*), or they can be updated on the target data (*fine-tuning*). Transfer learning has several practical advantages [Tan+18]. It reduces the computational effort to train a new model on a new dataset. It also enables learning an accurate model in few-shot settings, because models pre-trained on large datasets are able to extract complex features even for new input data. Some authors investigate how to pre-train the model in order to make it more transferable [Gei+18; Tam+17; KSL18]. Model generalization is favored by the quantity, quality, and diversity of its source training data [Oqu+23]. However, the parametric footprint of pre-trained models, typically in the range of hundreds of millions, is often too high for continual learning applications [HK22]. Smaller models can be obtained from larger models through knowledge distillation [HVD15; Tou+21].

Self-Supervised Learning (SSL) has recently gained interest thanks to its ability to produce diverse, reusable features for downstream tasks. SSL enables a model to learn from unlabeled data without relying on explicit annotations [JT20]. It leverages the inherent structure or information present within the data itself to create surrogate labeling tasks e.g. predicting missing image patches, image rotations, or colorizations. For example, MoCov3 [He+20b; CXH21] uses a contrastive loss function to obtain similar representations for two randomly augmented crops of the same input image. Recently, SSL methods trained on large datasets such as BYOL [Gri+20] and DINOv2 [Oqu+23] have provided efficient feature extractors, reusable for other tasks. We note that, while the reuse of pre-trained models as frozen feature extractors is easy, their fine-tuning in the presence of domain shift might be challenging [Kum+22]. This is important in the context of CIL since many existing models are based on fine-tuning.

We compare various pre-training methods to obtain the initial model of a CIL process (Subsec. 5.3.2). We consider (i) the case where the initial model is trained using only the initial batch of data and (ii) the case where an external dataset was available for pre-training. In the first case, the initial model is either obtained using classic supervised learning or using an SSL algorithm, here MoCov3 [CXH21]. In the second case, we start the EFCIL process with a model whose weights have been learned either in a supervised manner, in a self-supervised manner [Gri+20; Oqu+23], or through distillation [Tou+21]. This allows us to study the transferability of the resulting initial models, in combination with various CIL methods.

5.2.2 Class-Incremental Learning (CIL)

continual learning aims to build models that are able to continuously and adaptively learn about their environment. In CIL, learning a classification model is a sequential process, where each step in the sequence consists of integrating a set of new classes into the model [BPK21; Lan+19; Mas+21; Par+19]. In the exemplar-free setting, at a given stage in the process, the model must be able to recognize all the classes encountered so far, with access only to the current batch of classes or with limited access to past data samples.

The main challenge faced by CIL models is their tendency to forget previously acquired information when confronted with new information. This phenomenon is called *catastrophic forgetting* or *catastrophic interference*, as it is caused by the "interference" of new information with previous information [MC89; Fre99]. Forgetting may be reduced by storing examples from past classes, a strategy called *rehearsal* [Reb+17]. However, the availability of past data and the possibility to store it may be unrealistic in practice. Thus, we focus on exemplar-free CIL rather than rehearsal-based CIL.

Two main directions may be considered to deal with forgetting in artificial neural networks. A first family of CIL approaches lets the network grow as new capabilities must be learned, e.g. [WRH17]. In the extreme case, this approach can result in zero forgetting. But at the same time, it is not realistic to make the model grow infinitely. A second family of methods considers a network of constant size throughout the incremental process (except the classifier) and proposes various strategies for obtaining models which ensure a balance between stability, the need to preserve the performance of past classes, and plasticity, needed to recognize new classes. The weights of the initial model may be fine-tuned in combination with knowledge distillation between the previous model and the one which is currently learned. [LH16; Hou+19; JLM21; Dou+20; Zhu+21a; Zhu+21b; Zhu+22]. This type of approach favors plasticity over stability because models are retrained with all data of new classes and the incremental model incrementally learned on past classes. An alternative is to use a pre-trained model or to freeze the model learned in the initial incremental state and to train only a linear classification layer afterwards [BP18; HK20; Pet+23b]. Recent works propose to use a large pre-trained model combined with a k-NN classifier as a challenging baseline for continual learning algorithms [Jan+22; Pel22]. These works adapt linear probing, the basic transfer learning approach [KSL18], to an incremental context. They favor stability over plasticity since the feature extractor is not adapted during the incremental process [Mas+21]. The main challenge is that the initial model needs to be transferable to new classes in order to preserve good performance. Importantly, they are much faster than the fine-tuning-based methods because only the classification layer is trained.

Recent works propose to improve the learned representation by fine-tuning the model with a combination between cross-entropy loss and a self-supervised learning objective [Fin+22; Tan+23]. The authors of [GHK20] explore the use of a fixed feature extractor pre-trained in a self-supervised way. Another recent trend in CIL is to use a pre-trained model as an efficient starting point for the incremental process [Tia+23; Wu+22]. The authors of [Wan+22b] also propose a method based on dynamic prompting. In [Ost+22], pre-trained models are used to propose a compute-low method with a replay of past training samples. Using a pre-trained feature extractor is also interesting for cases where training data is scarce, as in few-shot CIL [Ahm+22].

The present work proposes a comprehensive study of training strategies for the initial model, with a focus on the interaction of these methods with different EFCIL algorithms. We experiment with transformer-based and CNN architectures, in combination with fine-tuning-based and transfer-learning-based EFCIL algorithms.

5.3 Problem statement

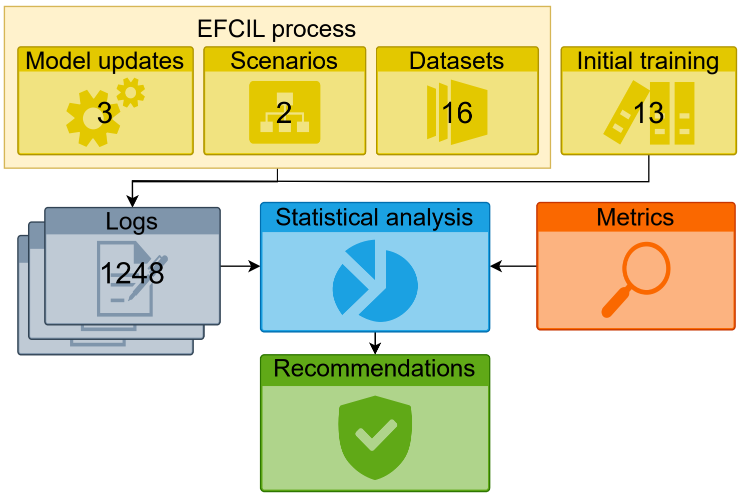


Fig. 5.1.: Overview of the proposed analysis framework of initial training strategies for EFCIL.

We summarize the proposed analysis framework in Figure 5.1. It combines a comprehensive modeling of the EFCIL process and initial training strategies as inputs for a statistical analysis that uses different EFCIL metrics. Recommendations for the design of EFCIL approaches are made based on the conclusions of the statistical analysis.

5.3.1 EFCIL process

Let us consider a dataset $\mathcal{D}$ split over K subsets, $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_K$, and an exemplar-free CIL algorithm $Incr$. A CIL process consists in learning a classification model incrementally over K non-overlapping steps using $Incr$. At each step $k \in [\![1, K]\!]$, the model is updated using $Incr$ and the data subset $\mathcal{D}_k$, whose associated set of classes is denoted by $\mathcal{C}_k$. The data subsets $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K$ composing the complete dataset $\mathcal{D}$ satisfy the following constraint: for $k, k' \in \{1, 2, \ldots, K\}$ with $k \neq k'$, $\mathcal{C}_k \cap \mathcal{C}_{k'} = \emptyset$, i.e. each class is only present in a single data subset. The use of an exemplar-free algorithm $Incr$ implies that when the training is performed at the k^{th} step, no example from any of the data subsets of the previous steps can be accessed. Although this is a more difficult setting, it is also more realistic in practice [HK22; BPK21].

Incremental model updates. The initial model $\mathcal{M}_1$ is obtained following one of the training strategies presented in 5.3.2. At the k^{th} step of the CIL process, $k \in [\![2, K]\!]$, the classification model $\mathcal{M}_k$ recovers the weights of the model $\mathcal{M}_{k-1}$ obtained in step $k-1$ and is updated using the data subset $\mathcal{D}_k$ and the algorithm $Incr$. Many EFCIL algorithms [JLM21] perform a full fine-tuning of the network weights at each incremental step, thus favoring plasticity. Alternatively, algorithms such as [HK20; Pet+23b] only retrain the classifier, thus favoring stability. As a compromise, it is also possible to freeze a part of the model and update only the last layers. We cover these three cases in our experiments.

Scenario. A scenario is characterized by the distribution of classes among the steps of the CIL process. We denote by b the proportion of the classes available in the initial step: $b = Card(\mathcal{C}_1)/Card(\mathcal{C})$. There are two commonly used scenarios [BPK21] (i) equal splitting of classes across the steps or (ii) half of the classes in the first step and the rest of the classes are divided equally between subsequent steps.

5.3.2 Training strategies for the initial model

In the following, we describe the main characteristics of the training strategies used in our experimental study to obtain the initial model of the incremental learning process. Further experimental settings are reported in Section 5.4.

Network architecture. So far, most CIL methods have been proposed in combination with a convolutional neural network, but visual transformer (ViT) networks have recently gained popularity in CIL [Dou+22]. In order to provide a fair comparison between the two types of architecture, we use a ResNet50 [He+16], and a ViT-Small [Dos+21] network, which contains a similar number of parameters (23.5M and 22.1M parameters respectively).

Model initialization. At the first step of the CIL process, the weights of the model may either be randomly initialized or transferred from a pre-trained model. In the second case, depending on the choice of the user, the dataset $\mathcal{D}^*$ used for pre-training may either be an auxiliary dataset (e.g. ILSVRC [Rus+15]), referred to as *source* dataset, or the first data subset $\mathcal{D}_1$ of the incremental process.

Label availability. We consider that all examples from the target dataset $\mathcal{D}$ are labeled, and we experiment with both supervised learning and self-supervised learning to obtain the initial model using $\mathcal{D}_1$. Labels may not be available for the external dataset $\mathcal{D}^*$. In this case, the training initialization is performed using a self-supervised pre-training algorithm (e.g., DINOv2 [Oqu+23]).

5.4 Experimental setting

We describe the experimental parameters and the metrics we use to evaluate EFCIL models. The combination of parameters results in 1,248 experiments in total (Figure 5.1).

5.4.1 Initial training strategies

We compare different strategies for training an initial model, as summarized in Table 5.1. We use Resnet50 [He+16] and ViT-S [Dos+21] networks, which are representative of CNNs and transformers and have similar sizes ($\sim$20M parameters). The training is done either using a self-supervised method (BYOL [Gri+20], DINOv2 [He+20b], MoCov3 [CXH21]) or a supervised one (DeiT and cross-entropy (CE)). We present results for pre-training with external data (i.e. ILSVRC [Rus+15] for BYOL, DeiT and CE; a 150M-images dataset + ILSVRC for DINOv2) and training on the first batch. We compare the effect of freezing the weights of (i) the pre-trained model or (ii) further optimizing the last layers of the model (e.g. the last convolutional block in ResNet50) on the initial data subset $\mathcal{D}_1$. The first type of experiment is denoted by the suffix "*-t*", the second by the suffix "*-ft*". In the case where the pre-training algorithm is applied to $\mathcal{D}_1$ and not to $\mathcal{D}^\star$, there is no suffix.

5.4.2 Target datasets

For a comprehensive evaluation and to account for the diversity of visual tasks, we evaluate the training strategies on 16 target datasets, sampled from publicly available datasets. They cover different domains (plants, animals, landmarks, food, faces, traffic signs etc.), and different types of images (natural, drawings, paintings). IMN100$_1$ and IMN100$_2$ consist of 100 classes randomly selected from ImageNet-21k [Den+09]. Flora is a thematic subset of ImageNet consisting of 100 classes belonging to the "flora" concept. IMN100$_1$, IMN100$_2$ and Flora have no mutual overlap and no overlap with ILSVRC [Den+09; Rus+15]. Amph100 and Fungi100, sampled from iNaturalist [Van+18], respectively contain 100 classes of amphibians and fungi, selected so as to avoid overlap with animal and fungi classes from ILSVRC. We also sample 100-class subsets from other popular datasets: WikiArt100 [SE15], Casia100 [Yi+14], Food100 [BGV14], Air100 [Maj+13], MTSD100 [MY16], Land100 [Wey+20], Logo100 [Wan+20] and Qdraw100 [HE17]. Finally, we consider three 1000-class subsets: Casia1k [Yi+14], Land1k [Noh+17], and iNat1k [Van+18]. The number of training images per dataset varies from 60 to 750. More details on the datasets are provided in the appendix of this thesis, at Chapter B.

5.4.3 Incremental learning

EFCIL scenario b. We experiment on two widely used CIL scenarios [Hou+19; BPK21]. In the first scenario, the classes are equally distributed over 10 steps, e.g. 10 classes per step for a 100-class dataset. In the second scenario, half of the classes are learned in the initial step, and the other half is equally distributed over 10 incremental steps, e.g. $50 + 10 \cdot 5$ classes for a 100-class dataset.

CIL algorithm $Incr$. We experiment with one fine-tuning based algorithm, namely BSIL[JLM21], which adds a balanced softmax without exemplars to LUCIR [Hou+19]. We also experiment with two fixed-representation-based algorithms, namely DSLDA [HK20] and FeTrIL [Pet+23b].

5.4.4 Metrics

The performance of EFCIL models can be evaluated in several ways [Mas+21], discussed below.

Average incremental accuracy $\overline{Acc}$. In EFCIL, a model trained over a K-step incremental process is commonly evaluated using the average incremental accuracy [Zhu+21b; Zhu+22; Zhu+21a; JLM21]. We denote it by $\overline{Acc}$ and compute it by:

$$\overline{Acc} = \frac{1}{K-1} \sum_{k=2}^{K} acc(\mathcal{M}_k, \bigcup_{i=1}^{k} \mathcal{D}_i) \tag{5.1}$$

where $acc(\mathcal{M}, \mathcal{D})$ is the accuracy of the model $\mathcal{M}$ on the dataset $\mathcal{D}$. Following common practice in CIL [Cas+18; Pet+23b; Zhu+22], $\overline{Acc}$ does not take the accuracy of the initial model into account.

Average forgetting F. Average forgetting, denoted here by F, is computed by:

$$F \ = b \times f(\mathcal{D}_1) + \frac{1-b}{K-1} \sum_{k=2}^{K} f(\mathcal{D}_k) \tag{5.2}$$

where $f(\mathcal{D}_k) = \max_{k' \in [\![k,K]\!]} acc(\mathcal{M}_{k'}, \mathcal{D}_k) - acc(\mathcal{M}_K, \mathcal{D}_k))$ is the difference between the best performance achieved on the data subset $\mathcal{D}_k$ during the EFCIL process and the final performance of the model on this data subset [Mir+22].

Initial accuracy Acc_1. To unskew the statistical models we present in Section 5.5, we consider the initial accuracy, defined as the accuracy of the first model on the first data subset $\mathcal{D}_1$ and denoted by Acc_1, i.e. $Acc_1 = acc(\mathcal{M}_1, \mathcal{D}_1)$.

Final accuracy Acc_K. The accuracy of the last model of the incremental learning process on the complete dataset $\mathcal{D}$ is denoted by Acc_K, i.e. $Acc_K = acc(\mathcal{M}_K, \mathcal{D})$.

$\overline{Acc}$ gives more weight to past classes since at each step, the model is evaluated on all seen classes. Consequently, a high average incremental accuracy does not guarantee a high accuracy on the latest classes, particularly when half of the classes are learned initially. Forgetting is complementary to accuracy, as it focuses on model stability. A low value for F indicates that, on average, the performance for a given class remains stable over the incremental process.

5.5 Analysis of results

We present a statistical analysis of the results from Table 5.1, which highlights the effects of pre-training strategies and of EFCIL algorithms on EFCIL performance. The statistical model and associated findings are presented below.

5.5.1 Modeling causal effects

Our objective is to identify the primary factors that influence the performance of EFCIL algorithms. To interpret causal effects, we employ multiple linear regressions using the Ordinary Least Squares (OLS) method, following established statistical and econometric practices [AP09; Gar+13]. In a linear regression, we aim to explain a target variable Y using explanatory variables X_i. The target variable is endogenous, i.e. determined by its relationship with other variables. If the outcome of a variable X_i is selected by the experimenter, it is said to be exogenous, i.e. not caused by other variables. For a given experiment, we denote by Y the target metric accuracy (endogenous), $Data$ the evaluation dataset (exogenous), $Train$ the initial training strategy (exogenous), and $Incr$

the incremental algorithm (exogenous). We also consider the initial accuracy Acc_1 as an endogenous variable that may influence performance and can be controlled in our regressions. Other parameters, such as the total number of classes or the dataset, are examined as potential predictors of a metric.

An OLS regression fits a model of the following form:

$$Y = \beta_0 + \beta_1 Train + \beta_2 Incr + \beta_3 Data + \ldots + \varepsilon, \tag{5.3}$$

where the intercept β_0 is a scalar and ε is assumed to be normally distributed Gaussian noise. Since $Train$, $Incr$, and $Data$ are categorical, we encode them as one-hot vectors. Thus, β_1, β_2, and β_3 are vectors of the same size as the number of possible categories for each variable. To emphasize the explanatory variables and to simplify notation, in the following we denote the above regression model (Eq. 5.3) as "$Y \sim Train + Incr + Data + \ldots$".

Under appropriate assumptions[1], the estimated coefficients can be interpreted as estimated causal effects. The statistical significance of these effects is assessed by examining the *p-value* of the associated Student t-test for each coefficient [Gar+13]. Following established statistical practices [Gar+13], we set the significance value at .05. The significance, sign, magnitude, and interpretation of each estimated coefficient depend on the regression model. In particular, introducing more exogenous variables can cause instability in the regression. Therefore, for each metric Y, we adopt the following methodology to select only the most influential factors:

1. We use multiple regression models to represent the evaluation metric Y as a linear combination of different variables, or of the product of these variables. We ensure that the chosen regressions exhibit no collinearity or numerical issues[2].

2. Subsequently, we select a regression model using the Akaike Information Criterion (AIC) [Aka98], which regularizes the likelihood of the model based on its degrees of freedom.

3. We interpret the regression coefficients, the coefficient of determination R^2, and examine the Q-Q plot of the residuals $\hat{\varepsilon}$ to verify their normality.

4. Next, we conduct an Analysis of Variance (ANOVA) [Gar+13] on the regression to obtain aggregated statistics on the categorical variables.

[1]Primarily, non-perfect collinearity among exogenous variables and the normality of the estimated residuals $\hat{\varepsilon}$

[2]We assess this by examining the smallest eigenvalue of the Gram matrix of the data $X^T X$. Although Ridge or Lasso regression could address these concerns, their coefficients are less interpretable than those of OLS.

5. Finally, we interpret the partial η^2 derived from the ANOVA as a measure of the importance of each variable.

A regression on a categorical variable requires the setting of a reference value for it. Therefore, the coefficient(s) associated with this categorical variable represent the causal effects of this variable *with respect to the reference level*. However, we want to compare all initial training strategies with each other to derive practical recommendations. Therefore, we use the following protocol to generate pairwise significant differences:

(1) perform the same regression multiple times using a different reference category;

(2) sum-up the pairwise comparisons in a double-entry matrix;

(3) since we are performing multiple tests, we need to adjust the significance threshold of each test using Bonferroni correction [Gar+13], which consists of dividing the *p-value* threshold by the number of tests;

(4) plot a heatmap of the pairwise comparisons between the choice of a parameter.

5.5.2 Metrics and confounding Factors

In Figure 5.2, we examine the relationship between the evaluation metrics defined in Subsection 5.4.4. We observe a strong positive correlation between $\overline{Acc}$ and Acc_K. There is a weak negative correlation between average incremental accuracy and forgetting, which is expected due to the inherent trade-off between stability (i.e. low forgetting) and plasticity in CIL (i.e. high performance on new classes). We note a significant correlation between average incremental accuracy and accuracy in the initial state. This correlation is expected since half of our experiments are done with half of the classes in the initial step. Additionally, the average incremental accuracy (Eq. 5.1) evaluates each model on each class, from the first occurrence of the class to the end of the incremental process, thus giving greater influence to earlier classes. Conversely, there is a weak correlation between forgetting and initial accuracy. This implies that the performance on the initial batch of classes does not significantly impact the model's stability throughout the incremental steps.

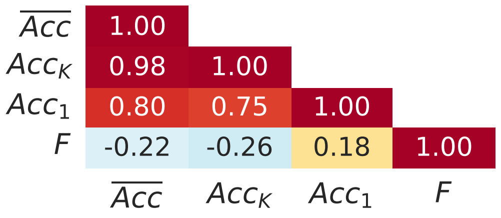


Fig. 5.2.: Correlation between the endogenous variables.

Based on these observations, we choose the average incremental accuracy $\overline{Acc}$ and the average forgetting F as the metrics of interest for our study, and include the effect of the initial accuracy in their models. Controlling the initial accuracy in a regression model is important to draw accurate conclusions: if pure accuracy is sought, then it can be left out of the model. However, the goal of CIL algorithms is not solely to be accurate on average, but rather to be accurate while preventing forgetting. Hence, to analyze the actual incremental contribution of each method, initial accuracy should be included in the regression.

5.5.3 Linear Regression

Variable selection. We use the Python module `statsmodels` for our linear regressions. We first consider a broad range of explanatory variables:

- Acc_1: the accuracy of the first state,
- $Data$: dummy variable for the type of target dataset,
- $Train$: dummy variable for the initial training strategy,
- $Incr$: dummy variable for the incremental method used,
- n_{mean}: the mean number of images per class in the experiment,
- $Small$: binary variable encoding if the training images are so small that they have to be up-scaled,
- $Width$: mean width of the images used for the experiment,
- B: binary variable encoding for the 2 possible CIL scenarios (i.e. either 10% or 50% of the total number of classes learned in the initial step of the process),
- N: the total number of classes,
- N_1: the number of images in the first state.

It has to be noted that some of these variables are highly collinear with each other since they are properties of the dataset of the experiment.

We first perform 1-variable regressions of the incremental accuracy $\overline{Acc}$ and the forgetting F. We identify the most important variables by looking at the R^2 of the regressions that have a sufficiently small $p-value$ (at the .05 threshold). Results are presented in tables 5.2

and 5.3. We select the four most important variables and use them to fit more complex linear regression models that combine these selected variables.

Variable	*p-value*	R^2
Acc_1	2.96e-240	0.63
$Train$	1.17e-87	0.33
$Data$	2.25-55	0.23
$Incr$	7.52e-29	0.11
n_{mean}	8.16e-20	0.07
$Small$	1.84e-05	0.02
$Width$	9.78e-03	0.01
B	1.05e-01	0.00
N	2.41e-01	0.00
N_1	2.87e-01	0.00

Tab. 5.2.: Variables predicting accuracy, sorted by decreasing importance

Variable	*p-value*	R^2
$Incr$	2.20e-222	0.62
$Train$	6.46e-15	0.08
Acc_1	7.71e-10	0.03
$Data$	2.66e-03	0.02
N	7.50e-04	0.01
B	3.43e-02	0.00
N_1	4.13e-02	0.00
n_{mean}	1.07e-01	0.00
$Small$	6.88e-01	0.00
$Width$	7.17e-01	0.00

Tab. 5.3.: Variables predicting forgetting, sorted by decreasing importance

Model selection. We perform linear regressions with many different combinations of the selected variables. We find that introducing product variables, such as $Train \times Incr$ with the intent of directly modeling the interactions between the initial training strategy and the incremental method, introduces collinearity problems. Therefore, we choose to study such interactions following the protocol presented in Section 5.5.

We select the following model:

$$\overline{Acc} \sim Incr + Train + Data. \tag{5.4}$$

The output of the regression is shown in Figure 5.4. To verify the quality of the regression, we also plot the residuals along with a Q-Q plot to verify their normality, as well as a scale-location plot to verify homoscedasticity (constant variance), and a residual vs. leverage plot to look for possible influential outliers. All of these diagnostics are shown in Figure 5.3.

5.5.4 Factors influencing incremental performance

This subsection presents the aggregated influence of the considered parameters. The models and findings presented in Table 5.4 are obtained with the methodology presented in Subsection 5.5.1.

Main influences. In Table 5.4, the most significant factor affecting average incremental accuracy is the choice of initial training strategy. However, upon controlling the impact of initial accuracy, the selected incremental algorithm has a greater importance. This distinction is primarily attributed to BSIL, which exhibits an average incremental accuracy 16 points below that of FeTrIL and DSLDA.

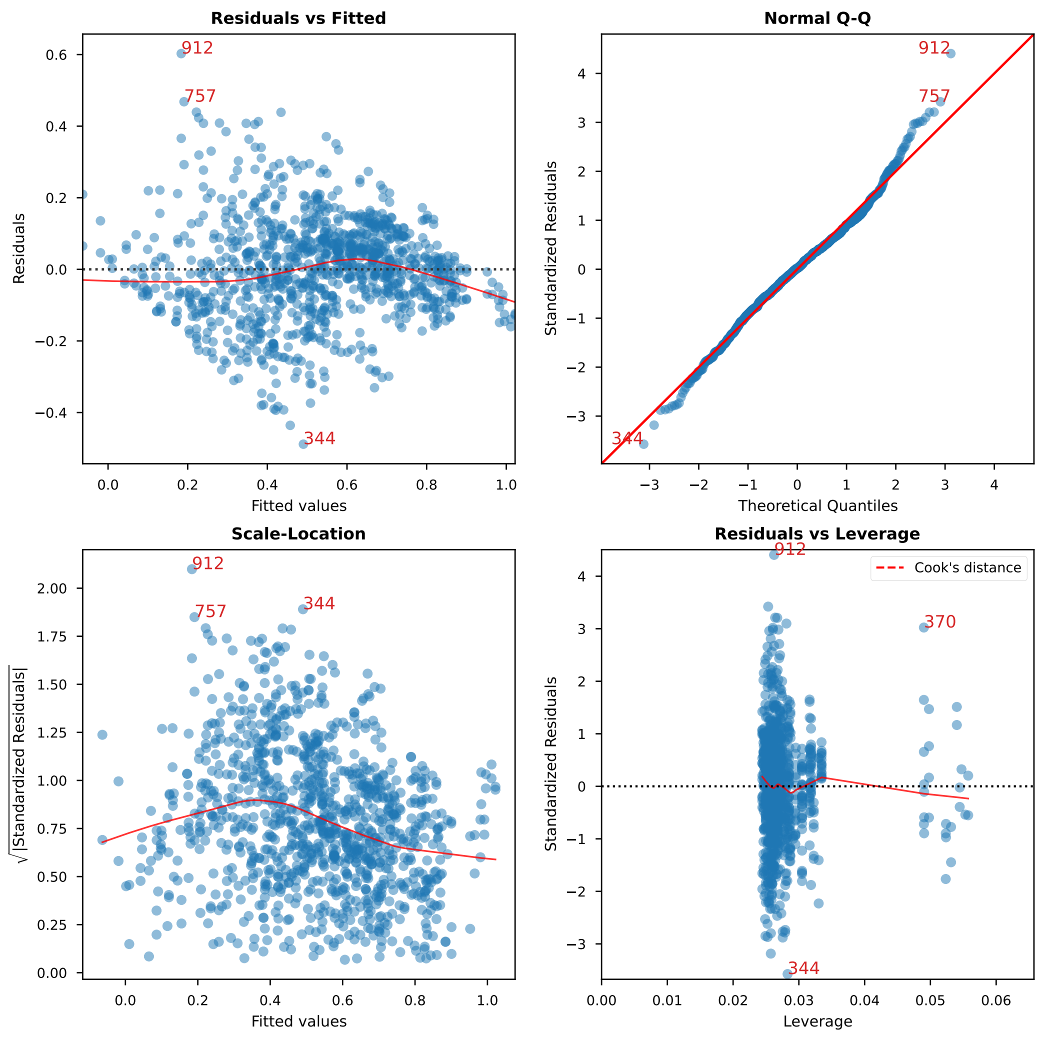
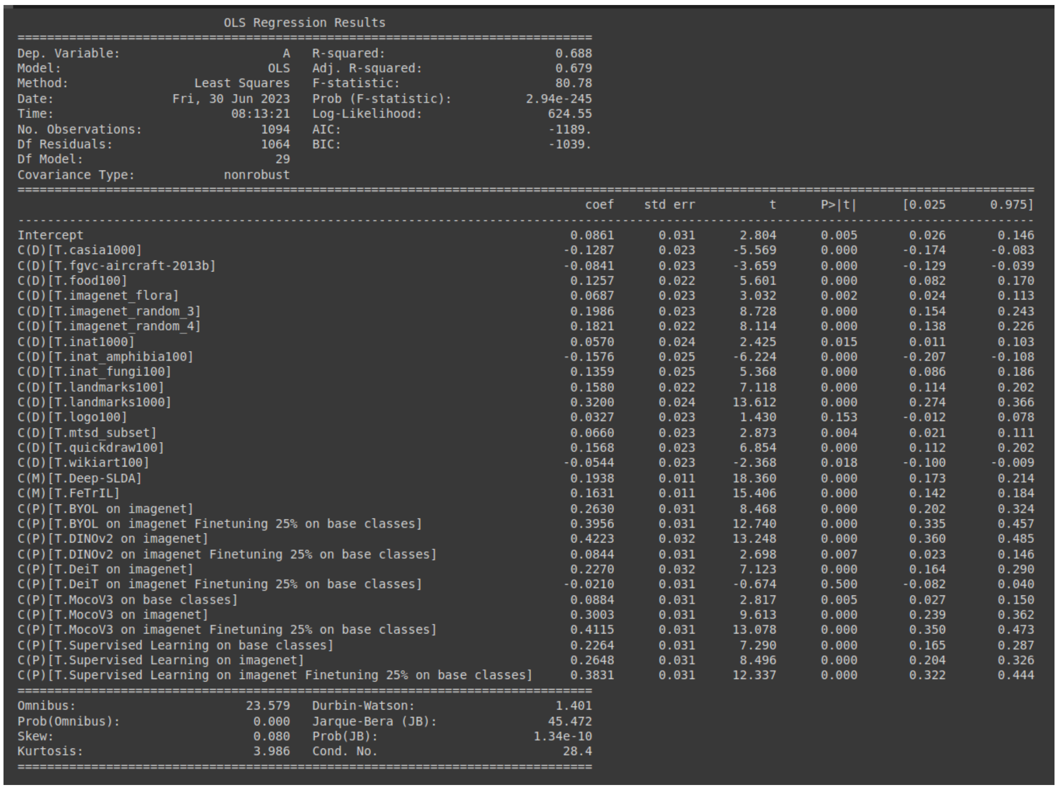


Fig. 5.3.: Diagnostics of the regression for the accuracy as in Equation 5.4.

Regarding forgetting, the incremental algorithm is the most influential parameter. Here, this effect is not driven by any specific outlier method. Further analysis shows that initial accuracy also plays a significant role in predicting the level of forgetting. The associated regression coefficient is .16 ($\pm$.02), indicating that a 1-point increase in initial accuracy results in a 16-point increase of forgetting.

Given that accuracy ranges between 0 and 1, a lower initial accuracy decreases the likelihood of experiencing high levels of forgetting. Hence, a trade-off arises concerning the initial accuracy: while its enhancement greatly improves the average incremental accuracy, it also appears to amplify forgetting. This should be taken into account when comparing CIL algorithms. From a research perspective, the incremental algorithm remains influential in the metrics, particularly when controlling for initial accuracy or focusing on forgetting. However, in practical applications of CIL, the final accuracy may be more important. Given its strong correlation with average incremental accuracy, increasing the initial accuracy becomes more advantageous in this case.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      A   R-squared:                       0.688
Model:                            OLS   Adj. R-squared:                  0.679
Method:                 Least Squares   F-statistic:                     80.78
Date:                Fri, 30 Jun 2023   Prob (F-statistic):          2.94e-245
Time:                        08:13:21   Log-Likelihood:                 624.55
No. Observations:                1094   AIC:                            -1189.
Df Residuals:                    1064   BIC:                            -1039.
Df Model:                          29
Covariance Type:            nonrobust
=========================================================================================================================================
                                                                              coef    std err          t      P>|t|     [0.025     0.975]
-----------------------------------------------------------------------------------------------------------------------------------------
Intercept                                                                   0.0861      0.031      2.804      0.005      0.026      0.146
C(D)[T.casia1000]                                                          -0.1287      0.023     -5.569      0.000     -0.174     -0.083
C(D)[T.fgvc-aircraft-2013b]                                                -0.0841      0.023     -3.659      0.000     -0.129     -0.039
C(D)[T.food100]                                                             0.1257      0.022      5.601      0.000      0.082      0.170
C(D)[T.imagenet_flora]                                                      0.0687      0.023      3.032      0.002      0.024      0.113
C(D)[T.imagenet_random_3]                                                   0.1986      0.023      8.728      0.000      0.154      0.243
C(D)[T.imagenet_random_4]                                                   0.1821      0.022      8.114      0.000      0.138      0.226
C(D)[T.inat1000]                                                            0.0570      0.024      2.425      0.015      0.011      0.103
C(D)[T.inat_amphibia100]                                                   -0.1576      0.025     -6.224      0.000     -0.207     -0.108
C(D)[T.inat_fungi100]                                                       0.1359      0.025      5.368      0.000      0.086      0.186
C(D)[T.landmarks100]                                                        0.1580      0.022      7.118      0.000      0.114      0.202
C(D)[T.landmarks1000]                                                       0.3200      0.024     13.612      0.000      0.274      0.366
C(D)[T.logo100]                                                             0.0327      0.023      1.430      0.153     -0.012      0.078
C(D)[T.mtsd_subset]                                                         0.0660      0.023      2.873      0.004      0.021      0.111
C(D)[T.quickdraw100]                                                        0.1568      0.023      6.854      0.000      0.112      0.202
C(D)[T.wikiart100]                                                         -0.0544      0.023     -2.368      0.018     -0.100     -0.009
C(M)[T.Deep-SLDA]                                                           0.1938      0.011     18.360      0.000      0.173      0.214
C(M)[T.FeTrIL]                                                              0.1631      0.011     15.406      0.000      0.142      0.184
C(P)[T.BYOL on imagenet]                                                    0.2630      0.031      8.468      0.000      0.202      0.324
C(P)[T.BYOL on imagenet Finetuning 25% on base classes]                     0.3956      0.031     12.740      0.000      0.335      0.457
C(P)[T.DINOv2 on imagenet]                                                  0.4223      0.032     13.248      0.000      0.360      0.485
C(P)[T.DINOv2 on imagenet Finetuning 25% on base classes]                   0.0844      0.031      2.698      0.007      0.023      0.146
C(P)[T.DeiT on imagenet]                                                    0.2270      0.032      7.123      0.000      0.164      0.290
C(P)[T.DeiT on imagenet Finetuning 25% on base classes]                    -0.0210      0.031     -0.674      0.500     -0.082      0.040
C(P)[T.MocoV3 on base classes]                                              0.0884      0.031      2.817      0.005      0.027      0.150
C(P)[T.MocoV3 on imagenet]                                                  0.3003      0.031      9.613      0.000      0.239      0.362
C(P)[T.MocoV3 on imagenet Finetuning 25% on base classes]                   0.4115      0.031     13.078      0.000      0.350      0.473
C(P)[T.Supervised Learning on base classes]                                 0.2264      0.031      7.290      0.000      0.165      0.287
C(P)[T.Supervised Learning on imagenet]                                     0.2648      0.031      8.496      0.000      0.204      0.326
C(P)[T.Supervised Learning on imagenet Finetuning 25% on base classes]      0.3831      0.031     12.337      0.000      0.322      0.444
==============================================================================
Omnibus:                       23.579   Durbin-Watson:                   1.401
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               45.472
Skew:                           0.080   Prob(JB):                     1.34e-10
Kurtosis:                       3.986   Cond. No.                         28.4
==============================================================================
```

Fig. 5.4.: Output of the regression for the accuracy

5.5.5 Comparison of initial training strategies

In Figure 5.5, we observe notable variations in accuracy among different initial training strategies, thus prompting the identification of three regimes:

1. **Strategies that surpass supervised learning without transfer:** MoCoV3-ft, DINOv2-t, BYOL-ft, SL(ResNet)-ft, MoCov3-t.

   These approaches exhibit superior performance by generating a robust latent space, whose features are transferable. MoCoV3-ft enhances its latent space by fine-tuning, enabling better generalization compared to other methods. DINOv2-t follows, leveraging its extensive self-supervised training on a very large amount of data. BYOL-ft and SL(ResNet)-ft closely follow, highlighting the advantage gained from additional adaptation steps on the target dataset following pre-training. MoCov3-t is fifth, showing that features generated through an adapted self-supervised method have a generalization capability that can be leveraged in CIL.

2. **Strategies that exhibit no significant improvement over supervised learning without transfer:** SL(ResNet)-ft, BYOL-t, SL(DeiT)-t.

Model	R^2	variable	η^2
$\overline{Acc} \sim Incr + Train + Data$	0.69	$Train$	0.32
		$Data$	0.24
		$Incr$	0.11
$\overline{Acc} \sim Acc_1 + Incr + Train + Data$	0.81	Acc_1	0.25
		$Incr$	0.22
		$Train$	0.10
		$Data$	0.06
$F \sim Incr + Train + Data$	0.71	$Incr$	0.61
		$Train$	0.06
		$Data$	0.03

Tab. 5.4.: ANOVA results for each considered regression. Variables are significant at $p < 0.05$ and ordered by decreasing importance.

Our analysis underlines the capability of well-designed self-supervised methods to outperform supervised pre-training approaches.

3. **Strategies that underperform compared to supervised learning without transfer:** MoCoV3, BYOL, DINOv2-ft, SL(DeiT)-ft.

   The inferior performance of self-supervised methods can be attributed to the limited initial data. Furthermore, the challenging nature of fine-tuning for transformer models contributes to the underwhelming outcomes observed in these models.

The analysis of the average forgetting, illustrated in Figure 5.6, indicates that the majority of pairwise initial training strategies exhibit no significant distinctions. However, DINOv2-t exhibits lower forgetting compared to other strategies, including SL (ResNet). This is particularly remarkable considering that DINOv2-t has the highest initial accuracy. Conversely, fine-tuned transfer models (DINOv2-ft, SL(DeiT)-ft) also display a lower forgetting, albeit primarily attributed to their inherently low initial accuracy, which leaves little room for further decline in their accuracy.

5.5.6 Further analysis of initial training strategies

We now inquire whether the preceding general analysis can be nuanced in specific scenarios. To this end, we perform the same analysis as in the previous section by performing the regression on subsets of the data.

Influence of the dataset. Regarding target datasets that are most different from the pre-training dataset, the benefit of pre-training with or without fine-tuning is lower due to the domain gap. It should be noted that specialized datasets such as Qdraw100 and Casia100

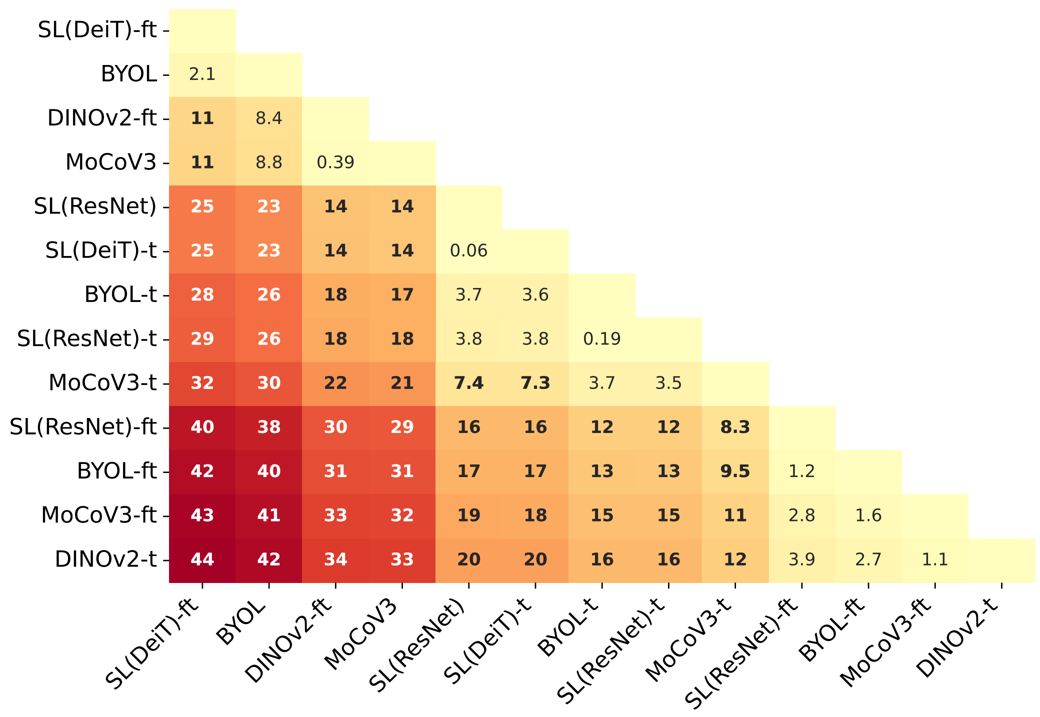


Fig. 5.5.: Accuracy gain by using strategy in row i over strategy in column j , e.g. "The accuracy of BYOL-ft is 17pts higher than SL(ResNet)". Only results in **bold** are statistically different.

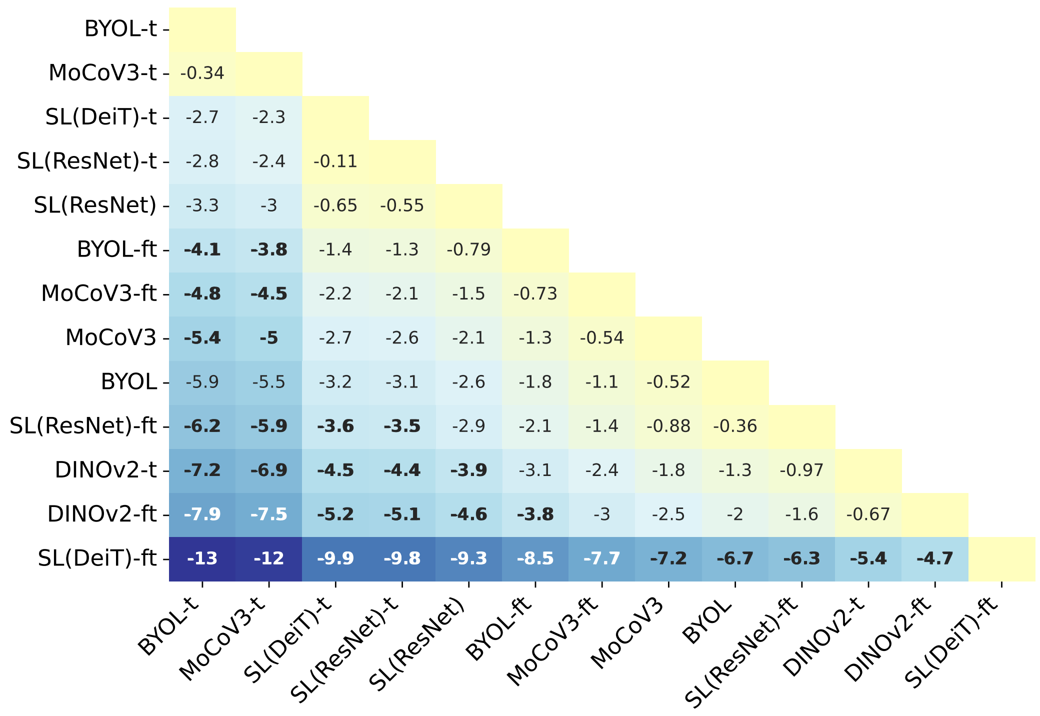


Fig. 5.6.: Overall pairwise comparisons on Forgetting. Only results in **bold** are statistically different.

also contain smaller images than those of ILSVRC. Whether the difference in performance is caused by a semantic gap or an image-size gap is unclear.

Influence of the incremental scenario. Regarding accuracy, we find that most differences among methods come from the scenarios with 50 initial classes or less. In scenarios containing 10 initial classes, all strategies that were previously not significantly better than SL(ResNet) start to outperform it. In scenarios with 50 initial classes, it becomes more difficult to precisely rank the top initial training strategies. In scenarios with 100 initial classes, no strategy is significantly better than any other one (which can come from the lower number of experiments with these scenarios).

Influence of Incremental method. We find that FeTrIL and DSLDA exhibit a similar pattern for $\overline{Acc}$ and F, contrary to BSIL. For FeTrIL and DSLDA, the differences between the best initial training strategies are less clear, but the general trend previously described still holds, in particular for the accuracy. The choice of the training strategy does not clearly impact the forgetting. On the other hand, BSIL is much more sensitive to the initial training strategy. Fine-tuned methods clearly outperform classical learning and plain transfer (except for DINOv2-t), whether it concerns the accuracy or the forgetting. Moreover, SL(ResNet) is a stronger baseline for BSIL than for the other methods when considering incremental accuracy.

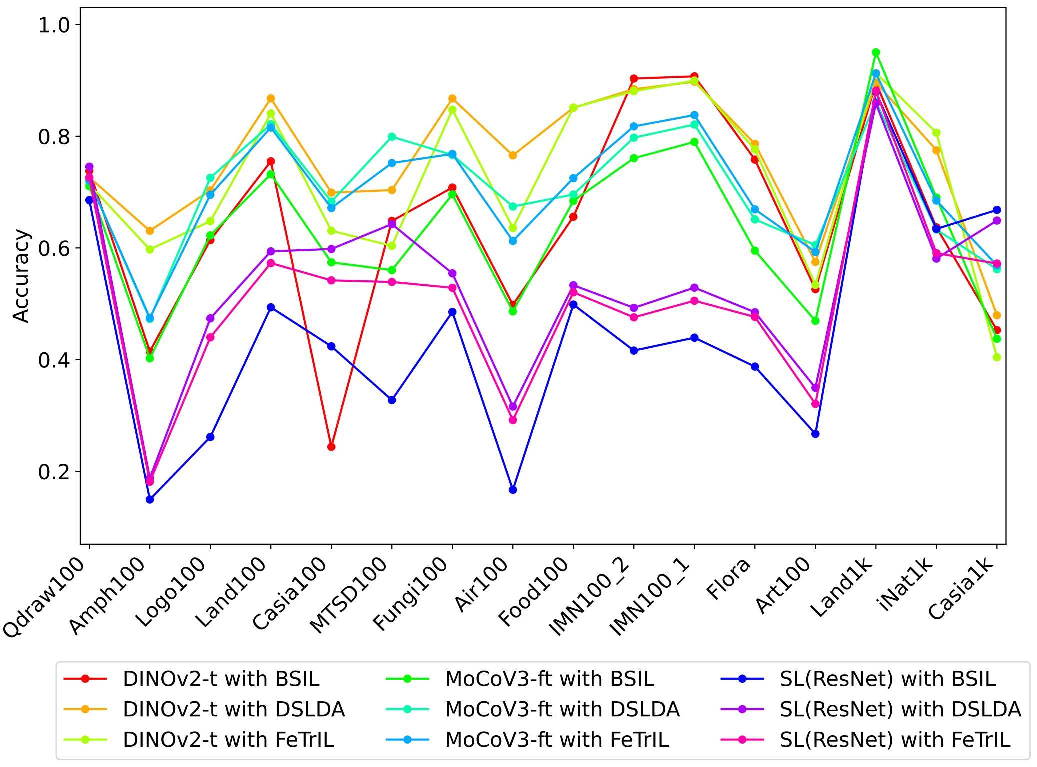


Fig. 5.7.: Interaction plot of the best strategies for different transfer types and for the 3 CIL algorithms. Similar slopes indicate similar behaviors. A change in slope indicates a change in behavior.

5.6 Discussion

We summarize our findings and propose recommendations for the design of EFCIL approaches.

Does the use of a model pre-trained on an external dataset $\mathcal{D}^{\star}$ always improve performance on the target dataset $\mathcal{D}$? Figure 5.7 highlights that no single initial training strategy outperforms the others on all datasets. As illustrated in Table 5.1, pre-training is clearly better on average, but there are exceptions. Intuitively, the use of a pre-trained model without fine-tuning (DINOv2-t in Figure 5.7), is clearly preferable for datasets such as IMN100$_1$ and Flora which are closely related to the dataset used for pre-training. Inversely, the supervised training method SL(ResNet) is better when the gap between the source and the target datasets is important, such is the case for Casia1k. MoCov3-ft is a good compromise since it leverages pre-training, but adapts the representation via partial fine-tuning. The initial training strategy should be selected by considering characteristics of the dataset such as: the number of classes, number of samples per class, domain gap with pre-training, and size of the initial batch of classes.

In the absence of an external dataset, is it better to train the initial model in a supervised way or with a self-supervised learning method? As shown in Figure 5.5, supervised learning on the initial data is better on average. However, self-supervised learning is better when the amount of data available initially is limited, making it difficult to train a supervised model effectively.

Should the pre-trained model be fine-tuned on the first batch of data, or frozen? Existing EFCIL works that use pre-trained transformers keep their weights fixed [Jan+22; Pel22; Wan+22b]. This might be explained by the fact that fine-tuning these models might be detrimental in transfer learning [Kum+22]. Inversely, the performance of CNN-based training strategies, such as BYOL or MoCov3, increases after partial fine-tuning. This is explained by the fact that the layers of CNNs are reusable across tasks, while fine-tuning the last layers with initial target data improves transferability in subsequent EFCIL steps.

How does the performance of EFCIL algorithms vary with initial training strategies? Table 5.1 and Figure 5.7 show that the performance of BSIL varies much more than that of DSLDA and FeTrIL. This is particularly clear for transformer models, where BSIL performance is strongly degraded when fine-tuning of pre-trained models is used. In contrast, the variation of performance for DSLDA and FeTrIL is much lower when testing partial fine-tuning and transfer strategies on top of pre-trained models. This suggests that both initial training strategies are usable in practice for transfer-learning based EFCIL algorithms.

What is the impact of using transformers versus convolutional neural networks? The averaged results presented in Table 5.1 and the detailed ones from Figure 5.7 show that the difference between the best training strategies based on transformers and on CNNs is small. This is particularly the case when CNNs are pre-trained in a self-supervised manner and then partially fine-tuned on the initial batch of target data. Our finding echoes those reported in recent comparative studies of the two types of neural architectures which conclude that there is no absolute winner [PTK22; Wan+23b]. The implication for EFCIL is that the use of both types of architecture should be explored in future works.

5.7 Conclusion

We perform an analysis of EFCIL in an evaluation setting that includes numerous and diverse classification tasks. We confirm the findings of existing comparative studies which have shown that no CIL algorithm is the best in all cases [BPK21; Mas+21; Fei+23] and that algorithms based on transfer learning provide accuracy and stability for EFCIL [HK20; Jan+22]. Our main finding is that the initial training strategy is the dominant factor influencing the average incremental accuracy, but that the choice of CIL algorithm is more important in preventing forgetting. Beyond the fact that there is no silver bullet approach to dealing with EFCIL, our in-depth statistical study quantifies the effect of different components of EFCIL approaches and thus enables informed decisions to be made when designing new methods or implementing EFCIL in practice.

6 Conclusion

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

— **Robert Frost**
The Road Not Taken (4/4)

6.1 General conclusion

Exemplar-Free Class-Incremental Learning (EFCIL) is the specialized subfield within the broader domain of Machine Learning and Artificial Intelligence, where a Machine Learning model needs to continuously adapt and expand its knowledge to accommodate new batches of classes, named states, while retaining the ability to recognize previously learned classes.

The *exemplar-free* aspect of EFCIL distinguishes it from other incremental learning methods that rely on retaining specific examples or exemplars from previous classes to aid in learning new classes. In EFCIL, the model does not have access to stored exemplars, making the learning process more challenging.

As discussed in Chapter 1, EFCIL is a complex and challenging domain that requires innovative solutions to enable continual learning of new classes while preserving previously acquired knowledge. Several key challenges contribute to the complexity of EFCIL, each of which requires careful consideration and specialized approaches.

First, EFCIL faces the problem of memory. The model continuously learns a large number of classes, without the possibility to recall any images of previously learned classes. The linear growth of the memory requirement with the number of classes would otherwise pose significant computational and storage problems. Therefore, it is critical to design the EFCIL model in a way that minimizes the impact on memory usage when new classes are added.

Second, computational requirements are a significant challenge in EFCIL. As the number of classes increases, the training time for each new class also increases, leading to computational bottlenecks. Techniques to mitigate feature drift, such as distillation, can further increase the computational cost of training. In this context, the development of resource-efficient learning algorithms capable of learning new classes with limited computational resources is critical. These algorithms should strike a balance between retaining past knowledge and efficiently acquiring knowledge from new classes.

The number of states that the model must learn is another critical factor affecting EFCIL performance. As the model accumulates knowledge about an increasing number of classes, the challenge of catastrophic forgetting becomes more pronounced. The model must retain information without revisiting past examples, which becomes increasingly difficult as the number of classes grows. To mitigate catastrophic forgetting, it is essential to explore techniques that allow the model to effectively retain some sort of past knowledge. Addressing the relationship between the number of states and the potential for catastrophic forgetting will be critical to the development of robust EFCIL algorithms.

The variability of data stream structures presents a significant hurdle to achieving efficient and adaptive EFCIL. The characteristics of the data stream, including data types, update frequency, and the size of incremental states, all affect the learning process. The diverse nature of data stream structures requires tailored class-incremental learning approaches that can handle different data types and adapt to different update frequencies. Moreover, these approaches should effectively use the information available in incremental states to optimally support the learning of new classes.

Finally, scenario variability has a significant impact on the performance of EFCIL models, particularly with respect to mitigating catastrophic forgetting. Different scenarios dictate the composition of the initial state, which affects the richness of the feature space from which the model learns. Investigating how to optimize the initial state composition and developing strategies to effectively adapt the model to different scenarios will be critical to improving the model's adaptability and generalization capabilities.

To that extent, we introduced different methods in Chapter 2. Those methods can be split into three categories: *Model-Growth based*, *Finetuning-Based*, and *Fixed-Model based*. Model-Growth based methods exhibit scalability, accommodating new classes with sufficient resources for model growth during training and inference. Additionally, they offer some mitigation of catastrophic forgetting as the model grows with new classes, retaining knowledge of past classes to some extent. However, Model-Growth based methods demand substantial memory and computational resources due to the model's growth with each new class, making them computationally expensive in long-term learning scenarios. Controlling the number of additional parameters for each new task is crucial to prevent the model from becoming excessively large and resource-intensive. Fine-tuning enables the model to adapt to new data, offering some level of plasticity to accommodate changes in the system. However, Fine-tuning has limitations in terms of timeliness, as it does not allow fast integration of new knowledge, requiring more than just training the classifier weight layer, which can be critical in time-sensitive applications. Additionally, it lacks stability, as it may lead to catastrophic forgetting of previously acquired knowledge, given that no previous knowledge is preserved in EFCIL methods. Fixed-Representation based methods accommodate a large number of classes without linearly increasing memory requirements, making them suitable for long-term learning scenarios. They are more stable since the representation remains fixed and unchanged during the training of new tasks, contributing to reduced computational demands and stability. Nonetheless, Fixed-Representation based methods have limited plasticity as the fixed representation in the initial state may become less transferable if incremental tasks change drastically.

6.2 Contributions

Within the scope of this thesis, our emphasis on enhancing efficiency, plasticity, and stability has led us to incorporate two Fixed-Representation based techniques alongside a statistical analysis. These methods have been carefully crafted and integrated to address key challenges in our research, ensuring a comprehensive exploration of the chosen factors while providing valuable insights into EFCIL.

6.2.1 PlaStIL: Plastic and Stable Memory-Free Class-Incremental Learning

The PlaStIL approach addresses a critical challenge in Class-Incremental Learning (CIL), which is finding the right balance between plasticity and stability. Plasticity refers to the ability of the model to learn new data, while stability pertains to retaining knowledge of previously learned classes. Traditional methods in CIL use memory buffers or maintain two separate deep models to achieve this balance, but these approaches can be inefficient in terms of memory usage and computational requirements.

PlaStIL proposes a novel solution that distributes a similar number of parameters as distillation-based methods but in a more efficient way. It freezes the feature extractor after the initial state and introduces several model tops to ensure high plasticity. By freezing the feature extractor, the model retains stability by training the oldest incremental classes with this unchanging extractor. On the other hand, it introduces plasticity to new classes using partially fine-tuned models and a specially designed plasticity layer. This approach allows PlaStIL to provide a balanced trade-off between plasticity and stability.

PlaStIL is evaluated on three large datasets, and the results demonstrate its superiority over existing methods in various configurations. By achieving better performance and effectiveness in class incremental learning, PlaStIL offers a promising advancement in addressing the plasticity-stability dilemma in CIL.

However, PlaStIL still requires a large number of parameters, two separate deep models in terms of size. The plasticity layer is not able to retain the knowledge of all past classes, only the last few ones and the model still requires storage equivalent to two separate deep models. To address these limitations, we introduce FeTrIL, which is a more efficient and stable approach to Exemplar-Free Class-Incremental Learning.

6.2.2 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning

FeTrIL, like PlaStIL, focuses on the plasticity-stability dilemma in Exemplar-Free Class-Incremental Learning (EFCIL). Traditional EFCIL methods prioritize either stability, using fixed feature extractors, or plasticity, through successive fine-tuning of the model. FeTrIL aims to strike a better balance between these two aspects by introducing a novel approach.

The key contribution of FeTrIL is the introduction of a pseudo-feature generator, which is an effective and simple component. This generator does not require storing exemplars of past classes but instead uses the centroid representations of those classes. Using these centroids, FeTrIL creates pseudo-features for past classes, which are combined with actual features of new classes. These combined features are then used to incrementally train a linear classifier that discriminates between all classes.

The advantage of this approach is its computational efficiency since only a minimal component of the deep model, the linear classifier, needs to be updated during the incremental process. This makes FeTrIL faster compared to mainstream methods that update the entire deep model.

FeTrIL is evaluated on three challenging datasets in different incremental settings, and it outperforms ten existing methods in most cases, demonstrating its effectiveness in exemplar-free class-incremental learning. By combining the strengths of fixed feature extractors with the simplicity and efficiency of the pseudo-feature generator, FeTrIL offers a novel and promising solution to the plasticity-stability dilemma in EFCIL.

As discussed in Chapter 2, and emphasized in PlaStIL and FeTrIL, the initial state is a critical component of the EFCIL process. The initial state is the starting point, and it can significantly impact the performance of the model. Therefore, we introduce a statistical analysis framework to quantify the relative contribution of Initial Training Strategies to incremental performance.

6.2.3 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning

This work focuses on studying how the initial model of the EFCIL process can be built in two ways: using only the first batch of the target dataset or incorporating pre-trained weights from an auxiliary dataset. This choice of initial training strategy can significantly impact the performance of the incremental learning model.

To gain deeper insights, this work conducts an extensive experimental study and introduces a statistical analysis framework to quantify the relative contribution of different factors to incremental performance. The main finding is that the initial training strategy has a significant impact on the average incremental accuracy, but the choice of the Class-Incremental Learning (CIL) algorithm plays a more critical role in preventing forgetting. For example, initializing the model with pre-trained weights from DinoV2 often provides a better feature space and then leads to better incremental performance. However, the choice of the CIL algorithm is more critical in preventing catastrophic forgetting. For example, the Finetuning-based method, is often more prone to catastrophic forgetting than the Fixed-Representation based methods. Though, there are exceptions to this rule, such as the study shows.

Based on these findings, the study provides practical recommendations for choosing the right initial training strategy for different incremental learning use cases. This analysis contributes to a better understanding of the factors influencing class-based incremental learning and offers valuable insights to optimize and deploy incremental learning methods effectively.

6.3 Future work and perspective

The research presented in this thesis lays the foundation for further advances in Exemplar-Free Class-Incremental Learning (EFCIL) and addresses critical challenges related to plasticity, stability, and efficiency. Building on these contributions, there are several avenues for future research and perspectives that can further advance the field of EFCIL:

- **Transfer learning and pretraining:** As demonstrated in Chapter 5, expanding the use of transfer learning and pretraining in EFCIL is promising. Investigating how to effectively adapt pre-trained models on large datasets to incremental learning environments could improve the initial training strategy and mitigate catastrophic forgetting.

- **Lifelong learning strategies:** Exploring lifelong learning approaches that enable continual learning of multiple tasks over extended periods of time and states could provide insights into how to more effectively handle the continuous stream of data in EFCIL. Such strategies could allow the model to periodically revisit past tasks to reinforce its knowledge without relying on exemplars.

- **Dynamic model architectures:** Designing dynamic model architectures that can grow and shrink based on task complexity and available resources could address the memory and computational demands associated with model growth-based methods discussed in Section 2.1. Such dynamic architectures could adapt to the incremental learning process and allocate resources more efficiently.

- **Handling data stream variability:** Investigating how to more effectively handle variability in data stream structures could improve the adaptability of EFCIL models to other Continual Learning scenarios, discussed in Chapter 2. Methods that can adapt to different data types, update frequencies, and incremental state sizes could lead to more versatile and robust EFCIL solutions.

- **Real-world applications:** Extending the evaluation of EFCIL methods to real-world applications and complex scenarios would provide practical insights into their effectiveness and potential limitations. Applying EFCIL to various domains, such as robotics, autonomous vehicles, or natural language processing, could shed light on its applicability in real-world settings.

- **Benchmark Datasets:** The development of benchmark datasets specifically designed for EFCIL could facilitate fair and comprehensive comparisons between different methods. Standardized evaluation protocols and datasets would allow researchers to assess the performance of their approaches under consistent conditions.

- **Incremental learning competitions:** Organizing competitions focused on EFCIL could encourage the development of innovative solutions and accelerate the advancement of the field. Such competitions could also facilitate the development of benchmark datasets and evaluation protocols.

- **Incremental learning libraries:** Developing open-source libraries for EFCIL could facilitate the adoption of incremental learning methods in real-world applications. Such libraries could provide a common framework for researchers and practitioners to develop and test their approaches.

In conclusion, the contributions presented in this thesis (PlaStIL, FeTrIL, and the statistical analysis framework) provide valuable insights into addressing the challenges of Exemplar-Free Class-Incremental Learning. The identified future work and perspectives open up exciting opportunities for further research and innovation, paving the way for more efficient, stable, and adaptive Exemplar-Free Class-Incremental Learning methods that can accommodate the continuous evolution of data and tasks in different domains.

Bibliography

[AAP10] Ignazio Aleo, Paolo Arena, and Luca Patané. „Incremental learning for visual classification using neural gas". In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2010, pp. 1–6 (cit. on p. 36).

[Abn+21] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. „Exploring the limits of large scale pre-training". In: *arXiv preprint arXiv:2110.02095* (2021) (cit. on pp. 45, 88).

[ACT17] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. „Expert Gate: Lifelong Learning with a Network of Experts". In: *Conference on Computer Vision and Pattern Recognition*. CVPR. 2017 (cit. on p. 37).

[Ahm+22] Touqeer Ahmad, Akshay Raj Dhamija, Steve Cruz, et al. „Few-shot class incremental learning leveraging self-supervised features". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3900–3910 (cit. on pp. 46, 91).

[Ahn+19] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. „Uncertainty-based continual learning with adaptive regularization". In: *Advances in neural information processing systems* 32 (2019) (cit. on pp. 37, 38).

[Aka98] Hirotogu Akaike. „Information Theory and an Extension of the Maximum Likelihood Principle". In: *Selected Papers of Hirotugu Akaike*. Ed. by Emanuel Parzen, Kunio Tanabe, and Genshiro Kitagawa. New York, NY: Springer New York, 1998, pp. 199–213 (cit. on p. 96).

[AKT19] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. „Task-Free Continual Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 11254–11263 (cit. on p. 38).

[Alj+18] 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)*. Sept. 2018 (cit. on p. 38).

[AP09] Joshua D Angrist and Jörn-Steffen Pischke. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press, 2009 (cit. on p. 95).

[BB07] Léon Bottou and Olivier Bousquet. „The tradeoffs of large scale learning". In: *Advances in neural information processing systems* 20 (2007) (cit. on p. 83).

[BC13] Oliver Beyer and Philipp Cimiano. „DYNG: Dynamic Online Growing Neural Gas for stream data classification." In: *ESANN*. 2013 (cit. on p. 37).

[BGV14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. „Food-101 – Mining Discriminative Components with Random Forests“. In: *European Conference on Computer Vision*. 2014 (cit. on pp. 94, 140).

[BGV92] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. „A Training Algorithm for Optimal Margin Classifiers“. In: *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*. Ed. by David Haussler. ACM, 1992, pp. 144–152 (cit. on p. 40).

[BP18] Eden Belouadah and Adrian Popescu. „DeeSIL: Deep-Shallow Incremental Learning“. In: *TaskCV Workshop @ ECCV 2018.* (2018) (cit. on pp. 40, 51–56, 60–64, 66, 71, 72, 74, 77, 78, 81, 83, 90, 141, 143).

[BPK20] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. „Initial Classifier Weights Replay for Memoryless Class Incremental Learning“. In: *British Machine Vision Conference (BMVC)*. 2020 (cit. on pp. 43, 53, 59, 61, 62, 141, 142).

[BPK21] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. „A comprehensive study of class incremental learning algorithms for visual tasks“. In: *Neural Networks* 135 (2021), pp. 38–54 (cit. on pp. 26, 51, 53–61, 67, 71–73, 75, 80, 85, 88, 90, 92, 94, 106).

[Cas+18] Francisco M Castro, Manuel J Marin-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. „End-to-end incremental learning“. In: *Proceedings of the European Conference on computer vision (ECCV)*. 2018, pp. 233–248 (cit. on pp. 43, 48, 55, 58, 69, 71, 94, 139).

[Cer+20] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. „Modeling the Background for Incremental Learning in Semantic Segmentation“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2020 (cit. on p. 44).

[Cha+18] Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. „Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence“. In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Vol. 11215. Lecture Notes in Computer Science. Springer, 2018, pp. 556–572 (cit. on pp. 26, 43, 53).

[Cha+19] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. „Efficient Lifelong Learning with A-GEM“. In: *International Conference on Learning Representations*. 2019 (cit. on p. 48).

[CV95] Corinna Cortes and Vladimir Vapnik. „Support-vector networks“. In: *Machine learning* 20.3 (1995), pp. 273–297 (cit. on pp. 54, 60).

[CXH21] Xinlei Chen, Saining Xie, and Kaiming He. „An empirical study of training self-supervised vision transformers“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9640–9649 (cit. on pp. 46, 89, 93).

[Den+09] Jia Deng, Wei Dong, Richard Socher, et al. „ImageNet: A large-scale hierarchical image database“. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. 2009, pp. 248–255 (cit. on pp. 58, 93, 94, 140).

[Dha+18] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. „Learning without Memorizing“. In: *CoRR* abs/1811.08051 (2018) (cit. on pp. 43, 53).

[Dha+21] Akshay Raj Dhamija, Touqeer Ahmad, Jonathan Schwan, et al. „Self-Supervised Features Improve Open-World Learning“. In: *arXiv preprint arXiv:2102.07848* (2021) (cit. on pp. 46, 53, 54, 68, 71, 73).

[DL19] Debasmit Das and CS George Lee. „A two-stage approach to few-shot learning for image recognition“. In: *IEEE Transactions on Image Processing* 29 (2019), pp. 3336–3350 (cit. on pp. 48, 73).

[Dos+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. „An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale“. In: *International Conference on Learning Representations*. 2021 (cit. on pp. 17, 18, 87, 92, 93).

[Dou+20] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. „Podnet: Pooled outputs distillation for small-tasks incremental learning“. In: *Computer vision-ECCV 2020-16th European conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XX*. Vol. 12365. Springer. 2020, pp. 86–102 (cit. on pp. 43, 51, 72, 77, 80, 90).

[Dou+21] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. „Plop: Learning without forgetting for continual semantic segmentation“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4040–4050 (cit. on p. 44).

[Dou+22] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. „DyTox: Transformers for Continual Learning With DYnamic TOken eXpansion“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2022, pp. 9285–9295 (cit. on p. 92).

[Fei+23] Eva Feillet, **Grégoire Petit**, Adrian Popescu, Marina Reyboz, and Céline Hudelot. „AdvisIL - A Class-Incremental Learning Advisor“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2023, pp. 2400–2409 (cit. on pp. 88, 106, 137).

[Fin+20] Enrico Fini, Stéphane Lathuiliere, Enver Sangineto, Moin Nabi, and Elisa Ricci. „Online continual learning under extreme memory constraints“. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer. 2020, pp. 720–735 (cit. on p. 43).

[Fin+22] Enrico Fini, Victor G Turrisi Da Costa, Xavier Alameda-Pineda, et al. „Self-supervised models are continual learners“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9621–9630 (cit. on pp. 44, 91).

[Fre99] Robert M French. „Catastrophic forgetting in connectionist networks“. In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135 (cit. on pp. 8, 26, 90).

[Fri94] Bernd Fritzke. „A growing neural gas network learns topologies“. In: *Advances in neural information processing systems* 7 (1994), pp. 625–632 (cit. on p. 36).

[Gao+22] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. „R-DFCIL: Relation-Guided Representation Learning for Data-Free Class Incremental Learning“. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*. Springer. 2022, pp. 423–439 (cit. on p. 45).

[Gar+13] James Gareth, Witten Daniela, Hastie Trevor, and Tibshirani Robert. *An introduction to statistical learning: with applications in R*. Spinger, 2013 (cit. on pp. 95–97).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016 (cit. on pp. 69, 83).

[Gei+18] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, et al. „Generalisation in humans and deep neural networks“. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, et al. Vol. 31. Curran Associates, Inc., 2018 (cit. on pp. 45, 89).

[GHK20] Jhair Gallardo, Tyler L Hayes, and Christopher Kanan. „Self-supervised training enhances online continual learning“. In: *British Machine Vision Conference (BMVC)*. 2020 (cit. on pp. 41, 91).

[GK17] Alexander Gepperth and Cem Karaoguz. „Incremental learning with self-organizing maps“. In: *2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*. IEEE. 2017, pp. 1–8 (cit. on p. 36).

[GKC19] Siavash Golkar, Micheal Kagan, and Kyunghyun Cho. „Continual Learning via Neural Pruning“. In: *Real Neurons & Hidden Units: Future directions at the intersection of neuroscience and artificial intelligence @ NeurIPS 2019*. 2019 (cit. on pp. 37, 38).

[Gri+20] Jean-Bastien Grill, Florian Strub, Florent Altché, et al. „Bootstrap your own latent-a new approach to self-supervised learning“. In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284 (cit. on pp. 46, 89, 90, 93).

[Hay+20] 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*. Springer. 2020, pp. 466–483 (cit. on pp. 41, 51, 53–56, 59, 61, 62, 66, 68, 141, 142).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Deep Residual Learning for Image Recognition“. In: *Conference on Computer Vision and Pattern Recognition*. CVPR. 2016 (cit. on pp. 60, 64, 77, 83, 87, 92, 93, 141, 142).

[He+18] Chen He, Ruiping Wang, Shiguang Shan, and Xilin Chen. „Exemplar-Supported Generative Reproduction for Class Incremental Learning“. In: *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*. 2018, p. 98 (cit. on p. 74).

[He+20a] 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 (CVPR)*. July 2020 (cit. on p. 44).

[He+20b] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. „Momentum contrast for unsupervised visual representation learning“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738 (cit. on pp. 46, 88, 89, 93).

[HE17] David Ha and Douglas Eck. „A Neural Representation of Sketch Drawings“. In: *CoRR* abs/1704.03477 (2017). arXiv: `1704.03477` (cit. on pp. 94, 140).

[HK20] 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*. 2020, pp. 220–221 (cit. on pp. 40, 51–56, 59, 61–64, 78, 87, 88, 90, 92, 94, 106, 141, 143).

[HK22] Tyler L. Hayes and Christopher Kanan. „Online Continual Learning for Embedded Devices“. In: *Conference on Lifelong Learning Agents (CoLLAs)*. Aug. 2022 (cit. on pp. 46, 69, 71, 83, 85, 88, 89, 92).

[Hou+19] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. „Learning a Unified Classifier Incrementally via Rebalancing“. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 831–839 (cit. on pp. 43, 44, 51–55, 58, 59, 61, 62, 69, 71, 72, 75, 77, 78, 80, 84, 85, 90, 94, 139, 141–143).

[Hu+21] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. „Distilling Causal Effect of Data in Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2021, pp. 3957–3966 (cit. on p. 44).

[Hun+19] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, et al. „Compacting, picking and growing for unforgetting continual learning“. In: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on p. 38).

[HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. „Distilling the Knowledge in a Neural Network“. In: *CoRR* abs/1503.02531 (2015) (cit. on pp. 46, 51, 53, 69, 71, 89).

[Isc+20] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. „Memory-efficient incremental learning through feature adaptation“. In: *European Conference on Computer Vision*. Springer. 2020, pp. 699–715 (cit. on p. 48).

[Jan+22] Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. „Pauljanson002/pretrained-cl: A Simple Baseline that Questions the Use of Pretrained-Models in Continual Learning (Accepted at DistShift workshop at Neurips 2022)“. In: (2022) (cit. on pp. 40, 88, 90, 105, 106).

[JK22] Hyundong Jin and Eunwoo Kim. „Helpful or Harmful: Inter-task Association in Continual Learning“. In: *European Conference on Computer Vision*. Springer. 2022, pp. 519–535 (cit. on p. 38).

[JLM21] Quentin Jodelet, Xin Liu, and Tsuyoshi Murata. „Balanced softmax cross-entropy for incremental learning“. In: *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II*. Springer. 2021, pp. 385–396 (cit. on pp. 21, 44, 87, 88, 90, 92, 94, 143).

[JS18] Khurram Javed and Faisal Shafait. „Revisiting Distillation and Incremental Classifier Learning“. In: *CoRR* abs/1807.02802 (2018) (cit. on pp. 51, 52, 61).

[JT20] Longlong Jing and Yingli Tian. „Self-supervised visual feature learning with deep neural networks: A survey“. In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 4037–4058 (cit. on pp. 46, 89).

[Jun+20] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. „Continual learning with node-importance based adaptive group sparse regularization“. In: *Advances in neural information processing systems* 33 (2020), pp. 3647–3658 (cit. on p. 38).

[Kan+22] Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, et al. „Forget-free continual learning with winning subnetworks“. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 10734–10750 (cit. on p. 38).

[KCR21] Abhishek Kumar, Sunabha Chatterjee, and Piyush Rai. „Bayesian structural adaptation for continual learning“. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5850–5860 (cit. on p. 38).

[Kem+18] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. „Measuring catastrophic forgetting in neural networks“. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 2018 (cit. on pp. 26, 51, 52, 69).

[Kir+17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, et al. „Overcoming catastrophic forgetting in neural networks“. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526 (cit. on pp. 77, 78).

[KK18] Ronald Kemker and Christopher Kanan. „FearNet: Brain-Inspired Model for Incremental Learning“. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018 (cit. on p. 40).

[KLH20] Zixuan Ke, Bing Liu, and Xingchang Huang. „Continual Learning of a Mixed Sequence of Similar and Dissimilar Tasks“. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 18493–18504 (cit. on p. 44).

[Kri09] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009 (cit. on pp. 76, 139).

[KSL18] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. „Do Better ImageNet Models Transfer Better?“ In: *CoRR* abs/1805.08974 (2018) (cit. on pp. 40, 45, 89, 90).

[Kum+22] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. „Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution“. In: *International Conference on Learning Representations*. 2022 (cit. on pp. 46, 87, 89, 105).

[Kur+21] Vinod K Kurmi, Badri N Patro, Venkatesh K Subramanian, and Vinay P Namboodiri. „Do Not Forget to Attend to Uncertainty while Mitigating Catastrophic Forgetting“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 736–745 (cit. on p. 44).

[Lan+19] Matthias De Lange, Rahaf Aljundi, Marc Masana, et al. „Continual learning: A comparative study on how to defy forgetting in classification tasks“. In: *CoRR* abs/1909.08383 (2019) (cit. on pp. 53, 54, 69, 71, 73, 90).

[LBE21] Seungwon Lee, Sima Behpour, and Eric Eaton. „Sharing Less is More: Lifelong Learning in Deep Networks with Selective Layer Transfer“. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 6065–6075 (cit. on p. 44).

[Lee+19] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. „Overcoming catastrophic forgetting with unlabeled data in the wild“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 312–321 (cit. on p. 44).

[Lee+20] 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 (cit. on p. 48).

[LH16] Zhizhong Li and Derek Hoiem. „Learning Without Forgetting“. In: *European Conference on Computer Vision*. ECCV. 2016 (cit. on pp. 43, 53, 59, 72, 83, 88, 90).

[Li+19] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. „Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting“. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3925–3934 (cit. on p. 38).

[Liu+20a] Xialei Liu, Chenshen Wu, Mikel Menta, et al. „Generative feature replay for class-incremental learning“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 226–227 (cit. on p. 48).

[Liu+20b] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. „Mnemonics Training: Multi-Class Incremental Learning Without Forgetting“. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 2020, pp. 12242–12251 (cit. on pp. 47, 77, 80).

[Liu+20c] Yu Liu, Sarah Parisot, Gregory Slabaugh, et al. „More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning“. In: *European Conference on Computer Vision*. Springer. 2020, pp. 699–716 (cit. on pp. 44, 72, 77, 78).

[Liu+22] Yu Liu, Xiaopeng Hong, Xiaoyu Tao, et al. „Model behavior preserving for class-incremental learning“. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022) (cit. on p. 45).

[LSS21] Yaoyao Liu, Bernt Schiele, and Qianru Sun. „Adaptive Aggregation Networks for Class-Incremental Learning“. In: *Conference on Computer Vision and Pattern Recognition*. CVPR. 2021 (cit. on pp. 77, 80).

[LY15] Ya Le and Xuan Yang. „Tiny imagenet visual recognition challenge“. In: *CS 231N* 7.7 (2015), p. 3 (cit. on pp. 76, 139).

[Maj+13] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. *Fine-Grained Visual Classification of Aircraft*. Tech. rep. 2013. arXiv: `1306.5151 [cs-cv]` (cit. on pp. 94, 140).

[Mar+15] Martin Abadi, Ashish Agarwal, Paul Barham, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015 (cit. on p. 61).

[Mas+21] Marc Masana, Xialei Liu, Bartlomiej Twardowski, et al. *Class-incremental learning: survey and performance evaluation on image classification*. 2021. arXiv: `2010.15277 [cs.LG]` (cit. on pp. 26, 51–56, 58, 60, 61, 67, 71, 72, 74, 78, 80, 83–85, 88, 90, 94, 106).

[MBB13] M Mermillod, A Bugaiska, and P Bonin. „The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects.“ In: *Frontiers in Psychology* 4 (2013), pp. 504–504 (cit. on pp. 26, 53, 71, 88).

[MBS93] Thomas M Martinetz, Stanislav G Berkovich, and Klaus J Schulten. „'Neural-gas' network for vector quantization and its application to time-series prediction“. In: *IEEE transactions on neural networks* 4.4 (1993), pp. 558–569 (cit. on p. 36).

[MC89] Michael Mccloskey and Neil J. Cohen. „Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem“. In: *The Psychology of Learning and Motivation* 24 (1989), pp. 104–169 (cit. on pp. 8, 26, 51, 52, 69, 71, 90).

[MDL18] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. „Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights“. In: *ECCV (4)*. Vol. 11208. Lecture Notes in Computer Science. Springer, 2018, pp. 72–88 (cit. on p. 38).

[Men+13] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. „Distance-based image classification: Generalizing to new classes at near-zero cost“. In: *IEEE transactions on pattern analysis and machine intelligence* 35.11 (2013), pp. 2624–2637 (cit. on pp. 54, 72).

[Mir+22] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, et al. „Wide neural networks forget less catastrophically“. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 15699–15717 (cit. on pp. 22, 95).

[ML18] Arun Mallya and Svetlana Lazebnik. „PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning“. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 7765–7773 (cit. on pp. 37, 38, 53).

[Mus] Constantine Dovrolis Mustafa B Gurbuz. „NISPA: Neuro-Inspired Stability-Plasticity Adaptation for Continual Learning in Sparse Networks“. In: *Proceedings of the 39th International Conference on Machine Learning* 162 () (cit. on p. 38).

[MY16] Ahmed Madani and Rubiyah Yusof. „Malaysian traffic sign dataset for traffic sign detection and recognition systems“. In: *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 8.11 (2016), pp. 137–143 (cit. on pp. 94, 140).

[MZ19] Umberto Michieli and Pietro Zanuttigh. „Incremental learning techniques for semantic segmentation“. In: *Proceedings of the IEEE/CVF international conference on computer vision workshops*. 2019 (cit. on p. 44).

[MZ21] Umberto Michieli and Pietro Zanuttigh. „Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 1114–1124 (cit. on p. 44).

[Ngu+19] Giang Nguyen, Tae Joon Jun, Trung Tran, Tolcha Yalew, and Daeyoung Kim. „ContCap: A scalable framework for continual image captioning“. In: *arXiv preprint arXiv:1909.08745* (2019) (cit. on p. 44).

[Noh+17] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. „Large-scale image retrieval with attentive deep local features“. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3456–3465 (cit. on pp. 58, 94, 139, 140).

[NSZ20] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. „What is being transferred in transfer learning?“ In: *Advances in neural information processing systems* 33 (2020), pp. 512–523 (cit. on pp. 56, 72).

[Oqu+23] Maxime Oquab, Timothée Darcet, Theo Moutakanni, et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2023 (cit. on pp. 45, 46, 88–90, 93).

[Ost+19] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. „Learning to remember: A synaptic plasticity driven framework for continual learning“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 11321–11329 (cit. on p. 38).

[Ost+22] Oleksiy Ostapenko, Timothee Lesort, Pau Rodriguez, et al. „Continual Learning with Foundation Models: An Empirical Study of Latent Replay“. In: *Proceedings of The 1st Conference on Lifelong Learning Agents*. Ed. by Sarath Chandar, Razvan Pascanu, and Doina Precup. Vol. 199. Proceedings of Machine Learning Research. PMLR, Aug. 2022, pp. 60–91 (cit. on pp. 46, 91).

[Par+19] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. „Continual Lifelong Learning with Neural Networks: A Review“. In: *Neural Networks* 113 (2019) (cit. on pp. 52, 71, 90).

[Pas+19] Adam Paszke, Sam Gross, Francisco Massa, et al. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, et al. Curran Associates, Inc., 2019, pp. 8024–8035 (cit. on p. 61).

[Ped+12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. „Scikit-learn: Machine Learning in Python“. In: *CoRR* abs/1201.0490 (2012) (cit. on pp. 72, 76).

[Pel22] Francesco Pelosin. „Simpler is better: off-the-shelf continual learning through pretrained backbones“. In: *arXiv preprint arXiv:2205.01586* (2022) (cit. on pp. 40, 88, 90, 105).

[Pet+23a] **Grégoire Petit**, Adrian Popescu, Eden Belouadah, David Picard, and Bertrand Delezoide. „PlaStIL: Plastic and Stable Memory-Free Class-Incremental Learning“. In: *Proceedings of The 2nd Conference on Lifelong Learning Agents*. Ed. by Sarath Chandar, Razvan Pascanu, and Doina Precup. Proceedings of Machine Learning Research. PMLR, 2023 (cit. on pp. 22, 27, 49, 71, 137).

[Pet+23b] **Grégoire Petit**, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. „FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2023, pp. 3911–3920 (cit. on pp. 28, 49, 87, 88, 90, 92, 94, 137, 143).

[Pet+24] **Grégoire Petit**, Michael Soumm, Feillet Eva, et al. „An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024 (cit. on pp. 28, 49, 137).

[POK05] Shaoning Pang, Seiichi Ozawa, and Nikola K. Kasabov. „Incremental linear discriminant analysis for classification of data streams“. In: *IEEE Trans. Syst. Man Cybern. Part B* 35.5 (2005), pp. 905–914 (cit. on p. 40).

[PTD20] 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*. Springer. 2020, pp. 524–540 (cit. on pp. 54, 56, 61, 67, 71, 85).

[PTK22] Francesco Pinto, Philip HS Torr, and Puneet K. Dokania. „An impartial take to the cnn vs transformer robustness contest“. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIII*. Springer. 2022, pp. 466–480 (cit. on p. 106).

[Pu+21] Nan Pu, Wei Chen, Yu Liu, Erwin M Bakker, and Michael S Lew. „Lifelong person re-identification via adaptive knowledge accumulation“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7901–7910 (cit. on p. 44).

[Qin+21] Qi Qin, Wenpeng Hu, Han Peng, Dongyan Zhao, and Bing Liu. „Bns: Building network structures dynamically for continual learning“. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20608–20620 (cit. on p. 38).

[Ran+17] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. „Encoder based lifelong learning“. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1320–1328 (cit. on p. 43).

[Rav+21] Leonardo Ravaglia, Manuele Rusci, Davide Nadalini, et al. „A TinyML Platform for On-Device Continual Learning With Quantized Latent Replays“. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 11.4 (2021), pp. 789–802 (cit. on pp. 69, 83, 85).

[Raz+14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. „CNN Features Off-the-Shelf: An Astounding Baseline for Recognition“. In: *Conference on Computer Vision and Pattern Recognition Workshop*. CVPR-W. 2014 (cit. on pp. 40, 51).

[RBV18] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. „Efficient Parametrization of Multi-Domain Deep Neural Networks“. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 2018, pp. 8119–8127 (cit. on p. 37).

[Reb+17] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. „iCaRL: Incremental Classifier and Representation Learning“. In: *Conference on Computer Vision and Pattern Recognition*. CVPR. 2017 (cit. on pp. 40, 43, 51–55, 58, 59, 61, 62, 69, 71–75, 77, 78, 83, 85, 88, 90, 139, 141).

[Rin97] Mark B Ring. „CHILD: A first step towards continual learning“. In: *Machine Learning* 28.1 (1997), pp. 77–104 (cit. on p. 88).

[RM19] Ricardo Ribani and Mauricio Marengoni. „A Survey of Transfer Learning for Convolutional Neural Networks“. In: *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*. 2019, pp. 47–57 (cit. on pp. 45, 89).

[RPR20] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. „Tree-CNN: A hierarchical Deep Convolutional Neural Network for incremental learning“. In: *Neural Networks* 121 (2020), pp. 148–160 (cit. on p. 37).

[RT17] Amir Rosenfeld and John K. Tsotsos. „Incremental Learning Through Deep Adaptation“. In: *CoRR* abs/1705.04228 (2017) (cit. on p. 37).

[Rud+17] Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. „The extreme value machine“. In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 762–768 (cit. on pp. 54, 73).

[Rus+15] Olga Russakovsky, Jia Deng, Hao Su, et al. „ImageNet Large Scale Visual Recognition Challenge“. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252 (cit. on pp. 58, 76, 80, 87, 93, 94, 139, 140).

[Rus+16] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, et al. „Progressive Neural Networks“. In: *CoRR* abs/1606.04671 (2016) (cit. on p. 37).

[SE15] Babak Saleh and Ahmed Elgammal. „Large-scale classification of fine-art paintings: Learning the right metric on the right feature“. In: *arXiv preprint arXiv:1505.00855* (2015) (cit. on pp. 94, 140).

[Ser+18] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. „Overcoming catastrophic forgetting with hard attention to the task“. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4548–4557 (cit. on pp. 38, 53).

[SF86] Jeffrey C Schlimmer and Douglas Fisher. „A case study of incremental concept induction“. In: *AAAI*. Vol. 86. 1986, pp. 496–501 (cit. on p. 69).

[Sha+14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. „CNN features off-the-shelf: an astounding baseline for recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 806–813 (cit. on pp. 71, 72).

[Shi+17] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. „Continual learning with deep generative replay“. In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 48).

[SKH21] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. „On learning the geodesic path for incremental learning“. In: *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*. 2021, pp. 1591–1600 (cit. on p. 43).

[Smi+21] James Smith, Yen-Chang Hsu, Jonathan Balloch, et al. „Always be dreaming: A new approach for data-free class-incremental learning“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9374–9384 (cit. on pp. 52–55, 58, 61, 69, 72–74, 77, 78, 80, 85).

[SSA17] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. „Incremental Learning of Object Detectors without Catastrophic Forgetting“. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. 2017, pp. 3420–3429 (cit. on p. 44).

[Tam+17] Youssef Tamaazousti, Hervé Le Borgne, Céline Hudelot, Mohamed El Amine Seddik, and Mohamed Tamaazousti. „Learning More Universal Representations for Transfer-Learning“. In: *CoRR* abs/1712.09708 (2017) (cit. on pp. 45, 89).

[Tan+18] Chuanqi Tan, Fuchun Sun, Tao Kong, et al. „A survey on deep transfer learning“. In: *International conference on artificial neural networks*. Springer. 2018, pp. 270–279 (cit. on pp. 45, 54, 71, 89).

[Tan+23] Chi Ian Tang, Lorena Qendro, Dimitris Spathis, et al. „Practical self-supervised continual learning with continual fine-tuning“. In: *arXiv preprint arXiv:2303.17235* (2023) (cit. on p. 91).

[Tao+20a] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. „Topology-preserving class-incremental learning“. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*. Springer. 2020, pp. 254–270 (cit. on p. 37).

[Tao+20b] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, et al. „Few-shot class-incremental learning“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12183–12192 (cit. on p. 37).

[Tia+23] Songsong Tian, Weijun Li, Xin Ning, et al. „Continuous transfer of neural network representational similarity for incremental learning“. In: *Neurocomputing* 545 (2023), p. 126300 (cit. on pp. 46, 91).

[Tou+21] Hugo Touvron, Matthieu Cord, Matthijs Douze, et al. „Training data-efficient image transformers & distillation through attention“. In: *International conference on machine learning*. PMLR. 2021, pp. 10347–10357 (cit. on pp. 46, 89, 90).

[Van+18] Grant Van Horn, Oisin Mac Aodha, Yang Song, et al. „The inaturalist species classification and detection dataset“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8769–8778 (cit. on pp. 59, 94, 139, 140).

[Ven+17] Ragav Venkatesan, Hemanth Venkateswara, Sethuraman Panchanathan, and Baoxin Li. „A strategy for an uncompromising incremental learner“. In: *arXiv preprint arXiv:1705.00744* (2017) (cit. on pp. 69, 71).

[Ver+21] Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. „Efficient Feature Transformations for Discriminative and Generative Continual Learning“. In: *CoRR* abs/2103.13558 (2021) (cit. on pp. 54, 67, 85).

[VT19] Gido M Van de Ven and Andreas S Tolias. „Three scenarios for continual learning“. In: *arXiv preprint arXiv:1904.07734* (2019) (cit. on p. 88).

[Wan+20] Jing Wang, Weiqing Min, Sujuan Hou, et al. „Logo-2K+: A large-scale logo dataset for scalable logo classification“. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34 (4). 2020, pp. 6194–6201 (cit. on pp. 94, 140).

[Wan+22a] Liyuan Wang, Xingxing Zhang, Kuo Yang, et al. „Memory Replay with Data Compression for Continual Learning“. In: *International Conference on Learning Representations*. 2022 (cit. on p. 48).

[Wan+22b] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, et al. „Learning to prompt for continual learning“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 139–149 (cit. on pp. 41, 46, 88, 91, 105).

[Wan+23a] Yabin Wang, Zhiheng Ma, Zhiwu Huang, et al. „Isolation and Impartial Aggregation: A Paradigm of Incremental Learning without Interference“. In: *2023 AAAI Conference*. 2023 (cit. on pp. 46, 54, 68).

[Wan+23b] Zeyu Wang, Yutong Bai, Yuyin Zhou, and Cihang Xie. „Can CNNs Be More Robust Than Transformers?“ In: (2023) (cit. on p. 106).

[Wel09] Max Welling. „Herding dynamical weights to learn“. In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*. 2009, pp. 1121–1128 (cit. on p. 75).

[Wey+20] Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. „Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2575–2584 (cit. on pp. 94, 140).

[WGL21] Guile Wu, Shaogang Gong, and Pan Li. „Striking a Balance Between Stability and Plasticity for Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1124–1133 (cit. on pp. 51–55, 58, 59, 61, 62, 88, 142).

[Wor+20] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, et al. „Supermasks in superposition“. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15173–15184 (cit. on p. 38).

[WRH17] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. „Growing a Brain: Fine-Tuning by Increasing Model Capacity“. In: *Conference on Computer Vision and Pattern Recognition*. CVPR. 2017 (cit. on pp. 37, 90).

[Wu+18] 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* 31 (2018) (cit. on p. 44).

[Wu+19] Yue Wu, Yinpeng Chen, Lijuan Wang, et al. „Large Scale Incremental Learning“. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 2019, pp. 374–382 (cit. on pp. 43, 51, 55, 58, 61, 69, 75).

[Wu+22] Tz-Ying Wu, Gurumurthy Swaminathan, Zhizhong Li, et al. „Class-Incremental Learning With Strong Pre-Trained Models“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 9601–9610 (cit. on pp. 46, 91).

[Xue+22] Mengqi Xue, Haofei Zhang, Jie Song, and Mingli Song. „Meta-attention for vit-backed continual learning“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 150–159 (cit. on p. 38).

[XZ18] Ju Xu and Zhanxing Zhu. „Reinforced continual learning“. In: *Advances in Neural Information Processing Systems* 31 (2018) (cit. on p. 38).

[Yi+14] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. „Learning face representation from scratch“. In: *arXiv preprint arXiv:1411.7923* (2014) (cit. on pp. 94, 140).

[Yoo+18] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. „Lifelong Learning with Dynamically Expandable Networks“. In: ICLR, 2018 (cit. on p. 38).

[Yos+14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. „How transferable are features in deep neural networks?“ In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 56).

[Yu+20] Lu Yu, Bartlomiej Twardowski, Xialei Liu, et al. „Semantic Drift Compensation for Class-Incremental Learning“. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 2020, pp. 6980–6989 (cit. on pp. 51, 53, 54, 69, 72–74, 77–79, 85).

[Zha+19] Mengyao Zhai, Lei Chen, Frederick Tung, et al. „Lifelong GAN: Continual Learning for Conditional Image Generation“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019 (cit. on p. 44).

[Zha+20] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. „Maintaining Discrimination and Fairness in Class Incremental Learning“. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 2020, pp. 13205–13214 (cit. on pp. 44, 69, 71).

[Zho+19] Peng Zhou, Long Mai, Jianming Zhang, et al. „M2KD: Multi-model and Multi-level Knowledge Distillation for Incremental Learning“. In: *CoRR* abs/1904.01769 (2019) (cit. on p. 51).

[Zhu+21a] Fei Zhu, Zhen Cheng, Xu-yao Zhang, and Cheng-lin Liu. „Class-Incremental Learning via Dual Augmentation“. In: *Advances in Neural Information Processing Systems* 34 (2021) (cit. on pp. 21, 44, 69, 71, 72, 74, 76–79, 83, 85, 88, 90, 94).

[Zhu+21b] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. „Prototype Augmentation and Self-Supervision for Incremental Learning“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 5871–5880 (cit. on pp. 21, 44, 59, 62, 69, 71, 72, 74, 76–79, 83–85, 88, 90, 94).

[Zhu+22] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. „Self-Sustaining Representation Expansion for Non-Exemplar Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9296–9305 (cit. on pp. 21, 44, 71–74, 76–79, 83–85, 88, 90, 94).

[ZYZ21] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. „Co-transport for class-incremental learning“. In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 1645–1654 (cit. on p. 44).

List of Figures

1.1 Illustration of the cross-validation procedure, with K folds. 13

1.2 Illustration of the working principle of a neuron. 15

1.3 Illustration of the working principle of an MLP. 16

1.4 Illustration of the working principle of a convolutional layer. 17

1.5 Illustration of the working principle of two common pooling layers. 17

1.6 Illustration of the ViT architecture. Source [Dos+21]. 18

1.7 Outlines of the five major challenges that are directly linked to Class-Incremental Learning. 20

1.8 Flowchart of the Classical Training Procedure for Machine Learning Models and description of the specificities of the CIL training. 25

2.1 Comparative illustration showing the unique characteristics of Generic Incremental Learning, Domain Incremental Learning, Task Incremental Learning, and (One-)Class Incremental Learning. Each of these learning approaches is shown with its specific context and learning paradigm, providing a clear distinction in terms of task adaptability, data domain change, class integration, and generic incremental adaptation. 32

2.2 Comparison of exemplar selection methods. Source [Liu+20b]. 47

3.1 Accuracy of past and new classes in exemplar-free CIL for three large-scale datasets with $K = 10$ incremental states. LUCIR [Hou+19] uses distillation to preserve past knowledge and favors plasticity. DeeSIL [BP18] transfers features from the initial frozen model to all subsequent states and focuses on stability. PlaStIL offers a better plasticity-stability balance. Note that the proportion of past classes increases as the incremental process advances and so does their weight in global accuracy. 52

3.2 PlaStIL overview using a toy example with $K = 4$ CIL states and $n = 2$ new classes learned per state. The global memory footprint is equivalent to that of distillation-based methods, but this memory is used differently. We assume that a model base and at most three model tops can be used. A base $\mathcal{B}$, is learned initially and then frozen, as is $\mathcal{T}_1$ which is needed to ensure stability. Initial classifier weights are trained using $\mathcal{T}_1$ in each state and reserved for future use. Classifier weights that are actually used in each state are highlighted in red. In state 4, the recent model tops ($\mathcal{T}_4$ and $\mathcal{T}_3$) are included to ensure plasticity. Classifier weights w_4^7 and w_4^8, associated with the new classes from state 4 are learned with features provided by $\mathcal{T}_4$. w_3^5 and w_3^6 were learned with $\mathcal{T}_3$ features in state 3, when they were new. w_1^1 to w_1^4 were learned with $\mathcal{T}_1$ features, in states 1 and 3. w_1^5 to w_1^8 are reserved for future use. $\mathcal{T}_2$ is discarded to keep the total memory footprint of PlaStIL bounded. *Best viewed in color.* 57

3.3 Incremental accuracy across all states for $K \in \{5, 10, 20\}$. Plots are presented for the best methods from Table 3.1. 63

3.4 Top-1 incremental accuracy of three versions of PlaStIL applied to DeeSIL when using a single model top with variable fine-tuning depth. DeeSIL is a limit case in which the whole feature extractor is frozen. *Best viewed in color.* 64

3.5 Top-1 accuracy gains obtained with the three variants of PlaStIL applied to DeeSIL with different thresholds for the rank of the top new class among the predictions generated with Equation 3.4. Results are shown for ILSVRC with $K = 10$ states. The corresponding percentage of supplementary inferences needed for each threshold is also plotted. Interesting gains are obtained starting with a recent class predicted in the second position, which requires approximately 25% of supplementary inferences for PlaStIL$_1$. *Best viewed in color.* 65

3.6 Effect of varying the number of additional tops on incremental accuracy, on ILSVRC with $K \in \{5, 10, 20\}$. As the number of tops increases, the accuracy of the model improves. However, this comes at the cost of increased memory usage on disk. *Best viewed in color.* 67

4.1 Illustration of the proposed pseudo-feature generation procedure. This toy example includes an initial state (3 classes) and two IL states (1 new class per state) in subfigures (a), (b) and (c). Subfigure (d) provides the actual features of all classes that would be available for classical learning. The illustration uses a 2D projection of actual features. Pseudo-features of past classes are generated by geometric translation of features of the new class added in each state with the difference between the centroids of the target past class and of the new class. While imperfect, the pseudo-feature generator produces a usable representation of past classes. *Best viewed in color.* 70

4.2 FeTrIL overview for a toy example with an initial state (3 classes) and two incremental states (1 class per state). The feature extractor $\mathcal{F}$ is trained in the initial state, using sets of data X_1, X_2, X_3, and then frozen afterward. The generator $\mathcal{G}$ uses features $f(C_n)$ of the new class extracted with $\mathcal{F}$ and prototypes of past classes $\mu(C_p)$ to generate pseudo-features of past classes $\hat{f}^t(C_p)$ in the t^{th} state. Prototypes ($\mu(C_i)$) are the centroids of all classes (past and new). They are learned when classes are first seen and then stored throughout the IL process. A linear classifier $\mathcal{L}$ is used to learn classification weights $w(C_i)$ for all seen classes (past and new). 73

4.3 Evolution of top-1 accuracy for an incremental process with $T = 10$ IL states. *Best viewed in color.* 81

4.4 Top-1 incremental accuracy of FeTrIL[1] for approximate training of the classification layer with different ratios for negative sampling. ova denotes a classical one-vs-all training procedure. 84

4.5 Top-1 incremental accuracy per state for past and new classes for TinyImageNet, with $T = 10$ incremental states for FeTrIL[1] and SSRE, the best compared method. An ideal method would provide high accuracy, but also similar performance for past and new classes. The accuracy of past and new classes is globally closer for FeTrIL[1] , which indicates that our method provides a better stability-plasticity balance than SSRE. Overall accuracy is better for FeTrIL[1] in Figure 4.3 because the contribution of new classes in each state diminishes during the CIL process. 85

5.1 Overview of the proposed analysis framework of initial training strategies for EFCIL. 91

5.2 Correlation between the endogenous variables. 97

5.3 Diagnostics of the regression for the accuracy as in Equation 5.4. 100

5.4 Output of the regression for the accuracy 101

5.5 Accuracy gain by using strategy in row i over strategy in column j , e.g. "The accuracy of BYOL-ft is 17pts higher than SL(ResNet)". Only results in **bold** are statistically different. 103

5.6 Overall pairwise comparisons on Forgetting. Only results in **bold** are statistically different. 103

5.7 Interaction plot of the best strategies for different transfer types and for the 3 CIL algorithms. Similar slopes indicate similar behaviors. A change in slope indicates a change in behavior. 104

B.1 Overall pairwise comparisons 145

B.2 Pairwise gain of accuracy per dataset 146

B.3 Pairwise gain of accuracy per method 147

B.4 Pairwise gain of accuracy per number of classes in the initial state 148

List of Tables

2.1 Sensibilities of Model-Growth based, Fixed-Representation based and Fine-Tuning based methods over complexity, memory, and accuracy. Green means that the type of method is well suited for the criterion, orange means that it is not well suited, and red means that it is not suited at all. 34

2.2 Sensibilities of Model-Growth based, Fixed-Representation based and Fine-Tuning based methods over timeliness, plasticity, and scalability. Green means that the type of method is well suited for the criterion, orange means that it is not well suited, and red means that it is not suited at all. 35

3.1 Average top-1 accuracy with three numbers of states K per dataset. PlaStIL is applied on top of DeeSIL and DSLDA. **Best results - in bold**, second best - underlined. 62

3.2 Average top-1 incremental accuracy in a dataset transfer learning configuration. Results are given for transferring between all pairs of initial and target datasets. All experiments are run with $K = 10$ states. **Best results are in bold**. 63

3.3 Comparison of PlaStIL on top of DeeSIL [BP18] with a version that knowing the final composition of the different states will only fine-tune the relevant states (PlaStIL +oracle). The average gains are of +0.2% with PlaStIL +oracle. 66

3.4 Average top-1 accuracy with three numbers of states K per dataset, comparison of PlaStIL$_1$ on top of DeeSIL [BP18] with REMIND [Hay+20] with different budgets for the storage of their compressed vectors (1, 2 and 4 times the size of a ResNet18 on disk). **Best results - in bold**, second best - underlined. 66

4.1 Average top-1 incremental accuracy in EFCIL with different numbers of incremental steps. FeTrIL^1 results are reported with pseudo-features translated from the most similar new class. "-" cells indicate that results were not available. "x" cells indicate that the configuration is impossible for that method. **Best results - in bold**, second best - underlined. 78

4.2 Comparison of SDC [Yu+20] with FeTrIL^1 using the evaluation protocol for two supplementary datasets used in [Yu+20]. **Best results in bold.** 79

4.3 Comparison of ABD [Smi+21] with FeTrIL using the authors' evaluation protocol. ImageNet50 includes 50 classes and 5 states of 10 classes. ImageNet100 includes 100 classes, with 20 states of 5 classes each. Note that [Smi+21] uses top-5 accuracy for ImageNet50 and top-1 for ImageNet100 and we present the same numbers. **Best results in bold.** 80

4.4 Comparison of FeTrIL with AANets [LSS21], applied on top of EBCIL baselines which store 20 exemplars of past classes to mitigate catastrophic forgetting. 80
4.5 Average top-1 CIL accuracy obtained with the variants of pseudo-feature selection from Subsection 4.3.2 for $T = 5$. We set $k = \{1, 5, 10\}$ for the similarity rank between the past and new classes to test the effect of class similarities. There are 10 (CIFAR-100 and ImageNet-Subset) and 20 (TinyImageNet) new classes per state from which to select features translation. 82

5.1 Performance of three EFCIL algorithms with different training strategies for the initial model averaged over 16 target datasets and two EFCIL scenarios. BSIL [JLM21] is a recent EFCIL algorithm that is representative of fine-tuning-based CIL works. DSLDA [HK20] and FetrIL [Pet+23b] adapt linear probing [Kum+22] for EFCIL. We present the averaged incremental accuracy ($\mu_{\overline{Acc}}$) and the number of cases (W) in which a combination of algorithm and initial training strategy performs best for a combination of target dataset and EFCIL scenario (see Sec.5.4). Initial training strategies are defined by: Arch- deep architecture used (ResNet50 (RN50) [He+16] or vision transformer (ViT-S) [Dos+21]); Method - initial training method; FT - fine-tuning on initial classes of the target dataset; Ext- use of an external dataset, such as ILSVRC [Rus+15]; Sup - type of supervision for the initial model: self-supervised (SSL) or supervised (SL). 87
5.2 Variables predicting accuracy, sorted by decreasing importance 99
5.3 Variables predicting forgetting, sorted by decreasing importance 99
5.4 ANOVA results for each considered regression. Variables are significant at $p < 0.05$ and ordered by decreasing importance. 102

B.1 Summary of datasets. μ is the mean number of train images per class and σ is the standard deviation 139
B.2 Summary of datasets. μ is the mean number of train images per class and σ is the standard deviation 139
B.3 Number of images per class in the train and test subsets of each target dataset. The average and the standard deviation of the number of images per class are denoted by μ and σ respectively. 140

A List of publications

Grégoire Petit, Michael Soumm, Feillet Eva, Adrian Popescu, David Picard, and Bertrand Delezoide. „An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024

Grégoire Petit, Adrian Popescu, Eden Belouadah, David Picard, and Bertrand Delezoide. „PlaStIL: Plastic and Stable Memory-Free Class-Incremental Learning“. In: *Proceedings of The 2nd Conference on Lifelong Learning Agents*. Ed. by Sarath Chandar, Razvan Pascanu, and Doina Precup. Proceedings of Machine Learning Research. PMLR, 2023

Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. „FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2023, pp. 3911–3920

Eva Feillet, **Grégoire Petit**, Adrian Popescu, Marina Reyboz, and Céline Hudelot. „AdvisIL - A Class-Incremental Learning Advisor“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2023, pp. 2400–2409

B Datasets details and implementation details

B.1 Datasets details

B.1.1 PlaStIL: Plastic and Stable Exemplar-Free Class-Incremental Learning

Dataset	#Train	#Test	μ(Train)	σ(Train)
$ILSVRC$ [Rus+15]	1,231,167	50,000	1231.2	70.2
$Landmarks$ [Noh+17]	374,367	20,000	374.4	103.8
$iNaturalist$ [Van+18]	300,00	10,000	300.0	0.0

Tab. B.1.: Summary of datasets. μ is the mean number of train images per class and σ is the standard deviation

The datasets used in evaluation are designed for three visual classification tasks: object, natural species, and landmark recognition. Their main statistics are in Table B.1.

B.1.2 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning

Dataset	#Train	#Test	μ(Train)	σ(Train)
CIFAR-100 [Kri09]	50,000	10,000	500.0	0.0
TinyImageNet [LY15]	100,000	10,000	500.0	0.0
ImageNet-Subset [Rus+15]	128856	5,000	1288.56	44.85
ILSVRC [Rus+15]	1,231,167	50,000	1231.2	70.2

Tab. B.2.: Summary of datasets. μ is the mean number of train images per class and σ is the standard deviation

The datasets used in evaluation are designed for visual classification tasks. Their main statistics are in Table B.2. Since the actual test subsets are not provided by the organizers of the ImageNet LSVRC competition, we follow common practice in incremental learning [Reb+17; Cas+18; Hou+19] and use the original validation subsets for the test phase.

B.1.3 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning

Dataset	μ_{train}	μ_{test}	σ_{train}	σ_{test}
Casia100	250.0	50.0	0.0	0.0
Food100	750.0	250.0	0.0	0.0
Land100	300.0	50.0	0.0	0.0
IMN100$_1$	340.0	60.0	0.0	0.0
IMN100$_2$	340.0	60.0	0.0	0.0
Flora	340.0	60.0	0.0	0.0
Logo100	80.0	15.0	0.0	0.0
Qdraw100	500.0	100.0	0.0	0.0
Art100	150.0	25.0	0.0	0.0
MTSD100	100.0	20.0	0.0	0.0
Air100	80.0	20.0	0.0	0.0
Fungi100	300.0	10.0	0.0	0.0
Amph100	300.0	10.0	0.0	0.0
Land1k	374.37	20.0	103.83	0.0
iNat1k	300.0	10.0	0.0	0.0
Casia1k	60.0	28.0	0.0	0.0

Tab. B.3.: Number of images per class in the train and test subsets of each target dataset. The average and the standard deviation of the number of images per class are denoted by μ and σ respectively.

We experiment with a wide variety of datasets in terms of domain, granularity, number of samples per class, and complexity of patterns to recognize. We select thirteen datasets containing 100 classes and three datasets containing 1000 classes as follows. The datasets IMN100$_1$ and IMN100$_2$ are obtained by randomly sampling 100 classes from ImageNet-21k [Den+09] which are not present in ILSVRC [Rus+15]. Flora is a thematic subset of ImageNet obtained by sampling 100 classes under the concept ‘flora‘, without intersection with ILSVRC. We also used 100-classes subsets of WikiArt [SE15] (Art100), Casia-align [Yi+14] (Casia100), Food101 [BGV14] (Food100), FGVC-Aircraft [Maj+13] (Air100), MTSD [MY16] (MTSD100), Google Landmarks v2 [Wey+20] (Land100), Logo2K [Wan+20] (Logo100) and Quickdraw [HE17] (Qdraw100). We build two fine-grained subsets from iNaturalist [Van+18] (2018 version) by selecting (i) amphibia species (Amph100) and (ii) fungi species (Fungi100) which do not intersect with the ILSVRC dataset. Finally, we also use three 1000-classes subsets of Casia-align (Casia1k), Google Landmarks v1 [Noh+17] (Land1k), and iNaturalist (iNat1k), respectively.

The average number of images per dataset is reported in Table B.3.

B.2 Implementation details

B.2.1 PlaStIL: Plastic and Stable Exemplar-Free Class-Incremental Learning

As already mentioned in Section 3.4.3, we used the authors' optimal parameters to run all baselines. A ResNet-18 model [He+16] and an SGD optimizer with $momentum = 0.9$ are used for all methods. We explicitly list the learning parameters of each method hereafter:

Learning from scratch
This type of learning is used to train models of the initial state (because it is not incremental), and also $Joint$, the upper bound method where all classes are learned with all their data at once.

Following [BPK20], $Joint$ and the first models of FT and SIW [BPK20] are run for 120 epochs using $batch\ size = 256$ and $weight\ decay = 0.0001$. The lr is set to 0.1 and is divided by 10 when the error plateaus for 10 epochs.

For REMIND [Hay+20], DSLDA [HK20], DeeSIL [BP18], Fixed$_{NCM}$ [Reb+17] and PlaStIL, we follow [Hay+20] and run the model for 90 epochs using $lr = 0.1$, $batch\ size = 128$, $weight\ decay = 0.00001$. The lr is set to its initial value decayed by 10 every 30 epochs. The lr is constrained to not decrease beneath 0.001.

For LUCIR, LwF, the first model is trained in the same manner as subsequent models (detailed below), following the authors of [Hou+19; Reb+17].

Incremental Learning
Here, we describe the hyper-parameters used to train the models incrementally for model-update-based methods.

- FT [BPK20] - IL models are trained for 35 epochs with $batch\ size = 256$, $momentum = 0.9$ and $weight\ decay = 0.0001$. The learning rate is set to $lr = 0.1/t$ at the beginning of each incremental state ($t \geq 2$) and is divided by 10 when the error plateaus for 5 consecutive epochs.

- LwF [Reb+17] - all models are trained for 60 epochs using $lr = 1.0$, $batch\ size = 128$, and $weight\ decay = 0.0001$. The learning rate is divided by 5 at epochs 20, 30, 40, and 50.

- LUCIR [Hou+19] - all models are trained for 90 epochs using $lr = 0.1$, $batch\ size = 128$ and $weight\ decay = 0.0001$. The lr is divided by 10 at epochs 30 and 60. The method-specific parameters are the same as those from the original paper [Hou+19] and can also be found once we release the codes and configuration files.

- SIW [BPK20] - is trained using the same hyper-parameters of FT following the authors.

- SPB-M [WGL21] - all models are trained for 90 epochs using $lr = 0.1$, $batch\ size = 128$ and $weight\ decay = 0.0001$. The lr is divided by 10 at epochs 30 and 60. The method-specific parameters are the same as those from the original paper [WGL21] and can also be found once we release the codes and configuration files. Note that SPB-M [WGL21] has no available code, we implemented it by modifying significantly the code of LUCIR [Hou+19], and verified its correctness on the results the authors provided in their paper.

- REMIND [Hay+20] - method-specific parameters are the same as those from the original paper, and run from the available code, using the advised version of pytorch

We reimplemented the method and verified its correct functioning using the protocol from the original paper.

B.2.2 FeTrIL: Feature Translation for Exemplar-Free Class-Incremental Learning

When implementations of compared methods were available, we first tested them using the protocol and datasets from the original paper to make sure that we reproduced their results. We then used the authors' optimal parameters to test these methods in our evaluation setting. Note that for the sake of fairness, all baselines were run using both training and validation sets (from Table B.2). A ResNet-18 model [He+16] and an SGD optimizer with $momentum = 0.9$ are used for all methods. We explicitly list the learning parameters of each method hereafter:

1. **Training the initial model:**

   This training regime is needed to obtain the initial model for each method, and also Joint training which can be considered the upper bound method where all classes are learned with all their data at once. We used the parameters provided by the authors as follows.

   Joint and the first models of FT and SIW are trained using the parameters from [BPK20]. Each model is learned for 120 epochs using $batch\ size = 256$ and $weight\ decay = 0.0001$. The lr is set to 0.1 and is divided by 10 when the error plateaus for 10 epochs.

The lr is set to its initial value decayed by 10 every 30 epochs. The lr is constrained to not decrease beneath 0.001.

For LUCIR, the first model is trained in the same manner as subsequent models (detailed below), following the original protocol from [Hou+19].

2. **Training the incremental models:**

   Here, we describe the hyper-parameters used to train the methods which were retrained for Chapter 4.

   - **LUCIR** [Hou+19] - all models are trained for 90 epochs using $lr = 0.1$, $batch\ size = 128$ and $weight\ decay = 0.0001$. The lr is divided by 10 at epochs 30 and 60. The method-specific parameters are the same as those from the original paper [Hou+19] and can also be found once we release the codes and configuration files.

   - **DeeSIL** [BP18] - the initial model is the same one used for FeTrIL. The training of linear classifiers is also done using the same parameters.

B.2.3 An Analysis of Initial Training Strategies for Exemplar-Free Class-Incremental Learning

Incremental learning algorithms

- **BSIL.** Our implementation of LUCIR [Hou+19] algorithm with a Balanced Cross-Entropy loss [JLM21] is based on the original repository of [Hou+19][1]. LUCIR was initially proposed as a CIL algorithm with rehearsal. In practice, as we focus on EFCIL, we set the size of LUCIR's memory buffer to zero.

- **DSLDA.** Our implementation is based on the original repository of [HK20][2].

- **FeTrIL.** Our implementation is based on the original repository of [Pet+23b][3].

Pre-training algorithms

[1]`https://github.com/hshustc/CVPR19_Incremental_Learning`
[2]`https://github.com/tyler-hayes/Deep_SLDA`
[3]`https://github.com/GregoirePetit/FeTrIL`

The pre-trained models are taken from the repositories indicated in the footnotes: DINOv2[4], BYOL[5], and DeiT[6]. We also used the method MoCov3[7] for training models with a ResNet50 architecture in a self-supervised manner on the initial data subset of each target dataset.

Fine-tuning

We use PyTorch[8] implementation of ResNet50 architecture and the ViT-Small transformer architecture from the checkpoints of DINOv2 and DeiT. When fine-tuning the models, in the case of ResNet50, we freeze the first 3 convolutional blocks and only update the parameters belonging to the last convolutional block, as well as the linear layer. In the case of ViT-Small, we freeze the blocks up to block 8 and update the blocks 9 to 11, as well as the linear layer. In both cases, the parameters are updated using a learning rate equal to one-tenth of the value of the base learning rate used to pre-train the model.

B.3 Influence of factors on accuracy

In this section, we provide additional information about the statistical study of Chapter 5.

Let us recall the overall pairwise comparisons in Figure B.1. We explore the effects of other variables by splitting the data with respect to a studied variable and report the regression results separately.

- Figure B.2 presents the results for each target dataset
- Figure B.3 presents the results for each incremental algorithm
- Figure B.4 presents the results depending on the number of classes in the first stats

[4]`https://github.com/facebookresearch/dinov2`
[5]`https://github.com/yaox12/BYOL-PyTorch`
[6]`https://github.com/facebookresearch/deit`
[7]`https://github.com/facebookresearch/moco-v3/tree/main`
[8]`https://pytorch.org/vision/main/_modules/torchvision/models/resnet.html#resnet50`

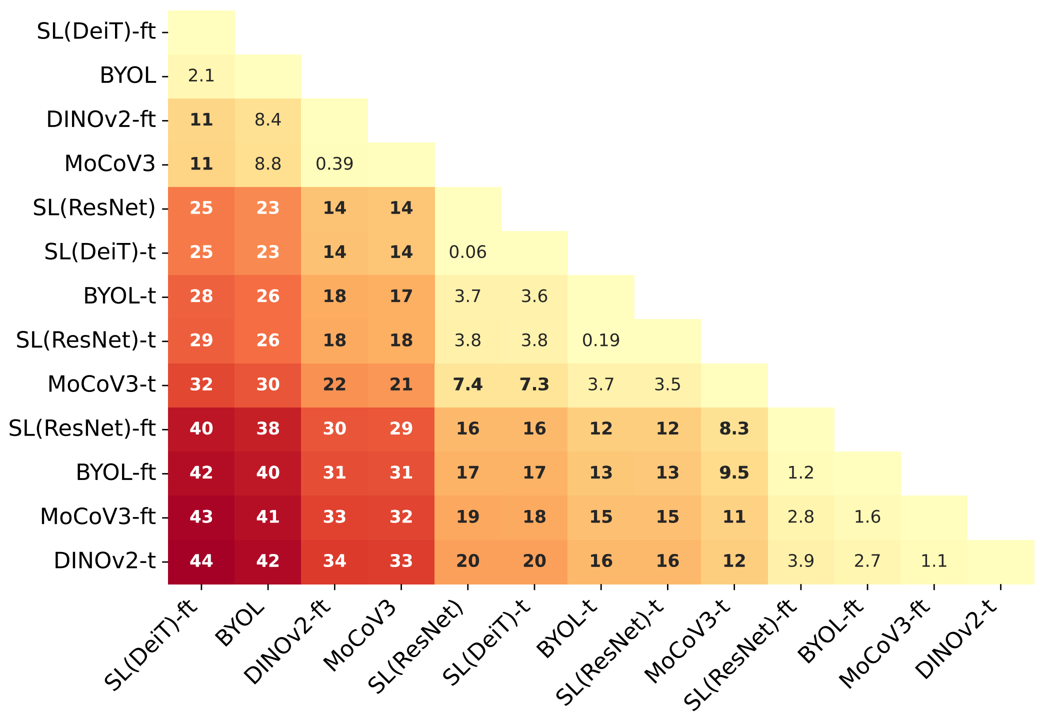


Fig. B.1.: Overall pairwise comparisons

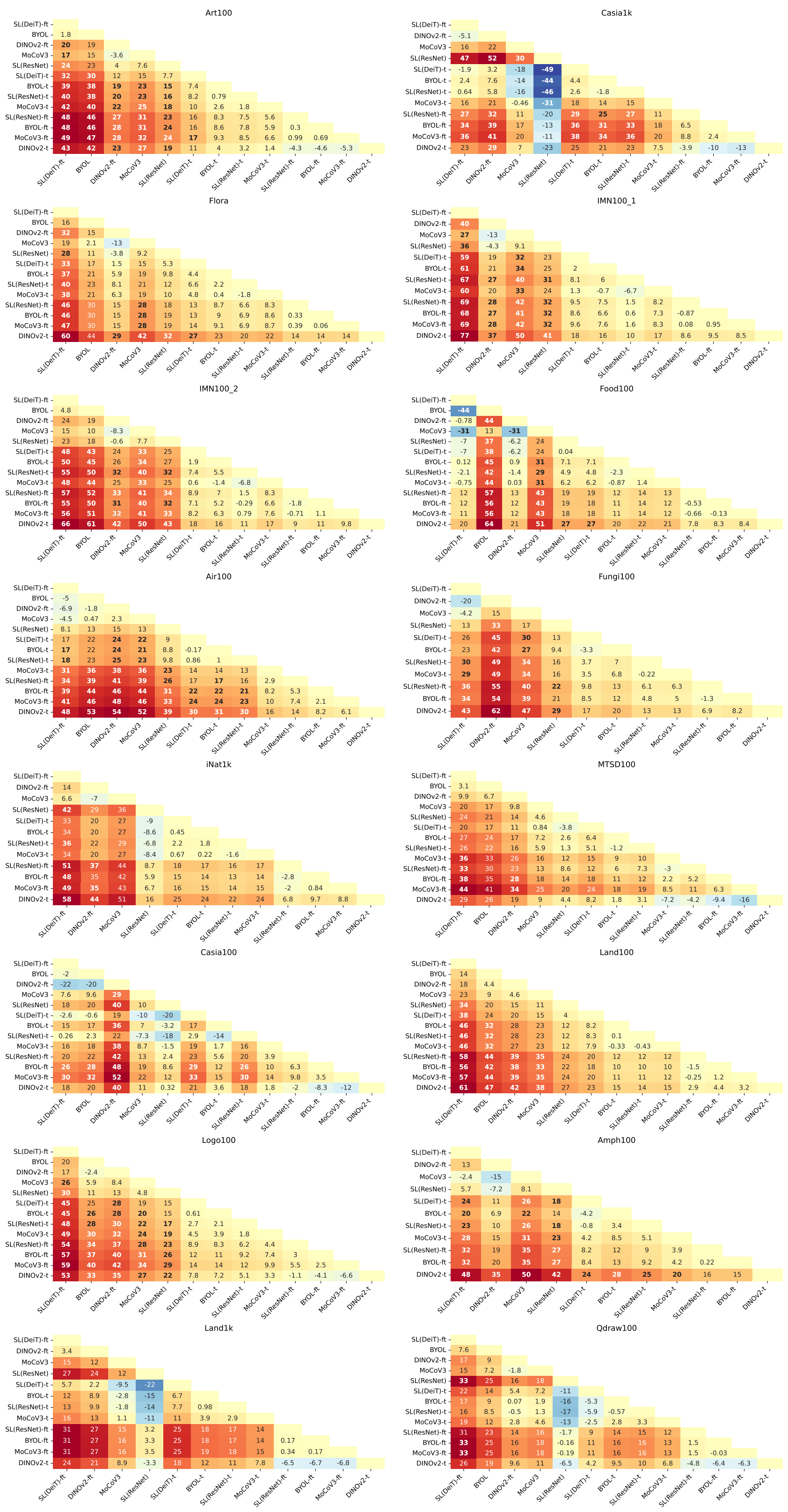
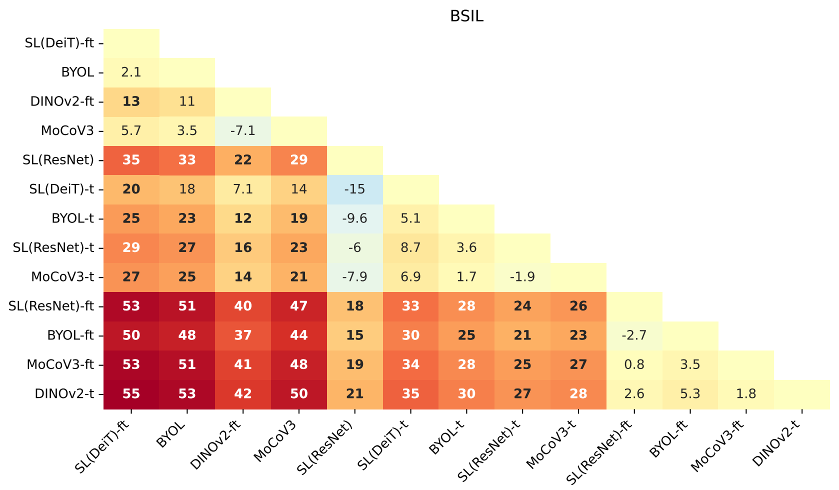
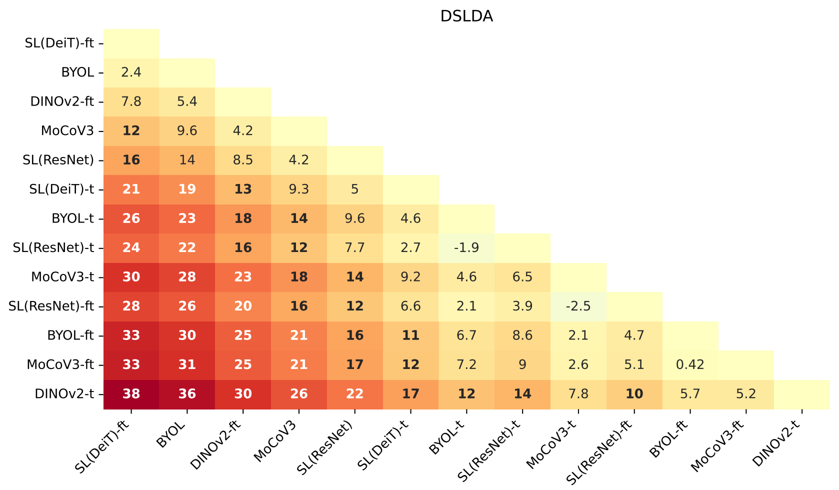


Fig. B.2.: Pairwise gain of accuracy per dataset





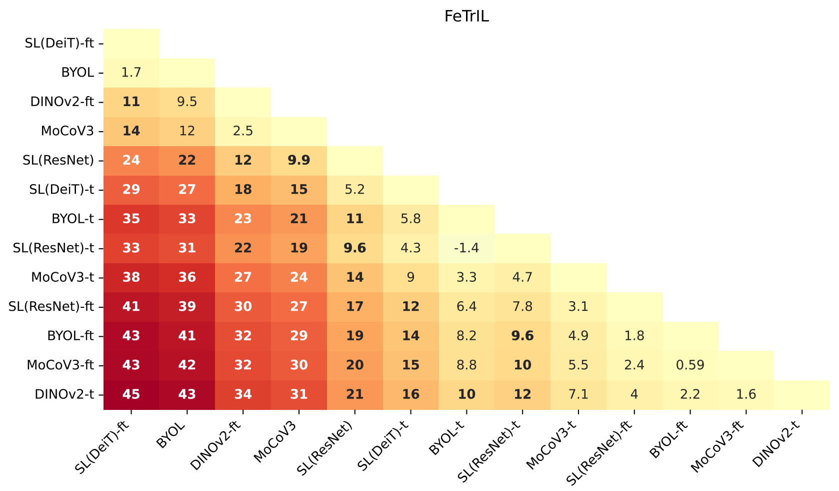


Fig. B.3.: Pairwise gain of accuracy per method

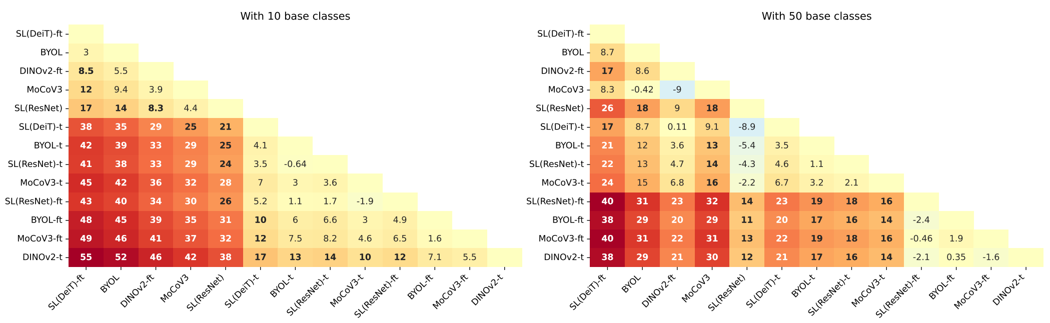


Fig. B.4.: Pairwise gain of accuracy per number of classes in the initial state

C Résumé en français

C.1 Introduction

Nous introduisons dans le chapitre 1, l'apprentissage incrémental de classes sans exemple (EFCIL). L'EFCIL est un domaine complexe et difficile qui nécessite des solutions innovantes pour permettre l'apprentissage continu de nouvelles classes tout en préservant les connaissances précédemment acquises. Plusieurs défis majeurs contribuent à la complexité de l'EFCIL, chacun d'entre eux nécessitant un examen attentif et des approches spécialisées.

Tout d'abord, l'EFCIL est confronté au problème de la mémoire. Le modèle apprend continuellement un grand nombre de classes, sans avoir la possibilité de rappeler les images des classes apprises précédemment. La croissance linéaire des besoins en mémoire avec le nombre de classes poserait sinon d'importants problèmes de calcul et de stockage. Il est donc essentiel de concevoir le modèle EFCIL de manière à minimiser l'impact sur l'utilisation de la mémoire lorsque de nouvelles classes sont ajoutées.

Deuxièmement, les exigences en matière de calcul constituent un défi important de l'EFCIL. Il est impératif d'apprendre de nouvelles classes efficacement sans compromettre la qualité des représentations apprises. Lorsque le nombre de classes augmente, le temps d'apprentissage pour chaque nouvelle classe augmente également, ce qui entraîne des goulets d'étranglement au niveau du calcul. Les techniques visant à atténuer la dérive des caractéristiques, telles que la distillation, peuvent encore augmenter le coût de l'apprentissage. Dans ce contexte, il est essentiel de développer des algorithmes d'apprentissage économes en ressources, capables d'apprendre de nouvelles classes avec des ressources informatiques limitées. Ces algorithmes doivent trouver un équilibre entre la conservation des connaissances antérieures (stabilité) et l'acquisition efficace de connaissances des nouvelles classes (plasticité).

Le nombre d'états que le modèle doit apprendre est un autre facteur critique affectant les performances de l'EFCIL. Au fur et à mesure que le modèle accumule des connaissances sur un nombre croissant de classes, le défi de l'oubli catastrophique devient plus prononcé. Le modèle doit conserver les informations sans revenir sur les exemples passés, ce qui devient de plus en plus difficile à mesure que le nombre de classes augmente. Pour atténuer l'oubli catastrophique, il est essentiel d'explorer des techniques qui permettent au modèle de conserver efficacement une certaine forme de connaissance du passé. La prise en compte de la relation entre le nombre d'états et le potentiel d'oubli catastrophique est essentielle pour le développement d'algorithmes EFCIL robustes.

La variabilité des structures des flux de données constitue un obstacle important à la réalisation d'un apprentissage EFCIL efficace et adaptatif. Les caractéristiques du flux de données, y compris les types de données, la fréquence de mise à jour et la taille des états incrémentaux, affectent toutes le processus d'apprentissage.

La diversité des structures des flux de données nécessite des approches d'apprentissage incrémental par classe adaptées, capables de gérer différents types de données et de s'adapter à différentes fréquences de mise à jour. En outre, ces approches doivent utiliser efficacement les informations disponibles dans les états incrémentaux pour soutenir de manière optimale l'apprentissage de nouvelles classes.

Enfin, la variabilité des scénarios a un impact significatif sur les performances des modèles EFCIL, en particulier en ce qui concerne l'atténuation des oublis catastrophiques. Différents scénarios dictent la composition de l'état initial, ce qui affecte la richesse de l'espace des caractéristiques à partir duquel le modèle apprend. L'étude de la manière d'optimiser la composition de l'état initial et l'élaboration de stratégies permettant d'adapter efficacement le modèle à différents scénarios sont essentielles pour améliorer les capacités d'adaptation et de généralisation du modèle.

C.2 État de l'art

Les méthodes de l'état de l'art de l'EFCIL peuvent être divisées en trois catégories : *basées sur la croissance du modèle*, *basées sur le réglage fin* et *basées sur un modèle fixe*. Les méthodes basées sur la croissance du modèle sont évolutives et permettent d'accueillir de nouvelles classes avec des ressources suffisantes pour la croissance du modèle au cours de la formation et de l'inférence. En outre, elles permettent d'atténuer le risque d'oubli catastrophique lorsque le modèle s'enrichit de nouvelles classes, en conservant dans une certaine mesure la connaissance des classes antérieures. Cependant, les méthodes basées sur la croissance du modèle nécessitent une mémoire et des ressources informatiques importantes en raison de la croissance du modèle avec chaque nouvelle classe, ce qui les rend coûteuses en termes de calcul et de memoire dans les scénarios d'apprentissage à long terme. Il est essentiel de contrôler le nombre de paramètres supplémentaires pour chaque nouvelle tâche afin d'éviter que le modèle ne devienne excessivement grand et gourmand en ressources. Le réglage fin permet au modèle de s'adapter aux nouvelles données, offrant un certain niveau de plasticité pour tenir compte des changements dans le système. Cependant, le réglage fin a des limites en termes de rapidité, car il ne permet pas l'intégration rapide de nouvelles connaissances, nécessitant plus qu'un simple entraînement de la couche de poids du classificateur, ce qui peut s'avérer critique dans les applications sensibles au temps. En outre, il manque de stabilité, car il peut conduire à un oubli catastrophique des connaissances acquises précédemment, étant donné qu'aucune connaissance antérieure n'est préservée dans les méthodes d'EFCIL. Les méthodes basées sur la représentation fixe permettent quant à elles de prendre en charge un grand nombre de classes sans augmentation linéaire des besoins en mémoire, ce qui les rend adaptées aux scénarios d'apprentissage à long terme. Elles sont plus stables puisque la représentation reste fixe et donc inchangée pendant l'apprentissage de nouvelles tâches, ce qui contribue à réduire les besoins de calcul et la stabilité. Néanmoins, les méthodes basées sur la représentation fixe ont une plasticité limitée car la représentation, fixée dans l'état initial, peut devenir moins transférable si les tâches incrémentales changent radicalement d'un état à l'autre.

Ainsi, en termes d'efficacité, de plasticité et de stabilité, nous avons choisi d'introduire deux méthodes *basées sur un modèle fixe* et une analyse statistique dans cette thèse.

C.3 PlaStIL : Apprentissage par classe sans mémoire, plastique et stable

L'approche PlaStIL s'attaque à un défi essentiel de l'EFCIL, qui consiste à trouver le bon équilibre entre plasticité et stabilité. La plasticité se réfère à la capacité du modèle à apprendre de nouvelles données, tandis que la stabilité se rapporte à la conservation de la connaissance des classes précédemment apprises. Les méthodes traditionnelles de CIL utilisent des tampons de mémoire ou maintiennent deux modèles profonds séparés pour atteindre cet équilibre, mais ces approches sont souvent inefficaces en termes d'utilisation de la mémoire et de besoins de calcul.

PlaStIL propose une nouvelle solution qui distribue un nombre similaire de paramètres que les méthodes basées sur la distillation, mais d'une manière plus efficace. Elle gèle l'extracteur de caractéristiques après l'état initial et introduit plusieurs têtes de modèle pour assurer une grande plasticité. En gelant l'extracteur de caractéristiques, le modèle conserve sa stabilité en formant les classes incrémentales les plus anciennes à l'aide de cet extracteur fixe. D'autre part, il introduit la plasticité dans les nouvelles classes en utilisant des têtes de modèles partiellement affinées. Cette approche permet à PlaStIL de fournir un compromis équilibré entre plasticité et stabilité.

La méthode est évaluée de manière approfondie sur trois grands ensembles de données, et les résultats démontrent son efficacité sur les méthodes existantes dans diverses configurations. En obtenant de meilleures performances et une plus grande efficacité dans l'apprentissage incrémental par classe, PlaStIL constitue une avancée prometteuse dans la résolution du dilemme plasticité-stabilité dans l'EFCIL.

C.4 FeTrIL : Translation des caractéristiques pour l'apprentissage incrémental par classe sans exemple

FeTrIL, comme PlaStIL, se concentre sur le dilemme plasticité-stabilité dans l'EFCIL. Les méthodes traditionnelles d'EFCIL privilégient soit la stabilité, en utilisant des extracteurs de caractéristiques fixes, soit la plasticité, en procédant à des ajustements successifs du modèle. FeTrIL vise à trouver un meilleur équilibre entre ces deux aspects en introduisant une nouvelle approche.

La principale contribution de FeTrIL est l'introduction d'un générateur de pseudo-caractéristiques, qui est un composant efficace et simple. Ce générateur ne nécessite pas de stocker des exemples de classes antérieures, mais utilise plutôt les représentations des centroïdes de ces classes. À l'aide de ces centroïdes, FeTrIL crée des pseudo-caractéristiques pour les classes passées, qui sont combinées avec les caractéristiques réelles des nouvelles classes. Ces caractéristiques combinées sont ensuite utilisées pour entraîner de manière incrémentale un classificateur linéaire qui fait la distinction entre toutes les classes.

L'avantage de cette approche est son efficacité en termes de calcul, puisque seul un composant minimal du modèle profond, le classificateur linéaire, doit être mis à jour au cours du processus incrémental. FeTrIL est donc plus rapide que les méthodes traditionnelles qui mettent à jour l'ensemble du modèle profond.

FeTrIL est évalué de manière approfondie sur trois ensembles de données difficiles dans différents contextes incrémentaux, et il surpasse dix méthodes existantes dans la plupart des cas, démontrant son efficacité dans l'EFCIL. En combinant les forces des extracteurs de caractéristiques fixes avec la simplicité et l'efficacité du générateur de pseudo-caractéristiques, FeTrIL offre une solution nouvelle et prometteuse au dilemme plasticité-stabilité dans l'EFCIL.

C.5 Analyse des stratégies d'initialisation pour l'apprentissage incrémental de classes sans exemple

Cette contribution aborde un défi fondamental de l'EFCIL, à savoir l'incorporation de nouvelles classes dans le modèle sans oublier les connaissances acquises précédemment. Contrairement à l'apprentissage incrémental traditionnel où les données antérieures peuvent être stockées dans une mémoire tampon, les méthodes EFCIL ne peuvent pas s'appuyer sur des exemples, ce qui rend la stratégie d'initialisation cruciale.

La contribution se concentre sur l'étude de la façon dont le modèle initial du processus EFCIL peut être construit de deux façons : en utilisant uniquement l'état initial de l'ensemble de données cible ou en incorporant des poids pré-entraînés à partir d'un ensemble de données auxiliaire.

Afin d'obtenir des informations plus approfondies, la contribution mène une étude expérimentale étendue et introduit un cadre d'analyse statistique pour quantifier la contribution relative des différents facteurs à la performance incrémentale. La principale conclusion est que la stratégie d'initialisation a un impact significatif sur la précision incrémentale moyenne, mais que le choix de l'algorithme d'apprentissage incrémental par classe (CIL) joue un rôle plus important dans la prévention de l'oubli.

Sur la base de ces résultats, l'étude fournit des recommandations pratiques pour choisir la bonne stratégie d'initialisation pour différents cas d'utilisation de l'EFCIL. Cette analyse contribue à une meilleure compréhension des facteurs influençant l'apprentissage incrémental basé sur les classes et offre des indications précieuses pour optimiser et déployer efficacement les méthodes d'EFCIL.