UNIVERSITÀ
DEGLI STUDI
FIRENZE

PHD PROGRAM IN SMART COMPUTING
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)

Efficient Architectures and Incremental Methodologies for Deep Learning

Simone Magistri

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

Efficient Architectures and Incremental Methodologies for Deep Learning

Simone Magistri

Advisor:

Prof. Andrew D. Bagdanov

Head of the PhD Program:

Prof. Stefano Berretti

Evaluation Committee:
Professor Simone Calderara, *Università degli Studi di Modena e Reggio Emilia*
Research Fellow Bartłomiej Twardowski, *Universitat Autònoma de Barcelona*

XXXVI ciclo — January 2024

Acknowledgments

As I reflect on the journey of my PhD, I am filled with immense gratitude for the wealth of experiences and knowledge I have gained. This period has been transformative, marked by intellectual growth, challenges, and the development of resilience. I would be lying if I said that my PhD journey was easy. My biggest challenge was understanding whether I was approaching the problem in the right way and determining if the several ideas I had deserved to be pursued, even if the initial results were not encouraging, or discarded. In this regard, the people I collaborated with during this journey were crucial. Words cannot describe my gratitude towards my collaborators, but I will try my best to express it in this brief paragraph.

I would like to express my deepest gratitude to my supervisor Andrew Bagdanov. Andrew Bagdanov, or perhaps better Andy, has been more than a supervisor. He taught me to reason, present, organize, and express ideas. Andy has spent a lot of time discussing ideas with me and analyzing my failures. His patience in discussing ideas, irrespective of their initial promise, and his encouragement to push the boundaries of conventional thinking have been invaluable. Thank you so much for the time dedicated and for the valuable teachings.

I am extremely grateful to Fabio Schoen. Fabio was my supervisor during my bachelor's and master's theses and welcomed me to the Global Optimization Laboratory (GOL) during my PhD period. I have always admired the passion that Fabio puts into teaching and his ability to instill a similar enthusiasm in his students. I believe he succeeded with me too, and I think that the lessons i learned from him have been fundamental in my growth as researcher.

I am also grateful to the friends at Verizon Connect. I collaborated with Verizon throughout my entire PhD journey and completed an internship under the Tuscany Region Program, for which I am thankful for giving me this opportunity. At Verizon, I learned to solve practical problems, analyze results, and organize work. In particular, I would like to thank Leonardo Taccari, who was also a member of the PhD Supervisory Committee, Tommaso Bianconcini, and Leonardo Sarti, with whom I collaborated both during the internship and throughout the entire PhD period. I am grateful to Marco Boschi, with whom I worked on the vehicle viewpoint estimation article, and to Samuele Salti from the University of Bologna for the useful suggestions he provided us.

Many thanks also go to Alessandro Piva, Daniele Baracchi, and Dasara Shullani from the LESC laboratory. Cooperating with them provided an opportunity to expand my knowledge in areas of artificial intelligence with which I was previously unfamiliar, namely Multimedia Forensics. Our collaboration resulted in work that I was fortunate enough to present at the IWBF conference in Barcelona.

In Barcelona, I also visited the Computer Vision Center (CVC) and met its mem-

bers. The CVC team was very hospitable, and in the short time I was there, thanks to the conversations we had, I believe I learned a lot, both theoretically and practically, about tackling problems. In particular, I must thank Joost Van de Weijer and Albin Soutif. I appreciated the stimulating meetings with Joost and Albin, where we discussed problems of Incremental Learning and interesting ideas for future work.

Now, let's move on to my friends at GOL, with whom I spent most of my PhD time. The first person I must thank is Tomaso Trinci. Tomaso arrived at the lab when I was in my second year, and he knew nothing about deep learning. I guided him for a while and tried to help him with what I knew. The amusing part is that, in a remarkably short period, Tomaso started to explain concepts to me. I will always remember the long daily conversations about Elastic Feature Consolidation, visualizing relevant directions, and trying to formalize the problem to understand the dynamics of training. The continuous discussions I had with Tomaso, Joost, Albin, and Andy, led us, to my immense satisfaction, to publish an article at ICLR 2024. Finally, I must thank the other members of GOL: Matteo Lapucci, Tommaso Aldinucci, Pierluigi Mansueto, Enrico Civitelli, and Francesco Carciaghi. Although optimization is not the primary focus of my research, your insightful feedback has been invaluable in helping me comprehend and address my challenges. Thank you from the bottom of my heart for the laughs, jokes, and idea exchanges during the PhD. I could not have undertaken this journey without all of you.

Abstract

Deep Neural Networks (DNNs) are fundamental to Artificial Intelligence (AI) applications, demonstrating remarkable performance across a diverse range of tasks in industries such as transportation, robotics, healthcare, and multimedia forensics. However, as these models have developed, the resources required for their training have increased significantly, which has dramatically impacted both costs and environmental sustainability. Currently, DNNs struggle to adapt to the dynamic and ever-changing nature of our world, often necessitating updates to address new situations or tasks. Traditionally, updating these models for novel situations involves re-training them from scratch, integrating both previous and current task data, a process known as *joint-incremental* training. This method exacerbates development costs and environmental impact. Relying solely on new task data for updates leads to *catastrophic forgetting* of previously acquired knowledge. To counter this, the main goal of incremental learning is to update these models effectively and efficiently, thereby mitigating catastrophic forgetting. Additionally, due to the high number of operations and parameters, deploying DNNs on edge devices is challenging; thus, they are often hosted on cloud servers for inference. This cloud-based inference presents challenges in terms of responsiveness and raises privacy concerns in real-world applications, such as intelligent vehicles.

This thesis tackles both the challenges of incremental learning and designing efficient DNNs for edge computing. For incremental learning, we propose a method named *Elastic Feature Consolidation*, which contributes meaningfully towards bridging the performance gap between existing incremental learning approaches and joint-incremental training, focusing on using only the current task data to prevent forgetting. We emphasize the importance of incremental learning in Multimedia Forensics by establishing a benchmark for *incremental social network identification* and assessing a broad range of incremental techniques, contributing to future research in this field. In terms of efficient architecture design, we develop *effective* and *lightweight vehicle viewpoint estimation* DNN models, optimized for edge applications in intelligent vehicles.

Contents

Chapter 1

Introduction

Over the past decade *Artificial Intelligence* (AI) has emerged as a foundational element in technological advancement, significantly influencing how companies and users interact with a wide range of applications such as transportation, robotics, biological applications and the creation of textual and multimedia content.

In the *transportation* sector AI is revolutionizing vehicle operation and their interaction with the environment, leading to a new era in this field. By processing data from an array of both visual and non-visual sensors, AI technologies are enhancing the functionalities of intelligent vehicles. This includes advances in autonomous driving [1], collision warning systems [2] and vehicle fleet management [3], leading to improved vehicle safety and greater efficiency in logistics management. Similarly, in *robotics* the integration of AI with sensor data is boosting robotic capabilities and refining decision-making processes in complex environments. This progress is evident in manufacturing [4] and assembly tasks in industries like aerospace [5] and packaging [6]. Additionally, the use of AI-driven drones in remote sensing has become more prevalent for environmental data collection and analysis [7].

AI has also made groundbreaking contributions to *medical* and *biological* fields. One notable example is *AlphaFold2* [8], which has demonstrated outstanding performance in predicting the 3D structures of proteins from their amino acid sequences. This capability is revolutionary for science and medicine, addressing challenges in protein structure identification. By reducing the time and cost of associated with understanding new proteins and biological processes, AlphaFold2 has the potential to revolutionize fields, such as drug discovery, enhancing the ability to design therapies and understanding complex biological systems.

AI is becoming increasingly influential for content creation, an area now commonly known as *Generative AI* [9]. This domain encompasses a broad spectrum of technologies dedicated to the production and use of textual and multimedia content. Central developments in this field include advanced AI chatbots like *LlaMA* [10] and *ChatGPT* [11]. These chatbots are capable of generating complex and coherent text,

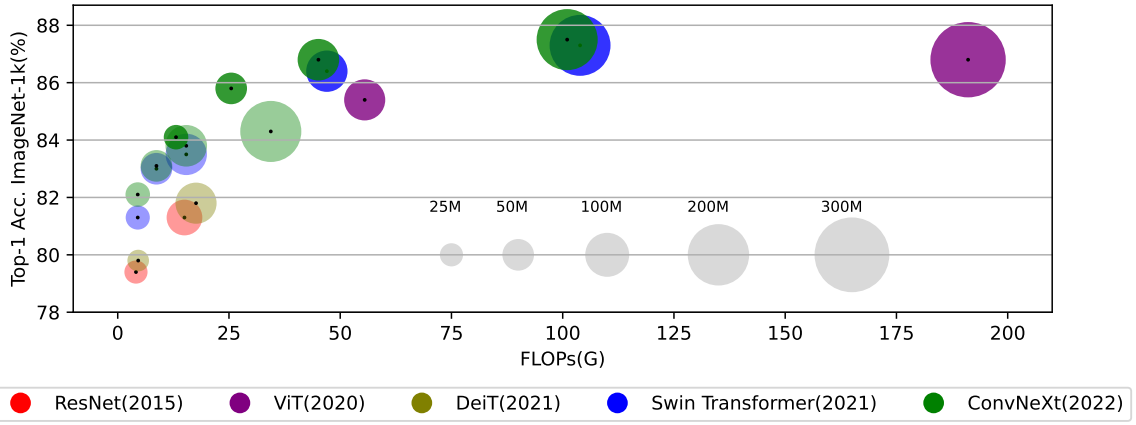


Figure 1.1: Deep Neural Networks for visual tasks. This figure illustrates the comparative performance of different variants in each family of recent vision-based models on the ImageNet-1K dataset, which contains 1.2 million images across 1,000 categories. It includes variants of the ResNet [14], ViT [15], DeIT [16], Swin Transformer [17], ConvNeXt [18] model families. The x-axis represents the GFLOPs of these models, and the y-axis show their Top-1 accuracy. The size of each circle indicates the number of parameters of the corresponding model. Darker bubbles denote models pre-trained on ImageNet-22K, a superset of ImageNet-1K with 14 million images and 21,841 classes. These results were extracted from [18].

but also are excellent at answering user queries and tackling complex tasks. Their influence extends beyond mere text generation, revolutionizing customer service, text editing, and communication strategies with more efficient and personalized interactions. In the area of multimedia content, Generative AI is widely employed in applications requiring realistic video and image content generation [12]. This is evident, for instance, in the gaming industry, where AI systems enhance user experience by generating dynamic and realistic environments.

However, the rise of intelligent systems producing highly realistic content has led to an increase in the fake digital content production, posing significant risks to users. These phenomena have serious implications, such as misinformation, identity theft and fraud. In response, the discipline of *Multimedia forensics* [13] has become increasingly crucial. This field, focusing on the analysis of digital content to prevent and detect criminal activities, is now heavily reliant on AI systems for identifying fake content and tracing the origin of counterfeit or illicit material.

Deep Neural Networks (DNNs) models have become integral to modern AI applications, handling data from various modalities, such as images and sequences, and achieving outstanding performance across a large variety of tasks. *Convolutional Neural Network* (CNN) [14] and *Vision Transformer* [15] models excel at visual tasks, including image/video classification, object detection, and visual content generation. Transformers [19] are also central to sequence-based tasks like natural lan-

guage processing, speech recognition and sequence identification. *Large Language Models* (LLMs) like GPT-4 and LlaMA-2, based on the Transformer architecture, have shown exceptional performance in language understanding and text generation. Additionally, innovative text-to-image models such as *Latent Diffusion Models* [20], are revolutionizing image generation by merging the strengths of both vision and sequence models, creating realistic and accurate visual content from textual descriptions.

However, despite the remarkable success of DNNs in terms of performance, they face several significant challenges, particularly regarding *adaptability* and *resource efficiency*. A primary issue is the *large number of parameters* and the *high volume of operations*. As models have advanced, both the number of parameters and the number of mathematical operations required for inference have increased exponentially. For instance, recent vision-based models have hundreds of millions of parameters and perform hundreds of billions of floating point operations per second (see Figure 1.1 for an overview of these models). The complexity is even more pronounced for Large Language Models, with parameters potentially reaching into the trillions. This increase in complexity necessitates vast amounts of data, on the order of billions or even trillions of data points. Some models, like Vision Transformers, require an expensive initial pre-training phase on a huge dataset to achieve optimal performance. This initial phase is usually performed on an auxiliary dataset and can rely upon self-supervised techniques [21]. The requirements mentioned above lead to long training times and the need for massive GPU clusters to train these models in a reasonable time. Consequently, this dependency on massive computational resources not only makes the process time-consuming and costly, it also raises concerns about the environmental impact and sustainability of the development of deep learning models [22, 23].

Another significant limitation of such models is their lack of *adaptability* to new tasks. The current learning paradigm of DNNs is *static*. DNNs are trained to perform a single task, using all the available examples in the training dataset before their deployment in real world applications. Consequently, after the learning phase the model remains unchanged and its application scope becomes fixed and eventually outdated and obsolete. This rigidity is a notable drawback in our dynamic and ever-evolving world. For instance, the rise of Internet of Things (IoT) smart devices across various sectors like transportation, healthcare, and robotics has led to a considerable increase in data acquisition rates and the nature of data itself is continuously changing. As another example, in Multimedia Forensics deep learning models are employed to identify fake content or unauthorized sharing of private media on social networks. However, these models quickly become ineffective at solving their tasks when the image generation techniques or social media platforms change. Therefore, a more *adaptive learning* paradigm is required, allowing DNNs

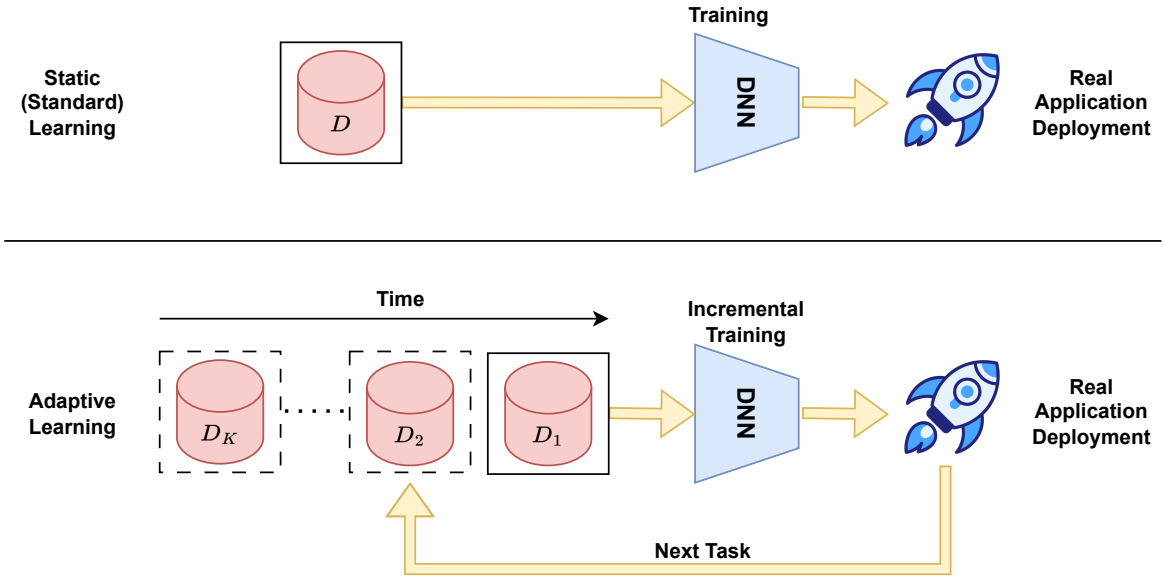


Figure 1.2: Static versus adaptive Learning. In static learning, the Deep Neural Network (DNN) is trained once on a specific set of task data and then deployed for its application. In adaptive learning, the DNN is incrementally trained on a sequence of task data, allowing it to be deployed multiple times in its applications.

to learn incrementally from data and adapt as new task data arrive over time (see Figure 1.2).

The current practices for managing multiple tasks over time involve either training a separate DNN for each task or reusing the initially-trained model to learn the new tasks. The first approach significantly increases memory usage and efficiency demands. Considering the resource-intensive characteristics of modern DNNs, this method quickly become unsustainable as the number of tasks grows. Additionally, it does not consider the relationships between tasks that often occur in real world applications. For instance, in an autonomous driving system correctly localizing vehicles can enhance the performance in determining their orientation. Moreover, using a separate model for each task can be impractical, especially when the choice of the appropriate model for testing is unclear, such as in scenarios where models are updated to recognize new object categories.

A different strategy, which is more efficient in terms of parameter usage and inference efficiency, involves exploiting the over-parameterization of DNNs to update an existing model to learn new tasks. However, sequentially training the model on the tasks, a strategy known as *fine-tuning*, brings its own challenges. In the standard training scheme, DNNs lack the capability to *incrementally* learn over time, and they suffer from a phenomenon known as *catastrophic forgetting* [24]: when a DNN, initially trained on one task, is subsequently trained exclusively on new tasks, it quickly forget how to perform the original task. The typical approach to solving this issue is

joint-incremental training, which involves retraining the network with both new and old task data. While this method is optimal in maintaining performance, it leads to several complications: the need to store an ever-increasing amount of data results in escalating costs and data storage challenges; the training duration expands as the number of tasks grows, becoming unsustainable in terms of cost and environment impact; and legal challenges emerge, especially with privacy laws like the European GDPR and the American CCPA, which grant individuals the right to request the deletion of personal data. The use of such sensitive data for training models means that if this data need to be removed, re-training the DNN with a mix of old and new data becomes impossible. Hence, a more efficient and privacy-preserving solution, which does not require storing all past data, is required.

The concept of training a DNN on a sequential set of tasks while retaining memory of older ones is closely related to the stability-plasticity dilemma [25]. This dilemma, significant in both neural network theory and cognitive science, poses the challenge of balancing the retention of learned information (*stability*) with the adaptation to new information (*plasticity*). In humans, this balance allows for the retention of past experiences while adapting to new situations. However, in standard learning, DNNs tend to exhibit high plasticity for learning new tasks but low stability for older ones. *Incremental learning* is an active research field aiming to develop DNN-based systems that achieve the right trade-off between stability and plasticity, thereby retaining knowledge of previously learned tasks while efficiently adapting to new tasks and addressing catastrophic forgetting.

Another significant limitation of DNNs is their deployment in real-world applications during the inference phase. Given the increasing number of operations and parameters characterizing these models, companies typically deploy them on cloud servers with powerful computational capabilities to provide services to their customers, such as voice assistance [26] or text generation [11, 10]. However, cloud-based inference suffers from issues with responsiveness, which can be a significant constraint in, for instance, autonomous driving systems, as well as from privacy concerns [27]. Therefore, any recent advancements in deep learning architectures focus on designing efficient architectures for moving the inference of DNNs to edge devices [28], although this may require sacrificing some performance.

In this thesis, we address both the challenges of incremental training and designing efficient DNNs for inference phases in edge-computing applications. In the area of incremental learning, we investigate the major causes of catastrophic forgetting and methods to mitigate it. Furthermore, we explore the application of incremental learning in the real-world scenario of Multimedia Forensics. Regarding the design of efficient architectures, we develop effective and lightweight vehicle viewpoint estimation DNN models, suitable for edge applications in the intelligent vehicle sector.

This thesis introduces three major original contributions, summarized as follows:

- **Elastic Feature Consolidation for Exemplar-Free Incremental Learning**: we propose *Elastic Feature Consolidation*, a novel method for exemplar-free class incremental learning that relies on a pseudo-metric in the feature space, named *Empirical Feature Matrix*, and leverages past DNN features, called prototypes, to incrementally train a model. Our method achieves notable improvements in terms of the stability-plasticity tradeoff with respect to recent methodologies.
- **Incremental Learning in Social Network Identification**: we propose realistic incremental scenarios for social network identification (namely *Incremental Social Version Classification* and *Incremental Social Platform Classification*). We develop a suitable architecture for this task and evaluate several incremental learning methods, including Elastic Feature Consolidation, to build a benchmark and draw attention to this topic for future research.
- **Vehicle Viewpoint Estimation from Monocular Cameras**: we propose neural networks that are both lightweight and effective for vehicle viewpoint estimation in intelligent vehicle systems, specifically designed for edge-computing applications. By combining object detection information and incorporating geometric constraints into the training loss, we significantly enhance the performance of viewpoint estimation. Although we have not yet evaluated incremental learning methodologies in the field of intelligent vehicles, we plan to address this in future work.

The rest of this thesis is organized as follows:

- In Chapter 2 we introduce some required background related to incremental learning, focusing on the main learning paradigms, the causes of forgetting, and recent advancements in this field. From a high-level perspective, we describe both exemplar-free and exemplar-based class incremental learning algorithms, and delve deeper into some foundational approaches that have inspired the most recent state-of-the-art methods.
- In Chapter 3, we introduce Elastic Feature Consolidation (EFC), an exemplar-free class incremental learning method that regularizes drift in directions highly relevant to previous tasks and employs prototypes to reduce forgetting. This method exploits a tractable second-order approximation of feature drift based on an Empirical Feature Matrix (EFM). The EFM induces a pseudo-metric in feature space of the CNNs, which we use to regularize feature drift in important directions and to update Gaussian prototypes used in a novel asymmetric cross-entropy loss. We perform an extensive evaluation of this method on several datasets, both in Warm-Start scenarios, characterized by a large number of classes in the first task, and Cold-Start scenarios, where insufficient data is available in the first task to learn a high-quality backbone.

- In Chapter 4, we introduce the problem of incremental Social Network Identification in Multimedia Forensics. Various forensic methods based on DNNs address the problem of Social Network Identification from images. However, the constant emergence of new social platforms and updates to existing ones render these methods ineffective. Therefore, we propose the evaluation of two realistic incremental learning scenarios: Incremental Social Platform Classification and Incremental Social Version Classification. We evaluate a wide range of incremental learning methodologies in these scenarios to understand the effectiveness of incremental learning solutions in updating a DNN when new social platforms emerge or existing platforms are updated.

- Finally, in Chapter 5, we present the problem of vehicle viewpoint estimation using monocular cameras. Estimating the viewpoint of vehicles from monocular images is a crucial component for autonomous driving systems and fleet management applications. We introduce lightweight and effective convolutional neural networks that employ logit smoothing techniques and utilize a Siamese loss to enforce geometric constraints during training. By utilizing both image and vehicle coordinate information, these networks provide good performance in vehicle viewpoint prediction in terms of accuracy, memory footprint, and inference time. The last two properties, in particular, make these models well-suited for deployment on edge devices.

Chapter 2

Incremental Learning Background

In the recent years, Deep Neural Networks (DNNs) have achieved remarkable success, often matching or even surpassing human performance in a wide range of tasks. Notable examples include excelling in vision tasks like object recognition, object detection, and image segmentation, as well as demonstrating proficiency in sequence prediction and the generation of textual, voice, and multimedia content. This exceptional capability has firmly established DNNs as a cornerstone in artificial intelligence (AI) applications.

However, the effectiveness of DNNs comes with its own set of challenges, primarily centered around two key aspects: *scalability* and *adaptability*. The inherent nature of DNNs demands an extensive dataset for robust generalization to unseen data. While amassing large amounts of data is feasible in some cases, practical limitations such as limited time and financial constraints or stringent data privacy regulations may make it impossible to collect a massive quantity of data. Moreover, a significant challenge arises when the distribution of some or all the data changes, requiring updates to the model. The laborious, expensive, and wasteful process of reusing the entire dataset for these updates, an approach known as *joint incremental* training, highlights fundamental scalability and privacy issues in the field of deep learning.

Furthermore, if a DNN is initially trained for a specific task and is transitioned to a different task by naively adapting the existing network to the novel task, an approach called *fine-tuning*, the new model encounters significant difficulties. Unlike the human brain, which exhibits remarkable plasticity in learning new tasks while maintaining stability in remembering old tasks, DNNs suffer from the so-called problem of *catastrophic forgetting* [24]. This means that when a DNN is trained on one task and then trained on one or more new ones, it quickly forgets how to perform the initial task.

Incremental Learning (sometimes referred to as Continual or Lifelong Learning) is a machine learning paradigm that treats DNNs as systems with the capability to

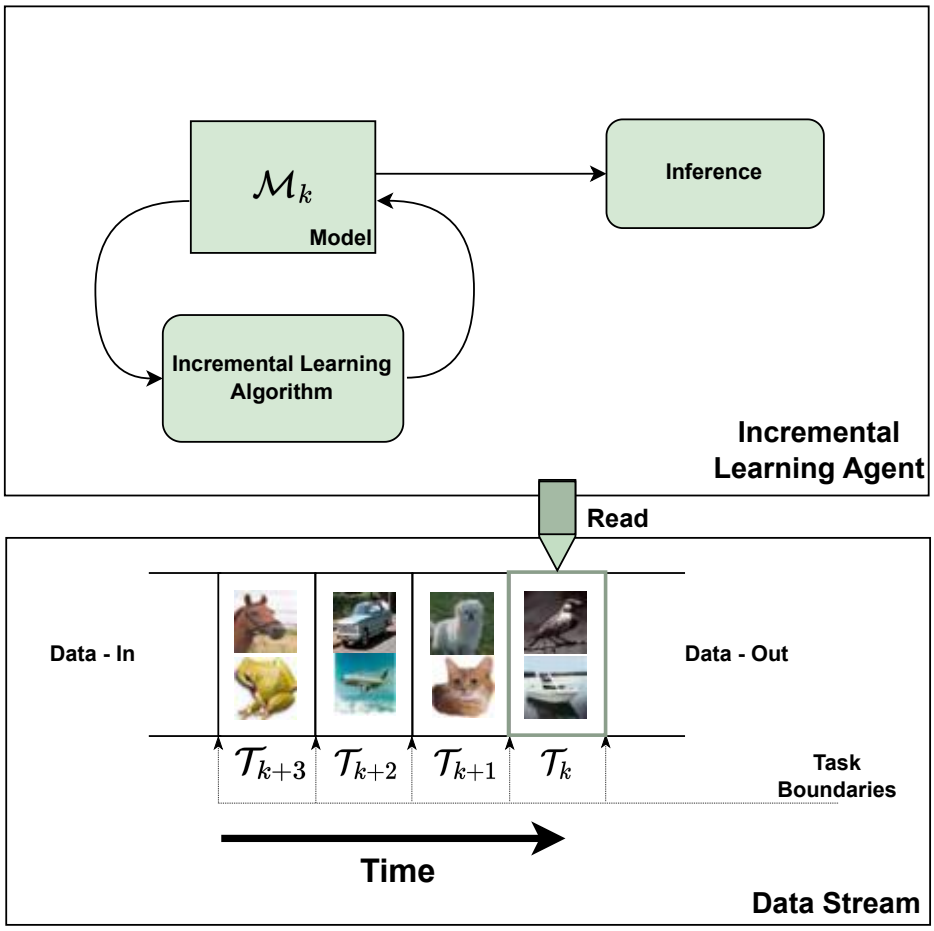


Figure 2.1: An Incremental Learning Agent. The agent processes a continuous data stream segmented into tasks, each representing a different classification problem, to train each model $\mathcal{M}_k$. Its primary goal is to acquire new knowledge at each task $\mathcal{T}_k$, while preserving previous learning to avoid catastrophic forgetting via an incremental learning algorithm. This is achieved without retaining all data from previous tasks. During the inference phase the agent uses the trained model to perform any of the learned tasks.

continuously acquire knowledge across a multitude tasks over time [29]. The core concept underlying incremental learning is that not all data is accessible from the initial stage, and that the network learns from an infinite data stream evolving over time. In this chapter, we describe incremental learning systems, the major causes of catastrophic forgetting and latest advances in this field.

2.1 Incremental Learning Systems

In incremental learning, the primary objective is for a learning agent (depicted in Figure 2.1) to continuously acquire new knowledge from a flowing *data stream* while

retaining previously acquired information, thus mitigating the issue of catastrophic forgetting. This process must be conducted efficiently, without the need to store and utilize all the data previously encountered.

More specifically, the learning agent interacts with the data stream, which is divided into discrete *tasks*. At each time step k, an *incremental learning algorithm* leverages the current task $\mathcal{T}_k$ to train a DNN model $\mathcal{M}_k$. The primary aim of this training is to learn the new task while preserving knowledge of previous ones (i.e. to effectively mitigate catastrophic forgetting). Once the training phase is complete, the agent can deploy the model for *inference*, enabling it to execute the tasks it has learned. Moreover, the agent transitions to acquire new knowledge by discarding old task data and proceeding to process the next task $\mathcal{T}_{k+1}$ in the data stream, subsequently re-initiating the incremental algorithm from the previously trained model.

In the following sections, we will provide a more detailed description of the components characterizing incremental agents, including the *data stream*, the *incremental learning algorithm*, and the *inference* phase.

2.1.1 Partitioning the Data Stream into Tasks

A widely accepted formalization within the incremental learning community involves the conceptual division of the evolving data stream into distinct tasks as time progresses. We can define a task as a tuple $\mathcal{T}_k = (\mathcal{X}_k, \mathcal{Y}_k)$, where $\mathcal{X}_k = (x_k^1, ..., x_k^{n_k})$ represent the data samples and $\mathcal{Y}_k = (y_k^1, ..., y_k^{n_k})$ represent their corresponding labels, with n_k the number of samples of task k. Incremental learning traditionally relies on labeled data, with most algorithms designed within a supervised framework. However, recent research [30, 31, 32] is exploring ways to adapt these algorithms for unsupervised settings, where data labels are not mandatory. For the purposes of this thesis, we maintain the assumption of working with labeled data and focus on the *supervised incremental learning* paradigm.

Regarding the content of a task, theoretically the samples $\mathcal{X}_k$ can encompass various data types, and the corresponding labels $\mathcal{Y}_k$ can be either discrete for classification tasks or continuous for regression tasks. However, it is important to note that the predominant body of research in the field of incremental learning primarily concentrates on image classification tasks, where the data samples are images, and the labels are discrete. Consequently, each task $\mathcal{T}_k$ is treated as a classification task, characterized by $\mathcal{C}_k$ classes. In this thesis, we also assume that the labels are discrete, thus focusing on *classification* tasks. Moreover, in the field of incremental learning, the most common assumption is that tasks are characterized by a disjoint set of classes, which means $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, $i \neq j$. In this thesis, we also adopt this assumption. However, it is worth mentioning that some studies explore incremental scenarios involving class overlap [33, 34].

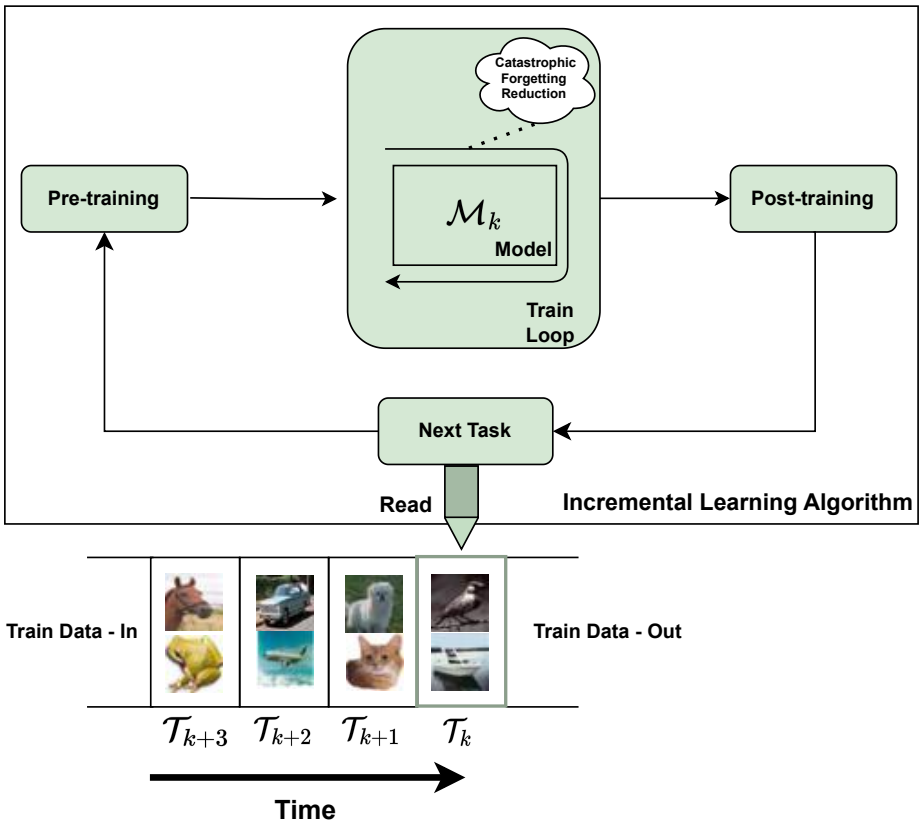


Figure 2.2: An Incremental Learning Algorithm. A model $\mathcal{M}_k$ is sequentially trained over a set of tasks, minimizing at the same time the catastrophic forgetting. Training involves three key phases: **Pre-training** for initial setup, **Train-loop** for iterative learning and minimizing forgetting, and **Post-training** for consolidating task-specific knowledge.

The concept of task division implies that, at each time step, the learning agent receives a batch of data, and these data segments are defined by what we refer to as *task boundaries*. In incremental learning, a common distinction is made between *training task boundaries* and *test task boundaries*, depending on whether the data stream is being used by the learning agent for training or inference. In the upcoming sections, we will explore how the incremental learning algorithm and the inference phase are impacted when the learning agent is aware of task boundaries during both training and inference, compared to situations where this awareness is lacking.

2.1.2 Incremental Learning Algorithms

The goal of an incremental learning algorithm is to sequentially train a Deep Neural Network, denoted as $\mathcal{M}_k$, on a set of K tasks, where ideally K can be infinite, while simultaneously mitigating catastrophic forgetting (as illustrated in Figure 2.2). The incremental model $\mathcal{M}_k$ at task k consists of a feature extraction backbone $f_k(\cdot;\theta_k)$,

shared across all tasks, with parameters θ_k updated during training, and a classifier.

In this section we describe the classifier and the feature extractor of the incremental model, then outline the primary operations of an incremental algorithm. Finally, we will discuss the impact of knowing the task boundaries during training on the learning algorithm.

Classifier. Common classifiers include the Softmax-based classifier and the Nearest Mean Classifier (NMC). We define $M^k = \sum_{j=1}^{k} |\mathcal{C}_j|$ as the cumulative count of classes encountered up to and including task $\mathcal{T}_k$. Here, $\mathcal{C}_j$ represents the set of new classes introduced in task j, and the summation accounts for the total number of unique classes seen over all tasks from 1 to k.

The *Softmax-based* classifier uses learned parameters $W_k \in \mathbb{R}^{n \times M^k}$, known as *head weights*. Here n is the feature space dimension. These parameters expand with each new task as additional classes are introduced. We use the notation $W_{k-1} \in \mathbb{R}^{n \times M^{k-1}}$ to represent the weights accumulated until the $k-1$ task, and $W_{k-1:k} \in \mathbb{R}^{n \times |\mathcal{C}_k|}$, the weights added for the last task. This classifier employs a Softmax activation function to output probabilities $p(y|x; \theta_k, W_k)$. The model's output for a sample x after task t is the composition of the feature extractor and classifier, expressed as:

$$\begin{aligned} \mathcal{M}_k(x; \theta_k, W_k) \equiv\ & p(y|x; \theta_k, W_k) \\ &= \text{softmax}(W_k^\top f_k(x; \theta_k)) \\ &= \text{softmax}([W_{k-1} \mid W_{k-1:k}]^\top f_k(x; \theta_k)) \\ &= \text{softmax}(z_k(x; \theta_k, W_k)) \in \mathbb{R}^{M^k}, \end{aligned} \tag{2.1}$$

where $z_k(x; \theta_k, W_k)$ represents the output *logits* of model $\mathcal{M}_k$ for sample x. The final prediction is obtained by computing the argmax of the Softmax output.

The *Nearest Mean Classifier* operates with *class means* or *prototypes*, denoted as $\mathcal{P}_{1:k} = \{p^1, .., p^{M^k}\}$, where each $p^i \in \mathbb{R}^n$. These prototypes are the average of feature vectors extracted from all examples of each class observed up to the current task. The model output for a sample x after task t is determined by computing a distance d between the feature vector $f_k(x; \theta_k)$ and all the accumulated prototypes:

$$\begin{aligned} \mathcal{M}_k(x; \theta_k) \equiv\ & d_{\mathcal{P}_{1:k}}(f_k(x; \theta_k)) \\ &= [d(f_k(x; \theta_k), p^1), \ldots, d(f_k(x; \theta_k), p^{M^k})] \in \mathbb{R}^{M^k}. \end{aligned} \tag{2.2}$$

Both the prototypes and the features are ℓ_2-normalized, which we do not write explicitly to avoid a cluttered notation. The final prediction is computed by considering the argmin of the distance between the normalized feature and prototypes.

From a notational perspective, for simplicity through the rest of this thesis, we will write $f_k(x) = f_k(x; \theta_k)$ and $z_k(x) = z_k(x; \theta_k, W_k)$, omitting the explicit mention of the dependence on the weights θ_k and W_k when it is not necessary for the discussion.

Feature Extractor. The prevalent practice for defining and initializing feature extractor f_k involves is to use ResNet-style architectures [14], with ResNet18 and ResNet32 being particularly popular choices. These architectures are favored primarily due to their fewer number of parameter relative to other models in the ResNet series. This characteristic enables thorough evaluation of various techniques and facilitate extended training across long sequence of tasks.

Regarding initialization, the standard practice for training the model involves starting from scratch. This means that the weights of the feature extractor are randomly initialized before the first task. This approach is preferred because random initialization, as opposed to initializing with pre-trained weights on a large dataset, ensures that the evaluation of incremental learning performance is free from the influence of previously acquired knowledge [35]. Consequently, this leads to a more fair and accurate assessment of the incremental algorithm's capabilities, particularly in terms of managing catastrophic forgetting and effectively learning new tasks.

However, recent developments of self-supervised learning [21] and the outstanding performance of *Vision Transformer* (ViT) [15] have significantly influenced incremental learning research. Unlike traditional CNNs, ViTs have demonstrated superior performance, albeit with a high demand for data and computational resources for effective training convergence. Consequently, recent studies are increasingly exploring the application of pre-trained ViT architectures in incremental learning [36, 37].

Training Phases. During the training process, at each distinct time step k, the agent reads the training task data and initializes the training of the model $\mathcal{M}_k$ (see Figure 2.2). This process is organized into three fundamental phases:

1. **Pre-training phase**. In this initial phase, the agent collects and stores essential information from the data necessary for initializing the model. This includes the number of classes $\mathcal{C}_k$ for a Softmax-based classifier and performs initial setup tasks, including parameter initialization and configuring network architecture.

2. **Train-loop phase**. During this phase, the model undergoes repeated training sessions employing specific techniques aimed at minimizing catastrophic forgetting. The choice of these techniques depends on the particular incremental learning strategy adopted. A more in-depth exploration of these methods will be detailed in Sections 2.3 and 2.4.

3. **Post-training phase**. In this phase, the agent focuses on preserving key information from the current task to mitigate catastrophic forgetting in subsequent phases. This preserved information, usually referred to as *meta-knowledge* [38], includes the weights of the model trained up to the current point, among other

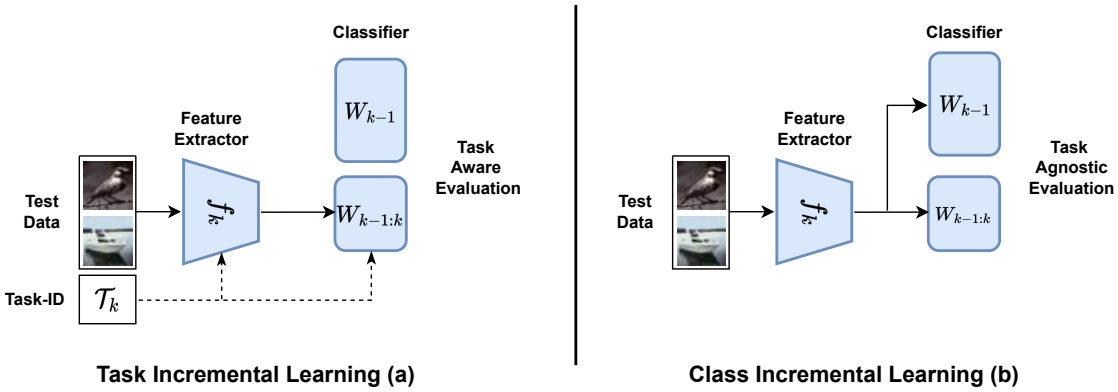


Figure 2.3: Task-incremental Learning versus Class-incremental Learning. In task-incremental learning, a scenario sometimes referred to as *task-aware*, the agent utilizes a task identifier associated with test data to select task-specific weights from the classifier and/or feature extractor for making predictions. Conversely, in class-incremental learning (sometimes called *task-agnostic*) the agent lacks information about test task boundaries and performs inference using all available weights.

pertinent data, depending on the learning system's design constraints. An example of this could be the retention of a subset of current task data known as **exemplars** (to be further discussed in Section 2.4). In scenarios utilizing a Nearest Mean Classifier, the model is updated with prototypes derived from the training data of the current task.

Following the training process, the agent becomes capable of performing all the tasks up to the current one and then proceeds to accumulate knowledge for the subsequent task in the sequence.

The transition between the post-training and the subsequent pre-training phase implies that the agent precisely understands when to shift from the current task to the next, i.e. the *knowledge of training task boundaries* during training. This understanding is critical for the agent to accumulate the meta-knowledge for the forthcoming task and to adjust the classifier to integrate new classes. In scenarios where this transition knowledge is externally provided, the learning framework is called *Task-based* incremental learning, as opposed to *Task-free* incremental learning. In the latter, the agent automatically determines the transition between tasks without requiring explicit knowledge of task boundaries.

2.1.3 Inference: Task-incremental versus Class-incremental Learning

At inference time, the learning agent utilizes the trained model to execute the learned tasks (as shown in Figure 2.3). Depending on the agent's *knowledge of the test task*

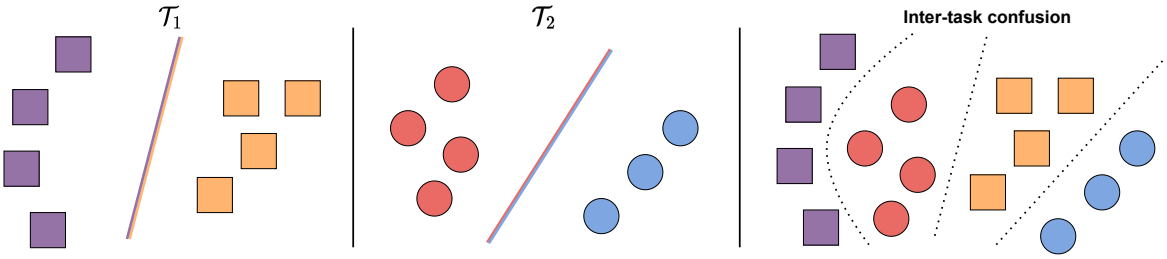


Figure 2.4: Inter-task confusion. In a scenario where the network is trained on two tasks, the decision boundaries (identified by color lines) are able to correctly classify the samples of the first and the second task (left and center figures). Unfortunately, the learned boundaries from the first two tasks are not able to differentiate between samples from different tasks (right figure).

boundaries, we can categorize two incremental learning scenarios:

- **Task-incremental Learning (task-IL):** In this scenario, the agent is aware of the test task boundaries and selects appropriate weights from the feature extractor or classifier head using a task identifier (Task-ID) for inference. When employing a Nearest Mean Classifier, the agent uses relevant task class means to classify the current samples. This evaluation setting is known as *Task Aware Evaluation*.

- **Class-incremental Learning (class-IL):** In contrast, here the agent does not have knowledge of the test task boundaries. During this type of inference, all parameters learned up to the current task are used. This evaluation method is referred to as *Task Agnostic Evaluation*.

Task-IL is generally less challenging compared to class-IL. Knowing the task-identifier allows for the training of specific weights for each task, and using these weights at the time of evaluation typically results in better performance and reduced forgetting. The objective in task-incremental learning is to achieve positive transfer across tasks, meaning that learning in one task enhances performance in another related task. On the other hand, the objective of class-IL is to learn a shared representation for objects that have never been seen together. This scenario is more challenging but also more realistic, as in many applications, such as Social Network Identification (discussed in Chapter 4), it is improbable to have the task-identifier available at inference time.

In the next section, we describe the main causes of catastrophic forgetting and we outline the main causes of forgetting for task-IL and class-IL scenarios.

2.1.4 Causes of Catastrophic Forgetting

Catastrophic forgetting is the phenomenon observed in neural networks where the learning new information leads to a loss of previously acquired knowledge [24, 25]. If a learning agent continuously trains the model on current task data, known as the *Fine-tuning* approach, it rapidly forgets previous tasks. A key goal of an effective incremental learning algorithm is to outperform Fine-tuning, while also maintaining lower storage and computational costs compared to Joint Incremental Training. Following Masana et al. [35], four main causes of catastrophic forgetting can be identified:

- **Weight Drift**. Training the network on a new task by minimizing a loss function can result in modifications to weights that are crucial for previous tasks. This results in a weight drift, which in turn induces forgetting of the previously learned information.

- **Activation Drift**. As weight drift occurs, it also affects the network activations. Maintaining stable activations across incremental steps can allows for weight adjustments without significantly altering layers activations.

- **Inter-task confusion**. Since incremental methods do not perform joint training on all the samples of all the classes, the learned decision boundaries are not optimally discriminative for different tasks (see Figure 2.4).

- **Task-recency bias**. When a network is trained on a sequence of task, the classifier output is not calibrated and the prediction on previous classes are biased toward to the new task ones.

In class-IL addressing all four causes of forgetting is crucial since the final classifier must effectively differentiate between classes from distinct tasks. On the other hand, task-IL requires the final classifier to discriminate between all the classes identified by the provided Task-ID. Consequently, the primary focus of task-IL is on mitigating weight and activation drift.

In the following section, we give an overview on the typical settings in incremental learning. Afterwards, we will delve into the various strategies employed to mitigate catastrophic forgetting.

2.2 Types of Incremental Learning

The objective of this section is to provide a comprehensive overview of the current incremental learning settings. These settings are classified based on the constraints of the learning system and the knowledge of task boundaries [39, 35, 40].

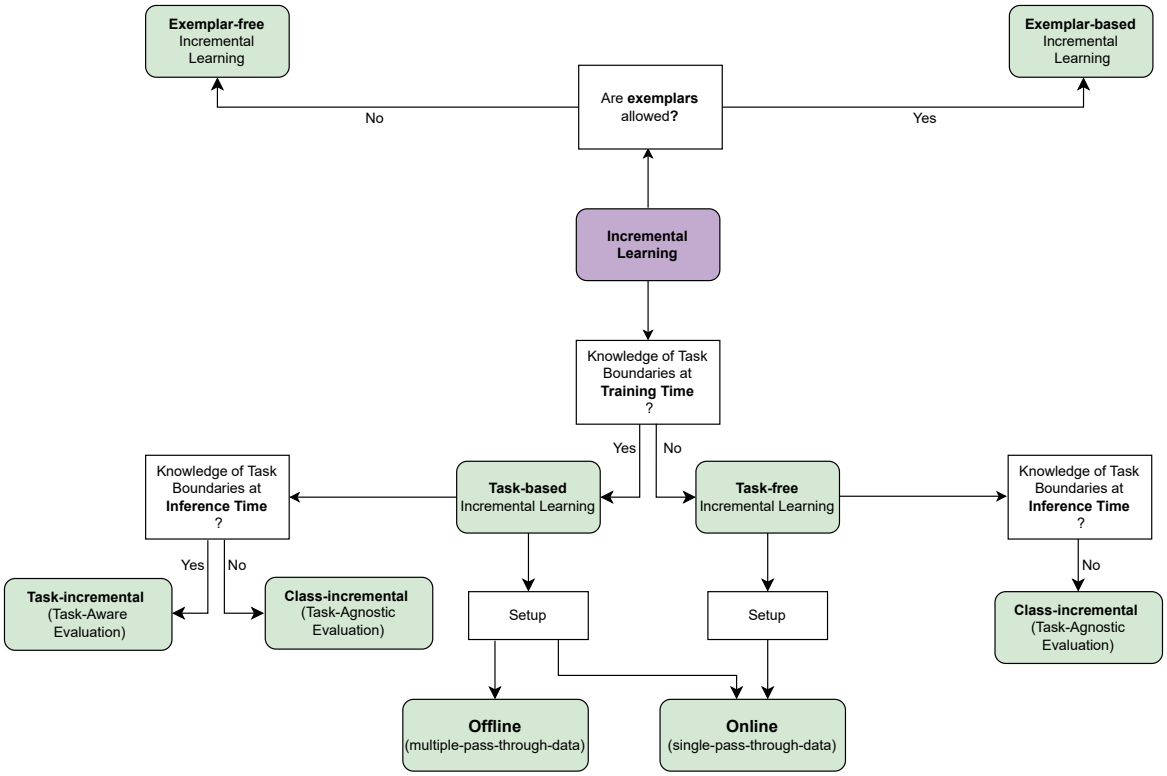


Figure 2.5: Diagram of incremental learning settings. In this diagram we distinguish between: Task-based and Task-free incremental learning methods according to the knowledge of task boundaries at training time; Class-incremental and Task-incremental according to the knowledge of task boundaries at inference time; and Offline and online systems according the number of training iterations performed on task data.

Offline versus Online. In terms of system constraints, we distinguish between *offline* and *online* incremental learning systems. *Offline* incremental learning allows for multiple training epochs on current task data during the train loop phase (see Figure 2.2). These systems are designed for low-frequency task transitions, conducting model inference only after extensive learning from the current task data. The primary goals are achieving high performance and minimizing forgetting after each task, with less focus on training duration and computational demands. In contrast, *online* incremental learning [41] assumes a single epoch per current task data. Ideally, this involves one forward and one backward pass in the train loop, though limited extra backward passes may be applied to enhance performance [42, 43]. This setting prioritizes frequent model updates (real-time) and optimal performance after a single exposure to data.

Task-based versus Task-free. Another distinction lies between *Task-based* and *Task-free* incremental learning, which depends on the awareness of training task boundaries [38]. *Task-based* incremental learning is further categorized into task-IL and

class-IL settings. This latter categorization is based on the knowledge of task boundaries at inference time, as discussed in Section 2.1.3. Conversely, *Task-free* incremental learning, commonly aligned with online scenarios due to frequent changes in task distribution, operates without awareness of training and test task boundaries and is thus categorized under class-IL settings. It is worth noting that the combination of the adjective *Task-based* with *offline* to describe an incremental learning method is relatively uncommon, as the latter term typically implies that the boundaries of the training task are already known. The term Task-based is more frequently used to differentiate from Task-free in the context of online continual learning. This thesis will also adopt this standard notation.

Exemplar-free versus Exemplar-based. All the aforementioned settings are further divided into *exemplar-based* and *exemplar-free* methods, depending on whether the learning system retains a subset of previous task data in a memory buffer (exemplars). Exemplar usage is common in incremental learning, as incorporating old task data into a network training on a current task helps control weight and activation drift. Additionally, exemplars help mitigate inter-task confusion and task-recency bias by training the network to categorize images from all observed classes. The primary challenge in exemplar-based methods lies in using minimal samples from previous classes and balancing them correctly with current task data.

Exemplar-based approaches are limited by the need for a memory budget to store exemplars. This memory requirement can become significant as memory needs increase with the number of tasks. Additionally, for certain applications, privacy regulations or concerns may not allow the storage of data samples or necessitate the deletion of data after a set period, rendering the storage of exemplars impractical. Lastly, employing exemplars means that additional forward and backward passes are required during training, leading to an increased training time. Exemplar-free methods, which avoid using exemplars throughout incremental learning steps, address these limitations, albeit often at the expense of a greater performance reduction.

In Figure 2.5 an overview of the above settings is provided. This thesis primarily explores *offline incremental learning* in both task-IL and class-IL settings. The following sections will delve into exemplar-based and exemplar-free methods, discussing approaches to address catastrophic forgetting.

2.3 Exemplar-free Incremental Learning

Exemplar-free incremental learning methods aim to mitigate catastrophic forgetting without storing data from previous tasks. Their main advantages include:

- **Privacy Preservation**: Suitable for scenarios where storing exemplars is not

feasible

- **Low Memory Requirement**: These methods are designed to use less memory compared to exemplar-based approaches, which is beneficial when memory consumption increases across tasks.
- **Reduced Training Time**: Unlike exemplar-based methods that require extra training passes due to exemplar rehearsal for each task, exemplar-free methods streamline the training process.

However, the primary challenge with exemplar-free methods is effectively controlling forgetting, as the network does not receive input from previous task data. This often leads to significant performance drops if compared to exemplar-based methods.

In this thesis, we focus primarily on two categories of exemplar-free methods. The first category is *regularization* methods, which constrain the network weights [44, 45, 46, 47, 48, 49] or activations [50, 51, 52] across the incremental learning steps. The second category is *prototype-based* methods, which employ prototypes to address task-recency bias and inter-task confusion in exemplar-free scenarios. Our focus on these methods is justified by two main reasons. Regularization methods, initially designed for Task-IL, are now extensively used in both prototype-based [53, 54] and exemplar-based methods [55, 56, 57, 58, 59] to control forgetting in the more challenging problem of class-IL. Prototype-based methods represent the latest advances in exemplar-free class-IL, achieving state-of-the-art results [60, 61, 62, 63, 64]. To provide a comprehensive understanding of exemplar-free incremental learning, this section will also examine other exemplar-free techniques, specifically mask-based[65, 66, 67, 68, 69] and gradient-based methods[70, 71, 72, 73, 74]. These have shown leading performance in Task-IL scenarios without relying on exemplars.

Before delving into the details of how exemplar-free methods work, we introduce some key notations. Focusing on softmax-based classifiers, introduced in Section 2.1, exemplar-free methods typically employ a cross-entropy loss to learn the features of the current task. We discuss two main implementations of the cross-entropy loss:

$$\mathcal{L}_t^{\text{ce}}(x,y)|_{\mathcal{C}_t} = \sum_{j=M^{t-1}+1}^{M^t} y^j \log\left(\frac{e^{z_t^j(x)}}{\sum_{k=M^{t-1}+1}^{M^t} e^{z_t^k(x)}}\right), \tag{2.3}$$

$$\mathcal{L}_t^{\text{ce}}(x,y)|_{\mathcal{C}_{1:t}} = \sum_{j=1}^{M^t} y^j \log\left(\frac{e^{z_t^j(x)}}{\sum_{k=1}^{M^t} e^{z_t^k(x)}}\right). \tag{2.4}$$

Here, $x \in \mathcal{X}_t$ and $y \in \mathcal{Y}_t$ denote a sample and its corresponding one-hot label for the current task t, M^t is the total number of classes up to and including task t and $z_t^j(x)$ represents the j-th logit output of the model $\mathcal{M}_t$ for the sample x.

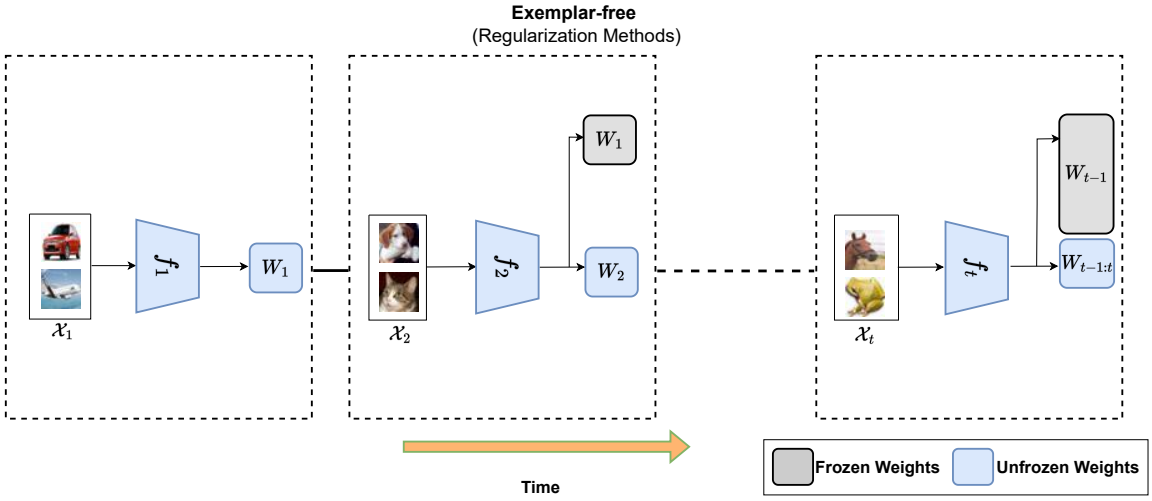


Figure 2.6: Exemplar-free Regularization Methods. The model receives only the current task data, and the weights of only the last classifier are updated. The training process combines a cross-entropy loss with a regularization loss.

The distinction between these two forms of cross-entropy loss lies in their application scope and impact on back-propagation. The first variant, $\mathcal{L}_t^{\text{ce}}(x,y)|_{\mathcal{C}_t}$ (Equation 2.3) is applied solely to the last task classifier, thereby affecting only its weights during back-propagation and keeping prior task classifiers' weights unchanged. In contrast, the second variant, $\mathcal{L}_t^{\text{ce}}(x,y)|_{\mathcal{C}_{1:t}}$ (Equation 2.4), is evaluated across all the task classifiers, which results in the training of all task classifier heads. As we will see in the subsequent sections, the choice of which variant to use depends on the specific method being employed.

In addition, in the remainder of this thesis, the arguments of the cross-entropy loss function $\mathcal{L}_t^{ce}(\mathbf{x},\mathbf{y})$ will be in bold when evaluated on a mini-batch of data. This implies that both Equations 2.3 and 2.4 are averaged over the components of a mini-batch. Adopting this notation will be particularly useful in discussion where it is important to emphasize the dependency on mini-batch processing.

2.3.1 Regularization Methods

To reduce catastrophic forgetting, these methods combine a cross entropy loss with a regularization loss $\mathcal{L}_t^{\text{reg}}$ to reduce weight or activation drift:

$$\mathcal{L}_t = \mathcal{L}_t^{\text{ce}}|_{\mathcal{C}_t} + \lambda_{reg}\mathcal{L}_t^{\text{reg}}, \tag{2.5}$$

where λ_{reg} controls the trade-off between stability in remembering the old task and the plasticity to learn the new one. It should be noted that the cross-entropy loss is evaluated only on the last classifier head, thus the previous task classifier weights are frozen (Figure 2.6). This strategy improves the performance of regularization

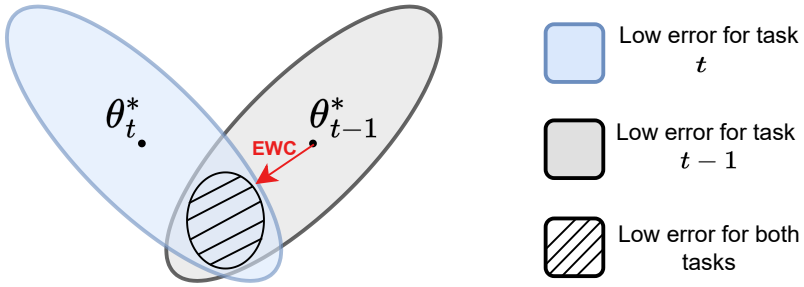


Figure 2.7: Elastic Weight Consolidation (EWC) [44]. The network weights are optimized using the Empirical Fisher Information Matrix, to achieve low error regions for both task t and task $t-1$.

methods (called as *label trick* by Zeno et al. [75]) and it is commonly employed in continual learning frameworks like FACIL [35].

We can distinguish two primary categories of regularization methods: *weight regularization*, which manages the drift in weights across tasks, and *functional* or *data regularization*, aimed at mitigating drift in activations. In the following sections, we offer a comprehensive overview of both weight and functional regularization methods. This includes in-depth descriptions of two foundational techniques: *Elastic Weight Consolidation* (EWC) [44] for weight regularization, and *Learning Without Forgetting* (LWF) [50] for functional regularization. We will conclude with a summary of the main limitations associated with these methods.

Weight Regularization. A naive approach to reduce weight drift across the incremental steps is to regularize the incremental training using the norm ℓ_2 to constraint weights update:

$$\mathcal{L}_t^{\ell_2} = ||\theta_t - \theta^*_{t-1}||_2^2, \tag{2.6}$$

where θ^*_{t-1} represent the (fixed) feature extractor weights after task $t-1$ and θ_t the weight of the feature extractor to optimize for the current task.

Kirkpatrick et al. [44] observed that adopting ℓ_2 regularizer (Equation 2.6), constraining all the weights equally, leaves little capacity for learning the current task. Hence, they proposed Elastic Weight Consolidation (EWC), which relaxes the ℓ_2 constraint with a quadratic constraint based on the *Empirical Fisher Information Matrix* (E-FIM):

$$F_i = \mathbb{E}_{x\sim\mathcal{X}_i}\left[\mathbb{E}_{y\sim p(y|x;\theta^*_i)}\left\{\left(\frac{\partial \log p(y|x;\theta^*_i)}{\partial \theta^*_i}\right)\left(\frac{\partial \log p(y|x;\theta^*_i)}{\partial \theta^*_i}\right)^T\right\}\right]. \tag{2.7}$$

The E-FIM provides an approximation of the curvature of the log-likelihood surface, denoted by $\log p(y|x;\theta^*_i)$, at its maximum likelihood estimate θ^*_i. This matrix

measures the sharpness of the likelihood function around its maximum. Parameters that exhibit high curvature, indicated by larger values of the E-FIM, are more important. This is because even little variations in these parameters can lead to substantial changes in the log-likelihood function and consequently on the model predictions.

Computing the full E-FIM for neural network weights is intractable, since it requires $\mathcal{O}(N^2)$ entries, where N is the number of network parameters, thus EWC proposes to approximate the E-FIM with a diagonal matrix reducing its entries to $\mathcal{O}(N)$. The regularization loss proposed by EWC is the following:

$$\mathcal{L}_t^{\text{E-FIM-orig}} = \lambda_{\text{E-FIM}} \sum_{i=1}^{t-1} (\theta_t - \theta_i^*)^T \text{diag}(F_i)(\theta_t - \theta_i^*). \tag{2.8}$$

Incorporating the Fisher Information matrix into regularization loss has the effect of forcing the network parameters to remain in a low-error region for previous models, penalizing updates to the most important weights. The rationale behind EWC is based on the *over-parametrization* property of neural networks. This implies that a solution for task t, θ_t^*, is likely to be found near the solution for the previous task θ_{t-1}^*. The mechanism of EWC is illustrated schematically in Figure 2.7.

The original formulation of EWC (Equation 2.8), necessitates preserving both a model and a diagonal Fisher information matrix for each task. This requirement results in increased computational and storage demands as the number of tasks grows. In contrast to this original approach, more recent implementations, such as EWC++ [45] and online-EWC [76], utilize only the most recent model θ_{t-1}^* for regularization purposes. Additionally, they employ a moving average of Fisher information matrices across tasks, enabling the storage of a single matrix upon completing each task. This approach, commonly integrated into incremental learning frameworks like Avalanche [77] and FACIL [35], is motivated by the fact that the latest model already incorporates constraints from the preceding task (for further details, refer to discussion in Huszr letters [78, 79]). Given these considerations, the final regularization loss is defined as follows:

$$\mathcal{L}_t^{\text{E-FIM}} = \lambda_{\text{E-FIM}} (\theta_t - \theta_{t-1}^*)^T diag(F_{t-1})(\theta_t - \theta_{t-1}^*), \tag{2.9}$$

where

$$F_{t-1} = \alpha \cdot \text{diag}(F_{t-1}) + (1 - \alpha) \cdot \text{diag}(F_{t-2}), \alpha \in \mathbb{R}. \tag{2.10}$$

In practice, the incremental learning algorithm based on EWC computes the current E-FIM during the post-training phase of each task. This computation involves a complete forward and backward pass using the current training data. The E-FIM is then updated, using Equation 2.10, integrating it with the previous E-FIM, which is initialized to zero for the first task. After this, the current model parameters are stored, and the algorithm proceeds to learn the next task using the following loss:

$$\mathcal{L}_t^{\text{EWC}} = \mathcal{L}_t^{\text{ce}}|_{\mathcal{C}_t} + \mathcal{L}_t^{\text{E-FIM}}. \tag{2.11}$$

Related Weight Regularization Methods. The concept of computing the importance score of each parameter for weight regularization has influenced various incremental learning methods. *Memory Aware Synapses* (MAS) [46] determine the importance score of parameters after each task by considering the magnitude of the gradients of the network's output relative to the weights. *Synaptic Intelligence* (SI) [47] calculates the importance score by assessing each weight(synapse) contribution to the final loss throughout the entire training trajectory. Recently, Benzing [80] demonstrated that both SI and MAS approximate the square root of the Fisher Information Matrix. Rwalk [45] suggests computing the Empirical Fisher Information Matrix (E-FIM) online during training, a method they refer to as EWC++. The overall approach combines the online E-FIM with a variant of the SI score to assess the importance of the weights.

However, the main limitation of E-FIM-based approaches is the assumption of a diagonal form of the Fisher matrix. This assumption is in practice unrealistic since it does not consider off-diagonal entries that represent how much the interaction between weights influences the log-likelihood. Therefore, some methods [49, 48] try to improve the diagonal E-FIM approximation, but at the cost of extra memory burden and computation. To overcome these limitations, other exemplar-free incremental learning methods rely on *functional regularization*, which scales better with the number of network parameters.

Functional Regularization. Incremental learning algorithms based on functional regularization mitigate the issue of activation drift. A noteworthy example of such techniques is *Learning Without Forgetting* (LwF) [50]. LwF tackles catastrophic forgetting by preserving the decision boundaries learned for the previous tasks. This is achieved without the need of exemplars from earlier tasks. During task t, the current training data are processed through both the current model $\mathcal{M}_t$ and the preceding model $\mathcal{M}_{t-1}$. Then, the *knowledge distillation* loss, introduced by Bucila et al. [81] and Hinton et al. [82], is employed to align the outputs for the previously learned classes in both models. The knowledge distillation loss is defined as follows:

$$\mathcal{L}_t^{\mathrm{KD}} = -\lambda_{\mathrm{KD}} \sum_{j=1}^{M^{t-1}} \left(\frac{e^{(z_{t-1}^j(x)/T)}}{\sum_{k=1}^{M^{t-1}} e^{(z_{t-1}^k(x)/T)}} \right) \log \left(\frac{e^{(z_{\mathrm{t}}^j(x)/T)}}{\sum_{k=1}^{M^{t-1}} e^{(z_t^k(x)/T)}} \right), \tag{2.12}$$

where $z_{\mathrm{t}}(x)$ and $z_{t-1}(x)$ represent the output logits from the current model and the previous (frozen) model for the sample $x \in \mathcal{X}_t$, respectively. M^{t-1} denotes the number of classes encountered up to and including task $t-1$, and $T \in \mathbb{R}$ is the standard temperature parameter used in knowledge distillation.

Operationally, LwF involves storing a trained model, denoted as $\mathcal{M}_{t-1}$, at the end of each task's training phase. During the training of the subsequent task, the model

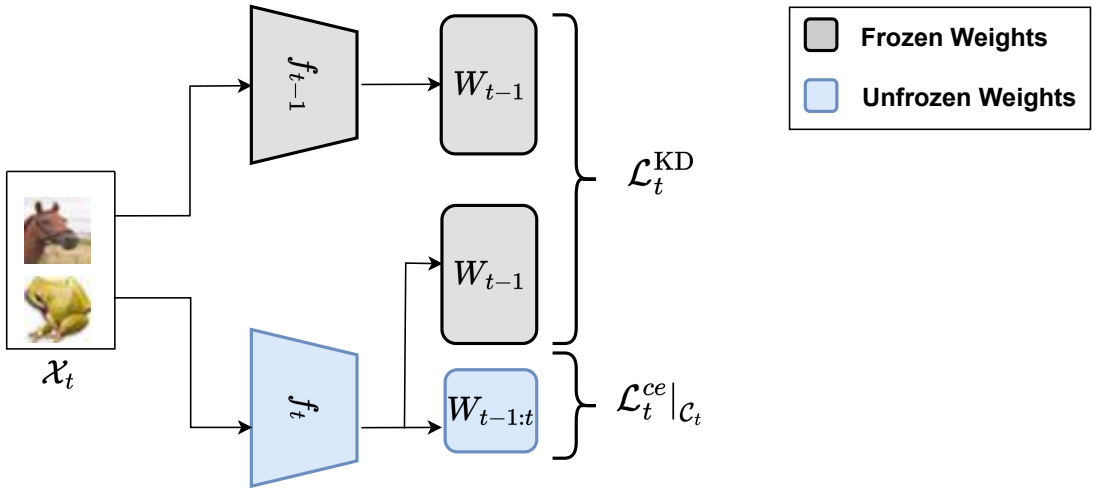


Figure 2.8: Learning without forgetting (LwF) [50]. The task data are fed into the current and previous models. The final loss is a combination of a cross-entropy and a knowledge distillation loss.

$\mathcal{M}_t$ is trained using a combination of knowledge distillation and cross-entropy losses:

$$\mathcal{L}_t^{\text{LwF}} = \mathcal{L}_t^{\text{ce}}|_{\mathcal{C}_t} + \mathcal{L}_t^{\text{KD}}. \tag{2.13}$$

A schematic visualization of LwF is provided in Figure 2.8.

Related Functional Regularization Methods. *Less-forgetting Learning* (LFL) [52] is closely related to LwF, with a focus on preserving the network's decision boundaries across tasks. Unlike LwF, LFL achieves this by introducing a regularization loss, based on *feature distillation*, aimed at align the features extracted by the current and previous models. *Learning Without Memorizing* (LwM) [51], on the other hand, proposes to constrain the current model's attention map, obtained via Grad-CAM [83], to match the attention map of the previous model. This approach focuses on ensuring that the attention regions of the network remain consistent across task transitions.

The main problem with functional regularization approaches is that the network is never trained to distinguish between novel and old classes. As a result, they achieve modest performance in terms of task-IL but low performance in class-IL due to the issues of task-recency bias and inter-task confusion. More recent exemplar-free techniques, referred to as *prototype-based* methods, try to overcome these issues by utilizing feature representations of past tasks.

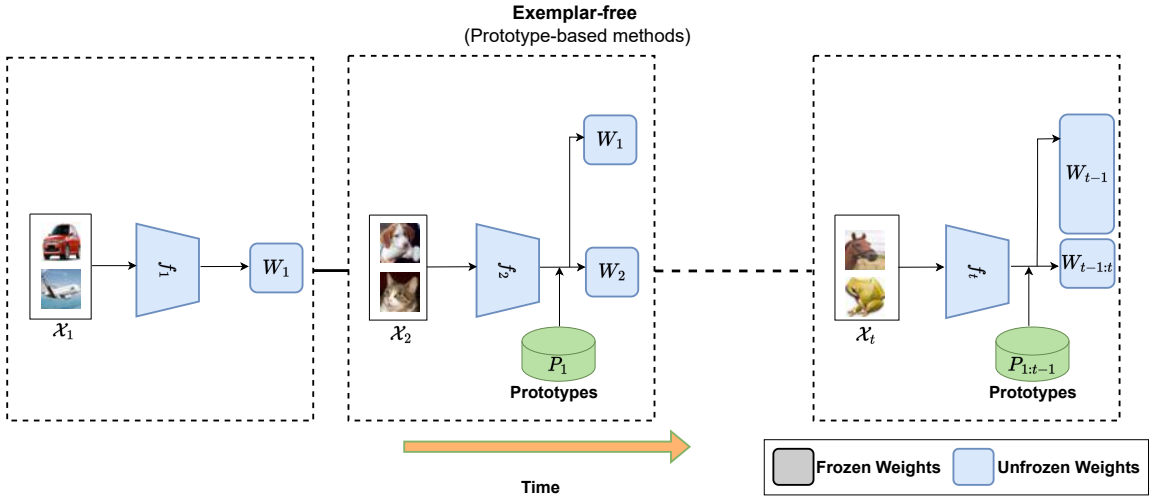


Figure 2.9: Exemplar-free Prototype-based Methods. The feature extractor receives only the current task data, and the final classifier is jointly trained on both prototypes and current task features to distinguish between old and new classes. The training process combines a cross-entropy loss evaluated on all the encountered classes and a regularization loss.

2.3.2 Prototype-based Methods

The primary challenge faced by regularization methods lies in their inability to mitigate task-recency bias and inter-task confusion problems in class-IL due to the fact that the final classifier is never trained to classify all the observed classes.

In order to overcome this limitation, recent methods have introduced *prototype rehearsal* as a method to leverage past-task deep features, enhancing the final classifier output and thereby mitigating task-recency bias. Generally speaking, prototypes are deep representations coming from the feature extractor from the previous task classes. A common solution is to define prototypes as class means, i.e., the average of CNN feature vectors for each class. Let us suppose we are at task t, and let us define the prototype for a class c in task $\mathcal{T}_k$, with $k = 1, \dots, t-1$, as:

$$p^c_{t-1} = \frac{1}{n_{c,k}} \sum_{x,y \in \mathcal{X}_k, \mathcal{Y}_k} \mathbb{1}[y = c] f_k(x), \tag{2.14}$$

where $\mathbb{1}[y = c]$ is the indicator function, equal to one if $y = c$ and zero otherwise, and $n_{c,k} = \sum_{y \in \mathcal{Y}_k} \mathbb{1}[y = c]$ represents the number of samples of class c within task $\mathcal{T}_k$. Moreover, let us define as $\mathcal{P}_{1:t-1}$ as the set of all prototypes accumulated up to and including task $t-1$.

Prototypes offer significant benefits, such as being computationally simple, privacy preserving and requiring less memory compared to exemplars. The size of each prototype depends on the feature space dimension, and their number scale linearly with the number of classes in the tasks.

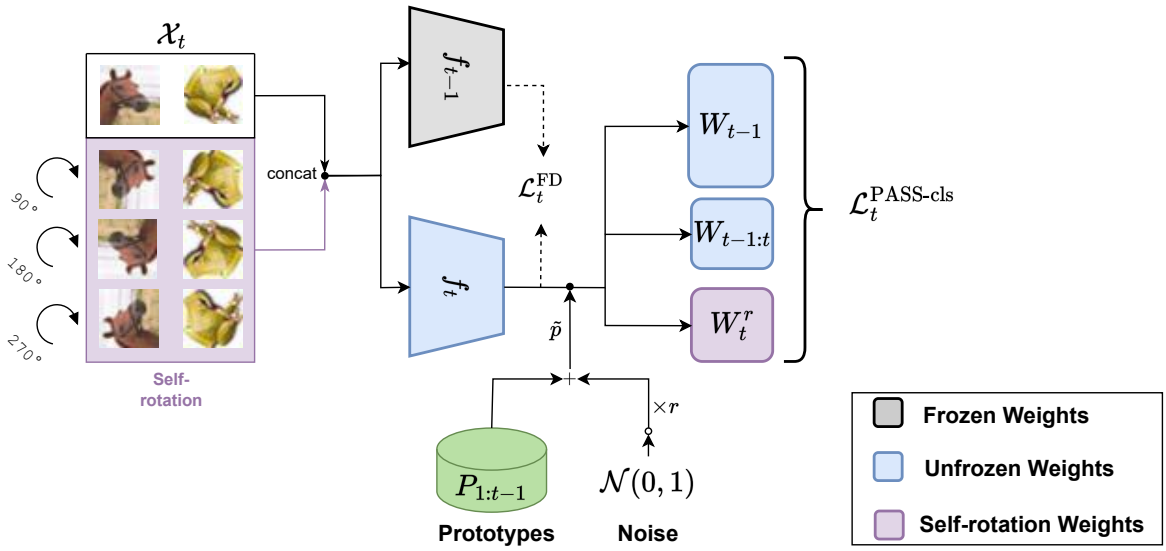


Figure 2.10: *Prototype Augmentation and Self-Supervision* (PASS) [53]. The feature extractor is trained using both the current task data and self-rotation. The final classifier is updated with weights W_t^r to accommodate the novel C_t^r classes introduced by self-rotation. The training loss is a combination of cross-entropy loss, evaluated on both prototype and current task data, as well as rotated data, along with feature distillation. Prototypes are perturbed using a Gaussian perturbation.

Prototype-based methods compute after each task the prototypes using Equation 2.14, store them and replay prototypes $\mathcal{P}_{1:t-1}$ in the next task. The training loss consists on a cross entropy loss evaluated on both prototypes and current feature representations and a regularization loss(see Figure 2.9).

There are several strategies, like *Gaussian perturbation* [53] and *oversampling* [60], that can be used to replay prototypes across the incremental learning steps. We introduce *Prototype Augmentation and Self-Supervision* (PASS) [53], a popular exemplar-free approach using prototypes for enhancing the performance of the final classifier in class-IL, and we provide an overview on the most recent techniques for leveraging prototypes.

Prototype Augmentation and Self-Supervision (PASS). Before prototype-based methods, most state-of-the-art exemplar-free incremental learning methods relied on generative models [84] to create pseudo-samples of previous classes, aiming to mitigate inter-task confusion and task-recency bias in class-IL. They primary issue with these generative methods was their complexity in training and frequent instability, which adversely affected overall performance.

Zhu et al. [53] proposed to leverage class prototypes (Equation 2.14), to balance the task classifier through the incremental learning steps. These prototypes, in contrast to generative models, do not require additional training for computation and

can be efficiently replayed across the training. To preserve the previously learned decision boundaries in subsequent tasks, PASS employs *Prototype Augmentation* which involves applying Gaussian perturbation to prototypes during the learning of new classes. Specifically, a prototype p^c_{t-1} is perturbed as follows:

$$\tilde{p}^c_{t-1} = p^c_{t-1} + e * r, \tag{2.15}$$

where e is drawn from a Gaussian distribution $\mathcal{N}(0,1)$ and r is a scaling factor controlling the uncertainty of the augmented prototype. The uncertainty and scale factor r are determined using the average variance of class feature representations. Notably, the authors of PASS empirically observed that the variance remains stable across the incremental learning steps, allowing to fix r after the initial task, eliminating the need to store a factor r per task. Further details on computing r are available in [53].

Another key contribution of PASS, is introducing self-supervision in incremental learning. It employs *self-rotation*, inspired by Lee et al. [85], to learn more generalizable and transferable features across tasks. In this method, each training sample in a task $\mathcal{T}_t$ is rotated by 90, 180 and 270 degrees, effectively creating three new classes for each original class. This expansion transforms the initial classification problem from $N_c^{t-1} + |\mathcal{C}_t|$ classes to $N_c^{t-1} + 4|\mathcal{C}_t|$ classes, where N_c^{t-1} is the number of classes encountered until task $t-1$:

$$x^r_t = \text{rotate}(x_t, \beta),\ \beta \in \{90, 180, 270\}. \tag{2.16}$$

Each rotated sample x^r_t receives a new label y^r_t and the final classifier is augmented accordingly to accommodate these new classes, collectively referred to as $\mathcal{C}^r_t$. The final classification loss of PASS is defined as:

$$\mathcal{L}^{\text{PASS-cls}}_t = \mathcal{L}^{ce}_t((\mathbf{x}_t, \mathbf{y}_t) \cup (\mathbf{x}^r_t, \mathbf{y}^r_t))|_{\mathcal{C}_{1:t} \cup \mathcal{C}^r_t} + \lambda_{\text{pr}}\, \mathcal{L}^{ce}_t(\tilde{\mathbf{p}}, \tilde{\mathbf{y}}_\mathbf{p})\big|_{\mathcal{C}_{1:t} \cup \mathcal{C}^r_t}, \tag{2.17}$$

where $(\mathbf{x}_t, \mathbf{y}_t) \sim \mathcal{X}_t, \mathcal{Y}_t$ represents a mini-batch of data and labels drawn from the current task; $(\mathbf{x}^r_t, \mathbf{y}^r_t)$ represent the rotations of the current mini-batch with the corresponding labels and $(\tilde{\mathbf{p}}, \tilde{\mathbf{y}}_\mathbf{p})$ is a mini-batch of prototypes and the corresponding labels, which are sampled from the prototype buffer $P_{1:t-1}$ and augmented as described in Equation 2.15. It should be noted that both the current task and the prototype batches are evaluated on all encountered classes $\mathcal{C}_{1:t}$ and new rotation classes $\mathcal{C}^r_t$. The term λ_{pr} is used to balance the loss on current task data and prototype loss.

Merely replying the prototypes is insufficient for achieving optimal performance in class-IL. As the learning progresses and new classes are introduced, the feature extractor is updated, causing the prototype representations to become outdated due to drifts in current feature representation. To counteract this issue, PASS proposes to employ *feature distillation* (FD), a regularization technique designed to regulate

and control the drift in the feature space:

$$\mathcal{L}_t^{\text{FD}} = \lambda_{\text{FD}} ||f_t(x) - f_{t-1}(x)||_2, \tag{2.18}$$

where f_t, f_{t-1} respectively represent the features extracted by the current and the previous (frozen) feature extractors from the current data. The overall training methodology, elaborated in Figure 2.10, involves training the network with augmented previous task prototypes and current task data, enhanced with rotations. The final training loss, combining feature distillation (Equation 2.18) and the classification loss defined in Equation 2.17, can be summarized as:

$$\mathcal{L}_t^{\text{PASS}} = \mathcal{L}_t^{\text{PASS-cls}} + \mathcal{L}_t^{\text{FD}}. \tag{2.19}$$

In the post-training phase of each task, the new prototypes are computed using Equation 2.14 and accumulated to be replayed in the next task.

Related Prototype-based Methods. SSRE [60] proposes replaying prototypes using *oversampling* in combination with feature distillation loss (Equation 2.18). Oversampling involves building a batch of prototypes by replicating the class means, which are computed using Equation 2.14, to achieve a fixed batch size. Furthermore, SSRE introduces a novel re-organization strategy to re-parameterize the feature extractor, aiming to retain the representation of older classes

Evanescent [54], aiming to overcome the issue of outdated prototypes as learning progresses, proposes the use of two generative models to model the drift of old class prototypes. These models are designed to learn two types of drifts, both contributing to the reduced effectiveness of old class prototypes. The first type of drift stems from the relationship between novel and past classes. The second type of drift arises when the feature extractor of the current model $\mathcal{M}_t$ is updated with data from the current task. Evanescent utilizes the drifts estimated by these generative models to infer up-to-date prototypes, termed *evanescent representations*, which are then replayed during the current training alongside the features of the current task. The training loss for model $\mathcal{M}_t$ combines cross-entropy loss, evaluated on both the current task and the evanescent representations, with feature distillation to ensure shareability of features across tasks.

Another method exploiting prototypes is *Semantic Drift Compensation* (SDC) [86]. SDC uses prototypes with a different objective. Instead of enhancing the softmax-based classifier with prototypes, it employs them for a Nearest Mean Classifier (as referenced in Equation 2.2). SDC utilizes an embedding network to learn the features of the current task via a *triplet loss*. It proposes several alternatives for regularizing the training loss, including feature distillation, elastic weight consolidation loss, and memory aware synapses losses. To correct feature space drift and update the prototypes, SDC suggests updating old class prototypes by performing a weighted average of the observed feature representation drift in the current task.

Given the challenges in estimating the drift of prototypes due to updates in the feature extractor, recent prototype-based methods have proposed freezing the feature extractor after the initial task and training only the classifier with both prototypes and current task features.

FeTrIL [62] follows this approach, freezing the feature extractor after learning the initial task. In subsequent tasks, it focuses on training only a linear classifier, based on a linear one-vs-rest SVM classifier [87], using both prototypes and the current task's features. Specifically, in each task, FeTrIL generates feature representations of old classes by applying a geometric translation to current task features, using the prototypes of old classes. These translated representations are referred to as *pseudo-features*. Finally, the linear classifier is trained using both these pseudo-features and the actual class representations.

Feature Covariance-Aware Metric (FeCAM) [63] also freezes the feature extractor and proposes using a Mahalanobis distance between the prototype and the features extracted from the backbone for the nearest mean classification. Notably, it employs the inverse of feature covariance of each class as the metric for the Mahalanobis distance. Similarly, *First Session Adaptation* (FSA) [64] suggests using prototypes for a Linear Discriminant Analysis (LDA) classifier [88], which is updated throughout the incremental learning steps.

The main drawback of prototype-based methods is that they require strong regularization of even the frozen feature extractor to prevent prototypes from becoming distant from the actual class means. This implies that such methods typically exhibit high stability in retaining information but low plasticity for learning new tasks. In Chapter 3, we will propose a method called *Elastic Feature Consolidation* [61], which aims to increase the plasticity of these methods.

2.3.3 Mask-based and Gradient Projection Methods

In this section, we provide a brief overview of exemplar-free incremental learning methods targeted for task-IL. In task-IL, unlike class-IL, the task identifier is known, allowing to select the corresponding classifier weights at inference time. We present two kinds of methods: *Mask-based* and *Gradient Projection* methods. The common idea behind these two types of methods is to reduce inter-task interference, which occurs when learning a new task interferes with the performance on previously learned tasks.

Mask-based methods aim to mask network weights or activations, dedicating a subset of network parameters or activations for each task. Piggyback [65] learns a binary mask per task while training the incremental model in an end-to-end fashion. PackNet [66], on the other hand, employs iterative pruning to learn a binary mask per task. *Hard Attention to the Task* (HAT) [67] uses attention masks to mask network weights, controlling gradient propagation in the network parameters. *Bit-*

Level Information Preserving (Bit) [68] employs weight quantization and keeps past information by bit-level freezing. SPACE [69] masks network activation per task, partitioning the representation into a *Core space*, representing the knowledge base from previous tasks, and a *Residual space*, which is used to learn the current task.

A more recent category of task-IL methods is the category of *Gradient Projection* methods. The main idea behind these methods is to retain memory of gradient information from past tasks, without using any exemplars, and update the model in orthogonal directions to minimize forgetting.

Adam-NSCL [70] proposes projecting gradient updates in the current task into an *approximate null space* for all previous tasks. This approximate null space is obtained after each task by computing the uncentered covariance of the training input features for each hidden layer. These covariance matrices are then decomposed via Singular Value Decomposition (SVD), and the eigenvectors corresponding to the smallest singular values are stored. The span of these eigenvectors constitutes the approximate null space for all previous tasks. In the subsequent task, the gradient updates generated by the Adam optimizer [89] are projected into this approximate null space.

Gradient Projection Memory (GPM) [71], inspired by SPACE's approach of splitting representations for old and new tasks, proposes partitioning the gradient space into two orthogonal subspaces: *Core Gradient Space* (CGS) and *Residual Gradient Space* (RGS). After completing each task, a random subset from the current training set is used to construct a representation matrix for the input of each hidden layer. These matrices are then decomposed using SVD, and the first top-k eigenvectors, which form the basis of the CGS, are stored. When training a new task, GPM projects the gradients orthogonally to the CGS. This projection has the objective to reduce interference with old tasks, thereby minimizing forgetting. The projection of the gradient update for a layer l with weights θ^l during task t can be easily obtained as:

$$\nabla_{\theta^l} \mathcal{L}_t^{ce} = \nabla_{\theta^l} \mathcal{L}_t^{ce} - M^l (M^l)^T (\nabla_{\theta^l} \mathcal{L}_t^{ce}), \tag{2.20}$$

where M^l is a matrix containing the bases of CGS for layer l stored at the end of the previous tasks.

Lin et al. [72] observes that purely performing gradient updates in orthogonal directions with respect to the previous task can be too restrictive for the optimization of the new task, as it does not take into account the correlation among tasks. Thus, they propose *Trust Region Gradient Projection* (TRGP), which modifies the gradient update of GPM (Equation 2.20) by adding a task-correlation based scaling of the weights of the old tasks.

Very recently, Zhao et al. [73] proposed a *space decoupling* (SD) strategy to decouple the space of feature representation into a pair of complementary spaces: a *stability space* and a *plasticity space*. They show that applying their decomposition to

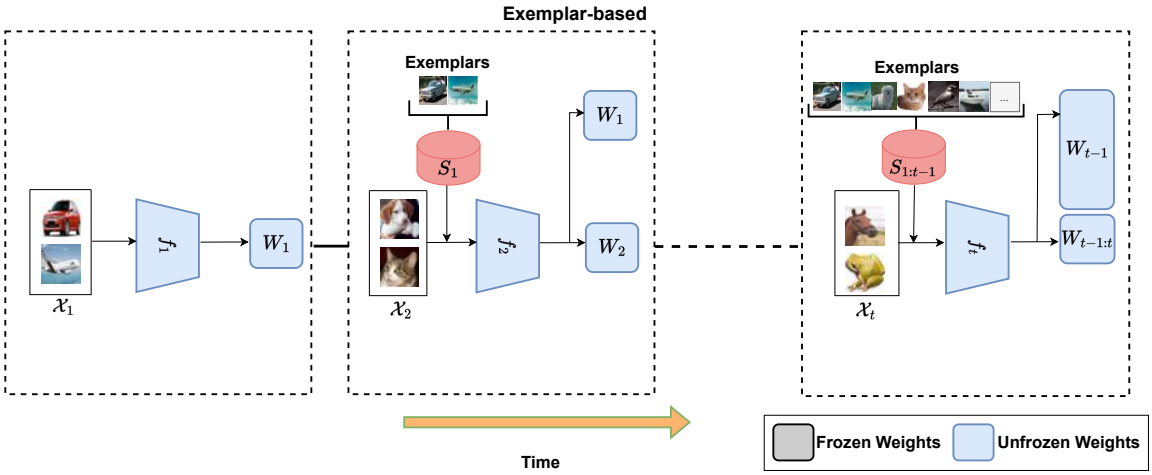


Figure 2.11: Exemplar-based Methods. The model is jointly train on exemplars from previous past classes and current task data. Exemplars can be stored in a *growing* memory buffer or in a *fixed* memory buffer. The training process combines a cross-entropy loss evaluated on all the encountered classes and a regularization loss.

the feature space before using gradient projection methods like GPM, Adam-NSCL, and TRGP achieves a better balance between stability and plasticity, thus improving overall performance.

Scaled Gradient Projection (SGP) [74] assigns importance scores to the bases of the Core Gradient Space, computed by GPM. Specifically, SGP utilizes the normalized eigenvalues derived from the SVD of the representation matrices. These normalized eigenvalues are employed in the orthogonal gradient update process for subsequent tasks (Equation 2.20), thereby enhancing the precision and effectiveness of the gradient updates by taking into account the importance of the bases of CGS.

2.4 Exemplar-based Incremental Learning

Exemplar-based methods utilize exemplars to mitigate catastrophic forgetting. These methods train the model using all samples of new classes, while simultaneously replaying a limited budget of old class samples in each incremental phase. As the final classifier is trained to recognize both old and new classes, their primary application lies in the class-IL setting.

The training loss for exemplar-based methods can be defined without loss of generality as:

$$\mathcal{L}_t = \mathcal{L}_t^{\text{ce}}|_{\mathcal{C}_{1:t}}\left((\mathbf{x}_t, \mathbf{y}_t) \cup (\mathbf{e}, \mathbf{y}_e)\right) + \lambda_{reg}\mathcal{L}_t^{\text{reg}}. \tag{2.21}$$

Here, $(\mathbf{x}_t, \mathbf{y}_t) \sim \mathcal{X}_t, \mathcal{Y}_t$ and $(\mathbf{e}, \mathbf{y}_e) \sim S_{1:t-1}$ represents a mini-batch of exemplars and their corresponding labels, sampled from the memory buffer $S_{1:t-1}$ which contains representative samples of old classes. Figure 2.11 provides an overview of

the exemplar-based methods' training process. It is important to note that $\mathcal{L}_{reg}$ is a regularization method designed to minimize activation or weight drift. As we will discuss in subsequent sections, knowledge distillation $\mathcal{L}_t^{KD}$, used by LwF, a regularization Exemplar-free approach (see Section 2.3.1), is commonly employed in many exemplar-based approaches.

Regarding the memory buffer, let's define its size as K. Two main types of memory buffers are commonly employed in the literature: a *growing* memory buffer, which expands in size with each new task as it accumulates a subset of data from each new task, and a *fixed* memory buffer, which maintains a constant size K by discarding excess exemplars as the number of tasks increases. Both strategies ensure a balanced distribution of classes in the memory buffer. The primary disadvantage of the first strategy is the linear increase in memory usage across tasks, while the main drawback of the second strategy is the reduction in the number of exemplars per class as new classes are introduced in successive incremental learning steps. Concerning the selection strategy for the memory buffer, a commonly used method is random selection, which involves uniformly and randomly picking exemplars from current classes. Alternative strategies include herding, as well as entropy- and distance-based selection methods (for further details refer to Masana et al. [35]).

Exemplar-based approaches are currently leading the way in performance for class-IL [59, 90, 91, 92]. However, they come with significant drawbacks, particularly in terms of the storage and computational resources required for managing the exemplar buffer and sampling process. Also, a notable concern is their lack of privacy preservation in contrast with exemplar-free methods.

The main goal of these methods is to balance exemplars and current task data in order to reduce task-recency bias and inter-task confusion, while at the the same time controlling activation drift across the steps. In the subsequent sections, we will provide a detailed discussion on how popular exemplar-based methods achieve these objectives, including specific algorithms and ways to integrate exemplar data into the learning process.

2.4.1 Incremental Classifier and Representation Learning (iCaRL)

ICaRL [55] is the first exemplar-based approach in incremental learning. It performs training by combining a cross-entropy loss with the use of knowledge distillation loss as a regularizer:

$$\mathcal{L}_t^{ICaRL} = \mathcal{L}_t^{\text{ce}}|_{\mathcal{C}_{1:t}}((\mathbf{x}_t, \mathbf{y}_t) \cup (\mathbf{e}, \mathbf{y}_e)) + \lambda_{kd}\mathcal{L}_t^{\text{KD}}. \tag{2.22}$$

Here, both the cross entropy loss and the knowledge distillation loss are evaluated on current task data and exemplars.

To reduce task-recency bias — the tendency of the network to classify all samples as belonging to the most recent class — ICaRL employs the Nearest Mean Classifier (as defined in Equation 2.2) during the inference phase. Specifically, after each task, it computes normalized prototypes for both the exemplars and the current task data that are used for the final classification prediction. Recomputing the prototypes for the exemplars ensures they are updated to reflect the feature space drift that occurs while learning a new task.

2.4.2 Bias Correction (BiC)

Wu et al. [56] conduct an in-depth analysis of the task-recency bias phenomenon, showing that it primarily arises from the last fully connected layer before the softmax-based classifier. They validate this hypothesis employing *linear probing* Davari et al. [93]. This method involves training an optimal linear classifier atop a frozen feature extractor following each incremental learning step, utilizing data from both prior and current tasks. Linear probing assesses the feature representations' quality throughout incremental learning and allows to quantify the extent of forgetting attributable to the bias in the last linear classifier. Their findings reveal that employing knowledge distillation and applying linear probing after each task significantly enhances model performance by mitigating task-recency bias.

Building on these findings, Wu et al. propose a *Bias Correction* (BiC) phase post each task to reduce the task-recency bias. Their training methodology comprises two phases for each task:

- **Phase 1 (Model Training).** The model $\mathcal{M}_t$ is trained using a combination of cross-entropy loss and knowledge distillation:

$$\mathcal{L}_t^{BiC} = \lambda_t \, \mathcal{L}_t^{\text{ce}}|_{\mathcal{C}_{1:t}} \left((\mathbf{x}_t, \mathbf{y}_t) \cup (\mathbf{e}, \mathbf{y}_e)\right) + (1 - \lambda_t)\mathcal{L}_t^{\text{KD}}, \tag{2.23}$$

  where $\lambda_t = M^{t-1}/M^t$, with M^t the number of classes accumulated up to and including task t, has the objective to balance the two terms. Both the cross-entropy loss and the knowledge distillation loss are evaluated on current task data and exemplars.

- **Phase 2 (Bias Correction).** This phase is applied after the training of the model $\mathcal{M}_t$ and involves training a linear model with two learnable parameters ($\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$), while maintaining the rest of the network fixed. The objective is to adjust the final classifier's output to counteract task-recency bias. Specifically, let us consider the output of the network with frozen weights, represented as follows:

$$\begin{aligned} z_t(x; \theta_t^{\text{frozen}}, W_t^{\text{frozen}}) &= [W_{t-1}^{\text{frozen}} \mid W_{t-1:t}^{\text{frozen}}]^\top f_t(x; \theta_t^{\text{frozen}}) \\ &= [o_{t-1}(x) | o_{t-1:t}(x)]. \end{aligned} \tag{2.24}$$

The BiC technique modifies the output z_t for the most recent classes by applying two learnable parameters, α and β. The adjusted output, $q_t(z_t; \alpha, \beta)$, is calculated as follows:

$$q_t(z_t; \alpha, \beta) = [o_{t-1}(x) | \alpha \cdot o_{t-1:t}(x) + \beta]. \tag{2.25}$$

Finally α and β are learned performing a training with a cross-entropy loss on a validation set of samples both belonging to the current and exemplar data sets.

Incremental Learning With Dual Memory (IL2M) [94], a method contemporary to BiC, also explores bias correction in exemplar-based incremental learning. The authors propose a technique to *rectify* the network output on previously learned classes after each new task, with the aim of reducing task-recency bias. This is achieved by utilizing both exemplars and new task data. In contrast to BiC, IL2M exclusively employs cross-entropy loss during training and does not incorporate knowledge distillation loss.

2.4.3 Learning a Unified Classifier via Rebalancing (LUCIR)

Hou et al. [95] observed that task-recency bias manifests concretely in the parameters of the last classifier layer, specifically in that the weights related to the over-represented classes of the current task become much larger than those for the under-represented old classes. Their method, named *Learning a Unified Classifier via Rebalancing* (LUCIR), mitigates this class unbalancing using four techniques: *cosine normalization*, *less-forget constraint*, *margin-ranking loss*, *class balance finetuning*.

Cosine normalization aims to eliminate the bias caused by the significant difference in the magnitude of the weights of the final classifier. It consists of ℓ_2 normalizing both the features and the weights of the classifier. Specifically, let $\overline{W}_t = (\overline{w}_t^1, \overline{w}_t^2, \ldots, \overline{w}_t^{N_c^t})$ be the normalized weights of the classifier at task t, where $\overline{w}_t^i$ represents the normalized class embeddings. The logits of the model at time t are defined as:

$$\overline{z}_t(x; \theta_t, \overline{W}_t) = (\langle \overline{w}_t^1, \overline{f}_t(x; \theta_t) \rangle, \langle \overline{w}_t^2, \overline{f}_t(x; \theta_t) \rangle, \ldots, \langle \overline{w}_t^{N_c^t}, \overline{f}_t(x; \theta_t) \rangle), \tag{2.26}$$

where $\overline{v} = v / \|v\|$ denotes the ℓ_2-normalized vector, and $\langle \overline{v}_1, \overline{v}_2 \rangle$ is the cosine similarity between two normalized vectors. Hence, the output of the model $\mathcal{M}_t$ is defined as:

$$\begin{aligned} \mathcal{M}_t(x; \theta_t, \overline{W}_t, \eta) \equiv\ & p(y | x; \theta_t, \overline{W}_t, \eta) \\ =\ & \text{softmax}(\eta \cdot \overline{z}_t(x; \theta_t, \overline{W}_t)), \end{aligned} \tag{2.27}$$

where η is a learnable scalar shared across tasks, introduced to control the peakiness of the softmax distribution since the range of $\langle \overline{v}_1, \overline{v}_2 \rangle$ is restricted to $[-1, 1]$.

To control the activation drift across tasks, they propose a variant of feature distillation aimed at controlling the drift of the normalized features. This regularizer, called the *less-forget constraint*, is defined as:

$$\mathcal{L}_t^{\text{less-forget}} = \lambda_t(1 - \langle \overline{f}_{t-1}(x), \overline{f}_t(x) \rangle), \tag{2.28}$$

where $\lambda_t = \lambda_{\text{base}} \sqrt{M^{t-1}/|\mathcal{C}_t|}$, M^{t-1} is the number of classes encountered up to and including task $t-1$, and λ_{base} is a fixed constant. The loss is zero when the angle α between the features extracted from the current and the previous model is 0° or 180° ($1 - \cos(\alpha) = 0 \iff \alpha = 2k\pi$). This loss encourages the orientation of features extracted by the current model to be similar to those by the original model.

Controlling feature drift via the less-forget constraint does not guarantee that old classes are well separated from the newest ones, since the new classes dominate the training set. To build a unified classifier for all classes and to force inter-class separation, LUCIR proposes a *margin-ranking loss*. Let e be an exemplar of class c, with $c = 1, \ldots, N_c^{t-1}$ and $\overline{w}_t^c$ its class embedding. The margin ranking loss is computed as:

$$\mathcal{L}_t^{\text{mr}} = \lambda_{mr} \sum_{k=1}^{K} \max(m - \langle \overline{w}_t^c, \overline{f}_t(e) \rangle + \langle \overline{w}_t^k, \overline{f}_t(e) \rangle, 0), \tag{2.29}$$

where m is a margin threshold, $\{\overline{w}_t^k\}_{k=1}^K$ are the top-K new class embeddings chosen as hard negatives for e. Specifically, these classes are selected during training as the new classes yielding the highest output score for sample e. This loss ensures that the new class embeddings, most similar to the exemplar, are maintained with significant separation, and it also guarantees alignment of the exemplar's feature representation with its corresponding class embedding.

The final loss of LUCIR is a combination of a cross-entropy loss and less-forget and margin-ranking losses:

$$\mathcal{L}_t^{\text{LUCIR}} = \mathcal{L}_t^{\text{ce}}|_{\mathcal{C}_{1:t}} ((\mathbf{x}_t, \mathbf{y}_t) \cup (\mathbf{e}, \mathbf{y}_e)) + \mathcal{L}_t^{\text{less-forget}} + \mathcal{L}_t^{\text{mr}}. \tag{2.30}$$

When all the training for the current task is completed, LUCIR performs a *class balance finetuning* using the stored exemplar samples. This involves fine-tuning the network with these exemplars, which slightly improves performance.

2.4.4 Latest Advances and State-of-the-art Methodologies

Some recent advanced exemplar-based methods, as we mentioned above, rely upon knowledge distillation. SS-IL [57] employs a separated softmax output layer in combination with *task-wise* knowledge distillation in order to reduce task-recency bias. *Menemonics* [58] proposes to combine cross-entropy loss and knowledge distillation during the training while at the same time they parametrized exemplars, named

mnemonics exemplars, that are updated during the training via backpropagation in a BiLevel Framework. *Dark Experience Replay* (DER++) [59] utilizes a cross-entropy loss on the current task data and additionally implements a distillation loss on exemplars that are sampled using reservoir sampling. This approach combines distillation and cross-entropy losses on the exemplars: the distillation loss is applied to the logits of the model to ensure consistency with previous outputs, and the cross-entropy loss is used to reinforce learning from previous tasks.

Inspired by LUCIR, other recent works explore cosine normalization for incremental learning. PODNet [96] proposes a modification to the LUCIR classifier. Instead of using a single embedding per class, it introduces a *Local Similarity Classifier* (LSC), which utilizes multiple embeddings per class to enforce a stronger inter-class separation. This approach effectively eliminates the need for the margin ranking loss. Additionally, PODNet regularize the feature representation drift by using a ℓ_2 constraints on the pooled intermediate representations of the network.

Adaptive Feature Consolidation (AFC) [90] adopts the LSC classifier of PODNet but changes the regularization loss. They propose to regularize the intermediate representation of the feature extractor by using an importance score computed at the end of each task. Specifically, after each task they compute the importance of the feature maps of each layer using both exemplars and current task data. In the next task, this importance score is used to constraint the feature maps drift across the task in a regularization loss.

Another group of exemplar-based approaches is the group of *dynamic expandable architectures*, which dynamically expand the network as the number of tasks increases. Each task is associated with dedicated sub-networks, and its weights are frozen when learning the new task. This category of methods is currently the state-of-the-art in exemplar-based class-IL in terms of performance, but with the limitation of increased storage and computational costs.

Dynamically Expandable Representation (DER) [91] trains a feature extractor per task using a cross-entropy loss in combination with an auxiliary classification loss on both exemplars and current task data. The objective of the auxiliary classification loss is to encourage the model to discriminate between old and new concepts. This auxiliary classifier takes as input both the old exemplar representations, to which a unique label is assigned, and the current task representations, to which the current task labels are assigned. To reduce the number of parameters used at inference time, DER introduces a channel-level mask to prune the network filters at the end of each task.

Huang et al. [97] observe that dynamically expanding the network does not take into account task correlations, leading to overall performance suffering from inter-task confusion and task-recency bias. To overcome these limitations, they propose TFCIL, a method that combines a *multi-level knowledge distillation* loss, which works

on both features and network outputs, with a *feature fusion module* to share the representations of task feature extractors. Furthermore, they introduce a *classifier rescoring* (CR) method to reduce task-recency bias caused by the unbalanced magnitude of the last weight classifier. Finally, a pruning method is applied to reduce the number of parameters used in inference

Recently, Zhou et al. [92] demonstrated that efficient and effective dynamic architectures can be designed by constructing specialized blocks for each task. Their approach, named MEMO, improves performance compared to DER, both in terms of memory efficiency and accuracy results.

Chapter 3

Elastic Feature Consolidation for Exemplar-free Incremental Learning

Deep neural networks achieve state-of-the-art performance on a broad range of visual recognition problems. However, the traditional supervised learning paradigm is limited in that it presumes all training data for all tasks is available in a single training session. The goal of Class-Incremental Learning (CIL) is to enable incremental integration of new classification tasks into already-trained models as they become available [35]. Class-incremental learning entails balancing model plasticity (to allow learning of new tasks) against model stability (to avoid catastrophic forgetting of previous tasks) [24]. Exemplar-based approaches retain a small set of samples from previous tasks which are replayed to avoid forgetting, while exemplar-free methods retain no samples from previous tasks. This latter category is of particular interest when retaining exemplars is problematic due to storage or privacy requirements.

Exemplar-free Class Incremental Learning (EFCIL) must mitigate catastrophic forgetting without recourse to stored samples from previous tasks. Approaches can be loosely grouped into those based on weight regularization and those based on functional regularization. *Elastic Weight Consolidation* (EWC) is an elegant and appealing approach based on a Laplace approximation of the previous-task posterior [44]. When training on a new task, an approximation of the Fisher Information Matrix is used to regularize parameter drift in directions of significant importance to previous tasks. This mitigates forgetting while maintaining more plasticity for learning the new task. EWC is a second-order approach, and as such requires ag-

Portions of this chapter were published in:

- [61] S. Magistri, T. Trinci, S.-C. Albin, J. V. de Weijer, and A. D. Bagdanov, "*Elastic feature consolidation for cold start exemplar-free incremental learning*," in The Twelfth International Conference on Learning Representations, 2024.

gressive approximations of the Fisher Information Matrix [48].

Functional regularization, especially through feature distillation, is a central component in recent state-of-the-art EFCIL approaches [53, 60, 86, 54]. Instead of regularizing weight drift, feature distillation regularizes drift in feature space to mitigate forgetting. Feature distillation is often combined with class prototypes, either learned [54] or based on class means [53], to perform pseudo-rehearsal of features from previous tasks which reduces the task-recency bias common in EFCIL [35]. Prototypes – differently than exemplars – offer a privacy-preserving way to mitigate forgetting. Feature distillation and prototype rehearsal reduce feature drift across tasks, but at the cost of model plasticity.

Existing EFCIL methods are predominantly evaluated in *Warm Start* scenarios in which the first task contains a larger number of classes than the rest, typically 50% or 40% of the entire dataset. For Warm Start scenarios* stability is more important than plasticity, since a good backbone can be already learned on the large first task. Current state-of-the-art approaches use strong backbone regularization [53] or indeed even *freeze* backbone after the large first task and focus on incrementally training the classifier [62].

In this chapter we consider EFCIL in the more challenging *Cold Start* scenario in which the first task is insufficiently large to learn a high-quality backbone and methods must be plastic and adapt their backbone to new tasks. EFCIL with Cold Starts faces two main challenges: an alternative to feature distillation is required, since the backbone must adapt to new data, and an exemplar-free mechanism is needed to adapt previous-task classifiers to the changing backbone.

We propose a novel EFCIL approach, which we call Elastic Feature Consolidation (EFC), that regularizes changes in directions in *feature space* most relevant for previously-learned tasks and allows more plasticity in other directions. In this chapter, we derive a pseudo-metric in feature space that is induced by a matrix we call the *Empirical Feature Matrix* (EFM). In contrast to the Fisher Information Matrix, the EFM can be easily stored and calculated since it does not depend on the number of model parameters, but only on feature space dimensionality. To address drift of the more plastic backbone, we additionally propose a *Prototype Replay Asymmetric Cross Entropy loss* (PR-ACE) that balances between new-task data and Gaussian prototypes during EFCIL. Finally, we propose an improved method to update the class prototypes which exploits the already-computed EFM. A visual overview of EFC and its components is given in Figure 3.1.

*We use the term Warm Start here, differently from [98], to distinguish continual learning starting from a large first task (Warm Start) from continual learning starting from a small first task (Cold Start).

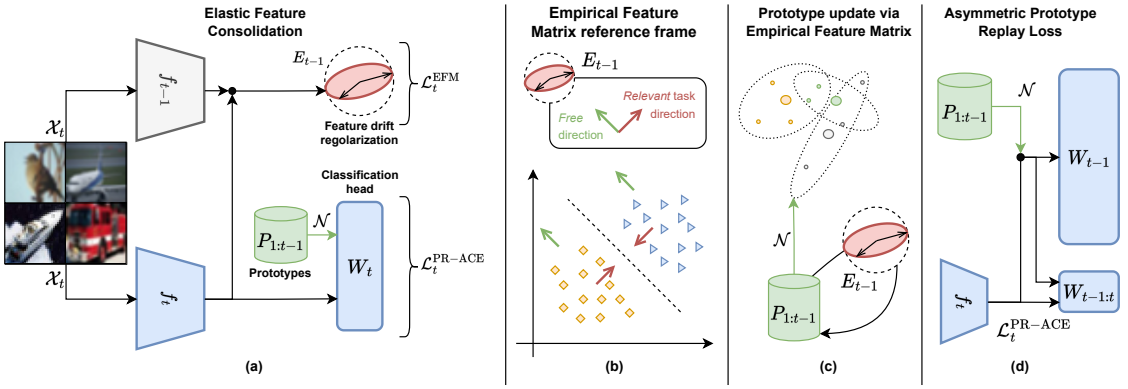


Figure 3.1: Elastic Feature Consolidation. (a) Architecture overview; (b) The Empirical Feature Matrix (EFM) measures how outputs vary with features and identifies important directions to mitigate forgetting (Section 3.2.3); (c) The EFM induces a pseudo-metric in feature space used to estimate prototype drift (Section 3.3.3); and (d) The Asymmetric Prototype Replay loss adapts previous task classifiers to the changing backbone by balancing new-task data and Gaussian prototypes (Section 3.3.2).

3.1 Related work

In this section, we briefly overview the most related works to our approach. We focus on **offline**, **class-incremental** learning problems. Class-incremental problems are distinguished from task-incremental problems in that no task information is available at test time. Offline incremental learning allows multiple passes on task datasets during training [55, 99, 35], while *online* incremental learning considers continuous data streams in which each sample is accessed just once during training [100, 101, 102].

Exemplar-based approaches rely on an exemplar memory of previous class samples that are replayed while learning new tasks. Hou et al. [95], Douillard et al. [96], Kang et al. [90] combine cosine normalization and distillation losses with exemplar-rehearsal to mitigate forgetting. Another category of exemplar-based approaches focuses on ResNet style architectures and expand sub-network structures across the incremental steps. AAnet [103] adds two residual blocks to mask layers and balance stability and plasticity. DER [91] train a single backbone for each incremental task, and more recently, MEMO [92] share generalized residual blocks and only extend specialized blocks for new tasks to improve performance and efficiency. The primary drawbacks of exemplar-based approaches are their lack of privacy preservation and potential high computational and storage costs, particularly when a growing memory buffer is utilized.

Exemplar-free approaches are less common in class-incremental learning. Early works computed importance scores for all weights and used them to perform weight

regularization [44, 48, 49]. Other works like Li and Hoiem [50], Jung et al. [52] use functional regularization methods, like knowledge and feature distillation, constraining network activations to match those of a previously-trained network. Unlike exemplar-based methods [91], most EFCIL approaches consider Warm Start scenarios in which the first incremental task is much larger that the others so employing strong regularization losses, reducing plasticity after the first task has less impact on the final performance. In particular, recent EFCIL approaches use feature distillation [104, 86], or even freeze the feature extractor after the first task [62, 105, 64]

On top of this decrease in plasticity, only using functional regularization is not enough to learn new boundaries between classes from previous tasks and classes from new ones. For this reason, regularization is often combined with *prototype rehearsal* [53, 60, 106]. Prototypes are feature space statistics (typically class means) used to reinforce the boundaries of previous-task classes without the need to preserve exemplars. Zhu et al. [53, 104] use prototype augmentation and self-supervised learning to learn transferable features for future tasks. Toldo and Ozay [54] learns and updates prototype representations during incremental learning. Zhu et al. [60] combine prototypes with a strategy to re-organize network structure to transfer invariant knowledge across the tasks. Petit et al. [62] fixes the feature extractor after the training the large first task and generates pseudo-samples to train a linear model discriminating all seen classes.

We propose an EFCIL approach that uses functional regularization and prototype rehearsal, but in contrast with most state-of-the-art methods, we also report results in *Cold Start* incremental learning scenarios that do not start with a large first task.

3.2 Regularization via the Empirical Feature Matrix

As a remark in this section, we provide the general EFCIL framework, introduced in Chapter 2, Section 2.1.2. Then, we derive a novel pseudo-metric in feature space, induced by a positive semi-definite matrix we call the *Empirical Feature Matrix* (EFM), used as a regularizer to control feature drift during class-incremental learning (see Figure 3.1b).

3.2.1 Exemplar-free Class-incremental Learning (EFCIL)

During class-incremental learning a model $\mathcal{M}$ is sequentially trained on K tasks, each characterized by a disjoint set of classes $\{\mathcal{C}_t\}_{t=1}^{K}$, where each $\mathcal{C}_t$ is the set of labels associated to the task t. We denote the incremental dataset as $\mathcal{D} = \{\mathcal{X}_t, \mathcal{Y}_t\}_{t=1}^{K}$, where $\mathcal{X}_t$ and $\mathcal{Y}_t$ are, respectively, the set of samples and the set of labels for task t. The incremental model $\mathcal{M}_t$ at task t consists of a feature extraction backbone $f_t(\cdot;\theta_t)$

shared across all tasks and whose parameters θ_t are updated during training, and a classifier $W_t \in \mathbb{R}^{n \times \sum_{j=1}^{t} |\mathcal{C}_j|}$ which grows with each new task. The model output at task t is the composition of feature extractor and classifier:

$$\mathcal{M}_t(x; \theta_t, W_t) \equiv p(y|x; \theta_t, W_t) = \text{softmax}(W_t^\top f_t(x; \theta_t)). \tag{3.1}$$

Directly training $\mathcal{M}_t$ (i.e. fine-tuning) with a cross-entropy loss at each task t results in backbone drift because at task t we see no examples from previous-task classes. This progressively invalidates the previous-task classifiers and results in forgetting. In addition to the cross-entropy loss, many state-of-the-art EFCIL approaches use a regularization loss $\mathcal{L}_t^{\text{reg}}$ to reduce backbone drift, and a prototype loss $\mathcal{L}_t^{\text{pr}}$ to adapt previous-task classifiers to new features.

3.2.2 Weight and Functional Regularization

Kirkpatrick et al. [44] showed that using ℓ_2 regularization to constrain weight drift reduces forgetting but does not leave enough plasticity for learning of new tasks. Hence, they proposed *Elastic Weight Consolidation* (EWC), which relaxes the ℓ_2 constraint with a quadratic constraint based on a diagonal approximation of the Empirical Fisher Information Matrix (E-FIM):

$$F_t = \mathbb{E}_{x \sim \mathcal{X}_t} \left[\mathbb{E}_{y \sim p(y|x;\theta_t^*)} \left\{ \left(\frac{\partial \log p(y)}{\partial \theta_t^*} \right) \left(\frac{\partial \log p(y)}{\partial \theta_t^*} \right)^T \right\} \right], \tag{3.2}$$

where θ_t^* is the model trained after task t. The E-FIM induces a pseudo-metric in parameter space [107] which, when to regularize learning, encourages parameters to remain in a low-error region for previous-task models:

$$\mathcal{L}_t^{\text{E-FIM}} = \lambda_{\text{E-FIM}} (\theta_t - \theta_{t-1}^*)^T F_{t-1} (\theta_t - \theta_{t-1}^*). \tag{3.3}$$

The main limitation of approaches of this type is need for approximations of the E-FIM (e.g. a diagonal assumption). This makes the computation of the E-FIM tractable, but in practice is unrealistic since it does not consider off-diagonal entries that represent influences of interactions between weights on the log-likelihood. To overcome these limitations, more recent EFCIL approaches [53, 60] rely on functional regularization which scale better in the number of parameters. In particular a common regularization loss proposed is feature distillation (FD):

$$\mathcal{L}_t^{\text{FD}} = \sum_{x \in \mathcal{X}_t} ||f_t(x) - f_{t-1}(x)||_2. \tag{3.4}$$

This type of isotropic regularizer using the ℓ_2 distance is too harsh a constraint for learning new tasks, and we propose instead to use a pseudo-metric in *feature space*.

This pseudo-metric is induced by a matrix which we call the *Empirical Feature Matrix* (EFM) that allows us to constrain directions in *feature space* most important for previous tasks, while allowing more plasticity in other directions when learning new tasks.

3.2.3 The Empirical Feature Matrix

Our goal is to regularize feature drift, but we do not want to do so isotropically as in feature distillation. Instead, we draw inspiration from how the E-FIM identifies important directions in parameter space and take a similar approach in feature space.

The local and empirical feature matrices. Let $f_t(x)$ be the feature vector extracted from input $x \in \mathcal{X}_t$ from task t and $p(y) = p(y|f_t(x); W_t)$ the discrete probability distribution over the set of classes $\{\mathcal{C}_j\}_{j=1}^{t}$. We define the *local feature matrix* as:

$$E_{f_t(x)} = \mathbb{E}_{y\sim p(y)}\left[\left(\frac{\partial \log p(y)}{\partial f_t(x)}\right)\left(\frac{\partial \log p(y)}{\partial f_t(x)}\right)^{\top}\right]. \tag{3.5}$$

The Empirical Feature Matrix (EFM) associated with task t is obtained by taking the expected value of Equation 3.5 over the entire dataset at task t:

$$E_t = \mathbb{E}_{x\sim\mathcal{X}_t}[E_{f_t(x)}]. \tag{3.6}$$

Both $E_{f_t(x)}$ and E_t are symmetric, as they are weighted sums of symmetric matrices, and positive semi-definite. Computing $E_{f_t(x)}$ requires a complete forward pass and a backward pass up to the feature embedding, however the next results provide an analytic formulation for $E_{f_t(x)}$ that avoids computing gradients.

Proposition. *Let $f_t(x)$ be the feature vector extracted from input $x \in \mathcal{X}_t$ from task t and $p(y) = p(y|f_t(x); W_t)$ the discrete probability distribution over the set of classes $\{\mathcal{C}_j\}_{j=1}^{t}$. Then, the following holds:*

$$E_{f_t(x)} = \mathbb{E}_{y\sim p(y)}[W_t(I_m - P)_y(W_t(I_m - P)_y)^{\top}], \tag{3.7}$$

where $m = \sum_{j=1}^{t} |\mathcal{C}_j|$, I_m the identity matrix of dimension $m \times m$, and P is a matrix built from the row of softmax outputs associated with $f_t(x)$ replicated m times.

Proof. To simplify the notation, we are going to prove the equation without explicitly specifying the task, as it is unnecessary for the derivation:

$$E_{f(x)} = \mathbb{E}_{y\sim p(y)}\left[\left(\frac{\partial \log p(y)}{\partial f(x)}\right)\left(\frac{\partial \log p(y)}{\partial f(x)}\right)^{\top}\right]. \tag{3.8}$$

We begin by computing the Jacobian matrix with respect to the feature space of the output function $g : \mathbb{R}^m \to \mathbb{R}^m$ which is the log-likelihood of the model on logits z:

$$g(z) = \log(\text{softmax}(z)) = [\log(\sigma_1(z)), \ldots, \log(\sigma_m(z))]^{\top},$$

where

$$\sigma_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}}.$$

The partial derivatives of g with respect to each input z_i are:

$$\frac{\partial \log(\sigma_i(z))}{\partial z_j} = \begin{cases} 1 - \sigma_i(z) & \text{if } i = j, \\ -\sigma_j(z) & \text{otherwise.} \end{cases} \tag{3.9}$$

Thus, the Jacobian matrix of g is:

$$J(z) = \begin{bmatrix} 1-\sigma_1(z) & -\sigma_2(z) & \dots & -\sigma_m(z) \\ -\sigma_1(z) & 1-\sigma_2(z) & & \vdots \\ \vdots & & \ddots & \\ -\sigma_1(z) & \dots & & 1-\sigma_m(z) \end{bmatrix} \tag{3.10}$$

$$= I_m - \mathbb{1}_m \cdot \begin{bmatrix} \sigma_1(z) \\ \vdots \\ \sigma_m(z) \end{bmatrix}^\top = I_m - P, \tag{3.11}$$

where $I_m \in \mathbb{R}^{m \times m}$ is the identity matrix and $\mathbb{1}_m$ represent the vector with all entries equal to 1.

Recalling Equation 3.8, we are interested in computing the derivatives of the log-probability vector with respect to the feature vector:

$$\frac{\partial \log p(y_1, \dots, y_m | f; W)}{\partial f} = \frac{\partial (Wf)}{\partial f} \frac{\partial (\log(\text{softmax}(z)))}{\partial z} \tag{3.12}$$

$$= WJ(z), \tag{3.13}$$

where $z = Wf$ for W the classifier weight matrix mapping features f to logits z. Combining this with Equation 3.11 we have:

$$\mathbb{E}_{y \sim p(y)} \left[\left(\frac{\partial \log p(y)}{\partial f} \right) \left(\frac{\partial \log p(y)}{\partial f} \right)^\top \right] = \mathbb{E}_{y \sim p(y)} [W(I_m - P)_y (W(I_m - P)_y)^\top], \tag{3.14}$$

where $p(y) = p(y|f(x); W)$, P is the matrix containing the probability vector associated with f in each row, and $(I_m - P)_y$ is the column vector containing the y_{th} row of the Jacobian matrix. Computing E_t using Equation 3.14 requires only a single forward pass of all data through the network, whereas a naive implementation requires an additional backward pass up to the feature embedding layer. □

Geometrical Properties and Loss with Empirical Feature Matrix. To gain additional insight into what the measures, we highlight some (information) geometrical

aspects of E_t. The Empirical Fisher Information Matrix provides information about the information geometry of parameter space. In particular, it can be obtained as the second derivative of the KL-divergence for small weight perturbations [108, 109]. Similarly, the Empirical Feature Matrix provides information about the information geometry of *feature* space. Since $E_{f_t(x)}$ is positive semi-definite, we can interpret it as a pseudo-metric in feature space and use it to understand which perturbations of features most affect the predicted probability:

$$\mathrm{KL}(p(y \mid f_t(x;\theta_t)+\delta;W_t) \;||\; p(y \mid f_t(x;\theta_t);W_t)) = \delta^T E_{f_t(x)}\delta + \mathcal{O}(||\delta||^3). \quad (3.15)$$

We now have all the ingredients for our regularization loss:

$$\mathcal{L}_t^{\mathrm{EFM}} = \mathop{\mathbb{E}}_{x\in\mathcal{X}_t}\Big[(f_t(x)-f_{t\text{-}1}(x))^T(\lambda_{\mathrm{EFM}}E_{t\text{-}1}+\eta I)(f_t(x)-f_{t\text{-}1}(x))\Big], \quad (3.16)$$

where $f_t(x)$ and $f_{t-1}(x) \in \mathbb{R}^n$ are the features of sample x extracted from the current model $\mathcal{M}_t$ and the model $\mathcal{M}_{t-1}$ trained on the previous task, respectively. In Equation 3.16 we employ a damping term $\eta \in \mathbb{R}$ to constrain features to stay in a region where the quadratic second-order approximation of the KL-divergence of the log-likelihood in Equation 3.15 remains valid [110]. However, to ensure that our regularizer does not degenerate into feature distillation, we must ensure that $\lambda_{\mathrm{EFM}}\mu_i > \eta$, where μ_i are the non-zero eigenvalues of the EFM. Additional analysis of these constraints and the spectrum of the EFM will be given in Section 3.4.4. In the next section we give empirical evidence of the effect of this elastic regularization.

The Regularizing Effect of the Empirical Feature Matrix. By definition, the directions of the eigenvectors of the EFM associated with strictly positive eigenvalues are those in which perturbations of the features have the most significant impact on predictions. To empirically verify this behavior, we perturb features in these *principal* directions and measure the resulting variation in probability outputs. Let $f_t = f_t(x) \in \mathbb{R}^n$ denote feature vector extracted from x and E_t the EFM computed after training on task t. We define the perturbation vector $\varepsilon_{1:k} \in \mathbb{R}^n$ such that each of the first k entries is sampled from a Gaussian distribution $\mathcal{N}(0,\sigma)$, where k is the number of strictly positive eigenvalues of the spectral decomposition of E_t. The remaining n-k entries are set to zero. Then, we compute the perturbed features vector $\tilde{f}_t$ as follows:

$$\tilde{f}_t = f + U_t^\top \varepsilon_{1:k}, \quad (3.17)$$

where U is the matrix whose columns are the eigenvectors of E_t.

In Figure 3.2 (left) we show that perturbing the feature space along the principal directions in this way, after training on the first task, results in a substantial variation in the probabilities for each class. Conversely, applying the complementary perturbation $\varepsilon_{k:n}$, i.e., setting zero on the first k directions and applying Gaussian noise on the remaining n-k directions, has no impact on the output. In Figure 3.2 (right) we

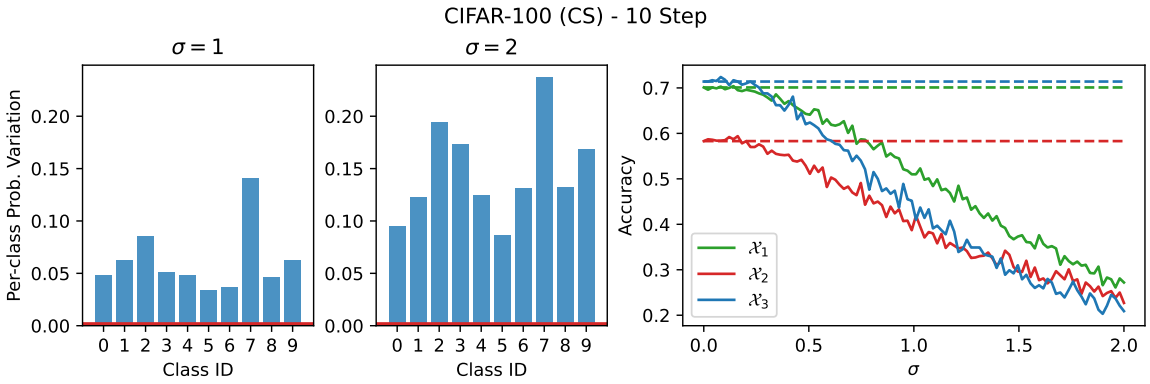


Figure 3.2: The regularizing effects of E_t. **Left**: Perturbing features in principal directions of E_1 results in significant changes in classifier outputs (in blue), while perturbations in non-principal directions leave the outputs unchanged (in red). **Right**: If we continue incremental learning up through task 3 and perturb features from all three tasks in the principal (solid lines) and non-principal (dashed lines) directions of E_3, we see that E_3 captures *all* important directions in feature space up through task 3. We conducted these experiments in the CIFAR-100 Cold Start (CS) 10 step scenario, distributing the classes into 10 equal splits across tasks. This analysis is also applicable to other dataset configurations. For more details on dataset settings see Section 3.4.1.

see that, after training three tasks, perturbing in the principal directions of E_3 significantly degrades performance across all tasks, while perturbations in non-principal directions continue to have no affect on the accuracy. This shows that E_3 captures the important feature directions for all previous tasks and that regularizing drift in these directions should mitigate forgetting.

3.3 Prototype Rehearsal for Elastic Feature Consolidation

In this section we first describe prototype rehearsal and then describe our proposed Asymmetric Prototype Rehearsal strategy (see Figure 3.1d). We also show how the Empirical Feature Matrix E_t defined above can be used to guide classifier drift compensation (see Figure 3.1c).

3.3.1 Prototype Rehearsal in EFCIL

EFCIL suffers from the fact that the final classifier is never jointly trained on the all seen classes. At each task t the classifier W_{t-1} is extended to include $|\mathcal{C}_t|$ new out-

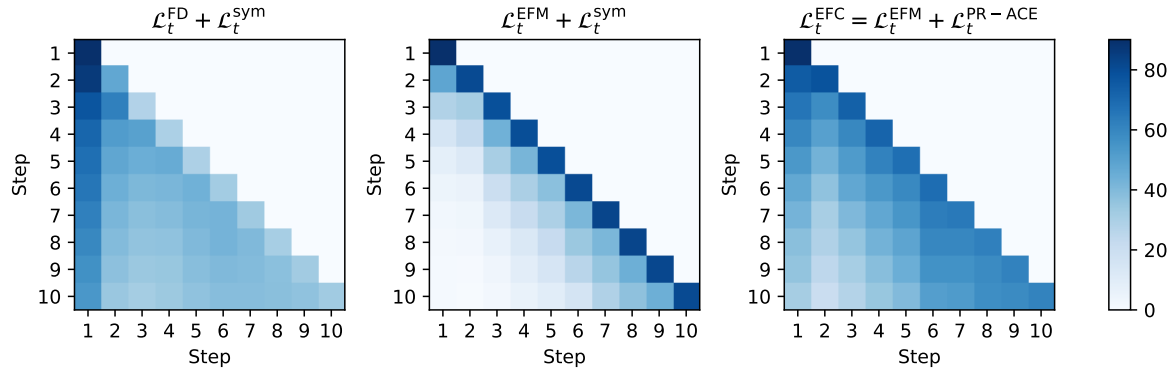


Figure 3.3: Accuracy after each incremental step on the Cold Start CIFAR-100 10-step scenario. Feature distillation with symmetric prototype loss (left) reduces forgetting at the cost of plasticity and new tasks are not learned. EFM regularization with symmetric loss (middle) increases plasticity at the cost of stability and previous tasks are forgotten. EFM regularization with asymmetric PR-ACE loss (right), balances current task data with prototypes and achieves better plasticity/stability trade-off.

puts. Thus, if classifier W_t is only trained on samples from classes $\mathcal{C}_t$ it will not be able to discriminate classes from previous tasks. This phenomenon is called *inter-task confusion* [35]. Moreover, even if a small number of exemplars is replayed, in an *exemplar-based* scenario, the task classifier outputs are not calibrated and the predictions on previous classes are biased toward to the newest task ones (*task-recency bias*).

A common *exemplar-free* solution is to replay only *prototypes* of previous classes [60, 53, 104, 62, 54]. At the end of task $t-1$, the prototypes p_{t-1}^c are computed as the class means of deep features and the set of prototypes $\mathcal{P}_{1:t-1}$ is stored.

During training of the next task the prototypes are perturbed and fed to the classification heads. Let $(\tilde{p}, y_{\tilde{p}}) \sim \mathcal{P}_{1:t-1}$ be a batch of perturbed prototypes and $(x_t, y_t) \sim (\mathcal{X}_t, \mathcal{Y}_t)$ a batch of current task data. The final classification loss, which we call the *symmetric* loss to distinguish it from the *asymmetric* loss we propose below, consists of two cross entropy losses evaluated on the all classification heads and can be written as:

$$\mathcal{L}_t^{\text{sym}} = \mathcal{L}_t^{\text{ce}}(\mathbf{x}_t, \mathbf{y}_t)\big|_{\mathcal{C}_{1:t}} + \lambda_{\text{pr}}\, \mathcal{L}_t^{\text{ce}}(\tilde{\mathbf{p}}, \mathbf{y}_{\tilde{\mathbf{p}}})\big|_{\mathcal{C}_{1:t}}, \tag{3.18}$$

where $\mathcal{L}_t^{\text{ce}}(\mathbf{x_t}, \mathbf{y_t})\big|_{\mathcal{C}_{1:t}}$ represents the cross-entropy loss evaluated on a mini-batch of data and computed over all encountered classes $\mathcal{C}_{1:t}$.

In our work, we use Gaussian prototypes to enhance the feature class means, as suggested by Zhu et al. [104]. To be more specific, we create augmented prototypes by sampling from a Gaussian distribution $\mathcal{N}(p_{t-1}^c, \Sigma^c)$. Here, Σ^c represents the class covariance matrix of feature representations.

3.3.2 Asymmetric Prototype Rehearsal

The symmetric prototype loss $\mathcal{L}_t^{\text{sym}}$ from Equation 3.18 above is able to balance old task classifiers with new ones only when feature space drift is small, for instance when using feature distillation (see Figure 3.3 (left)). But with the greater plasticity introduced by the EFM, the symmetric loss fails to effectively adapt old task classifiers to the changing backbone (see Figure 3.3 (middle)).

To address this, we propose an *Asymmetric Prototype Replay loss* (PR-ACE) to balance current task data and prototypes:

$$\mathcal{L}_t^{\text{PR-ACE}} = \mathcal{L}_t^{\text{ce}}(\mathbf{x}_t, \mathbf{y}_t)\big|_{\mathcal{C}_t} + \mathcal{L}_t^{\text{ce}}((\tilde{\mathbf{p}}, \mathbf{y}_{\tilde{\mathbf{p}}}) \cup (\hat{\mathbf{x}}_t, \hat{\mathbf{y}}_t))\big|_{\mathcal{C}_{1:t}}, \tag{3.19}$$

where $(\mathbf{x}_t, \mathbf{y}_t), (\hat{\mathbf{x}}_t, \hat{\mathbf{y}}_t) \sim (\mathcal{X}_t, \mathcal{Y}_t)$ and $\tilde{\mathbf{p}} \sim \mathcal{P}_{1:t-1}$ with $\mathbf{y}_{\tilde{\mathbf{p}}}$ the associated labels. $\mathcal{L}_t^{\text{ce}}(\mathbf{x}, \mathbf{y})\big|_{\mathcal{C}^t}$ is the cross entropy loss restricted to classes in $\mathcal{C}^t$.

The first term relies on the current task data, processed through the backbone and the current task classification head. It aligns the initially random head with the feature representation learned during previous tasks and aids learning of task-specific features. The second term calibrates the classification heads. It combines a batch of current task data with a batch of prototypes. Prototypes and current task data are uniformly sampled from all classes and used to train all task classifiers; consequently a larger proportion of prototypes is used compared to the current task data. In Figure 3.3 (right) we see that the addition of the PR-ACE loss achieves just this balance between stability and plasticity.

PR-ACE takes inspiration from the asymmetric loss proposed by Caccia et al. [111] for *online, exemplar-based* class incremental learning. In their work, the goal was to learn inter-task features with a loss term based on both exemplars and current task data passed through the current backbone. In contrast, for prototype rehearsal only *current task data* may be passed through the backbone.

3.3.3 Prototype Drift Compensation via Empirical Feature Matrix

Using fixed prototypes has the drawback that they eventually deviate from the previous class representations. Yu et al. [86] showed that the drift in *embedding networks* can be estimated using current task data. Their method, named SDC, works on the premise that we can effectively estimate the drift of mean feature embeddings from previous-task classes by looking at feature drift in the most closely related classes from the current task. After each task, they update the prior normalized class means via a weighted average of the observed feature representation drift in the current task. Subsequently these updated class means are used at test-time to perform the nearest class-mean classification.

Our aim is to update prototypes p_{t-1}^c used in for rehearsal in PR-ACE (Equation 3.19) across the entire training phase. As proposed by SDC, after each task we

update the prototypes:

$$\hat{p}^c_{t-1} = p^c_{t-1} + \frac{\sum_{x_i \sim \mathcal{X}_t} w_i \delta_i^{t-1}}{\sum_{x_i \sim \mathcal{X}_t} w_i}, \quad \delta_i^{t-1} = f_t(x_i) - f_{t-1}(x_i), \tag{3.20}$$

where δ_i^{t-1} is the feature drift of samples x_i between task $t-1$ and t.

We define the weights w_i using the EFM (Figure 3.1c). In Section 3.2.3 we showed that the EFM induces a pseudo-metric in feature space, providing a second-order approximation of the KL-divergence due to feature drift. We extend the idea of feature drift estimation to softmax-based classifiers by weighting the overall drift of our class prototypes. Writing $p^c = p^c_{t-1}$ and $f^i_{t\text{-}1} = f_{t-1}(x_i)$ to simplify notation, our weights are defined as:

$$\begin{aligned} w_i &= \exp\left(-\frac{(f^i_{t\text{-}1} - p^c) E_{t\text{-}1} (f^i_{t\text{-}1} - p^c)^\top}{2\sigma^2}\right) \\ &\approx \exp\left(-\frac{\mathrm{KL}(p(y|f^i_{t\text{-}1}; W_{t\text{-}1}) \,||\, p(y|p^c; W_{t\text{-}1}))}{2\sigma^2}\right), \end{aligned} \tag{3.21}$$

where $E_{t\text{-}1}$ is the EFM after training task $t-1$. Equation 3.21 assigns higher weights to the samples more closely aligned to prototypes in terms of probability distribution. Specifically, higher weights are assigned to samples whose softmax prediction matches that of the prototypes, indicating a strong similarity for the classifier.

Let $d^c_i = (f^i_{t\text{-}1} - p^c) E_{t\text{-}1} (f^i_{t\text{-}1} - p^c)^T$ be the distance according the EFM between current task feature f_i extracted from the model trained at task $t-1$ and the prototype p^c. However, since d^c_i is not bounded (similar to the KL-Divergence), the exponential function becomes unstable across incremental learning steps. To address this problem, we normalize all the distances d^c_i be fall in the interval $[0, 1]$ before applying the Gaussian kernel. In all our experiments, we use a fixed $\sigma = 0.2$ for the Gaussian kernel.

3.3.4 Elastic Feature Consolidation with Asymmetric Prototype Rehearsal

Figure 3.1 summarizes our approach, which is a combination of the EFM and PR-ACE losses:

$$\mathcal{L}^{\text{EFC}}_t = \mathcal{L}^{\text{EFM}}_t + \mathcal{L}^{\text{PR-ACE}}_t, \tag{3.22}$$

where $\mathcal{L}^{\text{PR-ACE}}_t$ is the asymmetric cross entropy loss (Equation 3.19) and $\mathcal{L}^{\text{EFM}}_t$ is the Empirical Feature Matrix loss from (Equation 3.16). After each task, our prototypes are updated using (Equation 3.20) equipped with the EFM (Equation 3.21) and new prototypes with the corresponding class covariances are computed. Finally, we compute the EFM E_t using the current task data for use when learning

Algorithm 1: Elastic Feature Consolidation

Data: $\mathcal{M}_1, E_1, \mathcal{P}_1$
for $t = 2, \dots T$ **do**
  `/* Iterate over epochs */`
  for *each optimization step* **do**
    sample $x_t \sim \mathcal{X}_t$, $\hat{x}_t \sim \mathcal{X}_t$
    sample gaussian proto $\tilde{p} \sim \mathcal{P}_{1:t\text{-}1}$
    compute $\mathcal{L}_t^{\text{PR-ACE}}$ (Equation 3.19)
    compute $\mathcal{L}_t^{\text{EFM}}$ (Equation 3.16)
    $\mathcal{L}_t^{\text{EFC}} \leftarrow \mathcal{L}_t^{\text{PR-ACE}} + \mathcal{L}_t^{\text{EFM}}$
    Update $\theta_t \leftarrow \theta_t - \alpha \frac{\partial \mathcal{L}_t^{\text{EFC}}}{\partial \theta_t}$
  end
  $\mathcal{P}_{1:t\text{-}1} \leftarrow \mathcal{P}_{1:t\text{-}1} + \Delta(E_{t\text{-}1})$ (Equations 3.20, 3.21)
  $\mathcal{P}_{1:t} \leftarrow \mathcal{P}_{1:t\text{-}1} \cup \mathcal{P}_t$
  $E_t \leftarrow \text{EFM}(\mathcal{X}_t, \mathcal{M}_t)$ (Equations 3.6, 3.7)
end

subsequent incremental tasks. The training procedure for the proposed method is described in Algorithm 1. The algorithm assumes that the first training has already been completed, as it does not require particular treatment.

Note that it is essential to combine $\mathcal{L}^{\text{EFC}}$ with $\mathcal{L}_t^{\text{PR-ACE}}$. E_{t-1} *selectively* inhibits drift in important feature directions, in contrast to Feature Distillation which inhibits drift in *all* directions (Figure 3.3 (left)). The resulting plasticity increases the risk of prototype drift, which in turn adapts previous-task classifiers to drifted prototypes (Figure 3.3 (middle)). Our use of $\mathcal{L}_t^{\text{PR-ACE}}$ to balance prototypes and new task samples, and our prototype drift compensation, counters this effect (Figure 3.3 (right)).

3.4 Experimental Results

In this section we compare Elastic Feature Consolidation with the state-of-the-art in EFCIL.

3.4.1 Datasets, metrics, and hyperparameters

We perform our experimental evaluation on three standard datasets. CIFAR-100 [112] consists of 60,000 images divided into 100 classes, with 600 images per class (500 for training and 100 for testing). Tiny-ImageNet [113] consists of 100,000 images divided into 200 classes, which are taken from ImageNet and resized to 64×64 pixels.

Table 3.1: Comparison with the state-of-the-art on CIFAR-100, TinyImageNet, and ImageNet-Subset.

		Warm Start				**Cold Start**			
		A^K_{step}		A^K_{inc}		A^K_{step}		A^K_{inc}	
	Method	**10 Step**	**20 Step**	**10 Step**	**20 Step**	**10 Step**	**20 Step**	**10 Step**	**20 Step**
CIFAR-100	EWC [44]	21.08 ± 1.09	13.53 ± 1.11	41.00 ± 1.11	31.79 ± 2.77	31.17 ± 2.94	17.37 ± 2.43	49.14 ± 1.28	31.02 ± 1.15
	LwF [50]	20.73 ± 1.47	11.78 ± 0.60	41.95 ± 1.30	28.93 ± 1.62	32.80 ± 3.08	17.44 ± 0.73	53.91 ± 1.67	38.39 ± 1.05
	PASS [53]	53.42 ± 0.48	47.51 ± 0.37	63.42 ± 0.69	59.55 ± 0.97	30.45 ± 1.01	17.44 ± 0.69	47.86 ± 1.93	32.86 ± 1.03
	Fusion‡ [54]	56.86	51.75	65.10	61.60	–	–	–	–
	FeTrIL [62]	56.79 ± 0.40	52.61 ± 0.81	65.03 ± 0.66	62.50 ± 1.03	34.94 ± 0.46	23.28 ± 1.24	51.20 ± 1.13	38.48 ± 1.07
	SSRE [60]	57.48 ± 0.55	52.98 ± 0.63	65.78 ± 0.59	63.11 ± 0.84	30.40 ± 0.74	17.52 ± 0.80	47.26 ± 1.91	32.45 ± 1.07
	EFC	**60.87** ± 0.39	**55.78** ± 0.42	**68.23** ± 0.68	**65.90** ± 0.97	**43.62** ± 0.70	**32.15** ± 1.33	**58.58** ± 0.91	**47.36** ± 1.37
TinyImageNet	EWC [44]	6.73 ± 0.44	5.96 ± 1.17	18.48 ± 0.60	13.74 ± 0.52	8.00 ± 0.27	5.16 ± 0.54	24.01 ± 0.51	15.70 ± 0.35
	LwF [50]	24.00 ± 1.44	7.58 ± 0.37	43.15 ± 1.02	22.89 ± 0.64	26.09 ± 1.29	15.02 ± 0.67	45.14 ± 0.88	32.94 ± 0.54
	PASS [53]	41.67 ± 0.64	35.01 ± 0.39	51.18 ± 0.31	46.65 ± 0.47	24.11 ± 0.48	18.73 ± 1.43	39.25 ± 0.90	32.01 ± 1.68
	Fusion‡ [54]	46.92	44.61	–	–	–	–	–	–
	FeTrIL [62]	45.71 ± 0.39	44.63 ± 0.49	53.95 ± 0.42	52.96 ± 0.45	30.97 ± 0.90	25.70 ± 0.61	45.60 ± 1.67	39.54 ± 1.19
	SSRE [60]	44.66 ± 0.45	44.68 ± 0.36	53.27 ± 0.43	52.94 ± 0.42	22.93 ± 0.95	17.34 ± 1.06	38.82 ± 1.99	30.62 ± 1.96
	EFC	**50.40** ± 0.25	**48.68** ± 0.65	**57.52** ± 0.43	**56.52** ± 0.53	**34.10** ± 0.77	**28.69** ± 0.40	**47.95** ± 0.61	**42.07** ± 0.96
Imagenet-Subset	EWC [44]	16.19 ± 2.48	10.66 ± 1.74	23.58 ± 2.01	18.05 ± 1.10	24.59 ± 4.13	12.78 ± 1.95	39.40 ± 3.05	26.95 ± 1.02
	LwF [50]	21.89 ± 0.52	13.24 ± 1.61	37.15 ± 2.47	25.96 ± 0.95	37.71 ± 2.53	18.64 ± 1.67	56.41 ± 1.03	40.23 ± 0.43
	PASS [53]	52.04 ± 1.06	44.03 ± 2.19	65.14 ± 0.36	58.88 ± 2.15	26.40 ± 1.33	14.38 ± 1.22	45.74 ± 0.18	31.65 ± 0.42
	Fusion‡ [54]	60.20	51.60	70.00	63.70	–	–	–	–
	FeTrIL [62]	63.56 ± 0.59	57.62 ± 1.13	71.87 ± 1.46	68.01 ± 1.60	36.17 ± 1.18	26.63 ± 1.45	52.63 ± 0.56	42.43 ± 2.05
	SSRE [60]	61.84 ± 0.93	55.19 ± 0.97	70.68 ± 1.37	66.73 ± 1.61	25.42 ± 1.17	16.25 ± 1.05	43.76 ± 1.07	31.15 ± 1.53
	EFC	**68.85** ± 0.58	**62.17** ± 0.69	**75.40** ± 0.92	**71.63** ± 1.13	**47.38** ± 1.43	**35.75** ± 1.74	**59.94** ± 1.38	**49.92** ± 2.05

ImageNet-Subset [114] is a subset of the original ImageNet dataset that consists of 100 classes. The images are resized to 224×224 pixels.

Each experiment is evaluated in two settings. The first is the *Warm Start* (WS) scenario commonly considered in EFCIL [53, 60, 62, 54] which uses a larger first task consisting of 50 classes for CIFAR-100 and ImageNet-Subset in the 10-step scenario, and 40 classes in the 20-step scenario. For Tiny-ImageNet, both the 10-step and 20-step scenarios use a large first task consisting of 100 classes. Regardless of 10-step or 20-step, the remaining classes after the first task are uniformly distributed among the subsequent tasks. The second setting, referred to as *Cold Start*(CS), uniformly distributes all classes among all tasks. We are especially interested in the Cold Start scenario since it requires backbone plasticity and is very challenging for EFCIL.

We measure using *per-step incremental accuracy* A^K_{step} and *average incremental accuracy* A^K_{inc}:

$$A^K_{\text{step}} = \frac{\sum_{i=1}^{K} |\mathcal{C}_i| a_i^K}{\sum_{i=1}^{K} |\mathcal{C}_i|}, \quad A^K_{\text{inc}} = \frac{1}{K} \sum_{i=1}^{K} A^i_{\text{step}}. \tag{3.23}$$

where a_i^K represents the accuracy of task i after training task K. We report both metrics to unify comparisons since some works report only per-step accuracy [53] and others only average incremental accuracy [62, 60].

Hyperparameter Settings. We use the standard ResNet-18 backbone [14] trained from scratch for all experiments. We train the first task of each state-of-the-art method

using the same optimizer, number of epochs and data augmentation. In particular, we train all the first task using self-rotation as performed by Zhu et al. [53], Toldo and Ozay [54] to align all the performance.

For the incremental steps of EFC we used Adam with weight decay of $2e^{-4}$ and fixed learning rate of $1e^{-4}$ for Tiny-ImageNet and CIFAR-100, while for ImageNet-Subset we use a learning rate of $1e^{-5}$ for the backbone and $1e^{-4}$ for the heads. We fixed the total number of epochs to 100 and use a batch size of 64. We set $\lambda_{\text{EFM}} = 10$ and $\eta = 0.1$ in Equation 3.16 for all the experiments. We ran all approaches using five random seeds and shuffling the classes in order to reduce the bias induced by the choice of class ordering [55, 35].

In Appendix A.1 we provide the optimization settings for the first tasks and for each state-of-the art method we evaluated.

3.4.2 Comparison with the state-of-the-art

In Table 3.1 we compare EFC with baselines and state-of-the-art EFCIL approaches. Specifically we consider EWC [44], LwF [50], PASS [53], Fusion [54], FeTrIL [62] and SSRE [60]. Our evaluation considers two key scenarios: Warm Start, commonly found in the EFCIL literature, and the more difficult Cold Start scenario. EFC significantly outperforms the previous state-of-the-art in both metrics across all scenarios of the considered datasets.

Notably, on the ImageNet-Subset Warm Start scenario EFC exhibits a substantial improvement of about 5% over FeTrIL in both per-step and average incremental accuracy. For SSRE, our incremental accuracy results are significantly higher compared to the results reported in the original paper.

This is attributable to the self-rotation we used as an initial task to align all results with PASS and Fusion (which both use self-rotation). This alignment is crucial when dealing with Warm Start scenarios, as performance on the large first task heavily biases the metrics (Equation 3.23) as discussed by [115, 116]. In Figure 3.4 and Figure 3.5 we provide per-step performance plots showing that all the evaluated methods begin from the same starting point, for both Warm Start and Cold Start scenarios

In Cold Start scenarios we see that methods relying on feature distillation, such as FeTrIL, SSRE and PASS, experience a significant decrease in accuracy compared to EFC. This drop in accuracy can be attributed to the fact that the first task does not provide a sufficiently strong starting point for the entire class-incremental process. Moreover, most of these approaches exhibit weaker performance even when compared to LwF, which does not use prototypes to balance the classifiers. On the contrary, the plasticity offered by the EFM allows Elastic Feature Consolidation to

‡Fusion results are those reported by [54] since no code is available to reproduce them.

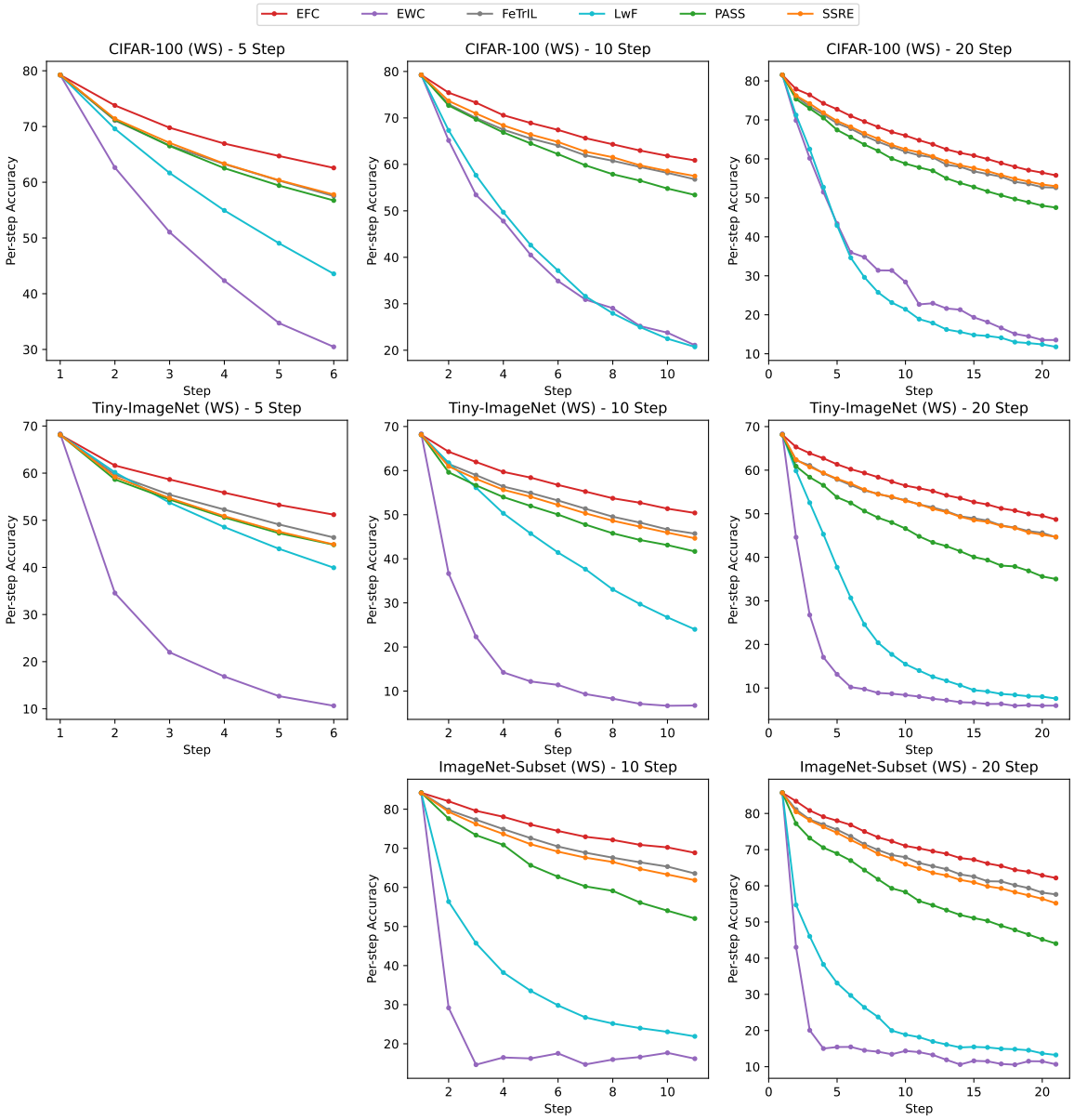


Figure 3.4: Warm Start (WS) per-step accuracy during the incremental learning. The plots compare recent EFCIL methods against EFC on Cifar-100, Tiny-ImageNet and ImageNet Subset for different incremental step sequences. All methods begin from the same starting point after training on the large first task, but EFC maintains more plasticity thanks to the EFM regularizer and is thus better at learning new tasks.

achieve excellent performance even in this setting, surpassing all other approaches by a significant margin.

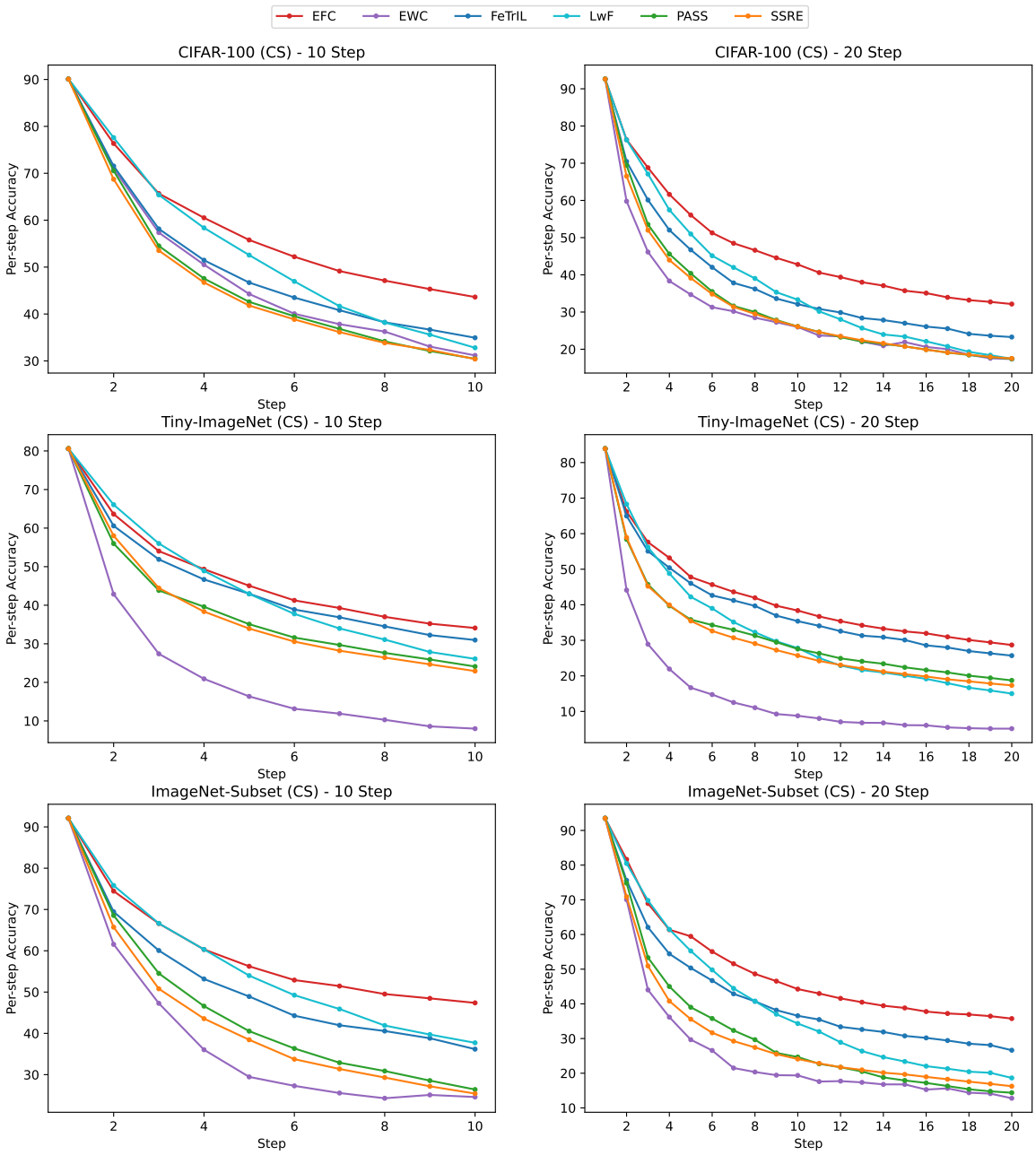


Figure 3.5: Cold Start (CS) per-step accuracy plots during incremental learning. The plots compare recent EFCIL methods with EFC on Cifar-100, Tiny-ImageNet and ImageNet subset for different incremental step sequences.All methods begin from the same starting point after training on the small first task, but EFC both maintains plasticity thanks to the EFM regularizer and is able to better update previous task classifiers to learn new tasks and mitigate forgetting.

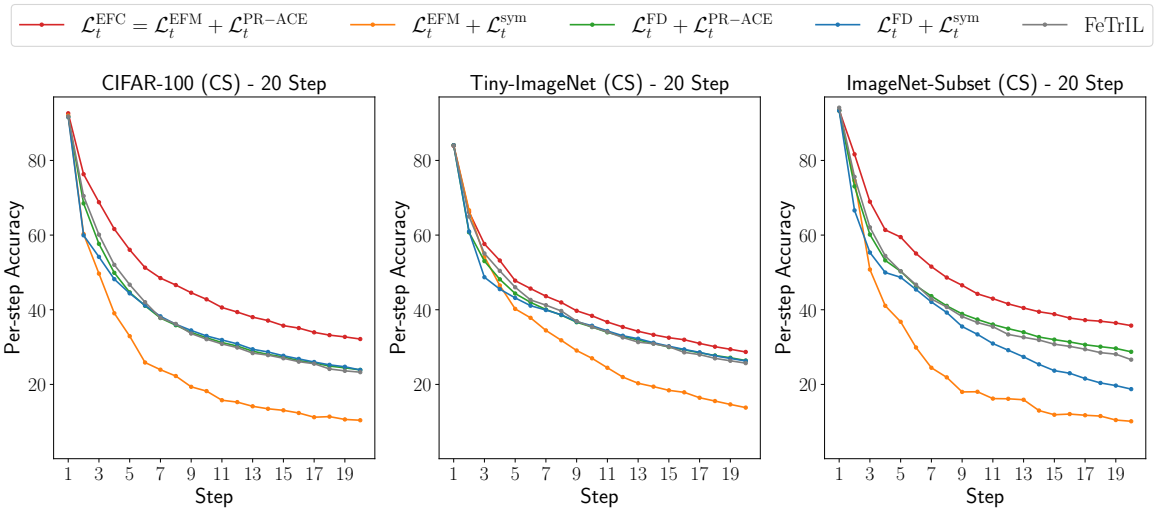


Figure 3.6: Ablation on regularization and prototype losses. We can clearly see that feature distillation as regularizer obtains similar performance to FeTrIL, which freezes the backbone after the first task, while EFC thanks to its plasticity obtains superior performance.

3.4.3 Ablation Study

Ablation on PR-ACE and Feature Distillation. We evaluated the performance of our regularizer in two settings: when combined with the proposed PR-ACE (Equation 3.19) and when combined with the standard *symmetric loss* (Equation 3.18). Figure 3.6 clearly demonstrates that the symmetric loss works well in conjunction with feature distillation. However, when used alongside the EFM, it fails to effectively control the adaptation of the old task classifier, resulting in an overall drop in performance. Additionally, it is evident from the results that feature distillation essentially degenerates into a method similar to FeTrIL, which freezes the backbone after the first task. These findings suggest that using feature distillation as a regularizer is equivalent to freezing the backbone after the first task and employing a suitable prototype rehearsal approach to balance the task classifier, pseudo-samples for FeTrIL, and Gaussian prototypes in our specific scenario.

To gain further insights, Figures 3.8 and 3.7 display the per-task classification accuracy following the final training session. These figures compare the EFC performance with FeTrIL and Feature Distillation methods in both Cold Start and Warm Start scenarios. These plots clearly show that EFC sacrifices some points of accuracy on the first tasks to maintain plasticity needed to learn new ones, while FD and FeTrIL, have high stability to remember first tasks, while little plasticity to learn the newest one.

Ablation on Prototype Update. In Table 3.2 we evaluate the effect of prototype updates (Equation 3.20 and Equation 3.21) on Cifar-100, Tiny-ImageNet and

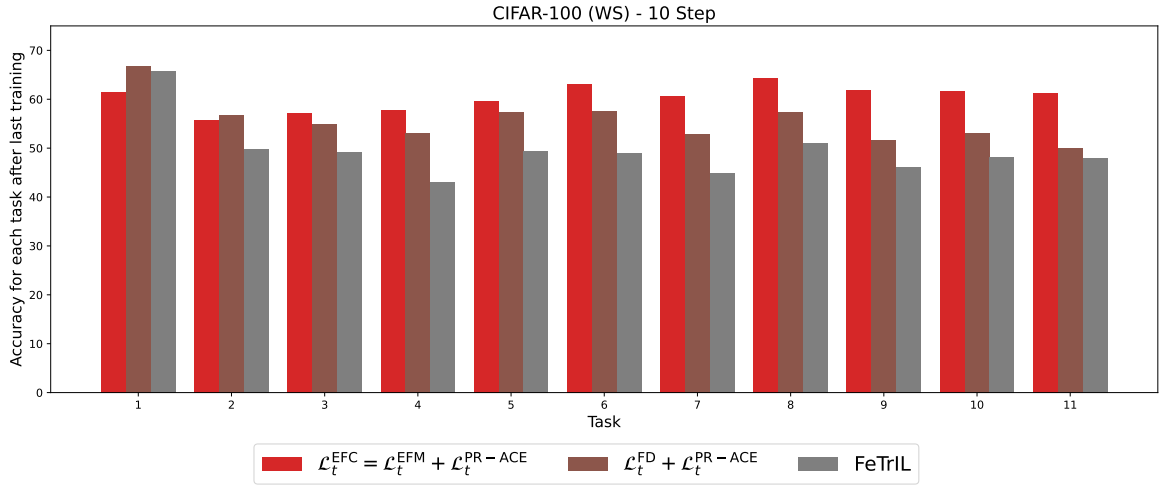


Figure 3.7: Per-task accuracy after the last training in the CIFAR-100(WS) 10 step scenario.

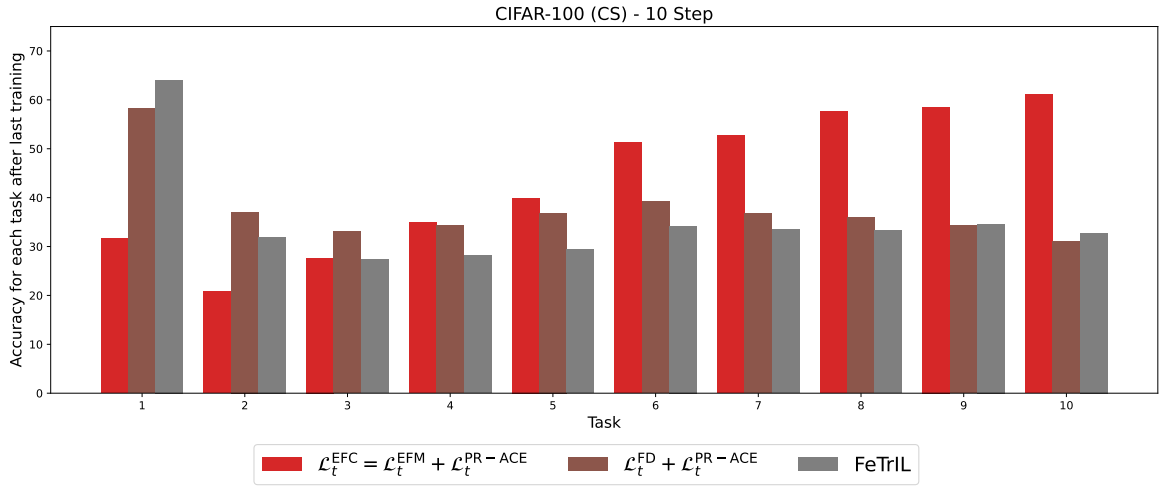


Figure 3.8: Per-task accuracy after the last training in the CIFAR-100(CS) 10 step scenario.

Table 3.2: Ablation on prototype update.

Dataset	Step	Update Proto (WS) ✗	Update Proto (WS) ✓	Update Proto (CS) ✗	Update Proto (CS) ✓
CIFAR-100	10	58.24 ± 0.62	**60.87** ± 0.39	39.28 ± 0.97	**43.62** ± 0.70
	20	52.60 ± 0.73	**55.78** ± 0.42	27.28 ± 1.48	**32.15** ± 1.33
Tiny-ImageNet	10	48.16 ± 0.53	**50.40** ± 0.25	31.23 ± 0.88	**34.10** ± 0.77
	20	45.70 ± 0.74	**48.68** ± 0.65	25.12 ± 0.23	**28.69** ± 0.40
ImageNet-Subset	10	66.92 ± 0.43	**68.85** ± 0.58	39.99 ± 2.20	**47.38** ± 1.43
	20	59.90 ± 1.09	**62.17** ± 0.69	29.62 ± 2.60	**35.75** ± 1.74

ImageNet-Subset. These results indicate that prototype updates enhance performance in both the Warm Start and the Cold Start experiments. The performance boost is more pronounced in the Cold Start scenario, where we achieve, on average on the three dataset, a substantial 5 and 7 points of improvement for the 10 and 20 steps respectively.

Comparing these results with those presented in Table 3.1, it is evident that incorporating prototype updates in some cases within the Cold Start Scenario enables surpassing the performance of existing methods, something that does not always happen when using fixed prototypes. For instance, FeTrIL (CS) Tiny-ImageNet - 20 step reaches 25.70 in accuracy compared to only 25.12 achieved by EFC with fixed prototypes.

3.4.4 Spectral Analysis of the Empirical Feature Matrix

In this section we conduct a spectral analysis on our Empirical Feature Matrix (EFM). Since Elastic Feature Consolidation (EFC) uses E_t to selectively regularize drift in feature space, it is natural to question *how* selective its regularization is. We can gain insight into this by analyzing how the rank of E_t evolves with increasing incremental tasks.

Let us consider a model incrementally trained on CIFAR-100 using EFC method up to the sixth and final task and compute the spectrum of the E_t for each $t \in [1, \ldots, 6]$. We chose a shorter task sequence for simplicity, but the same empirical conclusions hold regardless of the number of tasks. Recalling that $E_t \in \mathbb{R}^{n \times n}$ is symmetric, thus there exist U_t and $\Lambda_t \in \mathbb{R}^{n \times n}$ such that:

$$E_t = U_t^{-1} \Lambda_t U_t = U_t^{\top} \Lambda_t U_t, \tag{3.24}$$

where U_t is an orthogonal matrix and $\Lambda_t = \text{diag}(\lambda_1, .., \lambda_n)$ is a diagonal matrix whose entries are the eigenvalues of E_t. Since E_t is positive semi-definite, $\lambda_i \geq 0$ for each $i \in [1, \ldots, n]$. For our convenience we can rearrange the matrices Λ_t and U_t

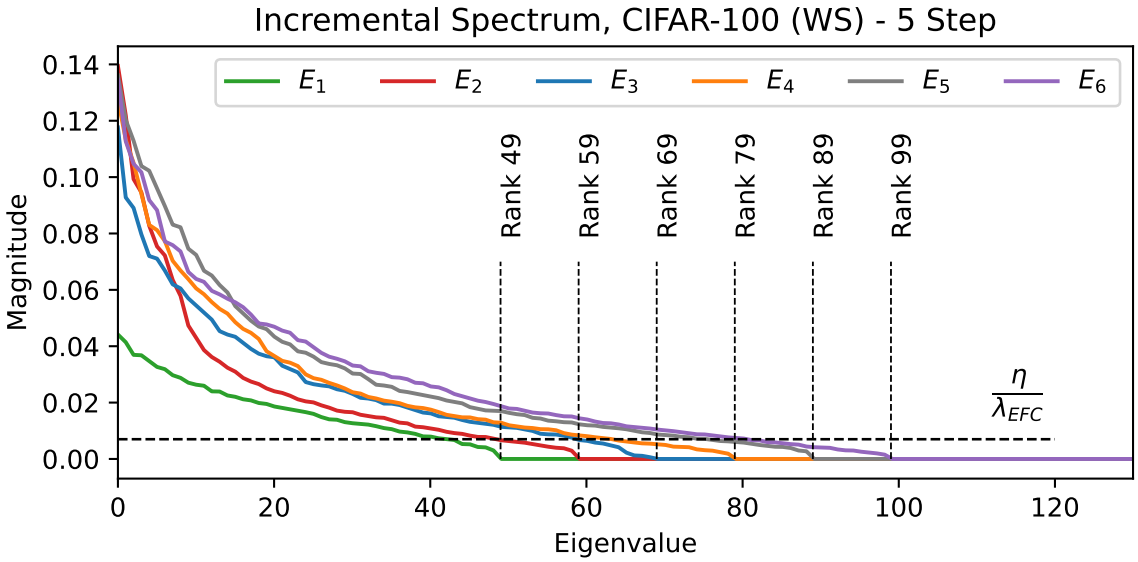


Figure 3.9: The spectrum of the Empirical Feature Matrix across incremental learning steps. For a better visualization of the spectrum in the analysis we considered a Warm Start 5-step scenario on CIFAR-100. The x-axis is truncated at the 120th eigenvalue.

of the decomposition such that the eigenvalues on the diagonal are ordered by the greatest to the lowest, with the first k being positive.

In Figure 3.9, we plot the spectrum of E_t at each $t \in [1, \ldots, 6]$. The plot shows that the number of classes on which the matrix is estimated corresponds to an elbow in the curve beyond which the spectrum of the matrix vanishes. This implies that the rank of the matrices is exactly equal to the number of observed classes.

Finally, we highlight that the results reported for EFC are obtained using hyperparameter values of $\lambda_{\text{EFM}} = 10$ and $\eta = 0.1$ in the regularization loss (Equation 3.16). In Figure 3.9, the horizontal line indicates that the plasticity constraints $\lambda_{\text{EFM}}\mu_i > \eta$ (described in Section 3.2.3 of the paper) are satisfied for nearly every $\mu_i > 0$. This observation clearly demonstrates that our regularizer effectively constrains the features in a non-isotropic manner, distinguishing it from feature distillation.

3.4.5 Computational and Storage Cost

In this section, we explore the storage costs and computational requirements for training EFC. For memory costs, we delve into the specific storage needs of EFC and examine strategies to reduce these expenses. In terms of computational requirements, we provide the training wall-clock time comparison between EFC and other

Table 3.3: Warm Start EFC performance with low-rank covariance approximations for Gaussian prototype sampling. On both datasets in all task sequences using fewer than 10% of the principal directions on average preserves 99% of the total variance and maintains the same accuracy as using full covariance matrices (full covariance with preserved variance = 1.00 given for reference).

	CIFAR-100			Tiny-ImageNet		
# Steps	**Preserved Variance**	**Avg # Components**	A^K_{step}	**Preserved Variance**	**Avg # Components**	A^K_{step}
5	0.90	11 ± 2	57.35	0.90	15 ± 3	49.02
5	0.95	18 ± 4	59.68	0.95	23 ± 4	49.72
5	0.99	39 ± 6	61.65	0.99	48 ± 6	50.85
5	1.00	512	62.57	1.00	512	51.19
10	0.90	11 ± 2	55.79	0.90	15 ± 3	47.79
10	0.95	18 ± 3	58.28	0.95	24 ± 4	48.77
10	0.99	38 ± 6	60.66	0.99	48 ± 6	50.54
10	1.00	512	60.87	1.00	512	50.40
20	0.90	11 ± 2	50.67	0.90	15 ± 3	46.64
20	0.95	18 ± 3	52.95	0.95	23 ± 4	47.49
20	0.99	38 ± 6	54.91	0.99	48 ± 6	48.39
20	1.00	512	55.78	1.00	512	48.68

EFCIL methods.

Storage Costs. In terms of storage efficiency, Elastic Feature Consolidation requires to store a single EFM that is used as pseudo-metric to regularize the drift and a covariance matrix per class to replay the prototype across the incremental learning steps. This last requirement results in a linear increase in storage as the number of classes grows. We discuss how this memory cost can be mitigated by applying a *low rank decomposition* of the class covariance matrices and two more aggressive strategies, named *Each Task* and *Last Task*, further mitigating storage costs.

A spectral analysis of the covariance matrices reveals that for all classes in all datasets and all task sequences the spectra are highly concentrated in the first 30-50 eigenvalues and are thus well-approximated by *low-rank* reconstructions. To verify this we ran experiments with low-rank reconstructions and give results for CIFAR-100 and Tiny-ImageNet in Table 3.3. On average, using only 30 eigenvectors is sufficient to preserve 99% of the total variance and maintain the same EFCIL accuracy.

To further mitigate storage costs, we consider two more aggressive strategies: the *Each Task* strategy, which employs a single covariance matrix per task for proto-

Table 3.4: Mitigating storage costs with proxy covariance matrices. We can save only one covariance matrix per task (**Each Tas**k) to use for prototype generation, which results in storage that grows linearly in number of tasks. Alternatively, we can store only the covariance matrix from the **Last Task**, which results in a constant storage cost. Both proxies sacrifice some EFCIL accuracy in the Warm Start scenario, but results are still comparable with the state-of-the-art (compare with results in Table 3.1).

		Warm Start				**Cold Start**			
		$A^K_{\textbf{step}}$		A^K_{inc}		A^K_{step}		A^K_{inc}	
	Variant	**10 Step**	**20 Step**	**10 Step**	**20 Step**	**10 Step**	**20 Step**	**10 Step**	**20 Step**
CIFAR100	Each Task	59.37 ± 0.18	54.97 ± 0.86	67.34 ± 0.38	65.12 ± 0.92	**45.26** ± 0.62	**33.73** ± 2.38	**60.11** ± 0.85	**48.96** ± 1.90
	Last Task	58.22 ± 0.80	52.79 ± 0.10	66.79 ± 0.49	63.89 ± 0.86	43.72 ± 0.38	31.43 ± 2.48	59.46 ± 0.99	47.27 ± 1.53
	EFC	**60.87** ± 0.39	**55.78** ± 0.42	**68.23** ± 0.68	**65.90** ± 0.97	43.62 ± 0.70	32.15 ± 1.33	58.58 ± 0.91	47.36 ± 1.37
Tiny	Each Task	49.00 ± 0.48	47.26 ± 0.18	56.31 ± 0.51	55.06 ± 0.39	**34.81** ± 0.56	**29.39** ± 0.52	47.84 ± 0.39	**42.53** ± 0.17
	Last Task	48.00 ± 0.43	45.64 ± 0.51	55.85 ± 0.54	54.12 ± 0.37	34.36 ± 0.14	28.57 ± 0.32	47.50 ± 0.24	42.11 ± 0.11
	EFC	**50.40** ± 0.25	**48.68** ± 0.65	**57.52** ± 0.43	**56.52** ± 0.53	34.10 ± 0.77	28.69 ± 0.40	**47.95** ± 0.61	42.07 ± 0.96
ImageNet	Each Task	66.53 ± 0.48	60.55 ± 0.73	74.10 ± 0.62	70.36 ± 1.24	**48.49** ± 2.18	**36.31** ± 3.35	**61.19** ± 2.46	**50.52** ± 2.22
	Last Task	65.46 ± 0.69	59.20 ± 0.69	73.42 ± 0.72	69.59 ± 1.40	47.21 ± 1.57	35.43 ± 3.00	60.50 ± 2.16	50.22 ± 1.94
	EFC	**68.85** ± 0.58	**62.17** ± 0.69	**75.40** ± 0.92	**71.63** ± 1.13	47.38 ± 1.43	35.75 ± 1.74	59.94 ± 1.38	49.92 ± 2.05

type generation and requires storage that scales linearly with the number of tasks; and the *Last Task* strategy, which stores only the covariance matrix from the most recently learned task and thus requires constant storage. In Table 3.4, we present the results of these strategies. It is noteworthy that storing a per-class task covariance significantly impacts the Warm Start performance. However, in the Cold Start scenario, storing a covariance per task appears to be less beneficial, in fact the differences between variants of our method are practically negligible. We hypothesize that this outcome arises because, in the Cold Start (CS) scenario, representations are more susceptible to changes, making per-class fixed covariance less crucial. Overall, even with these proxies, EFC for both Cold-Start and Warm-Starts, results remain comparable to or even better than the state-of-the-art.

Computational Costs. In terms of computational efficiency, the calculus of the EFM can be derived without computing backward operations, but only a forward pass of the entire training set at the end of each task (Equation 3.7).

For the training time, we conducted a wall-clock time analysis to assess the computational costs of EFC. The results of this analysis are presented in Figure 3.10. LwF is the fastest methods since it does not rely upon prototype rehearsal. Among the prototype-based methods, EFC is comparable to SSRE in terms of computational time and it is much faster than PASS which is more time-consuming since applies self-rotation across the incremental steps. In the plot we do not report FeTrIL, relying upon SVM for the training. Currently, FeTrIL is the fastest state-of-the-art EFCIL approach since it freezes the backbone and updates only the classifier.

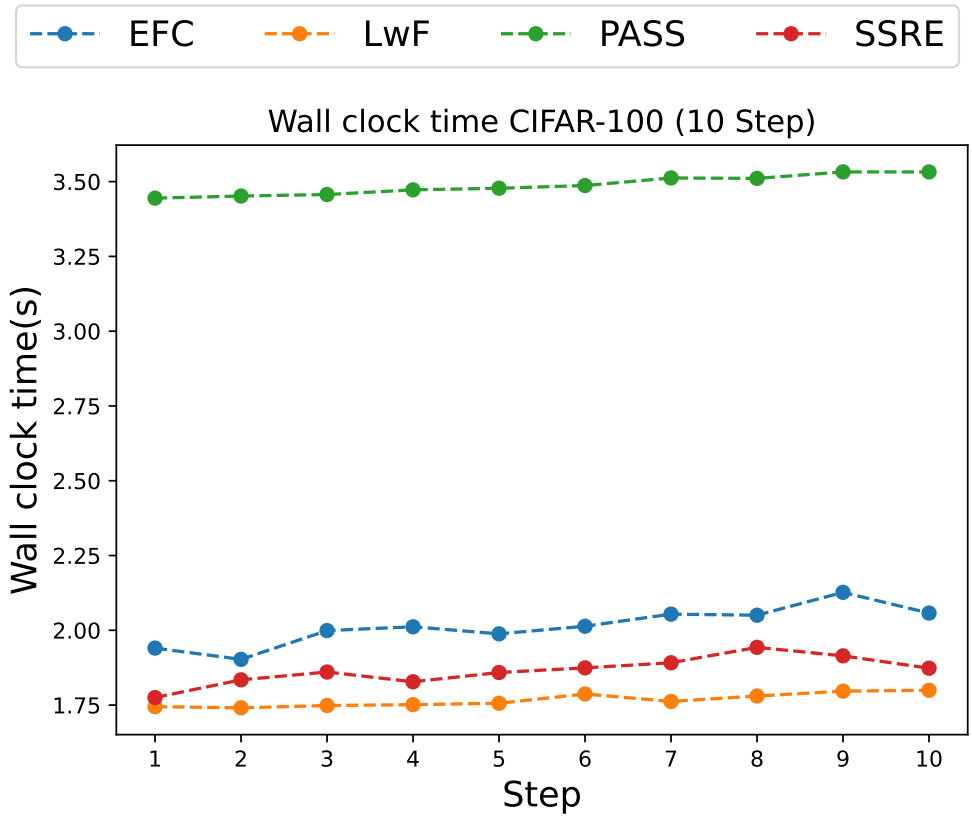


Figure 3.10: Wall clock timing comparison. We compute the per-epoch time averaged over 100 epochs. These timings are obtained using a single NVIDIA GeForce RTX 3090.

3.5 Conclusions

In this chapter, we introduced Elastic Feature Consolidation which regularizes feature drift in directions in feature space important to previous-task classifiers. We derive an Empirical Feature Matrix that induces a pseudo-metric used to control feature drift and allow more plasticity than feature distillation in directions unimportant for previous tasks. We further use this Empirical Feature Matrix in an Asymmetric Prototype Replay loss and to update prototypes across incremental learning steps. The results of our study reveal that the additionally plasticity introduced by EFC allows for significantly outperforming the state-of-the-art on both Cold Start and Warm Start Exemplar-free class incremental learning, achieving a better stability-plasticity trade-off.

However, EFC faces challenges: prototypes updates using the drift estimated from current class samples may not accurately reflect, real class means, potentially leading to forgetting over long task sequences. Also, estimating covariance drift remains an open question. Moreover, EFC requires storage that grows linearly in the

number of classes, but this growth can be mitigated using low-rank approximations or proxies for class covariance matrices.

Chapter 4

Incremental Learning for Social Network Identification

Multimedia content such as images and videos have become one of the primary means by which information is shared between Internet users. Unfortunately, this also includes content used to perpetrate crimes such as cyber bullying, incitement to hatred, and revenge porn. As a result, determining the origin of multimedia content is of great interest not only to law enforcement agencies but also to the general public. As the number of images and videos stored in seized devices can easily reach into the thousands, such analysis can often only be performed by automatic tools. This problem has been addressed by the multimedia forensics community through a number of techniques capable of analyzing different aspects of content history. Among these, discovering the social network from which content was downloaded has become of great interest in the last few years [119]. Knowledge about the social network of origin can then be used to guide further analyses, which can ultimately lead to the complete reconstruction of content history.

Unfortunately, the identification of social networks is a daunting task due to their black box nature. Their inner workings are closely guarded by parent companies who consider them proprietary information and researchers are consequently forced to depend on hidden clues embedded in shared media that arise from the processing performed by social platforms. The processing chain that multimedia content undergoes ends with a compression algorithm to reduce file size as much as possible while maintaining maximum visual quality [120]. When a picture is

Portions of this chapter were published in:

- [117] S. Magistri, D. Baracchi, D. Shullani, A. D. Bagdanov, and A. Piva, "*Towards continual social network identification*," in 11th International Workshop on Bio- metrics and Forensics, 2023, pp. 1–6.
- [118] S. Magistri, D. Baracchi, D. Shullani, A. D. Bagdanov, and A. Piva, "*Continual learning for adaptive social network identification*," Pattern Recognition Letters, vol. 180, pp. 82–89, 2024.

taken, the vast majority of smartphones and cameras store the resulting file in JPEG format. A similar procedure, which can also include resizing, renaming, and editing all or part of the metadata [121], occurs when content is shared on a social platform, resulting in a *double* JPEG compression trace. Numerous studies propose methodologies for *Social Network Identification* (SNI) that rely on factors such as JPEG quantization tables, pixel resolution, and image metadata [122]. Some researchers exploit the distribution of *Discrete Cosine Transform* (DCT) coefficients [123, 124, 125, 126] as well as Discrete Wavelet Transform coefficients (DWT) [127]. Moreover, the distinctive fingerprint of Photo Response Non-Uniformity (PRNU) noise [128], renowned for its camera ballistics capabilities, has also been taken into account for image-based SNI [129, 130, 131]. Researchers have also investigated the importance of the container structure of multimedia content [132, 133, 134, 135] to detect a specific SN platform in the content history. Today, the task of social network identification (single or multiple) is addressed predominantly through deep learning techniques based on Convolutional Neural Networks (CNNs) [119, 136], using all or a combination of the above-mentioned fingerprints to exploit both spatial and meta information of the content itself.

Despite the remarkable SNI results reported for existing CNN-based methods, the task of keeping them current and effective poses a significant challenge due to the ever-changing nature of the social network landscape. Indeed, as companies fiercely compete to attract new users to their platforms, the software responsible for managing these networks undergoes constant updates to incorporate new features and enhance existing ones. This, in turn, leads to modifications in the traces left on shared content, consequently requiring the update of previously trained models. Additionally, new economic players consistently strive to enter the market by proposing new platforms, hoping to address gaps in existing products and establish themselves in this growing industry. As data-driven methods are usually designed to classify among a predetermined set of possibilities, incorporating additional platforms inevitably requires training a new model. Because of the phenomenon known as *catastrophic forgetting*, existing models cannot be easily updated by solely finetuning them on new data; indeed, when a CNN is initially trained on one task and subsequently trained on one or more new tasks, it quickly loses its ability to perform the initial task [24]. A naive solution to avoiding catastrophic forgetting, called Joint Incremental Training, consists of jointly training the network on the new data along with the old ones. The main problem with joint training is that it is expensive to re-train the network with the entire dataset each time new data become available. Furthermore, it may not always be possible to retrieve data from previous tasks due to privacy considerations or because they are simply no longer available.

Incremental learning approaches strive to reduce catastrophic forgetting by making efficient use of limited data from past tasks. In the context of multimedia foren-

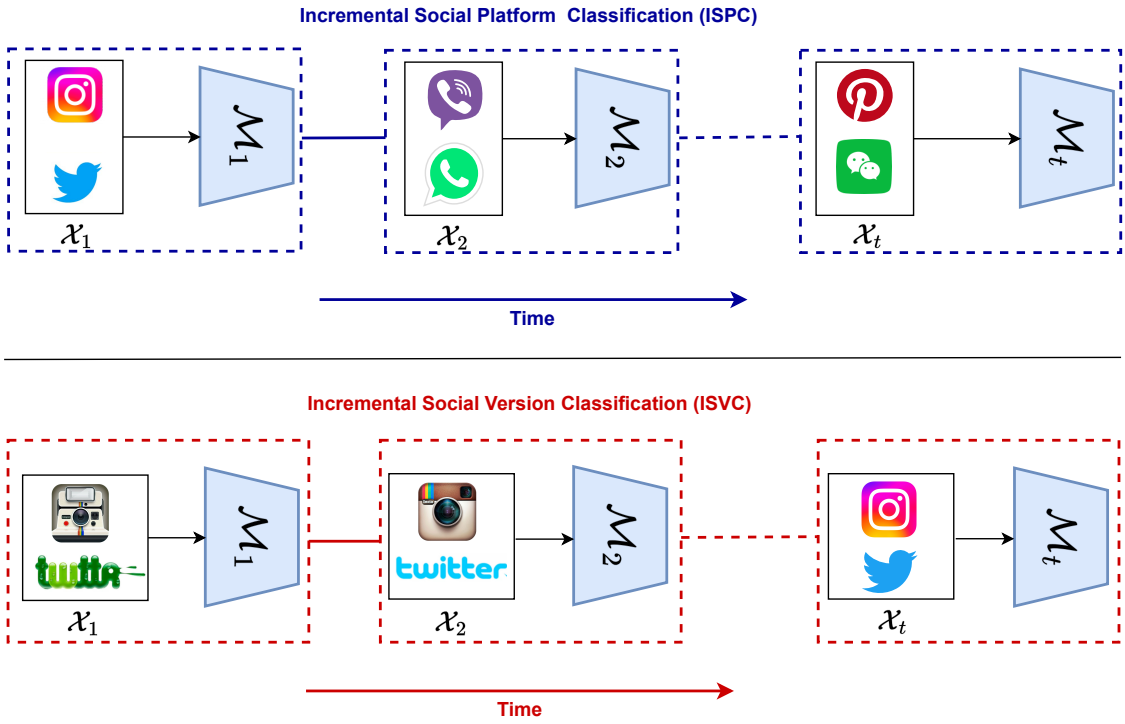


Figure 4.1: Incremental Social Platform Classification (ISPC) and Incremental Social Version Classification (ISVC) Scenarios. In the ISPC scenario (in blue), new social network platforms are introduced over time to update a model $\mathcal{M}_t$ that must classify all encountered platforms. In the ISVC scenario (in red), new versions of social networks are introduced over time, and the model should still recognize social networks while distinguishing between different versions. See Section 4.2 for further details.

sics, a first attempt at applying incremental learning techniques (although not for social network identification) was performed by Marra et al. [137]. This work showcased the efficacy of iCaRL [55] in expanding the capabilities of a network for GAN-generated image identification.

In this chapter, we introduce a *multi-modal* architecture for social network identification, combining both image pixels with features based on the Discrete Cosine Transform (DCT). Its objective is to identify the social network from which an image has been downloaded.

Furthermore, we propose two incremental learning scenarios for social network identification, depicted in Figure 4.1:

- *Incremental Social Platform Classification* (ISPC) which is focused on updating a model to accurately classify *newly introduced* social media platforms across the time.

- *Incremental Social Version Classification* (ISVC) which is focused on update a

model in order to accommodate novel *versions* of the original set of social networks on which it was trained. This update entails not only the ability to handle these new versions but also to differentiate between them.

We perform an extensive experimental evaluation of incremental learning techniques, both exemplar-based and exemplar-free, including Elastic Feature Consolidation, introduced in Chapter 3. Additionally, we investigate how the number of exemplars from past tasks affects the results of exemplar-based techniques. Our experiments demonstrate that, by employing a limited memory budget of image patches, existing incremental learning methods can approach *Joint Incremental Training* performance in both ISPC and ISVC scenarios.

4.1 Related Work

In this section we briefly review the incremental learning approaches we evaluated for incremental social network identification.

Incremental learning methods can be roughly grouped into two macro-categories: *exemplar-free* approaches [44, 45, 50] which do not store exemplars from past tasks and only add extra terms to the training loss to incorporate knowledge from past tasks during the training of new ones, and *exemplar-based* approaches [55, 56, 94], which rely on a small subset of representative samples (exemplars) from previous tasks.

Exemplar-free methods include *Elastic Weight Consolidation* (EWC) [44] and *Riemannian Walk* (RWalk) [45] which define a weight regularization loss based on the Fisher Information Matrix to prevent network weights from drifting away from the previous task model when learning new task classes. LwF, instead, uses Knowledge Distillation [82] to discourage predictions from drifting when learning new tasks [50].

Knowledge Distillation (KD) has been employed as a regularization technique by many exemplar-based methods [55, 56, 57]. Moreover, exemplar-based approaches place a significant emphasis on tackling the challenge of imbalanced data between exemplars and current-task data. The imbalance between the number of exemplars from past task classes and number of training samples for current-task classes results in a task-recency bias towards classifying images into classes of the current task [35]. To mitigate this task-recency bias, methods such as BIC [56] and IL2M [94] rectify the network outputs. More recently, SS-IL [57] was proposed which employs a separated softmax output layer in combination with task-wise knowledge distillation in order to reduce task-recency bias. Techniques like iCaRL [55] avoid this bias by using a nearest-mean rule in feature space for classification instead of relying on classification heads trained with the cross-entropy loss.

The primary challenge faced by initial attempts at exemplar-free methods lies in their inability to mitigate task-recency bias due to absence of exemplars. Recent advancements have introduced *prototype rehearsal* as a method to leverage past-task deep features, enhancing the final classifier output and thereby mitigating task-recency bias.

Prototypes, or class-means, are computed as the averages of feature vectors for each class in previous tasks. The storage of these prototypes aligns with the privacy requirements inherent in exemplar-free methods. In FeTrIL [62] a fixed feature extractor was proposed and the training is performed only on the last classifier using both previous task prototypes and current task features. Finally, EFC [61], introduced in Chapter 3, uses the Prototype Rehearsal Asymmetric Cross-entropy loss (PR-ACE) along with the Empirical Feature Matrix (EFM) to selectively regularize feature space drift and prevent catastrophic forgetting while maintaining enough plasticity to still learn new tasks.

4.2 Two Scenarios for Incremental Social Network Identification (SNI)

Given the perpetual state of change and evolution in the social network landscape, we believe that the application of incremental learning techniques can significantly enhance social network identification systems. A notable advantage of Incremental SNI methods is that they eliminate the need for maintaining a continually expanding dataset containing both old and new data. Such approaches would address concerns related to efficiency and privacy, as managing massive datasets is complicated and sensitive content need not be retained indefinitely. Additionally, the capability to update a model by training it solely on new data would offer significant time-efficiency advantages compared to retraining the entire model from scratch, thus making the process of building an updated model more cost-effective and energy-efficient. To demonstrate these advantages, we envision two practical scenarios arising from real-world social network identification tasks (depicted in Figure 4.1).

In the first scenario, which we call *Incremental Social Platform Classification* (ISPC), we hypothesize the emergence of new social networks over time. Since existing models could not have possibly been trained on these new platforms, they are bound to misclassify content coming from them, associating images with one of the pre-existing social networks. In this case our goal is to update the model to make it capable of classifying both the platforms on which it was originally trained on as well as newly-introduced ones.

In the second scenario, which we call *Incremental Social Version Classification* (ISVC), we hypothesize the release of new versions of existing social networks. These up-

dates may significantly alter the processing pipeline used to produce media content, leading to a drop in classification accuracy for models trained on older datasets. We therefore have the aim to update the network to make it correctly classify both media content produced by older version of the available social platforms and media content produced by the updated versions. Moreover, by modifying the model to make it capable of classifying both the social platform and its version, we could leverage this additional information as a clue on the temporal origin of the content.

In the ISVC scenario the goal is to classify images according to one of two possible social platforms (e.g. Instagram, Twitter) from which they were downloaded, and we assume that those platforms undergo updates over time to incorporate new features. In this case, the first task involves classifying images from the original versions of the social networks. The second task entails classifying images from the first update of those platforms, and so on for subsequent tasks. For ISPC, on the other hand, we assume that completely new social platforms are introduced over time. In this case, the primary task involves classifying images from an initial set of social platforms (e.g. Instagram, Twitter). The second task then entails classifying images from the new platforms (e.g. Viber, WhatsApp), and each subsequent task then involves handling an additional set of social networks. In both scenarios, our goal is to update a classifier to handle subsequent tasks while still retaining its ability to classify the previous ones.

4.3 A Benchmark Dataset for Incremental Social Network Identification

In existing literature, there is no focus on assessing incremental learning in the domain of social network identification. This results in a lack of datasets tailored for evaluating scenarios such as Incremental Social Platform Classification (ISPC) and Incremental Social Version Classification (ISVC). To address this gap, we have created two new benchmark dataset, each containing sufficient classes for task partitioning. These benchmarks are specifically designed to assess the performance of incremental learning in these scenarios.

In the ISPC scenario, we use *Smartphone Images* (SI) [138]. This dataset consists of a variety of indoor and outdoor images captured by multiple smartphones. We consider 14 social networks split into 4 tasks. Specifically, we allocated five social networks for the first task and three for each of the remaining three tasks. We do not use a fixed class order for the tasks, but we use different random seeds for each run in order to reduce the bias induced by the choice of class ordering [35].

For the ISVC scenario, we collected a dataset, called *Social Update* (SU), which contains different versions of four major social platforms: Facebook, Instagram,

Table 4.1: Smartphone Images and Social Update dataset statistics.

Dataset	#Classes	Split	#Patches	#Images	#Devices
Smartphone Images	14	Train	462k	17k	13
		Valid	92k	2k	2
		Test	131k	5k	4
Social Update	4/15	Train	574k	26k	52
		Valid	78k	3k	7
		Test	138k	6k	13

Twitter, WhatsApp. This dataset was created by gathering images from multiple existing datasets including SI, IPLAB [122] and FODB [139]. SU encompasses a diverse range of both indoor and outdoor scenes, captured using smartphones and cameras, and subsequently shared on social media. Moreover, we incorporate SocialNews [140], a dataset consisting of images shared by news organizations and influencers on social networks. From this last dataset we only have images from Facebook, Instagram, and Twitter since WhatsApp was not available. As a result, the dataset it characterized by 4 social network platforms and a total of 15 versions. For SU, we adopted a specific task division where each task focuses on a particular social network version, corresponding to a specific dataset. The tasks are ordered based on the chronological sequence of their publication date. The first task is based on the SI dataset (containing data released in 2015), followed by the IPLAB dataset (2016), the FODB dataset (2021), and finally the SocialNews dataset (2023). As a result, the first three tasks consist of four social networks each, while the last task only includes three. We use the SU dataset for two distinct objectives. The first is to assess the capability of an incrementally trained network to accurately classify a social network after an update, which entails a 4-class classification problem. The second objective entails evaluating the performance of the network in terms of social version classification, which is a 15-class classification problem.

To ensure a fair evaluation and eliminate any biases stemming from the acquisition device, we divided the SI and SU dataset into three separate sets (training, validation, and test). These sets were carefully constructed to ensure there is no overlap in the devices used. In Table 4.1 we report the overall statistics of the two datasets.

4.4 Social Network Identification Architecture

We propose a dual-branch architecture for assessing incremental learning in social network identification (depicted in Figure 4.2). This architecture comprises a fea-

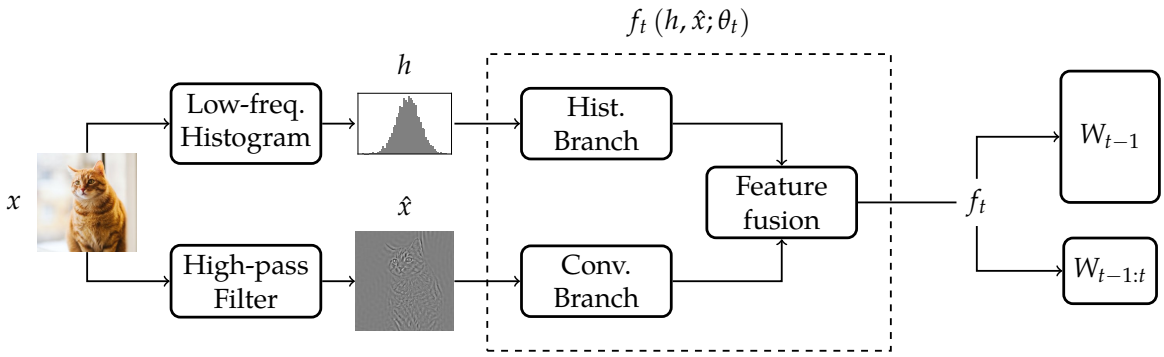


Figure 4.2: Model architecture for Incremental Social Network Identification. The network consists of two parallel branches: the top branch is a multilayer perceptron taking as input the histograms of quantized DCT coefficients, and the lower branch is a ReNet-18 backbone taking as input an image after high-pass filtering. The two representations are fused via concatenation before the classification heads. See Section 4.4 for details.

ture extractor $f_t(\cdot;\theta_t)$ and a softmax-based classifier with weights W_t that are updated through the incremental learning steps. The feature extractor handles both image pixel information and DCT-based features. Our choice to incorporate these inputs into the network is based on their proven capability to represent the traces left by social networks on images. This approach is supported by Amerini et al. [131] which employed a dual-branch architecture similar to the ours in terms of input representation, yet differing in the structure of the models

The architecture operates on fixed-size image patches, obtained by dividing high-resolution images into non-overlapping sections of size 256×256 pixels. In the pre-processing phase, these patches undergo transformations to provide two distinct representations of the image signal to the model. The first is a *low-frequency histogram*, generated from the DCT of the image patch. The second is a *high-pass filtered* image, created by converting the image patch to gray-scale and then applying a high pass filter.

The *low-frequency histogram* representation, originally proposed by Amerini et al. [131], is obtained dividing the image patch x into non-overlapping 8×8 pixel blocks, aligned with the JPEG grid. These blocks are transformed into the DCT frequency domain, and the first 9 AC coefficients of each block, with values between -50 and 50, are utilized to compute 9 histograms. These histograms are concatenated to create a feature vector h, which is fed into the *Histogram Branch*. In our architecture, this branch is a classic feed-forward network, consisting of three ReLU layers, with 512, 256, and 256 neurons respectively, interleaved with two dropout layers.

The *high-pass filtered* image is generated by transforming x into a residual image $\hat{x}$ using a high-pass filter. We propose to use a filter, removing the DCT coefficients corresponding to the lowest 1250 frequencies. The residual image $\hat{x}$ is then pro-

cessed by the *Convolutional Branch*, which employs a ResNet-18 [141] convolutional neural network.

It is worth noting that the two representations, h and $\hat{x}$, offer complementary information to the network. This approach is based on the observation that social network recompression traces frequently occur in medium frequencies. The histogram representation h is designed to enhance low/medium frequencies, whereas the filtered image $\hat{x}$ concentrates on enhancing medium/high frequencies.

The output of the Histogram and Convolutional branches are then concatenated and fused via a *Fusion Layer*, consisting of a single fully connected ReLU layer with 512 neurons. Finally, the output of the fusion layer is fed in a softmax-based classifier. As previously mentioned, the architecture operates on patches of high-resolution images, thus the final softmax score is computed per patch. To derive the overall score for the full images, we employ two strategies: *softmax averaging*, which averages the softmax probabilities across the patches, and *majority voting*, which determines the prediction by selecting the most frequent outcome among the patch predictions.

4.5 Experimental Results

In this section, we assess our architecture's performance in a non-incremental setting for comparison with a state-of-the-art counterpart and subsequently, we evaluate the performance of our architecture in both the Incremental Social Platform and Incremental Social Version scenarios.

4.5.1 Experimental Settings, metrics and hyperparameters

We train all the models using Adam [89] with an initial learning rate of 10^{-3} which was decayed when the validation loss did not improve for 20 epochs. For each epoch, we randomly sampled one crop per image in order to reduce the training time. All patches were evaluated during the validation and test phases. Training was stopped when the learning rate reached 10^{-6} or when 200 epochs were reached. We average the performance of the models using 5 seed, each applying a different random initialization of the weights.

We ran all the incremental learning experiments using FACIL [35]. We consider 5 exemplar-free methods (EWC [44], LwF [50], RWalk [45], FeTrIL [62], EFC [61]) and 4 exemplar-baed methods (BIC [56], iCaRL [55], IL2M [94], SS-IL [57]). All experiments were run five times initializing the weights with different random seeds and, for the ISPC scenario, randomizing class order as mentioned in Section 4.2. For the first task of each method, we use the optimization settings mentioned above,

while for the subsequent tasks in Appendix B, we report the optimization settings and the hyperparameters of each incremental learning method evaluated.

For exemplar-based methods we used a fixed-size memory $\mathcal{M}$ with a capacity of K, containing randomly sampled image patches. We chose to save only patches and not entire high resolution images for two reasons. Firstly, since the patches have a fixed size of 256×256, they incur a lower memory burden. Secondly, saving only patches can be useful for applications where the full image content cannot be saved due to privacy concerns. After each new task, we use a rebalancing procedure for the patches stored in $\mathcal{M}$. We randomly discard patches from previous tasks to ensure a uniform distribution of exemplar patches per class. By ensuring this uniform distribution, we maintain a constant overall memory dimension K.

Metrics. We consider class-IL settings, thus we report the performance in terms of task agnostic (TAG) performance, where the task-ID is not available at inference time. To measure the performance, we provide the average accuracy of the model across the tasks, for both patches and images, after the final task [35]:

$$A_T = \frac{1}{T} \sum_{i=1}^{T} a_{i,T} \tag{4.1}$$

where $a_{i,T}$ is the accuracy obtained on task i after learning task T.

All the evaluated incremental learning approaches except iCaRL compute the global decision on images by averaging the softmax outputs for patches, since improves the prediction performance as we will see in the next Section 4.5.2. For iCaRL, after training each task, the feature vectors belonging to every patch class are extracted and their mean is computed. At inference time, test patches are classified according to the nearest class mean, while images are classified according to the minimum average distance of their patches to the class mean.

4.5.2 Performance of the SNI Architecture

Before evaluating incremental learning methods for social network identification, we first assess the performance of our architecture in a non-incremental setting. This involves training our proposed architecture on all the 14 classes of Smartphone Images. We then compare the accuracy of our model with that proposed by Amerini et al. [131], and we conduct an ablation study on the choices made for the input representation of the Convolutional Branch in our architecture.

In Table 4.2, we present both patch-level and image-level accuracy, along with the number of parameters, for our architecture compared to that of Amerini et al. [131]. For image-level accuracy, we employ both softmax-averaging and majority voting methods. While majority voting was originally proposed by Amerini et al. [131] to combine predictions from multiple patches, we find that averaging the softmax

Table 4.2: Comparison on Smartphone Images with the state-of-the-art. The proposed architecture obtains a higher accuracy both on patch- and image-level classification problems. At the same time, our network is smaller (with respect to the number of parameters) compared to the model proposed for SNI in [131].

Method	**Patch Accuracy ↑**	**Image Accuracy ↑**		**# Parameters ↓**
		Avg Softmax	**Majority Vote**	
Amerini et al. [131]	63.1 (±1.2)	72.9 (±1.6)	71.7 (±1.4)	73.3 M
Proposed architecture	**64.6** (±1.2)	**75.4** (±0.6)	**74.7** (±0.8)	**12.2 M**

Table 4.3: Performance comparison on Smartphone Images using only the convolutional branch after preprocessing with different filters.

Image pre-processing	**Accuracy ↑**	
	Patch	**Image**
No filter	38.1 (±6.0)	48.3 (±7.5)
Mihcak Filter [142]	49.7 (±2.8)	62.6 (±3.7)
Ours (DCT High-pass filter)	**55.1** (±2.9)	**67.9** (±3.2)

predictions improves the performance of both models. Overall, our model demonstrates superior results in both patch and image-level accuracy while maintaining a less complex structure in terms of number of parameters.

Ablation on Convolutional Branch. As previously mentioned, the effectiveness of using a histogram representation to detect social network traces, along with image-level information, was demonstrated by Amerini et al. [131]. Unlike them, relying on the filter by Kivanc Mihcak et al. [142] for image representation, we propose a different filtering pre-processing based on a DCT high-pass filter. To validate our approach, we trained three classifiers using only the convolutional branch as feature extractor: one with non-filtered images, one with images processed using the Mihcak filter, and one with our DCT high-pass filter, as described in Section 4.4. The results, as shown in Table 4.3, clearly indicate that filtering the images improves classification performance. Notably, our proposed DCT-based filtering further enhances the accuracy of the convolutional branch. It is also important to highlight that using both branches, rather than the convolutional branch alone, improves accuracy by about 9% at both patch and image levels. This improvement is evident when comparing the results in Table 4.2 and Table 4.3.

Table 4.4: **Incremental Social Platform Classification (ISPC) on 14 classes** in Average TAG accuracy on patches and images after the last task with and without exemplars ($K = 500$ when using exemplars). We highlight the best-performing methods, both exemplar-free and exemplar-based, in **bold** and underline the second-best approaches.

Method	Accuracy ↑ 14 classes w/o exemplars		Accuracy ↑ 14 classes w/ exemplars	
	Patch	Image	Patch	Image
Finetuning (**LB**)	23.3 (± 6.4)	22.5 (± 3.3)	49.0 (± 5.2)	52.1 (± 7.8)
EWC [44]	27.3 (± 2.1)	25.3 (± 2.2)	53.3 (± 5.2)	56.0 (± 7.0)
LwF [50]	26.0 (± 6.7)	25.6 (± 7.3)	48.8 (± 5.9)	51.1 (± 8.0)
RWalk [45]	28.2 (± 8.4)	25.8 (± 7.5)	51.3 (± 5.4)	53.9 (± 7.4)
FeTrIL [62]	36.4 (± 3.4)	40.3 (± 5.8)	-	-
EFC [61]	**39.4** (± 3.7)	**46.6** (± 6.0)	-	-
BIC [56]	-	-	**56.4** (± 6.1)	**63.3** (± 3.6)
iCaRL [55]	-	-	54.7 (± 5.8)	62.5 (± 3.1)
IL2M [94]	-	-	49.4 (± 1.8)	52.0 (± 1.8)
SS-IL [57]	-	-	46.4 (± 8.3)	52.4 (± 5.2)
Joint Incremental (**UB**)	67.7 (± 4.8)	73.0 (± 4.2)	67.7 (± 4.8)	73.0 (± 4.2)

4.5.3 Incremental Social Network Identification Evaluation.

We evaluate our proposed architecture on the Incremental Social Platform Classification (ISPC) and the Incremental Social Version Classification (ISVC), depicted in Figure 4.1. For each incremental scenario, the lower-bound (**LB**) baseline for comparison is *Finetuning* which simply consists of training the network on the new task data, while the upper bound (**UB**) is *Joint Incremental* training which consists of retraining the network on new task data along with all data from previous tasks.

Incremental Social Platform Classification (ISPC). We present the performance evaluation of incremental learning methodologies on the ISPC scenario, where new social network platforms appear over time. In Table 4.4 we report performance in average accuracy after the last task for this scenario. We also provide results for exemplar-based extensions of EWC, LwF, and RWalk in which a small number of training samples for each task are retained as exemplars and replayed when training on a new one.

Most exemplar-free techniques fail to achieve satisfactory performance, demonstrating only modest improvement over Finetuning. Only two approaches (EFC and FeTrIL) are capable of significantly increasing accuracy with respect to the lower bound. This outcome comes from the incorporation of prototypes, which are rein-

Table 4.5: **Incremental Social Version Classification (ISVC) on 15 classes** in TAG accuracy on patches and images after the last task with and without exemplars ($K = 500$ when using exemplars). We highlight the best-performing methods, both exemplar-free and exemplar-based, in **bold** and underline the second-best approaches.

Method	Accuracy ↑ 15 classes w/o exemplars		Accuracy ↑ 15 classes w/ exemplars	
	Patch	Image	Patch	Image
Finetuning (**LB**)	14.2 (± 3.7)	12.0 (± 3.9)	27.0 (± 1.5)	28.4 (± 2.7)
EWC [44]	20.1 (± 4.2)	14.4 (± 3.7)	22.3 (± 1.5)	25.8 (± 2.9)
LwF [50]	19.5 (± 0.4)	11.2 (± 0.5)	29.4 (± 2.2)	31.9 (± 3.1)
RWalk [45]	18.6 (± 1.5)	12.7 (± 1.6)	24.4 (± 4.6)	26.0 (± 3.2)
FeTrIL [62]	41.0 (± 0.6)	45.5 (± 0.7)	-	-
EFC [61]	**48.5** (± 2.1)	**51.9** (± 0.7)	-	-
BIC [56]	-	-	40.9 (± 3.5)	44.3 (± 3.5)
iCaRL [55]	-	-	40.9 (± 2.9)	**50.7** (± 2.3)
IL2M [94]	-	-	24.2 (± 4.4)	27.3 (± 4.4)
SS-IL [57]	-	-	**45.6** (± 1.3)	47.9 (± 2.0)
Joint Incremental (**UB**)	63.3 (± 1.5)	66.3 (± 0.9)	63.3 (± 1.5)	66.3 (± 0.9)

troduced during training to strike a balance between the previous and current task classifiers, effectively alleviating task-recency bias. In the exemplar-based setting, by incorporating $K = 500$ exemplars, iCaRL and BIC are the top performing methods reducing the performance gap with Joint Incremental by half. Note that the relatively high standard deviations are due to the random ordering of classes, as certain social networks exhibit similar characteristics.

Incremental Social Version Classification (ISVC). We present the performance evaluation of incremental learning methodologies on ISVC scenario, where the version of the social networks changes over time. In Table 4.5 and Table 4.6 we report performance for both exemplar-free and exemplar-based methods on this scenario for 15 and 4 classes respectively. As detailed in Section 4.3, in the 15-class setting, each distinct *version* of a social network platform is treated as an individual class. Conversely, in the 4-class setting, examples are only labelled with the originating social platform without considering the specific version. We emphasize that we did not train separate models for the 4-class case. Instead, we obtained the results by performing *a posteriori* remapping of the network outputs and disregarding any information pertaining to the version. As expected, predicting the social network version along its type (ISVC-15 classes) is a significantly more challenging setting. Indeed, most methods show a drop of more than 20 points in TAG accuracy with

Table 4.6: **Incremental Social Version Classification (ISVC) on 4 classes** in average TAG accuracy on patches and images after the last task with and without exemplars ($K = 500$ when using exemplars). We highlight the best-performing methods, both exemplar-free and exemplar-based, in **bold** and underline the second-best approaches.

Method	Accuracy ↑ 4 classes w/o exemplars		Accuracy ↑ 4 classes w/ exemplars	
	Patch	Image	Patch	Image
Finetuning (**LB**)	39.6 (± 7.0)	43.0 (± 6.1)	53.4 (± 1.2)	57.4 (± 1.2)
EWC [44]	41.5 (± 8.4)	45.6 (± 10.1)	47.2 (± 2.1)	53.5 (± 2.6)
LwF [50]	39.3 (± 0.1)	45.2 (± 0.2)	56.3 (± 1.9)	61.3 (± 2.6)
RWalk [45]	35.3 (± 3.1)	37.2 (± 3.4)	49.7 (± 5.0)	54.3 (± 3.9)
FeTrIL [62]	69.3 (± 0.6)	72.9 (± 0.6)	-	-
EFC [61]	**71.4** (± 0.4)	**73.7** (± 0.8)	-	-
BIC [56]	-	-	**70.4** (± 1.8)	**72.2** (± 1.5)
iCaRL [55]	-	-	63.3 (± 1.1)	67.1 (± 1.0)
IL2M [94]	-	-	49.5 (± 6.8)	55.2 (± 5.6)
SS-IL [57]	-	-	66.5 (± 1.2)	67.6 (± 1.6)
Joint Incremental (**UB**)	85.9 (± 0.8)	87.8 (± 0.5)	85.9 (± 0.8)	87.8 (± 0.5)

respect to ISVC-4 classes.

Moreover, we note that in the ISVC scenario the two prototype-based exemplar-free methods (FeTrIL and EFC) achieve superior results compared to exemplar-based approaches (with memory size $K = 500$). This outcome underscores the efficacy of prototypes as a viable solution when it is not possible to store exemplars from previous tasks. Notably, EFC emerges as the top-performing method, reducing the performance gap in image classification with Joint Incremental by approximately 14%.

Closing The Gap with Joint Incremental Training. We investigate the impact of memory size K on exemplar-based approaches and compare performance with FeTril and EFC, which are exemplar-free but achieved competitive results in the ISVC scenario. Moreover, we examine how far current incremental learning solutions are from joint incremental training (**UB**). In Figure 4.3, we give the performance of all approaches in all the SNI scenarios for K ranging from 100 to 2000.

Difference in results between ISPC 14-class and ISVC 15-class scenarios is to be expected. Indeed, while the number of classes is comparable in both scenarios, there is a key distinction. In the first case, each class represents an entirely different social network, whereas in the second case multiple classes represent different versions of the same platform. Moreover, while ISPC 14-classes only accounts for a single

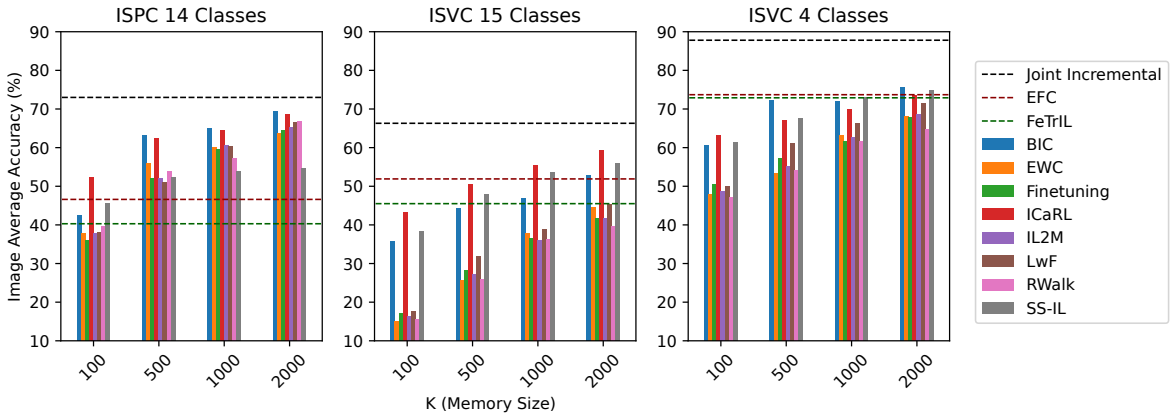


Figure 4.3: Image average accuracy as a function of memory size in the ISPC and ISVC scenarios.

dataset with fairly homogeneous data, ISVC 15-classes uses a mix of four different datasets acquired using multiple devices and following different protocols. iCaRL and BIC consistently outperform the other approaches in all the considered scenarios. SS-IL obtains competitive performance in ISVC scenario, while it obtains poor results for the ISPC scenario. Our conjecture is that this outcome can be attributed to the task-wise knowledge distillation of SS-IL, which could potentially necessitate a greater number of exemplar samples for effective learning in this context.

Results show that exemplar-based approaches are capable of reaching performance comparable to that of Joint Incremental in both the ISPC 14-class scenario and ISVC 15-class scenario while retaining only a fraction of the examples (see Table4.4 and Table4.5). However, there is a larger performance gap observed in the ISVC 4-class scenario compared to the other scenarios. EFC and FeTrIL perform worse than exemplar-based approaches when the memory size increases, however they still achieve competitive performance in the ISVC 4-class scenario. Finally, it is worth noting that iCaRL performs exceptionally well when the memory is limited to storing only $K = 100$ image patches. This highlights the effectiveness of iCaRL even with a significantly reduced memory size.

4.6 Conclusions

In this chapter we introduce incremental learning for social network identification. We considered two practical situations where updating an existing model would be valuable: Incremental Social Platform Classification, which involves accommodating newly introduced platforms, and Incremental Social Version Classification, which entails handling updated versions of existing social networks. To evaluate the effectiveness of incremental updating, we conducted extensive experiments with exemplar-free and exemplar-based incremental learning methods to incrementally

update a state-of-the-art network.

Remarkably, exemplar-free methods based on prototypes provide a viable solution when saving previous tasks exemplars is not feasible, for instance due to privacy concerns. Exemplars-based approaches achieve the largest improvement over fine-tuning in all considered scenarios by retaining only a fraction of the original training patches. Even though incremental learning methods are not yet able to reach the performance obtained by Joint Incremental Training, the reported results shows that recent techniques are rapidly closing the gap with the upper bound. This extensive evaluation serves as an initial benchmark, providing a foundation for researchers to further explore incremental social network identification in their studies.

As future work, exemplar-based methods could be further improved by employing a patch selection strategy based on the distribution of DCT coefficients. Moreover, considering the large gap between joint incremental and incremental learning approaches in the ISVC scenario, we hypothesize that there exist features shared across different tasks that are not currently taken into account by exemplar-based methods. To address this, future research efforts could concentrate on expanding incremental learning approaches to identify and incorporate these inter-task features.

Chapter 5

Vehicle Viewpoint Estimation from Monocular Images

Vehicle viewpoint estimation consists in estimating the vehicle azimuth or yaw angle, i.e., the rotation of the vehicle around the axis perpendicular to the road plane with respect to a reference point of view, like a camera (see Figure 5.1). Vehicle viewpoints enrich the semantic information about the road scene provided by traditional perception tasks, such as object detection, semantic segmentation and depth estimation, and lead to a better understanding of the road scene [144, 145]. Furthermore, viewpoint estimation enables accurate prediction of future vehicle motion [146], thus it is of considerable interest for applications in which prediction or classification of road maneuvers are key, like driver warning systems and risk estimation in the context of fleet management [147].

Convolutional Neural Networks (CNNs) are widely used for viewpoint estimation, but training them requires a substantial volume of annotated data. This data is often collected using a full suite of sensors, such as *Light Detection and Ranging* (LiDAR) or multiple cameras [148]. LiDAR sensors are excellent for obtaining high-quality annotations, enabling CNNs to achieve state-of-the-art results in determining vehicle orientation [149]. However, the high cost associated with LiDAR sensors presents a notable disadvantage compared to more economical vision-based instruments, such as monocular cameras (e.g. dash-cams). As a result, LiDAR-based solutions for vehicle viewpoint estimation, despite the high accuracy, are not economically viable for applications like fleet management, where equipping a large

---

Portions of this chapter were published in:

- [143] S. Magistri, M. Boschi, F. Sambo, D. C. de Andrade, M. Simoncini, L. Kubin, L. Taccari, L. de Luigi, and S. Salti, "*Lightweight and effective convolutional neural networks for vehicle viewpoint estimation from monocular images*," IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 1, pp. 191– 200, 2023.

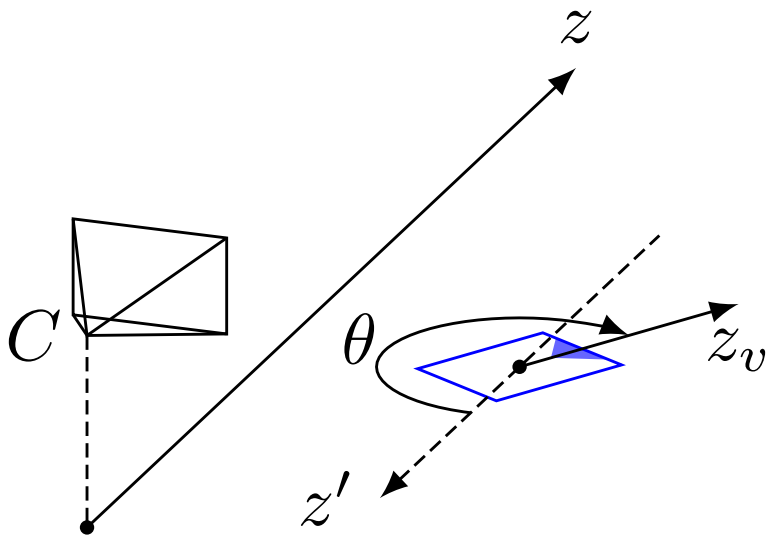
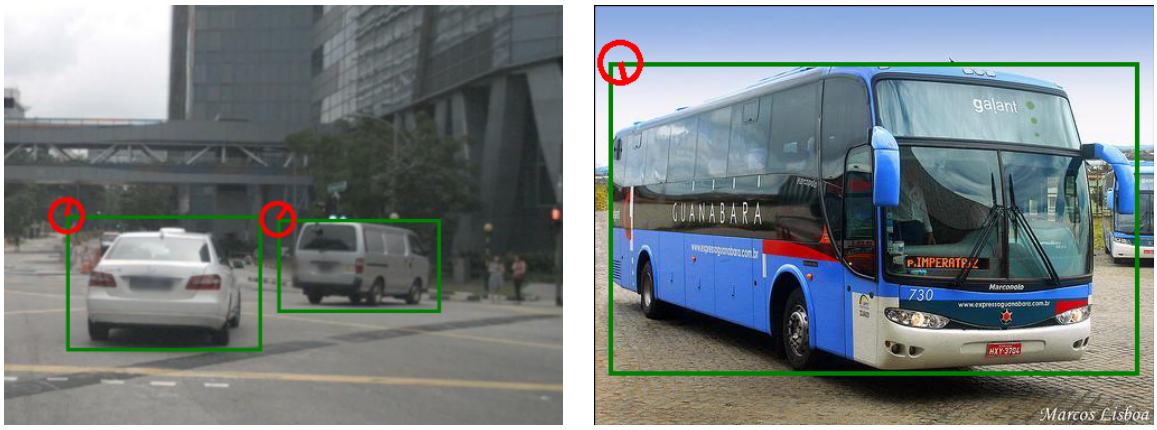


Figure 5.1: Definition of azimuth θ for the blue vehicle, whose front is denoted by the solid triangle. C is the camera and z is the direction of the optical axis on the road plane. z' is the global z translated so to pass through the blue vehicle center flipped to face towards the camera and z_v is pointing in the direction of the vehicle front from the vehicle center: θ is the angle from z' to z_v measured clockwise.

number of vehicles with expensive hardware is impractical and unsustainable.

Therefore, our reasearch is focused on developing a vehicle viewpoint estimation solution relying on a single *monocular camera*, which is mounted inside the windshield of a moving vehicle. This approach is more cost-effective and suitable for after-market fleet-management applications. Estimating viewpoint of vehicles from a single monocular image introduces inherent challenges, including radial distortion, motion blur, occlusion and the absence of depth information. Moreover, in a realistic scenario, cameras are not mounted exactly in the same spot for every vehicle, leading to non-fixed points of view and lack of some extrinsic camera parameters, namely camera height, roll and pitch. In light of this, a solution that generalizes to multiple vehicle makes and models and that is independent of the camera mounting position is required.

In this chapter, we build upon our previous work [150], in which we developed an efficient CNN-based model for predicting a vehicle's viewpoint from its image. Extending this research, we introduce *Lightweight Convolutional Neural Networks*, specifically designed for estimating a vehicle's viewpoint using monocular cameras. These models process not just the vehicle's image, but also its 2D bounding box information. We assume the 2D box information is externally provided to our models, separating the viewpoint estimation task from the vehicle detection task. This approach is justified by the fact that training a detector jointly with a viewpoint estimator results in a model that is both less versatile and more challenging to



(a) nuScenes. (b) Pascal3D+.

Figure 5.2: Samples from nuScenes (5.2a) and Pascal3D+ (5.2b) datasets. The viewpoint of each vehicle (i.e. azimuth or yaw angle) is identified by the red circle placed in the top-left corner of each bounding box.

maintain or adapt.

Our developed models incorporate geometric constraints into both the training loss and the model output, achieving state-of-the-art performance in the vehicle viewpoint estimation task. Thanks to the low number of parameters and reduced inference time, our models are suitable for deployment on edge devices. To validate our approach, we conduct extensive evaluations on a well-known viewpoint estimation benchmark, Pascal3D+ [151], and on nuScenes [148] dataset, a large-scale autonomous driving dataset. Examples from these datasets are presented in Figure 5.2.

5.1 Related work

The existing literature on viewpoint estimation starting from monocular images can be broadly classified into two categories: methods that rely on 3D object detection and those based on 2D object detection. 3D object detection solves the task of defining a 3D bounding box around an object within a 2D image [152, 153]. This approach allows the extraction of several information, including object localization, viewpoint and depth details. However, the main problems concerning these methods are the need to estimate depth or 3D information and the requirement of accurate camera parameters for correct localization. Furthermore, they may require higher computational resources if compared to 2D object detectors because of extra computation for 3D position estimation [154].

On the other hand, methods for vehicle viewpoint estimation based on 2D detection techniques rely on existing 2D bounding boxes to predict azimuth [155] or they

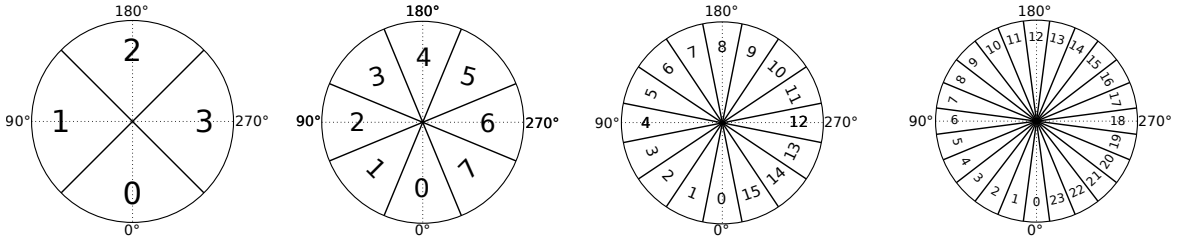


Figure 5.3: Standard Evaluation Protocol of Pascal3D+. Viewpoints (or azimuth) bins are grouped into 4, 8, 16 and 24 bins, representing coarser to finer quantization scales. The frame of reference assumes slice 0 facing towards the camera. The final classification accuracy is reported for all the quantization scales.

jointly train an object detector with a viewpoint estimator network [156]. A significant portion of the literature addresses the viewpoint estimation problem using a classification approach, which was shown by Massa et al. in [155, 157] to outperform the regression one. Ghodrati et al. [158] leverage image features extracted by CNNs to estimate a discretized object viewpoint. Tulsiani and Malik [159] directly estimated the viewpoint from the object images, using a CNN based on VGG16 architecture [160], taking into account the object class. Zhou et al. [161] reduce the prediction error of [159], changing the convolutional backbone to the more recent residual architecture ResNet18 [14]. Su et al. [162] introduced a discretization into 360 bins for viewpoint prediction and they proposed a geometric structure aware loss function. Divon and Tal [156] proposed a CNN-based architecture that jointly solved detection, classification and viewpoint estimation, introducing a loss function based on the idea of the Siamese Networks [163]. Magistri et al. [150] proposed a Multi-Task loss taking into account viewpoint errors at different granularities. More recently, Xiao et al. [164] used a class-agnostic CNN to estimate object poses and introduced a contrastive learning method based on a contrastive loss and pose-aware data-augmentations to improve the performance.

5.2 Lightweight and Effective Convolutional Neural Networks For Vehicle Viewpoint Estimation

In this section, we present the proposed methodology for addressing the challenge of estimating vehicle viewpoints from monocular images. We specifically focus on the viewpoint problem by decoupling it from the vehicle detection task, assuming the availability of detection boxes. This assumption is made because training a vehicle detector alongside a viewpoint estimator often results in a model that is less versatile and more challenging to mantain or adapt.

The current literature has predominantly addressed the problem of viewpoint estimation as a classification problem instead of regression one, as demonstrated to

be superior by Massa et al [155, 157]. Therefore, in this chapter, we delve into this classification-based approach.

Naively tackling this challenge involves training a standard CNN, such as Resnet-50 [14], on vehicle images. The CNN extracts vehicle features, which are then fed to a classification layer, characterized by a fully connected layer with 360 output units and a Softmax function generating viewpoint probabilities. During training, the loss function employed is a supervised cross-entropy loss, comparing discretized ground-truth azimuth labels with the corresponding viewpoint probabilities. However, this approach has three main limitations:

1. **Image Distortion and Truncated Vehicles**. CNN performance is adversely affected by standard image-processing that does not preserve aspect ratio, and it performs poorly on partially visible (truncated) vehicles, as discussed in [165].

   Convolutional neural networks take fixed-dimensional images as input, which leads to a common pre-processing practice of resizing images to fixed dimensions. However, when images are resized without preserving their original aspect ratio, the content of vehicle images may become distorted, posing a challenge for accurate viewpoint estimation.

   Furthermore, addressing the inherent challenge of truncated vehicles is complicated due to the limited presence of truncated vehicle samples in existing literature datasets. Consequently, such samples are under-represented during CNN training, leading to a deacrease in performance of truncated vehicle viewpoint estimation. Enhancing performance in estimating viewpoints for partially visible vehicles is crucial for various safety-critical applications, including car crash identification, given that truncated vehicles are often the closest to the camera's field of view and thus demanding high-performance solutions.

2. **Intrinsic Ambiguity of Image Appearance Information**. Relying solely on appearance information is insufficient to predict viewpoint. The vehicle's appearance strongly depends on its distance from the camera's optical axis. For instance, vehicles placed in the same direction as the camera's optical axis but at varying distances exhibit identical azimuth labels, despite having different appearances. This concept is illustrated in Figure 5.6. Providing the position and the size of the vehicle in the camera frame and thus implicitly the distance of the vehicle from the camera's optical axis can further improve the performance.

3. **Geometric Constraints**. The standard cross entropy loss is unaware of geometric constraints on viewpoints: nearby viewpoints should be more correlated than more distant ones. In a classification approach, continuous view-

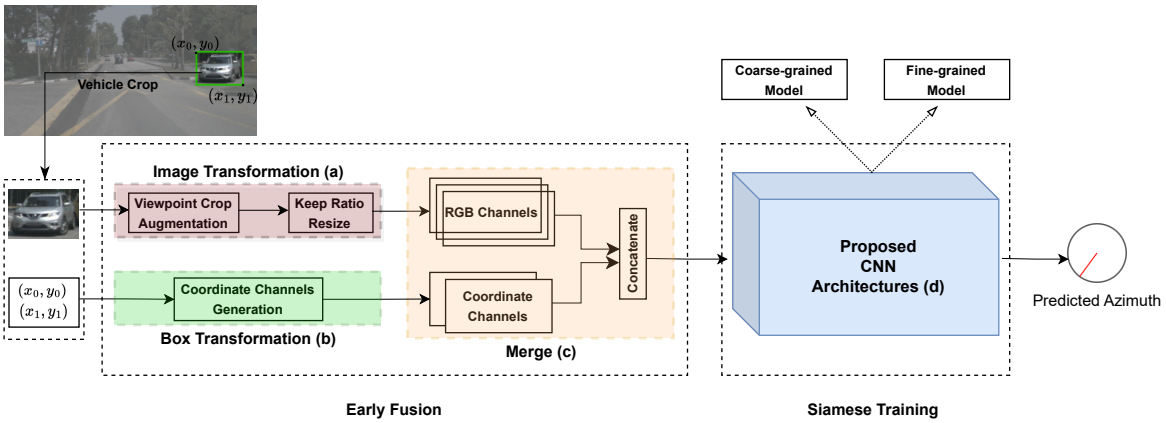


Figure 5.4: Proposed method. A vehicle image is cropped from the whole camera frame and, along with its bounding box coordinates, is pre-processed by an Early Fusion pipeline (see Section 5.3). The image undergoes two transformations (a) (see Section 5.3.1): the *viewpoint crop* augmentation, designed to improve performance on partially visible vehicles, and the *KeepRatio* resize, which resizes the vehicle image to a fixed dimension while preserving the original aspect ratio. The bounding box coordinates are transformed into two additional channels using a *Coordinate Channels Generation* method (b) based on CoordConv representation (see Section 5.3.2). Subsequently, the image and coordinate representations are merged (c) and fed into our *Proposed CNN Architectures* (d). Specifically, we propose two architectures, the *fine-grained* and the *coarse-grained* models (see Section 5.4), which are trained in a Siamese fashion (see Section 5.5) in order to predict vehicle azimuth.

points are quantized into a fixed number of bins, typically 360 bins to cover the full 360 degrees. However, this quantization process may result in nearby viewpoints being assigned to different bins, breaking down the natural order of bins and, consequently, impacting the performance. Moreover, a common practice for evaluating viewpoint models performance involves grouping the viewpoint space into larger bins, as outlined in the standard evaluation protocol of Pascal3D+ (Figure 5.3). This means that, unlike the traditional classification paradigm, in situations where the network fails to classify the correct viewpoint, it is preferable that the model predicts a viewpoint closer to the ground truth rather than one that is faraway.

Taking into account the above considerations, we have designed *lightweight* and *effective* CNNs for vehicle viewpoint estimation. Our methodology, depicted in Figure 5.4, involves cropping the vehicles according to their bounding box information from the whole camera frame. Subsequently, the images and the bounding box coordinates are passed through an early fusion pipeline (see Section 5.3), working on both the vehicle image and its bounding box coordinates.

The vehicle image is augmented using a *viewpoint crop augmentation*, performing

a random crop targeted to improve performance on truncated vehicles, and it is resized while preserving the original aspect ratio (see Section 5.3.1). The bounding box coordinates, denoted as $(x_0, y_0), (x_1, y_1)$, are modified via a *Coordinate Channel Generation Method* (see Section 5.3.2) to become two 2D matrices, which can be concatenated to the vehicle RGB channels and fed into CNN-based models.

We propose two CNN-models based on the MobileNetV2 feature extractor [166], which thanks to its reduced number of parameters, makes the overall method *lightweight*. The proposed models, named *fine-grained* and *coarse-grained*, are designed to address geometric constraints that viewpoint prediction should be subject to. Specifically, they employ smoothing techniques aimed at increasing correlation among nearby viewpoints (see Section 5.4). Moreover, they are trained using a *Siamese* approach to enforce these constraints (see Section 5.5). In the next sections, we provide additional details on each step of the proposed approach.

5.3 Early Fusion of Vehicle Images and Detection Data

In this section we describe the early fusion approach for pre-processing both the vehicle images and the box coordinates (see Figure 5.4 (a), (b)). After the pre-processing, the image and the coordinate channels generated are concatenated (see Figure 5.4 (c)) and fed inside the proposed CNN architectures.

5.3.1 Image Transformation

Viewpoint crop augmentation. Traditional approaches on viewpoint estimation are highly sensitive to truncated vehicles [165, 159]. In the context of vehicle viewpoint estimation from a single on-board monocular camera, vehicles can be heavily truncated, since they can be really close to it or placed on the margins of the camera field of view. For this reason we propose a task-specific data augmentation for vehicle viewpoint estimation, that we name *viewpoint crop*, with the aim of improving performance on truncated vehicles. The proposed data augmentation is performed online during training and chooses with probability 25 % one of four cropping options, each designed to simulate different truncations of vehicles captured by a moving camera:

1. The vehicle is directly in front of the camera and very close to it (Figure 5.5(a)), simulated by cropping only the image bottom side;

2. The camera is very close to a vehicle placed on its left side, i.e. part of the vehicle is just beyond the left edge of the field of view of the camera (Figure 5.5(b)), simulated by cropping the left and bottom image side.

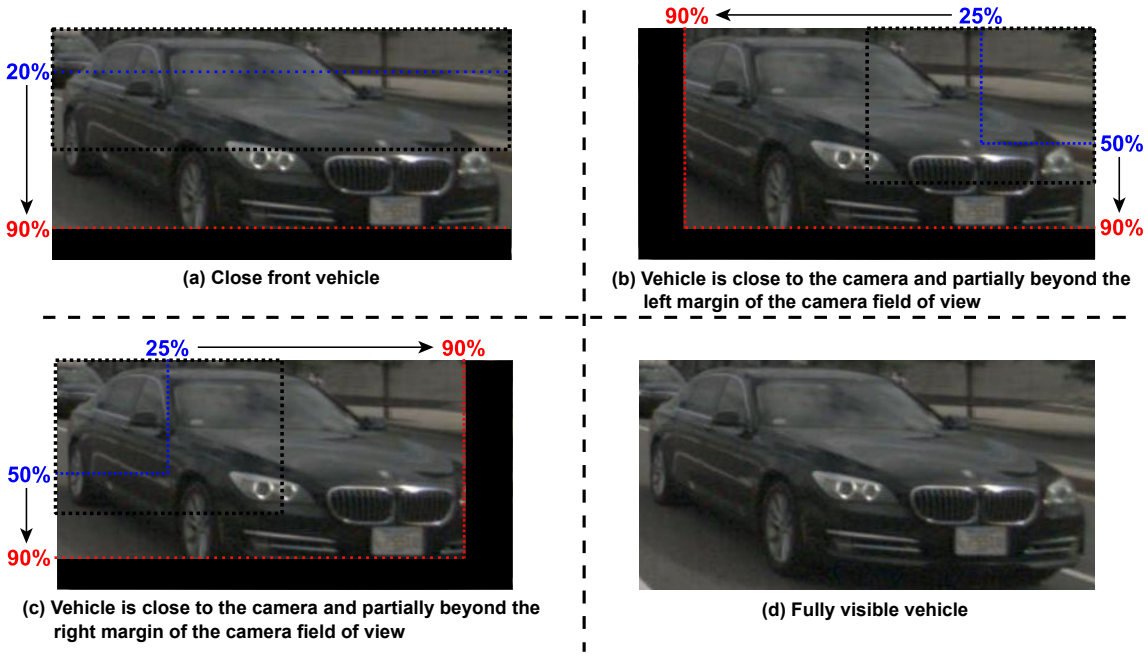


Figure 5.5: The *viewpoint crop* data augmentation. The blue and the red dashed lines represent respectively the minimum and the maximum dimensions of the crop for each option, while the black dashed line represents a possible crop applied during training.

3. The mirrored version of the previous case, simulating vehicles on the right side of the camera (Figure 5.5(c)).

4. The vehicle is fully visible (Figure 5.5(d)).

The crop target sizes are extracted as $w \sim \mathcal{U}(w_{min}, w_{max})$ and $h \sim \mathcal{U}(h_{min}, h_{max})$, where $\mathcal{U}$ stands for the uniform probability distribution and w_{min}, h_{min}, w_{max}, h_{max} represent respectively the lower and the upper bounds of the percentage of the vehicle original width and height. In Figure 5.5, we provide the bounds of each cropping option.

KeepRatio Resize. Convolutional Neural Networks typically operate under the assumption of fixed image sizes. For instance, when training CNNs on the Imagenet dataset for image classification [114], a standard practice is to resize images to a fixed resolution of 224×224 pixels, forming a square. In our specific case, resizing vehicle crops extracted from full images to 224×224 pixels without considering their original aspect ratio may lead to content distortion and pose challenges in accurately estimating their viewpoint.

Hence, we propose a different approach, referred to as *KeepRatio*: the images are resized so that the largest side is 224 pixels and zero-padding is subsequently

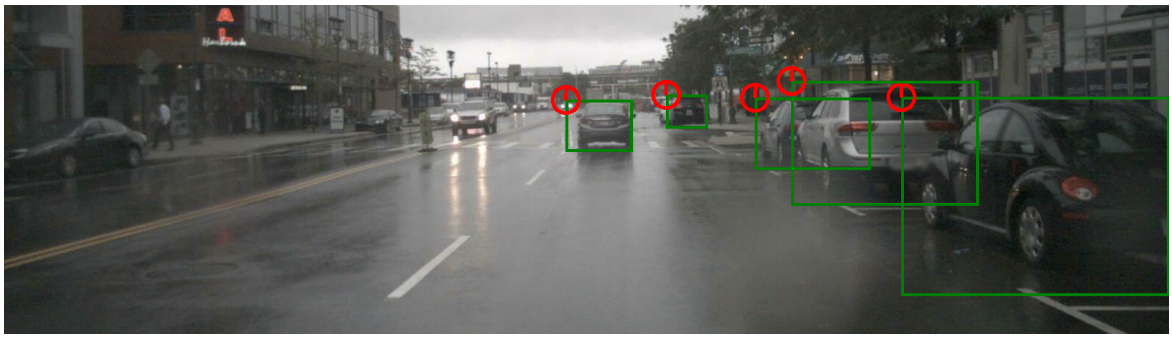


Figure 5.6: Example from the nuScenes dataset of the intrinsic ambiguity of viewpoint and appearance. Vehicles at different horizontal distances from the camera's optical axis may share the same azimuth but exhibit distinct appearances. The vehicle positioned in the center of the image displays only its rear, whereas for vehicles on the right side of the image, both their rear and left-side are visible.

applied to center the image within the target 224×224 square. This method aims to maintain the original aspect ratio while ensuring compatibility with CNNs.

5.3.2 Box Transformation via Coordinate Channels Generation

Estimating viewpoint reasoning only on the appearance of the vehicle within the bounding box suffers of an intrinsic ambiguity. Indeed, a vehicle with the same azimuth angle with respect to the camera exhibit different appearance onto the image plane of the camera according to its horizontal distance with respect to the camera optical axis. For instance, a vehicle whose azimuth is 180°, i.e., moving in the same direction of the camera, will have only its rear visible in the image if it is in front of the camera, but both the rear and the left side if it is in the adjacent right lane [167] (see Figure 5.6).

To enable the network to estimate a consistent azimuth angle despite variations in appearance, our proposal involves providing the bounding box coordinates together with the cropped vehicle as input to the network. To achieve this, we modify the Coordinate Channels Generation method, which was first introduced in [168] for encoding image pixel coordinates, to encode the bounding box coordinates as additional input channels for CNNs. Specifically, to a color image represented as a 3D tensor with dimensions for channels, width and height, we add two additional channels C_x and C_y. The channel C_x encodes the X coordinates along the horizontal axis of the image, and C_y encodes the Y coordinates along the vertical axis of the image. Each channel is constructed using as values the coordinates of the elements themselves for that dimension, i.e., for C_x the first column will be filled with 0, the second with 1 and so on, while for C_y the first row is all 0, the second all 1, and so on; both coordinates channels are then normalized to $[-1, 1]$. The result is a tensor with five channels, three carrying color information and two for the coordinates.

In our context, we apply the coordinate channels generation method to the whole camera frame. The resulting C_x and C_y channels are cropped according to the bounding box coordinates and then are interpolated to match the dimension of the vehicle image after the *KeepRatio* resize, described in Section 5.3.1. Finally, these two extra channels are concatenated with the vehicle's RGB image (as shown in Figure 5.4(c)) and fed into a CNN. The input layer of this CNN is modified to include a CoordConv layer. The CoordConv operates like a standard convolutional layer, accepting a 5-channel tensor as input.

5.4 Proposed Convolutional Neural Network Architectures

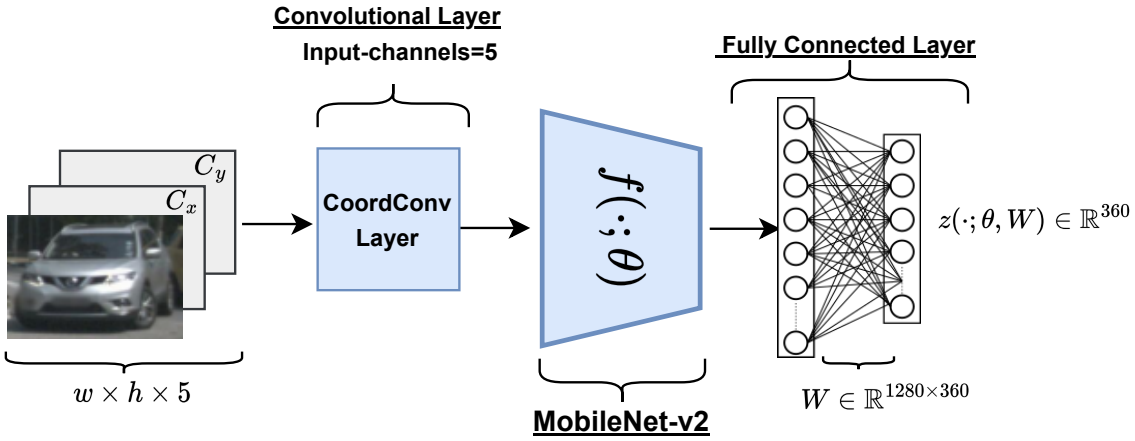
We propose architectures based on CNNs to tackle the problem of vehicle viewpoint estimation. These architectures accept a 5-channel tensor as input, with 3 channels dedicated to RGB image, preprocessed via the viewpoint crop and *KeepRatio* augmentations, and the remaining two channels C_x an C_y encoding coordinate data, obtained via the Coordinate Channel Generation method (see Figure 5.4 (d) for reference).

The developed architectures share a base network that includes a *CoordConv* layer [168], a feature extractor $f(\cdot;\theta)$ parameterized by weights θ and based on MobileNetV2 architecture, and a fully connected layer with weights $W \in \mathbb{R}^{1280\times360}$ (see Figure 5.7a). The CoordConv layer operates as a standard convolutional layer processing the 5-channel input. The output from this layer is then input into MobileNetV2, with the modification of omitting its initial convolutional layer, which is typically designed for three-channel inputs. This feature extractor outputs a 1D representation of a fixed size of 1280, which is then processed through a fully connected layer. This layer performs a linear transformation to produce the *logits* $z(\cdot;\theta,W) \in \mathbb{R}^{360}$, representing the network's output.

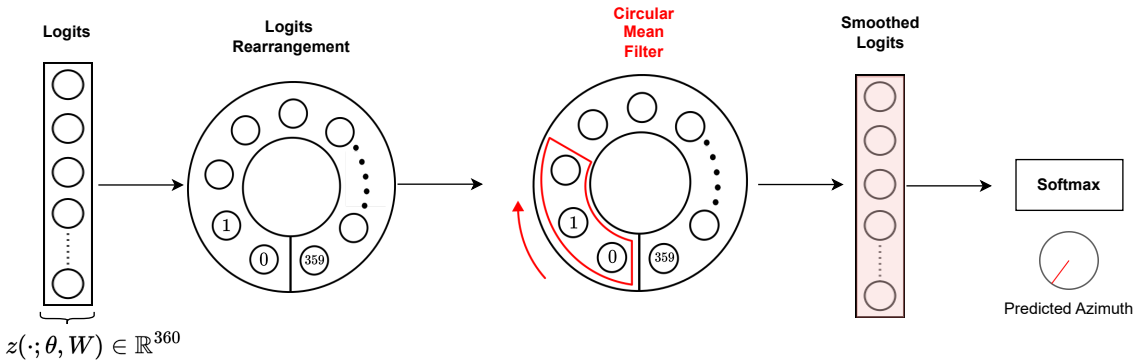
The models we propose, named as the *fine-grained* (refer to Figure 5.7b) and *coarse-grained* models (refer to Figure 5.7c), diverge in terms of the specific smoothing technique applied to the network logits. The purpose of these smoothing techniques is to improve correlation among nearby viewpoints. In the following sections, we provide a comprehensive description of these two models.

5.4.1 Fine-grained Model

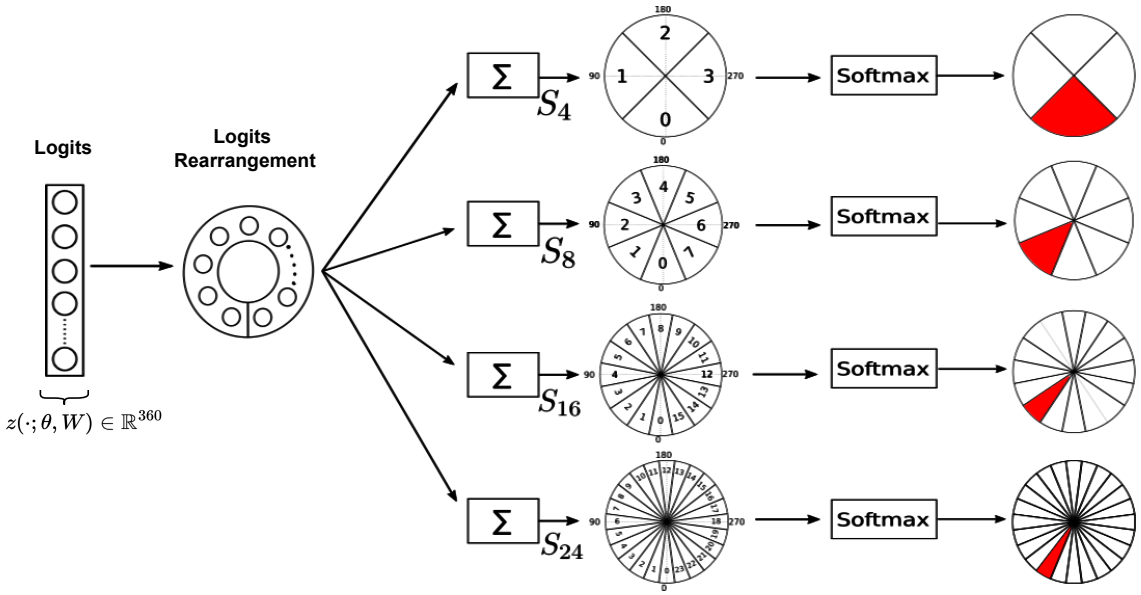
The fine-grained model, as illustrated in Figure 5.7b, utilizes a circular moving average filter with a stride of 1 and a window size of 15, on the network logits. We refer to it as *circular* because of the padding strategy employed, which involves values wrapping around at the edges. Given the logits $z(\cdot;\theta,W) = (l_0, l_1, \ldots, l_{359})$, a 7



(a) Base network shared by the two models.



(b) Fine-grained model. The model has a single head for azimuth angle prediction with 1-degree precision. Such precision is achieved by smoothing the logits with a circular mean filter and allows the predicted azimuth to be used as-is or discretized with large bins.



(c) Coarse-grained model. The output of the model consists of 4 heads, which sum in a circular fashion the network logits. Each head obtains a prediction at a different discretization level. S_α represents the summation performed for α bins with $\alpha \in \{4, 8, 16, 24\}$.

Figure 5.7: Proposed CNN architectures.

element wide padding is introduced as follows:

$$z^{\text{pad}}(\cdot;\theta,W) = (\underbrace{l_{353},\dots,l_{359}}_{\text{padding}},\underbrace{l_0,l_1,l_2,\dots,l_{359}}_{z(\cdot;\theta,W)},\underbrace{l_0,l_1,\dots,l_6}_{\text{padding}}). \tag{5.1}$$

The resulting vector contains $360+14$ elements. After the filter is applied, z^{pad} is reduced to a cardinality of $360+14-(15-1)=360$, which matches the original logits z. As a result, this model achieves azimuth prediction with an angular precision of 1°, hence earning its designation as *fine-grained model*.

The intuition behind the usage of a circular moving average filter comes from the error analysis of viewpoint estimation methods by Redondo-Cabrera et al. [165]. They show that a consistent portion of the errors made by the viewpoint models concentrates on nearby and opposite viewpoint predictions with respect to the ground truth viewpoints, i.e., the prediction error e usually satisfies either $15^\circ \leq e \leq 30^\circ$ or $e > 160^\circ$. In order to correct for these errors, we would like the viewpoint probabilities distribution of the network to be unimodal [169], i.e. the probability mass should gradually decrease on both sides of the viewpoint that has most of the mass and should be centered on the correct viewpoint. A circular mean filter has the objective of smoothing the 360 network logits in order to filter isolated peaks and to gradually decrease the probability mass function away from the overall highest logit.

The loss function used to train this model is the categorical cross entropy loss between the smoothed logits and the ground truth azimuth discretized into 360 sectors.

5.4.2 Coarse-grained Model

The *coarse-grained* model (see Figure 5.7c), inspired by the Multi-Task model proposed in our previous work [150], consists of four sums of the network logits, resulting in multiple output heads. The summations are designed to map the network logits to the four discretization levels defined in Figure 5.3. The summation for α bins, where $\alpha \in \{4,8,16,24\}$, can be thought of as a mono dimensional circular convolution with a unitary filter with window and stride fixed to $\lceil \frac{360}{\alpha} \rceil$. Similarly to the *fine-grained* model, applying these filters results in assigning less importance to isolated peaks in the predicted probability mass function.

The loss function adopted to train this model is made up of the sum of four categorical cross entropy losses between the ground truth azimuth discretized into 4, 8, 16 and 24 bins and the corresponding logits summation. This kind of loss is designed to take into account errors at different granularities, to smooth isolated peaks in the probability mass of the network probability distribution and to train a single model able to predict viewpoint at different quantizations.

5.5 Siamese Training

A Siamese network [163] consists of two or more copies of the same convolutional neural networks sharing learnable weights. This architecture is particularly effective for viewpoint estimation, since enhancing standard neural network training by adding geometrical constraints within the training loss.

In this research, we have adapted the Siamese loss, initially introduced by [156] for the joint training of a 2D object detector and a viewpoint estimator. We propose a Siamese approach for a model trained independently from a 2D object detector, taking both the bounding box and the vehicle image as input.

Formally, we consider the triplet (X, c, β) where X is the cropped vehicle image, $\beta \in [0, 360)$ is the azimuth label, and $c = (x_0, y_0, x_1, y_1)$ denotes the bounding box coordinates in the source image, using the standard top-left bottom-right coordinate system. During training, the network processes two images: X and its horizontally flipped version X_{flp}. Flipping X modifies both the bounding box coordinates and the ground truth azimuth label associated with X_{flp}. We define β_{flp} as the mirrored version of β on the Y-axis, and $c_{flp} = (w - x_0, y_0, w - x_1, y_1)$ as the flipped bounding box coordinates in an image of width w. Additionally, let define as C^x_{flp} and C^y_{flp} the extra channels obtained from c_{flp} via the Coordinate Channels Generation method, described in Section 5.3.2.

We propose the following Siamese training loss:

$$\mathcal{L}_s = \mathcal{L}(x, \beta) + \mathcal{L}(x_{flp}, \beta_{flp}) + \lambda D(z, flip(z^{flp})), \tag{5.2}$$

where:

- $x = X||C^x||C^y$ represents the concatenation of the RGB image with the C^x and C^y coordinate channels in the early fusion branch, as illustrated in Figure 5.4(c). Similarly, $x_{flp} = X_{flp}||C^x_{flp}||C^y_{flp}$ represents the concatenation of the flipped image X_{flp} with the flipped coordinate channels.

- $\mathcal{L}(x, \beta)$ and $\mathcal{L}(x_{flp}, \beta_{flp})$ denote the loss on x with ground-truth β and on x_{flp} with ground-truth β_{flp}, respectively. The specific implementation of these losses depends on the model used for training. For the fine-grained model, $\mathcal{L}$ is a categorical cross-entropy with the ground-truth azimuth discretized into 360-sectors (see Section 5.4.1). For the *coarse-grained* model, described in Section 5.4.2, the loss $\mathcal{L}$ is a summation of four categorical cross-entropy with the ground-truth azimuth labels discretized into 4, 8, 16 and 24 bins.

- $D : \mathbb{R}^{360} \times \mathbb{R}^{360} \mapsto \mathbb{R}$ is a distance function measuring the difference between the network output $z = z(x; \theta, W)$ on the input x and the network output $z^{flp} = z(x_{flp}; \theta, W)$ on the flipped input x_{flp}. Both outputs are parameter-

ized by shared feature extractor weights θ and head weights W. $\lambda \in \mathbb{R}$ is a hyperparameter scaling the distance term in the loss function.

The flip operator, defined as $flip : \mathbb{R}^{360} \mapsto \mathbb{R}^{360}$, maps $y = (y_0, y_1, \ldots, y_{359})$ to $flip(y) = (y_0, y_{359}, y_{358}, \ldots, y_1)$ i.e. it swaps logits between an angle β and its correspondent $-\beta$ under the horizontal flip while leaving the first logit, corresponding to 0°, fixed.

The rationale behind this loss function is to constrain the network such that the distance between its output on the input x and the *flip* of the network output on x_{flp} approaches to zero. In our evaluation, we used as D function either the squared ℓ_2 distance:

$$D(u, v) = \|u - v\|_2^2, \tag{5.3}$$

or the angular distance

$$D(u, v) = \frac{1}{\pi} \arccos \frac{u \cdot b}{\|u\| \, \|v\|}, \tag{5.4}$$

where $u, v \in \mathbb{R}^n$.

5.6 Experimental Results

In this section, we introduce the dataset and the experimental settings for our methodology. We proceed to compare our *fine-grained* and *coarse-grained* models (introduced in Section 5.4) with state-of-the-art approaches. Subsequently, we perform an ablation study to assess the impact of each component of our methodology: the early fusion of vehicle image and detection data (Section 5.3) and the Siamese training loss (Section 5.5).

5.6.1 Dataset

The nuScenes dataset [148] is a large-scale autonomous driving dataset that encompasses a comprehensive suite of vehicular sensor data, along with 3D bounding box and viewpoint annotations for vehicles in road scenes. Each image in the nuScenes dataset is captured from one of six cameras mounted on a moving vehicle at various positions: front, front-right, front-left, back-right, back-left, and back. Consequently, the elevation and tilt angles of the vehicles in these images remain relatively consistent across each image.

To amass a substantial collection of images from nuScenes, we used vehicle images captured by all six cameras. Since nuScenes provides only 3D bounding boxes for vehicles, we first projected these onto the image plane to create 2D bounding boxes, and then cropped the vehicle images, retaining the azimuth annotation. For

Table 5.1: Train, validation, and test distribution for Pascal3D+ and nuScenes datasets. The percentage of vehicle in each split and the total number of images is reported.

		%					
Dataset	**Split**	**Bike**	**Bus**	**Car**	**Motorbike**	**Truck**	**Total**
nuScenes	Train	1.7	3.0	74.9	1.7	18.6	$\sim$ 400k
	Valid	2.4	3.4	74.8	2.3	17.1	$\sim$ 35k
	Test	2.1	3.5	72.8	2.2	19.5	$\sim$ 90k
Pascal3D+	Train	14.9	11.9	52.5	13.9	6.8	$\sim$ 10k
	Valid	19.8	14.1	46.1	17.5	2.5	$\sim$ 1k
	Test	18.6	14.7	44.4	17.8	4.5	$\sim$ 2k

training and testing, we selected only those bounding boxes containing sufficient data to accurately infer orientation, in accordance with the criteria outlined in [150]. The official training set of nuScenes was divided, with 650 road scenes allocated for training and 50 for the validation set. We used the official validation set as our test set.

The Pascal3D+ dataset [151] is a general object viewpoint estimation dataset that includes 12 object classes. It comprises samples from Pascal VOC 2012 [170] and a subset of ImageNet [114], each enriched with 3D annotations (azimuth, elevation, and tilt). In Pascal3D+, images are captured by a single fixed camera at various positions, resulting in significant variations in elevation and tilt of objects across different images. For our purpose, we focus exclusively on vehicle objects. We used the provided 2D bounding boxes to extract tight crops around the vehicles and retrieve their azimuth angle. Furthermore, we manually identified and labeled trucks in the Pascal3D+ dataset to maintain consistency with the nuScenes vehicle classes.

For the Pascal3D+ dataset, we split the official training set into training and validation subsets, dividing it 50/50 and stratifying by vehicle type. The official validation set is used for testing, including vehicles annotated as *difficult*, *occluded*, and *truncated*.

Table 5.1 provides the statistics of nuScenes and Pascal3D+ datasets, stratified by vehicle type.

5.6.2 Metrics and Hyperparameters

Our experimental results are reported on the nuScenes and Pascal3D+ dataset. For each model, we measure the *accuracy per vehicle* class, the *average accuracy* across

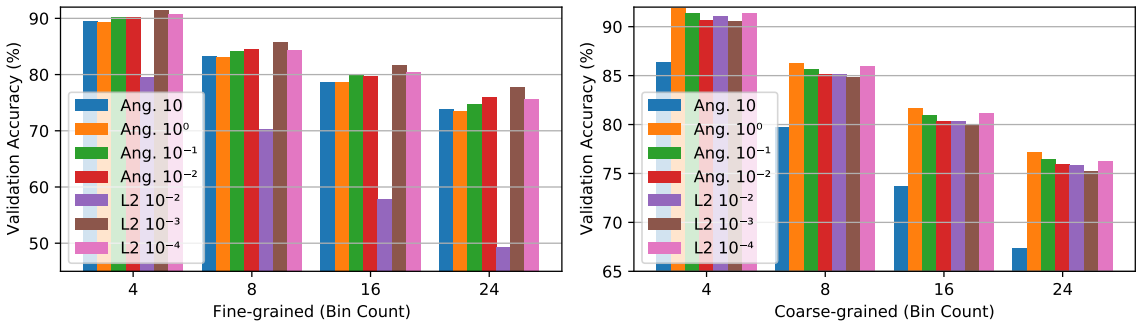


Figure 5.8: Performance of the *fine-grained* and *coarse-grained* models on the nuScenes validation set varying the distance type and λ value of the Siamese training loss (Equation5.2).

vehicle classes and the *overall accuracy* for each discretization level (i.e. 4, 8, 16, 24 bins) on the test set.

Training settings. For both the fine-grained model and the *coarse-grained* model, we use MobileNetV2 [166] as convolutional bakcbone, starting from pre-trained ImageNet weights. The models are trained using Adam [89] as optimizer with learning rate 10^{-3} and weight decay set to 10^{-4}. During training, the learning rate is reduced by a factor 0.1 when no improvement on the overall accuracy on the validation set for each discretization level is detected for three epochs. Each training is performed setting a maximum number of epochs to 100 with early stopping when the learning rate reaches 10^{-5} and the validation set accuracy stops to improve.

The images fed into the models are resized to 224×224 pixels using the *KeepRatio* and *viewpoint crop* image augmentations, described in Section 5.3.1. As an additional data augmentation step, we apply random horizontal flipping with probability 50 % to all the training images and their respective coordinates, except when the Siamese approach is employed.

Siamese hyperparameters. We evaluated different λ values and distance type for the Siamese training loss (Equation 5.2). Since the distance metrics have different scales, we perform a grid-search over two different search ranges: $\lambda \in \{10^k\}, k \in \{-2, -1, 0, 1\}$ for the angular distance and $\lambda \in \{10^k\}, k \in \{-4, -3, -2\}$ for the ℓ_2 distance. In our tests, illustrated in Figure 5.8), we verified that both our proposed models (i.e., *coarse-grained* and *fine-grained*) are pretty robust with respect to these two parameters, except for the ℓ_2 distance with $\lambda = 10^{-2}$ that involves a significant performance drop when used to train the *fine-grained* model. From this analysis we fix the distance type and λ to the values gaining the best performance on the validation set, i.e., the angular distance with $\lambda = 10^0$ for the *coarse-grained* model and the ℓ_2 distance with $\lambda = 10^{-3}$ for the *fine-grained* one, which gained the best performance on the validation set.

Table 5.2: State of the art comparison on **nuScenes** dataset. We report the accuracy per vehicle class, the average accuracy across classes and the overall accuracy."ResNet18 CS" refers to "ResNet18 Class Specific" [161, 159].

		%						
Bins	**Architecture**	**Bike**	**Bus**	**Car**	**Motorbike**	**Truck**	**Avg**	**Total**
4	Render for CNN [162]	57.35	79.59	89.92	66.65	79.49	74.60	86.33
	ResNet18 CS [161, 159]	59.95	82.80	90.95	73.57	81.23	77.70	87.74
	PoseContrast [164]	64.36	85.19	91.80	71.16	83.79	79.26	88.98
	Multi-Task [150]	65.66	**86.20**	92.26	73.36	83.65	80.23	89.40
	Coarse-grained Model	65.56	84.93	**93.83**	**74.23**	84.66	80.64	90.71
	Fine-grained Model	**66.23**	85.47	93.77	73.67	**85.02**	**80.83**	**90.75**
8	Render for CNN [162]	40.88	73.65	81.95	54.35	68.72	63.91	77.61
	ResNet18 CS [161, 159]	49.87	76.86	83.11	60.40	71.05	68.26	79.35
	PoseContrast [164]	48.36	81.21	85.35	60.81	75.10	70.17	81.89
	Multi-Task [150]	54.75	80.61	85.46	62.40	75.24	71.69	82.15
	Coarse-grained Model	**55.74**	**82.36**	89.38	**66.03**	79.58	**74.62**	86.01
	Fine-grained Model	**55.74**	82.04	**89.81**	63.78	**79.77**	74.23	**86.29**
16	Render for CNN [162]	29.87	62.62	71.27	36.58	57.86	51.64	66.73
	ResNet18 CS [161, 159]	29.71	66.88	73.31	41.75	60.85	54.50	69.05
	PoseContrast [164]	37.82	70.15	74.88	45.39	64.04	58.46	71.18
	Multi-Task [150]	39.95	71.23	76.17	44.52	65.76	59.53	72.52
	Coarse-grained Model	**44.47**	74.86	82.60	**54.15**	70.64	65.34	78.58
	Fine-grained Model	44.16	**75.68**	**83.83**	51.33	**71.73**	**65.35**	**79.64**
24	Render for CNN [162]	22.81	56.07	64.96	27.36	50.52	44.34	60.13
	Resnet18 CS [161, 159]	21.82	59.98	67.38	31.97	53.57	46.94	62.70
	PoseContrast [164]	26.08	65.03	70.73	37.19	59.10	51.63	66.59
	Multi-Task [150]	29.19	63.57	70.11	35.45	58.09	51.28	65.92
	Coarse-grained Model	34.81	72.57	79.03	43.55	67.07	59.41	74.77
	Fine-grained Model	**36.99**	**73.33**	**80.42**	**44.77**	**67.92**	**60.70**	**76.05**

5.6.3 Comparison with the State-of-the-art

We assess the performance of our models on Pascal3D+ and nuScenes, comparing them with ResNet18CS [159, 161], Render for CNN [162], PoseContrast [164] and our previous Multi-Task model [150]. ResNet18CS is the model defined by Tulsiani and Malik [159] but modified to use Resnet18 as feature extractor. This adaptation leads to a slight improvement in performance, as shown by Zhou et al. [161]. Both our *coarse-grained* and *fine-grained* models use MobileNetV2 as feature extractor due to its superior efficiency and accuracy on vehicle viewpoint estimation, when compared to other feature extractors, such as DenseNet and ResNet. Further details are provided in the upcoming Section 5.6.4.

To facilitate the comparison with state-of-the-art approaches, we use available

Table 5.3: State of the art comparison on **Pascal3D+** dataset. We report the accuracy per vehicle class, the average accuracy across classes and the overall accuracy."ResNet18 CS" refers to "ResNet18 Class Specific" [161, 159].

		%						
Bins	**Architecture**	**Bike**	**Bus**	**Car**	**Motorbike**	**Truck**	**Avg**	**Total**
4	Render for CNN [162]	68.5	81.5	68.4	73.2	N/A	72.9	74.3
	ResNet18 CS [161, 159]	73.5	82.9	71.3	76.1	67.4	74.3	74.1
	PoseContrast [164]	72.7	80.4	73.6	75.5	68.6	74.2	74.5
	Multi-Task [150]	71.6	**84.0**	**80.0**	74.9	**76.7**	77.5	78.0
	Coarse-grained Model	74.9	**84.0**	76.1	77.0	**76.7**	77.8	77.2
	Fine-grained Model	**76.3**	83.6	77.3	**80.8**	**76.7**	**79.0**	**78.7**
8	Render for CNN [162]	58.6	70.5	58.3	62.5	N/A	62.5	63.2
	ResNet18 CS [161, 159]	62.3	73.3	61.0	63.7	61.6	64.8	63.5
	PoseContrast [164]	60.6	**74.0**	66.0	65.5	54.7	64.1	65.6
	Multi-Task [150]	60.3	71.2	**70.2**	66.4	**68.6**	67.3	67.8
	Coarse-grained Model	64.8	72.6	67.6	65.8	60.5	66.3	67.2
	Fine-grained Model	**65.1**	71.5	68.7	**70.8**	65.1	**68.2**	**68.6**
16	Render for CNN [162]	39.4	63.3	46.5	45.1	N/A	48.6	49.7
	ResNet18 CS [161, 159]	45.4	65.8	48.8	44.8	38.4	48.6	49.5
	PoseContrast [164]	45.6	66.9	55.6	**50.2**	43.0	52.3	53.9
	Multi-Task [150]	42.5	68.0	57.9	42.8	**50.0**	52.2	53.5
	Coarse-grained Model	47.9	66.9	**58.5**	46.0	43.0	52.5	54.9
	Fine-grained Model	**48.7**	**68.7**	58.0	44.8	45.4	**53.1**	**55.0**
24	Render for CNN [162]	34.9	55.9	40.8	36.0	N/A	41.9	42.6
	ResNet18 CS [161, 159]	33.2	**60.9**	45.9	36.6	32.6	41.8	43.5
	PoseContrast [164]	37.8	60.1	50.0	37.2	37.2	44.5	46.4
	Multi-Task [150]	32.7	56.2	49.3	36.3	**41.9**	43.3	44.6
	Coarse-grained Model	38.3	59.4	**50.7**	33.0	40.7	44.4	46.1
	Fine-grained Model	**40.6**	60.5	49.2	**41.6**	40.7	**46.5**	**47.5**

pre-trained weights when possible, such as Render for CNN trained on Pascal3D+*. In cases where pre-trained weights are not available, we retrain the models only to predict the azimuth angle, maintaining the original training settings and employing our performance evaluation. When training both our models and state-of-the-art models on the nuScenes dataset, we enrich the training set with Pascal3D+ images, as in our preliminary tests we verified that this improves model generalization. Conversely, for training on Pascal3D+, we exclusively use images from the original dataset to ensure a fair comparison with state-of-the art approaches.

In Table 5.2 and Table 5.3, we presents our results for both Pascal3D+ and nuScenes. Our *fine-grained* model is the top performer among state-of-the-art approaches and

*This model was trained without the *truck* class as it is not part of the official Pascal3D+ dataset.

also shows a big improvement over our previous *Multi-Task* model [150], which did not incorporate the viewpoint crop, the *KeepRatio* pre-processing, the CoordConv layer and the Circular Smoothing Filter. The *coarse-grained* model is the second top-performing methods, except for 24 bins on Pascal3D+ where it obtains comparable performance with PoseContrast.

Considering Table 5.2 and Table 5.3, our models show considerable performance improvements on nuScenes compared to Pascal3D+, when measured against other methods. This enhanced performance is largely attributable to the CoordConv layer, which significantly boost performance on nuScenes but not on Pascal3D+, where vehicles are typically centered in the image (see the upcoming Section 5.6.4 for additional details). Notably, the impact of the CoordConv layer is more pronounced with finer quantization levels.

Finally, it is important to recognize that different types of vehicles pose different difficulty levels in correctly estimating the viewpoint as can be seen in Table 5.2 and Table 5.3. This is especially true for bicycles and motorbikes, for which is challenging to recognize the front from the rear, especially if the handlebar is occluded [156].

5.6.4 Ablation Study

In this section we discuss the impact on the performance of the proposed design decisions for the image transformations (Section 5.3.1), the early fusion of vehicle image with detection information via the Coordinate Channel Generation method (Section 5.3.2 and the proposed fine-grained and coarse grained model trained with the Siamese loss (Section 5.4 and Section 5.5). For this analysis, we use the nuScenes dataset, since it is more representative of the task we want solve, i.e., vehicle viewpoint estimation from road view monocular images

Image transformations: SquareResize vs KeepRatio. In Section 5.3.1, we outline that resizing an image to a square of 224×224 pixels can distort the content because the aspect ratio is not preserved. We refer to this approach as *SquareResize*. In Figure 5.9, we compare the performance of *SquareResize* against *KeepRatio*, which resizes the images while maintaining the original aspect ratio. The results shows that preserving the original aspect ratio using *KeepRatio* provides a consistent improvement in accuracy compared to *SquareResize*, for both fine-grained and coarse-grained models, especially on finer discretization levels.

Image transformations: Viewpoint crop. Moreover, in the same section, we develop viewpoint crop augmentation to improve the performance on vehicles that are closer to the camera and, therefore, potentially truncated. In Table 5.4, the results show that using *viewpoint crop* provides a noticeable, albeit small, improvement in the overall model's accuracy.

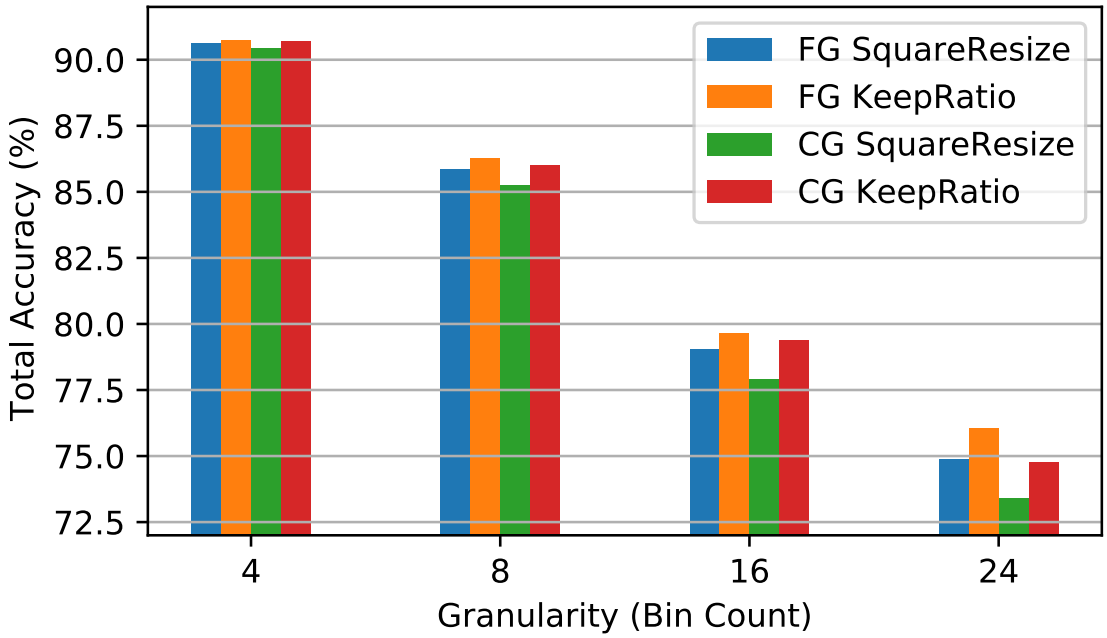


Figure 5.9: *KeepRatio* and *SquareResize*. We report the overall accuracy for both the *fine-grained* (FG) and the *coarse-grained* (CG) model on nuScenes. Both the models benefit from *KeepRatio* pre-processing, especially on finer discretization levels.

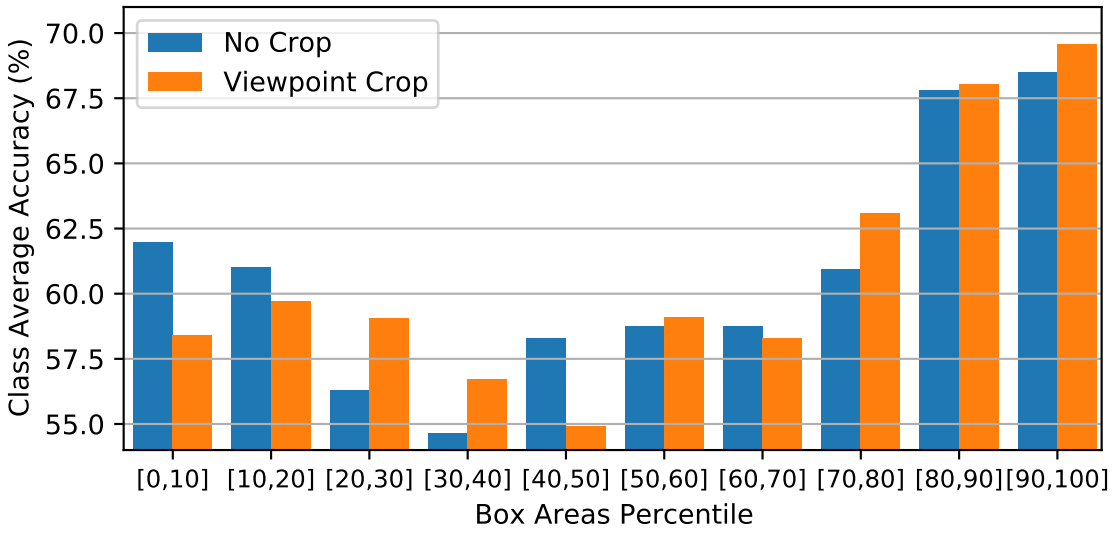


Figure 5.10: Viewpoint crop image tranformation. A bar pair in the plot represents the 24 bins average accuracy on the classes considering only the box areas included in the percentile range of the bin. Percentile values are computed for each vehicle class. This setup ensures that each bin contains one tenth of the whole dataset while preserving the class ratios, detailed in Table 5.1).

Table 5.4: Ablation study on Viewpoint crop, CoordConv, and Siamese loss on the nuScenes dataset. Vehicle class average and overall accuracies are reported

	Viewpoint crop	CoordConv	Siamese	Avg	Total
Coarse-grained (24 bin)	✓	✓	✓	**59.41**	**74.77**
	✗	✓	✓	59.30	74.09
	✓	✗	✓	51.18	65.69
	✓	✓	✗	55.88	73.08
Fine-grained (24 bin)	✓	✓	✓	**60.70**	**76.05**
	✗	✓	✓	**60.70**	75.82
	✓	✗	✓	51.26	66.49
	✓	✓	✗	55.18	72.59

This slight improvement is attributed to the scarcity of truncated vehicles in the nuScenes dataset, when compared to the overall number of vehicle images, making the impact of this augmentation not immediately evident. To further understand the benefits of viewpoint crop augmentation for truncated vehicles, we analyze the model's performance, stratifying by the area of the vehicle's bounding box in the nuScenes test set. This approach is based on our empirical finding that a larger box area is more likely to contain a cropped vehicle. In Figure 5.10, we compute the average accuracy for each vehicle class in the nuScenes test set, focusing on specific ranges of box areas. The results clearly show that viewpoint crop augmentation substantially enhances accuracy for larger boxes, which often represent nearby and possibly occluded vehicles. Enhancing model performance for nearby vehicles is particularly important for a wide range of applications, such as safety-critical scenarios like car crash identification, which typically demand high performance for vehicles that are closest to the camera.

Box Transformation: Coordinate Channels as Input. We discuss the usage of Coordinate Channels as input to our model via the CoordConv layer, as described in Section 5.3.2. In Table 5.4 we show the effect of the use of vehicle position in the original image as an additional input. Our experiments designate this as the most significant factor in improving the accuracy of the network, resulting in a notable increase of about 10% in total accuracy and about 9% average accuracy for both models.

However, we also recognize a limitation in using a CoordConv layer. This is because the network learns to associate specific coordinates with certain vehicle aspects, implying that a substantial change in camera positioning between training and testing can lead to a significant decline in performance. For instance, when testing our models trained on the nuScenes dataset on the Pascal3D+ dataset, the total accuracy for the fine-grained model at 24 bins was 35.71%, which is a decrease of ap-

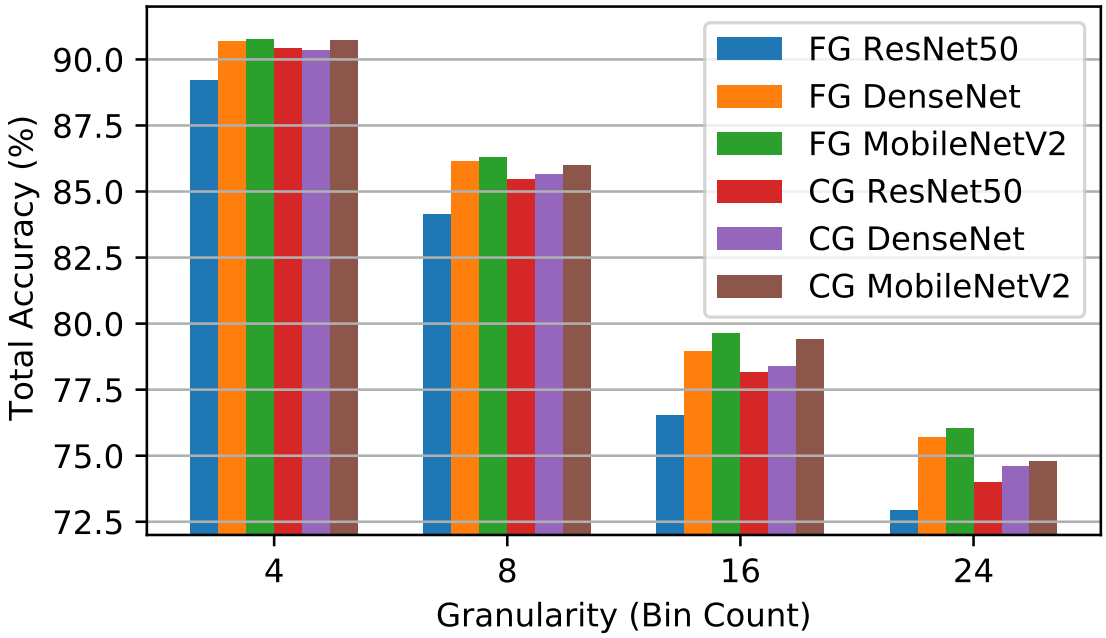


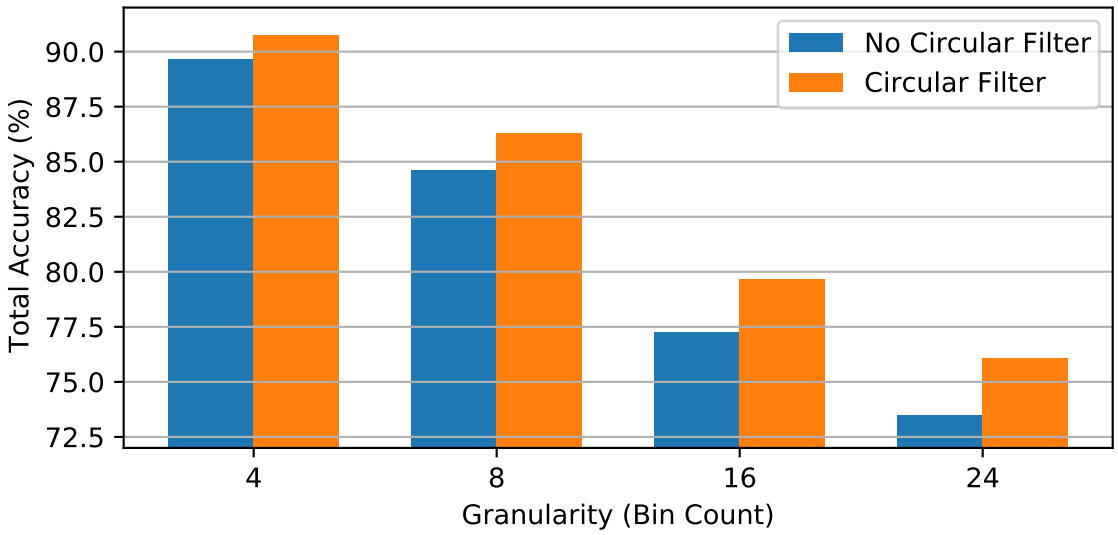
Figure 5.11: Performance of different backbones for the *fine-grained* (FG) and the *coarse-grained* (CG) model on nuScenes. MobileNetV2 obtain higher performance for both the fine-grained and the coarse-grained models for each discretization level. Higher improvement are achieved at finer bin granularities (16 and 24 bins)

.

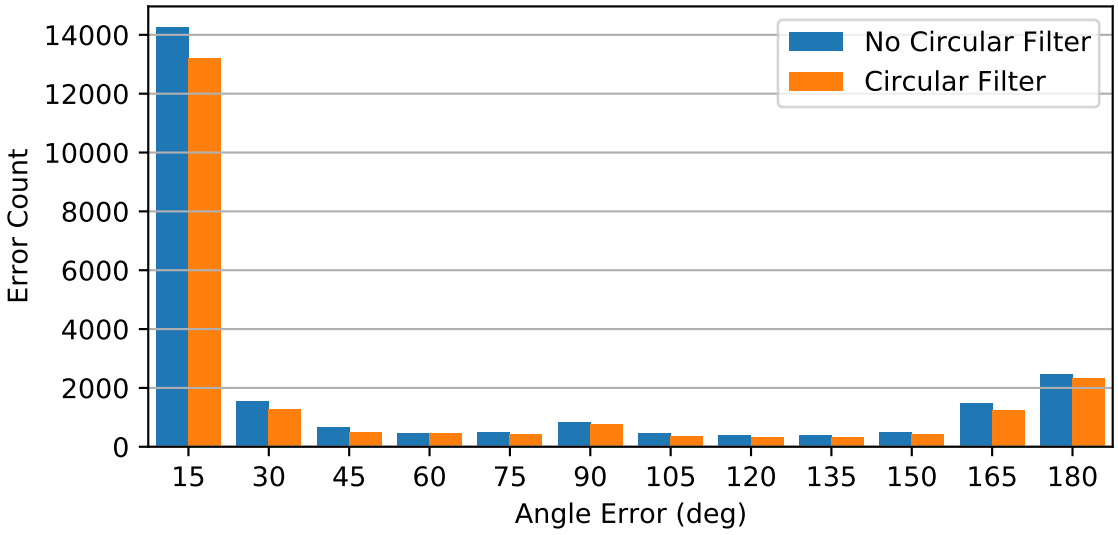
proximately 10% compared to models trained solely on Pascal3D+ (see Table 5.3). In contrast, training without the CoordConv layer on nuScenes resulted in a smaller drop in performance on Pascal3D+, achieving 44.68% accuracy. This underscores the limitation of using vehicle coordinates in the training process.

Siamese Loss. In Table 5.4, we demonstrate the effects of training using a Siamese loss. Both models benefit from this training method, showing a significant increase in both average and overall accuracy. The fine-grained model shows an approximate increase of 5% in overall accuracy and 4% in class average accuracy when trained with the Siamese loss. Meanwhile, the coarse-grained model exhibit a more modest improvement, with a gain of about 2% in overall accuracy and 4% in class average accuracy.

Feature Extractor and Logit Smoothing. Finally, in this section, we discuss the architectural choices of our models, including both the feature extractor and the logit smoothing techniques presented in Section 5.4. As regards the feature extractor, we consider three convolutional neural networks, with different size and structure, to evaluate the performance of both the *fine-grained* and *coarse-grained* models: ResNet50 [14], DenseNet [171] and the chosen MobileNetV2 [166]. In Figure 5.11



(a) Performance impact in terms of accuracy, higher is better.



(b) Distribution of errors granularity at 24 bins, lower is better.

Figure 5.12: The influence of the mean circular filter, using a window size of 15 and a stride of 1, on the fine-grained model's performance on nuScenes dataset. In Figure 5.12a we report the overall performance when the circular filter is both added and removed. In Figure 5.12b, we report the count of errors at various angular thresholds, measured as deviations from the ground truth.

we show the overall total accuracy of both models, varying the feature extractor. Our results show that, for each discretization level the performance drops when more complex models are used, which seems counter intuitive. We conjecture that, for this task, the reduced number of parameters helps to prevent overfitting, an hypothesis supported by the increasing accuracy when the size of the backbone decreases from the larger ResNet50 (24.2 M parameters) to DenseNet(7.3 M) and finally to the chosen MobileNetV2 (2.7 M).

Regarding the *logit smoothing* techniques employed, in our previous work [150] we demonstrated that summing the network logits, as done by the *coarse-grained* model, before predicting azimuth at different levels enhances performance compared to multiple single task models. Now, we evaluate how significantly the circular smoothing filter improves performance compared to its absence in the fine-grained model. In Figure 5.12a, we compare the accuracy of the fine-grained model with and without the application of the circular mean filter before the softmax activation function. Our results indicate that smoothing the network logits notably enhances the performance. To understand the source of this improvement, we analyze the errors made by the network. Specifically, in Figure 5.12b we categorize the errors according to the angular deviation from the ground truth. This analysis confirms our hypothesis regarding logit smoothing and aligns with the error analysis provided by Redondo-Cabrera et al. [172], as described in Section 5.4.1. Circular Smoothing significantly increases accuracy compared to the model trained without it, particularly by reducing errors in the bins close to the true value and in those corresponding to a viewpoint flip, i.e., an error near 180°.

5.6.5 Memory Requirements and Inference Timing

In this section we discuss about efficiency of our proposed models. In Table 5.5, we provide the inference time per image and the number of trainable parameters for our models, which are identical due to the shared feature extractor architecture based on MobileNetV2.

Since the nuScenes dataset contains on average around 20 vehicles per image, we measure the timing by fixing the batch size to 20 vehicle crop images and averaging the time over 300 executions. These tests are conducted on an AMD Ryzen 7 3700X 8-Core CPU and a GeForce RTX 2070 GPU. Our models outperform other methods in terms of memory requirements and, thanks to their small memory footprint, are well-suited for deployment on edge devices.

Of note is that a smaller number of parameters does not lead to faster inference times, which is also influenced by how the parameters are used, e.g. in sequential operations generally slower instead of faster parallel processes or convolutions. This is the case for Render for CNN and ResNet18 CS when compared to our models; this behavior has also been reported in benchmarks for different models [173].

Table 5.5: Memory requirements and inference time per image on the nuScenes dataset for the fine-grained and coarse-grained models. Timing results are averaged over 300 executions, each feeding 20 vehicle crops into the network.

	CPU Time (ms)	**GPU Time** (ms)	**# Parameters**
Render for CNN [162]	74.51 ± 7.71	5.43 ± 0.03	64.2M
ResNet18 CS [161, 159]	192.31 ± 14.61	10.82 ± 0.03	11.2M
PoseContrast [164]	662.67 ± 28.85	33.71 ± 0.05	25.8M
Coarse-grained Model Fine-grained Model	277.10 ± 10.91	12.05 ± 0.02	2.7M

5.7 Conclusions

In this chapter we presented two lightweight deep learning models able to predict the viewpoint of vehicles from monocular images, taken, e.g., from a camera mounted on a vehicle driving on the road. We have shown that the *fine-grained* model improves the *coarse-grained* one, which was inspired by the Multi-Task model presented in our previous work [150], and outperforms state-of-the-art results. We achieved this result by means of several contributions.

First, we show that applying smoothing techniques to the network output can noticeably improve vehicle viewpoint estimation performance. Specifically, applying a circular mean filter before the network output, as done for the *fine-grained*, provides better results than summing the network logits, i.e. the approach of the *coarse-grained* model. Additionally, we show that adding geometrical constraints to the training loss by means of a Siamese network further improves the results.

We also introduce the task-specific data augmentation technique *viewpoint crop* with the aim of improving its performance on truncated vehicles at test time, which is an important trait in practical deployments of a viewpoint estimation algorithm, e.g. to estimate azimuth of involved vehicles in the case of car crashes.

Finally, we point out an ambiguity of vehicle viewpoint prediction from monocular images and propose an effective solution to it in the form of CoordConv layers [168]. Both the *fine-grained* and *coarse-grained* model considerably improved their performance on the nuScenes dataset.

Experimental results on the nuScenes dataset show that a small convolutional backbone, like MobileNetV2 [166], is able to obtain results which are comparable to or even better than the ones obtained by much more complex backbones. Thanks to the speed and the size of the backbone, our model is suitable to be deployed on edge devices. We obtain state-of-the-art results also on Pascal3D+, although the absolute performance is lower than on nuScenes, likely because the fixed point of view of the camera with respect to the road eases vehicle viewpoint estimation.

Possible directions for future research involve the use of videos to train and test our proposed models. We believe that temporal information from videos can improve the quality of vehicle viewpoint estimation by imposing constraints on the rigid object motion through time. Another path is to further evaluate the limitation to a specific camera and positioning imposed by the use of a CoordConv layer and how to compensate the differences between the train and test setup.

Chapter 6

Conclusions and Future Work

In this dissertation we explored incremental learning methodologies and the design of efficient CNNs for edge computing applications. The work reported in this dissertation consists of three primary novel contributions:

- **A state-of-the-art approach to exemplar-free incremental learning**. In Chapter 3 we introduced a novel incremental learning method, called *Elastic Feature Consolidation* (EFC), an exemplar-free approach that addresses catastrophic forgetting by relying solely on current task data. This offers advantages in terms of privacy preservation and computational efficiency for learning long sequences of tasks. EFC, thanks to the Empirical Feature Matrix (EFM) and the Prototype Asymmetric Cross-Entropy Loss (PR-ACE), demonstrates significant improvement in terms of both stability and plasticity compared to recent state-of-the-art methods.

  This work represents a first step in understanding training dynamics in Incremental Learning, particularly in the challenging Cold Start Scenario where the initial task does not contain enough classes to learn a high quality backbone. We identify that refining prototype design and updating methods during incremental learning steps could further enhance the stability of EFC. Another promising direction for future research is the exploration of training dynamics using pre-trained feature extractors, particularly in light of the remarkable achievements of pre-trained Vision Transformers. This approach could offer valuable insights into the integration of advanced pre-trained models within the context of incremental learning.

- **A novel application of incremental learning to Social Network Identification**. In Chapter 4, we emphasized the importance of developing incremental learning strategies for real-world applications, particularly in the field of multimedia forensics. We examined two realistic incremental learning scenarios for social network identification: *Incremental Social Update Classification* and *In-*

cremental Social Version Classification. Our analysis demonstrated how recent incremental learning methods perform in these scenarios and highlighted that current solutions still lag behind *Joint Training* performance. Our findings indicate that while significant progress has been made, the journey to completely closing this gap is ongoing, with recent advancements contributing to narrowing it.

Future work should focus on further improving incremental learning methodologies, both in general and in application-specific settings, to enable their practical use in real-world applications. For instance, we observed that relying on DCT-based features improves the performance of the social network identification task. How to effectively incorporate DCT features for regularization during training in incremental social network identification remains an open problem and should be further investigated in future research. Other interesting multimedia forensics applications for incremental learning include *camera source identification* and *text* and *multimedia fake detection*. Similar to the challenge in social network identification, these applications also rely on Deep Neural Networks for their effectiveness. However, these methods can become outdated over time due to new camera vendor types or emerging techniques for generating fake materials.

- **Lightweight models for vehicle orientation estimation**. In Chapter 5 we concentrated on edge-computing applications, designing lightweight and effective DNNs for vehicle viewpoint estimation in intelligent vehicles. Our results showed that by combining coordinate and image pixel information, it is possible to develop lightweight and effective Convolutional Neural Networks for this task. However, we remark that our models are built on top of an existing object detector that provides vehicle coordinates. We chose not to train an object detector specifically for viewpoint estimation, as it is difficult to maintain and adapt this for real-world industrial use. Object detectors are used for many different tasks in industry, and training them for just one task, like viewpoint estimation, might reduce the effectiveness of other task specific models, based as well on DNNs, using its information, such as detecting stop signs, identifying people, or recognizing crashes. Training a single model for all these tasks is impractical due to several challenges: the training data is available at different time steps, data may become unavailable due to privacy concerns, and not all tasks consistently have labeled data. Incremental learning could greatly improve how the intelligent vehicle industry uses deep learning, creating fewer models and making the deep learning models deployment and usage more efficient, cost-effective, and environmentally friendly. Currently, research in this application field is limited, as is the number of benchmark datasets avail-

able for incremental learning in this context. We plan to further investigate incremental learning in this scenario in the future.

Appendix A

Elastic Feature Consolidation for Exemplar-free Incremental Learning

A.1 Training Settings and Hyperparameters

For all the methods, we use the standard ResNet-18 [14] backbone trained from scratch.

First Task Optimization Details. We use the same optimization settings for both the Warm Start and Cold Start scenarios. We train the models on CIFAR-100 and on Tiny-ImageNet for 100 epochs with Adam [89] using an initial learning rate of $1e^{-3}$ and fixed weight decay of $2e^{-4}$. The learning rate is reduced by a factor of 0.1 after 45 and 90 epochs (as done in [53, 54]). For ImageNet-Subset, we followed the implementation of PASS [53], fixing the number of epochs at 160, and used Stochastic Gradient Descent with an initial learning rate of 0.1, momentum of 0.9, and weight decay of $5e^{-4}$. The learning rate was reduced by a factor of 0.1 after 80, 120, and 150 epochs. We applied the same label and data augmentation (random crops and flips) for all the evaluated datasets. For the first task of each dataset, we use self-rotation as performed by [53, 54].

Incremental Steps. Below we provide the hyperparameters and the optimization settings we used for the incremental steps of each state-of-the-art method we tested.

- **EWC [44]**: We used the implementation of [35]. Specifically, we configured the coefficient associated to the regularizer as $\lambda_{\text{E-FIM}} = 5000$ and the fusion of the old and new importance weights is done with $\alpha = 0.5$. For the incremental steps we fix the total number of epochs to 100 and we use Adam optimizer with an initial learning rate of $1e^{-3}$ and fixed weight decay of $2e^{-4}$. The learning rate is reduced by a factor of 0.1 after 45 and 90 epochs.
- **LwF [50]**: We used the implementation of [35]. In particular, we set the temperature parameter $T = 2$ as proposed in the original work and the parameter

associated to the regularizer λ_{LwF} to 10. For the incremental steps we fix the total number of epochs to 100 and we use Adam optimizer with an initial learning rate of $1e^{-3}$ and fixed weight decay of $2e^{-4}$. The learning rate is reduced by a factor of 0.1 after 45 and 90 epochs.

- **PASS [53]**: We follow the implementation provided by the authors. It is an approach relying upon feature distillation and prototypes generation. Following the original paper we set $\lambda_{\text{FD}} = 10$ and $\lambda_{\text{pr}} = 10$. In the original code, we find a temperature parameter, denoted as T, applied to the classification loss, which we set to $T = 1$. As provided in the original paper, for the incremental steps we fix the total number of epochs to 100 and we use Adam optimizer with an initial learning rate of $1e^{-3}$ and fixed weight decay of $2e^{-4}$. The learning rate is reduced by a factor of 0.1 after 45 and 90 epochs

- **SSRE [60]**: We follow the implementation provided by the authors. It is an approach relying upon feature distillation and prototypes generation. Following the original paper, we set $\lambda_{\text{FD}} = 10$ and $\lambda_{\text{pr}} = 10$. In the original code, we find a temperature parameter, denoted as T, applied to the classification loss, which we set to $T = 1$. Following the original code, for the incremental steps we fixed the total number of epochs to 60 and used Adam Optimizer with an initial learning rate of $2e^{-4}$ and fixed weight decay of $5e^{-4}$. The learning rate is reduced by a factor of 0.1 after 45 epochs.

- **FeTrIL [62]**: We follow the official code provided by the authors. During the incremental steps, it uses a Linear SVM classifier working on the pseudo-features extracted from the frozen backbone after the first task. We set the SVM regularization $C = 1$ and the tolerance to 0.0001 as provided by the authors.

Symmetric Loss Hyperparameter. We fix the value of $\lambda_{\text{pr}} = 10$ in Eq. 3.18 as reported in the literature [60, 53, 54].

Appendix B

Incremental Learning for Social Network Identification

B.1 Training Settings and Hyperparameters

We conducted all experiments using FACIL [35], with a fixed maximum of 200 epochs per task. Each experiment was repeated five times with different random weight initializations, and for the ISPC scenario, we randomized the class order.

For methods already implemented in FACIL, we trained each task with Adam [89], starting with a learning rate of 10^{-3}, which was decayed if the validation loss did not improve for 20 consecutive epochs. We selected the best model for the current task based on validation loss, which was then used for subsequent tasks. To reduce training time, we randomly sampled one crop per image for each epoch.

For most of the methods implemented in FACIL, default hyperparameters were used, except for EWC, where we set λ_{E-FIM} to 500 to reduce the weight of the regularizer, and for LwF and BIC, where we set the temperature of knowledge distillation to $T = 1$.

In addition to the existing methods in FACIL, we incorporated SS-IL [57], FeTrIL [62], and EFC [61]. For SS-IL [57], we followed the hyper-parameters provided in the original work and used the same training settings as described above. For FeTrIL [62], we utilized its implementation based on fully connected layer, which allowed end-to-end training and yielded comparable performance to the SVM classifier, as reported by the authors. We train the first task using the optimization protocol used for other FACIL methods. In subsequent tasks, following the training protocol of the authors, we trained the classifier on pseudo-features using Stochastic Gradient Descent (SGD) with a momentum of 0.9, weight decay of 10^{-4}, initial learning rate of 10^{-1}, and decay after 10 epochs. The number of epochs was fixed at 50. For EFC [61], we set $\lambda_{\text{EFM}} = 10$ and the damping hyperparameter $\eta = 1$. We trained the first task using the optimization protocol used for other FACIL methods. For

subsequent tasks, we train the model for 200 epochs, using Adam with a learning rate of 10^{-5} for the backbone and 10^{-4} for the heads, as reported in Chapter 3.

Appendix C

Publications

Peer-reviewed Journal Articles

1. [143] S. Magistri, M. Boschi, F. Sambo, D. C. de Andrade, M. Simoncini, L. Kubin, L. Taccari, L. de Luigi, and S. Salti, "*Lightweight and effective convolutional neural networks for vehicle viewpoint estimation from monocular images*," IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 1, pp. 191–200, 2023.

2. [118] S. Magistri, D. Baracchi, D. Shullani, A. D. Bagdanov, and A. Piva, "Continual learning for adaptive social network identification," Pattern Recognition Letters, vol. 180, pp. 82–89, 2024.

Peer-reviewed Conference Papers

1. [61] S. Magistri, T. Trinci, S.-C. Albin, J. V. de Weijer, and A. D. Bagdanov, "*Elastic feature consolidation for cold start exemplar-free incremental learning*," in The Twelfth International Conference on Learning Representations, 2024.

2. [117] S. Magistri, D. Baracchi, D. Shullani, A. D. Bagdanov, and A. Piva, "*Towards continual social network identification*," in 11th International Workshop on Bio- metrics and Forensics, 2023, pp. 1–6.

Patents

1. [174] S. Magistri, F. Sambo, D. C. De Andrade, F. Schoen, M. Simoncini, L. Bravi, S. Caprasecca, L. Kubin, and L. Taccari, "Systems and methods for utilizing a deep learning model to determine vehicle viewpoint estimations," Dec. 20 2022, US Patent 11,532,096.

2. [175] T. Bianconcini, L. Sarti, L. Taccari, F. Sambo, F. Schoen, E. Civitelli, and S. Magistri, "Systems and methods for utilizing machine learning for vehicle detection of adverse conditions," Mar. 16 2023, US Patent App. 17/447,510.

Bibliography

[1] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *arXiv preprint arXiv:2306.16927*, 2023.

[2] J. Fang, J. Qiao, J. Xue, and Z. Li, "Vision-based traffic accident detection and anticipation: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2023.

[3] L. Taccari, F. Sambo, L. Bravi, S. Salti, L. Sarti, M. Simoncini, and A. Lori, "Classification of crash and near-crash events from dashcam videos and telematics," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2460–2465.

[4] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018, special Issue on Smart Manufacturing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612518300037

[5] J. Song, D. Rondao, and N. Aouf, "Deep learning-based spacecraft relative navigation methods: A survey," *Acta Astronautica*, vol. 191, pp. 22–40, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576521005671

[6] Z. Wang, S. Hirai, and S. Kawamura, "Challenges and opportunities in robotic food handling: A review," *Frontiers in Robotics and AI*, vol. 8, p. 789107, 2022.

[7] J. Jung, M. Maeda, A. Chang, M. Bhandari, A. Ashapure, and J. Landivar-Bowles, "The potential of remote sensing and artificial intelligence as tools to improve the resilience of agriculture production systems," *Current Opinion in Biotechnology*, vol. 70, pp. 15–22, 2021, food Biotechnology Plant Biotechnology. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0958166920301257

[8] J. M. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Zídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. A. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, pp. 583 – 589, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235959867

[9] Y. Liu, Z. Yang, Z. Yu, Z. Liu, D. Liu, H. Lin, M. Li, S. Ma, M. Avdeev, and S. Shi, "Generative artificial intelligence and its applications in materials science: Current situation and future perspectives," *Journal of Materiomics*, vol. 9, no. 4, pp. 798–816, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352847823000771

[10] GenAI and Meta, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[11] OpenAI, "Gpt-4 technical report," 2023.

[12] S. W. Kim, B. Brown, K. Yin, K. Kreis, K. Schwarz, D. Li, R. Rombach, A. Torralba, and S. Fidler, "Neuralfield-ldm: Scene generation with hierarchical latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 8496–8506.

[13] I. Amerini, A. Anagnostopoulos, L. Maiano, L. R. Celsi *et al.*, "Deep learning for multimedia forensics," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 4, pp. 309–457, 2021.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[16] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[17] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[18] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 976–11 986.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[20] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 684–10 695.

[21] R. Balestriero, M. Ibrahim, V. Sobal, A. S. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, A. Schwarzschild, A. G. Wilson, J. Geiping, Q. Garrido, P. Fernandez, A. Bar, H. Pirsiavash, Y. LeCun, and M. Goldblum, "A cookbook of self-supervised learning," *ArXiv*, vol. abs/2304.12210, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258298825

[22] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 610–623. [Online]. Available: https://doi.org/10.1145/3442188.3445922

[23] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," *ArXiv*, vol. abs/2007.03051, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:220381235

[24] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[25] W. C. Abraham and A. Robins, "Memory retention - the synaptic stability versus plasticity dilemma," *Trends in Neurosciences*, vol. 28, no. 2, pp. 73–78, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166223604003704

[26] V. Këpuska and G. Bohouta, "Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home)," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 2018, pp. 99–103.

[27] S. Singh, R. Sulthana, T. Shewale, V. Chamola, A. Benslimane, and B. Sikdar, "Machine-learning-assisted security and privacy provisioning for edge computing: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 236–260, 2022.

[28] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing ai to edge: From deep learning's perspective," *Neurocomputing*, vol. 485, pp. 297–320, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221016428

[29] Z. Chen, B. Liu, R. Brachman, P. Stone, and F. Rossi, *Lifelong Machine Learning*, 2nd ed. Morgan & Claypool Publishers, 2018.

[30] E. Fini, V. G. T. da Costa, X. Alameda-Pineda, E. Ricci, K. Alahari, and J. Mairal, "Self-supervised models are continual learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[31] A. Gomez-Villa, B. Twardowski, K. Wang, and J. van de Weijer, "Plasticity-optimized complementary networks for unsupervised continual learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, pp. 1690–1700.

[32] A. Gomez-Villa, B. Twardowski, L. Yu, A. D. Bagdanov, and J. van de Weijer, "Continually learning self-supervised representations with projected functional regularization," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2022, pp. 3866–3876. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPRW56347.2022.00432

[33] D. Maltoni and V. Lomonaco, "Continuous learning in single-incremental-task scenarios," *Neural Networks*, vol. 116, pp. 56–73, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608019300838

[34] J. Xie, S. Yan, and X. He, "General incremental learning with domain-aware categorical representations," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 14 331–14 340. [Online]. Available: https://doi.org/10.1109/CVPR52688.2022.01395

[35] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2022.

[36] V. V. Ramasesh, A. Lewkowycz, and E. Dyer, "Effect of scale on catastrophic forgetting in neural networks," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=GhVS8_yPeEa

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

[38] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 246–11 255, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:54462131

[39] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1185–1197, 2022.

[40] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[41] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6470–6479.

[42] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, "Online continual learning with maximal interfered retrieval," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019,

pp. 11 849–11 860. [Online]. Available: http://papers.nips.cc/paper/9357-online-continual-learning-with-maximal-interfered-retrieval.pdf

[43] Y. Zhang, B. Pfahringer, E. Frank, A. Bifet, N. J. S. Lim, and Y. Jia, "A simple but strong baseline for online continual learning: Repeated augmented rehearsal," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 14 771–14 783. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/5ebbbac62b968254093023f1c95015d3-Paper-Conference.pdf

[44] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114

[45] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[46] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[47] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 3987–3995. [Online]. Available: http://proceedings.mlr.press/v70/zenke17a.html

[48] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. López, and A. D. Bagdanov, "Rotate your networks: Better weight consolidation and less catastrophic forgetting," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 2262–2268.

[49] H. Ritter, A. Botev, and D. Barber, "Online structured laplace approximations for overcoming catastrophic forgetting," *ArXiv*, vol. abs/1805.07810, 2018.

[50] Z. Li and D. Hoiem, "Learning without forgetting," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 614–629.

[51] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5133–5141, 2018.

[52] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetting learning in deep neural networks," *arXiv preprint arXiv:1607.00122*, 2016.

[53] F. Zhu, X.-Y. Zhang, C. Wang, F. Yin, and C.-L. Liu, "Prototype augmentation and self-supervision for incremental learning," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 5867–5876.

[54] M. Toldo and M. Ozay, "Bring evanescent representations to life in lifelong class incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 16 732–16 741.

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

[56] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.

[57] H. Ahn, J. Kwak, S. F. Lim, H. Bang, H. Kim, and T. Moon, "Ss-il: Separated softmax for incremental learning," *IEEE/CVF International Conference on Computer Vision*, pp. 824–833, 2020.

[58] Y. Liu, Y. Su, A.-A. Liu, B. Schiele, and Q. Sun, "Mnemonics training: Multi-class incremental learning without forgetting," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 245–12 254.

[59] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: A strong, simple baseline," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[60] K. Zhu, W. Zhai, Y. Cao, J. Luo, and Z.-J. Zha, "Self-sustaining representation expansion for non-exemplar class-incremental learning," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 9286–9295.

[61] S. Magistri, T. Trinci, S.-C. Albin, J. V. de Weijer, and A. D. Bagdanov, "Elastic feature consolidation for cold start exemplar-free incremental learning," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=7D9X2cFnt1

[62] G. Petit, A. Popescu, H. Schindler, D. Picard, and B. Delezoide, "Fetril: Feature translation for exemplar-free class-incremental learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3911–3920.

[63] D. Goswami, Y. Liu, B. Twardowski, and J. van de Weijer, "Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[64] A. Panos, Y. Kobe, D. O. Reino, R. Aljundi, and R. E. Turner, "First session adaptation: A strong replay-free baseline for class-incremental learning," in *International Conference on Computer Vision (ICCV)*, 2023.

[65] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 67–82.

[66] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[67] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International conference on machine learning*. PMLR, 2018, pp. 4548–4557.

[68] Y. Shi, L. Yuan, Y. Chen, and J. Feng, "Continual learning via bit-level information preserving," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 674–16 683.

[69] G. Saha, I. Garg, A. Ankit, and K. Roy, "Space: Structured compression and sharing of representational space for continual learning," *IEEE Access*, vol. 9, pp. 150 480–150 494, 2021.

[70] S. Wang, X. Li, J. Sun, and Z. Xu, "Training networks in null space of feature covariance for continual learning," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021, pp. 184–193.

[71] G. Saha, I. Garg, and K. Roy, "Gradient projection memory for continual learning," in *International Conference on Learning Representations*, 2020.

[72] S. Lin, L. Yang, D. Fan, and J. Zhang, "Trgp: Trust region gradient projection for continual learning," in *International Conference on Learning Representations*, 2021.

[73] Z. Zhao, Z. Zhang, X. Tan, J. Liu, Y. Qu, Y. Xie, and L. Ma, "Rethinking gradient projection continual learning: Stability/plasticity feature space decoupling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 3718–3727.

[74] G. Saha and K. Roy, "Continual learning with scaled gradient projection," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. [Online]. Available: https://doi.org/10.1609/aaai.v37i8.26157

[75] C. Zeno, I. Golan, E. Hoffer, and D. Soudry, "Task-agnostic continual learning using online variational bayes with fixed-point updates," *Neural Computation*, vol. 33, no. 11, pp. 3139–3177, 2021.

[76] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *International conference on machine learning*. PMLR, 2018, pp. 4528–4537.

[77] A. Carta, L. Pellegrini, A. Cossu, H. Hemati, and V. Lomonaco, "Avalanche: A pytorch library for deep continual learning," *Journal of Machine Learning Research*, vol. 24, no. 363, pp. 1–6, 2023. [Online]. Available: http://jmlr.org/papers/v24/23-0130.html

[78] F. Huszár, "Note on the quadratic penalties in elastic weight consolidation," *Proceedings of the National Academy of Sciences*, vol. 115, pp. E2496 – E2497, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3415694

[79] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Reply to huszár: The elastic weight consolidation penalty is empirically valid," *Proceedings of the National Academy of Sciences*, vol. 115, no. 11, pp. E2498–E2498, 2018. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1800157115

[80] F. Benzing, "Unifying importance based regularisation methods for continual learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 2372–2396.

[81] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06. New York, NY, USA: Association

for Computing Machinery, 2006, p. 535–541. [Online]. Available: https://doi.org/10.1145/1150402.1150464

[82] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. [Online]. Available: https://arxiv.org/abs/1503.02531

[83] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[84] X. Liu, C. Wu, M. Menta, L. Herranz, B. Raducanu, A. D. Bagdanov, S. Jui, and J. v. de Weijer, "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.

[85] H. Lee, S. J. Hwang, and J. Shin, "Self-supervised label augmentation via input transformations," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5714–5724.

[86] L. Yu, B. Twardowski, X. Liu, L. Herranz, K. Wang, Y. Cheng, S. Jui, and J. van de Weijer, "Semantic drift compensation for class-incremental learning," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6980–6989, 2020.

[87] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[88] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI Commun.*, vol. 30, pp. 169–190, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3906277

[89] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[90] M. Kang, J. Park, and B. Han, "Class-incremental learning by knowledge distillation with adaptive feature consolidation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 071–16 080.

[91] S. Yan, J. Xie, and X. He, "Der: Dynamically expandable representation for class incremental learning," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3013–3022.

[92] D.-W. Zhou, Q.-W. Wang, H.-J. Ye, and D.-C. Zhan, "A model or 603 exemplars: Towards memory-efficient class-incremental learning," in *The Eleventh International Conference on Learning Representations*, 2022.

[93] M. Davari, N. Asadi, S. Mudur, R. Aljundi, and E. Belilovsky, "Probing representation forgetting in supervised and unsupervised continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 712–16 721.

[94] E. Belouadah and A. Popescu, "Il2m: Class incremental learning with dual memory," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 583–592.

[95] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 831–839.

[96] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. 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 16*. Springer, 2020, pp. 86–102.

[97] B. Huang, Z. Chen, P. Zhou, J. Chen, and Z. Wu, "Resolving task confusion in dynamic expansion architectures for class incremental learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 1, 2023, pp. 908–916.

[98] J. Ash and R. P. Adams, "On warm-starting neural network training," *Advances in neural information processing systems*, vol. 33, pp. 3884–3894, 2020.

[99] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 374–382.

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

[101] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," *Neurocomputing*, vol. 469, pp. 28–51, 2022.

[102] A. Soutif-Cormerais, A. Carta, A. Cossu, J. Hurtado, V. Lomonaco, J. Van de Weijer, and H. Hemati, "A comprehensive empirical evaluation on online continual learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3518–3528.

[103] Y. Liu, B. Schiele, and Q. Sun, "Adaptive aggregation networks for class-incremental learning," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 2544–2553.

[104] F. Zhu, Z. Cheng, X.-y. Zhang, and C.-l. Liu, "Class-incremental learning via dual augmentation," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 14 306–14 318.

[105] E. Belouadah and A. Popescu, "Deesil: Deep-shallow incremental learning." in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[106] J. Smith, Y.-C. Hsu, J. Balloch, Y. Shen, H. Jin, and Z. Kira, "Always be dreaming: A new approach for data-free class-incremental learning," in *International Conference on Computer Vision (ICCV)*, 2021.

[107] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[108] J. Martens, "New insights and perspectives on the natural gradient method," *J. Mach. Learn. Res.*, vol. 21, pp. 146:1–146:76, 2014.

[109] L. Grementieri and R. Fioresi, "Model-centric data manifold: the data through the eyes of the model," *CoRR*, vol. abs/2104.13289, 2021. [Online]. Available: https://arxiv.org/abs/2104.13289

[110] J. Martens and I. Sutskever, "Training deep and recurrent networks with hessian-free optimization," in *Neural Networks*, 2012.

[111] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, and E. Belilovsky, "New insights on reducing abrupt representation change in online continual learning," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=N8MaByOzUfb

[112] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Technical report*, 2009.

[113] J. Wu, Q. Zhang, and G. Xu, "Tiny imagenet challenge," *Technical report*, 2017.

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

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

[116] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 233–248.

[117] S. Magistri, D. Baracchi, D. Shullani, A. D. Bagdanov, and A. Piva, "Towards continual social network identification," in *11th International Workshop on Biometrics and Forensics*, 2023, pp. 1–6.

[118] S. Magistri, D. Baracchi, D. Shullani, A. D. Bagdanov, and A. Piva, "Continual learning for adaptive social network identification," *Pattern Recognition Letters*, vol. 180, pp. 82–89, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865524000540

[119] C. Pasquini, I. Amerini, and G. Boato, "Media forensics on social media platforms: a survey," *EURASIP Journal on Information Security*, vol. 2021, no. 1, pp. 1–19, 2021.

[120] P. Bestagini, M. Fontani, S. Milani, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro, "An overview on video forensics," in *European Signal Processing Conference*, 2012, pp. 1229–1233.

[121] A. Castiglione, G. Cattaneo, and A. De Santis, "A forensic analysis of images on online social networks," in *International Conference on Intelligent Networking and Collaborative Systems*. IEEE, 2011, pp. 679–684.

[122] O. Giudice, A. Paratore, M. Moltisanti, and S. Battiato, "A classification engine for image ballistics of social data," in *19th International Conference on Image Analysis and Processing*. Springer, 2017, pp. 625–636.

[123] J. He, Z. Lin, L. Wang, and X. Tang, "Detecting doctored jpeg images via dct coefficient analysis," in *European Conference on Computer Vision*. Springer, 2006, pp. 423–435.

[124] T. Pevny and J. Fridrich, "Detection of double-compression in jpeg images for applications in steganography," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 247–258, 2008.

[125] I. Amerini, T. Uricchio, and R. Caldelli, "Tracing images back to their social network of origin: A cnn-based approach," in *IEEE Workshop on Information Forensics and Security*. IEEE, 2017, pp. 1–6.

[126] D. Shullani, D. Baracchi, M. Iuliani, and A. Piva, "Social network identification of laundered videos based on dct coefficient analysis," *IEEE Signal Processing Letters*, vol. 29, pp. 1112–1116, 2022.

[127] Manisha, A. Karunakar, and C.-T. Li, "Identification of source social network of digital images using deep neural network," *Pattern Recognition Letters*, vol. 150, pp. 17–25, 2021.

[128] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[129] A. Castiglione, G. Cattaneo, M. Cembalo, and U. Ferraro Petrillo, "Experimentations with source camera identification and online social networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 4, no. 2, pp. 265–274, 2013.

[130] R. Caldelli, I. Amerini, and C. T. Li, "Prnu-based image classification of origin social network with cnn," in *26th European Signal Processing Conference*. IEEE, 2018, pp. 1357–1361.

[131] I. Amerini, C.-T. Li, and R. Caldelli, "Social network identification through image classification with cnn," *IEEE access*, vol. 7, pp. 35 264–35 273, 2019.

[132] T. Gloe, "Forensic analysis of ordered data structures on the example of jpeg files," in *IEEE International Workshop on Information Forensics and Security*. IEEE, 2012, pp. 139–144.

[133] S. Verde, C. Pasquini, F. Lago, A. Goller, F. G. De Natale, A. Piva, and G. Boato, "Multi-clue reconstruction of sharing chains for social media images," *arXiv preprint arXiv:2108.02515*, 2021.

[134] M. Iuliani, D. Shullani, M. Fontani, S. Meucci, and A. Piva, "A video forensic framework for the unsupervised analysis of mp4-like file container," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 635–645, 2018.

[135] P. Yang, D. Baracchi, M. Iuliani, D. Shullani, R. Ni, Y. Zhao, and A. Piva, "Efficient video integrity analysis through container characterization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 947–954, 2020.

[136] K. Rana, G. Singh, and P. Goyal, "Snrcn2: Steganalysis noise residuals based cnn for source social network identification of digital images," *Pattern Recognition Letters*, vol. 171, pp. 124–130, 2023.

[137] F. Marra, C. Saltori, G. Boato, and L. Verdoliva, "Incremental learning for the detection and classification of gan-generated images," in *IEEE International Workshop on Information Forensics and Security*. IEEE, 2019, pp. 1–6.

[138] SmartData Research Group, "Smartphone images dataset," 2015. [Online]. Available: http://smartdata.cs.unibo.it/datasets

[139] B. Hadwiger and C. Riess, "The forchheim image database for camera identification in the wild," in *Pattern Recognition. ICPR International Workshops and Challenges*. Springer, 2021, pp. 500–515.

[140] D. Baracchi, D. Shullani, M. Iuliani, D. Giani, and A. Piva, "Uncovering the authorship: Linking media content to social user profiles," 2024, preprint available at https://lesc.dinfo.unifi.it/.

[141] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[142] M. Kivanc Mihcak, I. Kozintsev, and K. Ramchandran, "Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising," in *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings.*, vol. 6, 1999, pp. 3253–3256 vol.6.

[143] S. Magistri, M. Boschi, F. Sambo, D. C. de Andrade, M. Simoncini, L. Kubin, L. Taccari, L. de Luigi, and S. Salti, "Lightweight and effective convolutional neural networks for vehicle viewpoint estimation from monocular images," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 191–200, 2023.

[144] Q. Li, H. Lu, X. Liu, X. Huang, C. Song, S. Huang, and J. Huang, "Optimized 3d street scene reconstruction from driving recorder images," *Remote Sensing*, vol. 7, pp. 9091–9121, 07 2015.

[145] N. Cornelis, B. Leibe, K. Cornelis, and L. V. Gool, "3D Urban Scene Modeling Integrating Recognition and Reconstruction," *International Journal of Computer Vision*, vol. 78, pp. 121–141, 2007.

[146] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, J. Schneider, D. Bradley, and N. Djuric, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," in *ICRA*, 2020, pp. 10 563–10 569.

[147] M. Simoncini, D. C. de Andrade, S. Salti, L. Taccari, F. Schoen, and F. Sambo, "Two-stream neural architecture for unsafe maneuvers classification from dashcam videos and gps/imu sensors," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[148] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[149] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, and N. Sun, "Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3434–3440, 2018.

[150] S. Magistri, F. Sambo, F. Schoen, D. C. d. Andrade, M. Simoncini, S. Caprasecca, L. Kubin, L. Bravi, and L. Taccari, "A lightweight deep learning model for vehicle viewpoint estimation from dashcam images," in *ITSC*, 2020, pp. 1–6.

[151] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *IEEE winter conference on applications of computer vision*. IEEE, 2014, pp. 75–82.

[152] Z. Liu, Z. Wu, and R. Tóth, "Smoke: Single-stage monocular 3d object detection via keypoint estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 996–997.

[153] A. Simonelli, S. Rota Bulo, L. Porzi, M. Lopez Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection: From single to multi-class recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[154] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3d object detection methods for autonomous driving applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3782–3795, 2019.

[155] F. Massa, M. Aubry, and R. Marlet, "Convolutional neural networks for joint object detection and pose estimation: A comparative study," *CoRR*, vol. abs/1412.7190, 2014. [Online]. Available: http://arxiv.org/abs/1412.7190

[156] G. Divon and A. Tal, "Viewpoint estimation—insights & model," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[157] R. M. Francisco Massa and M. Aubry, "Crafting a multi-task cnn for viewpoint estimation," in *BMVC*, E. R. H. Richard C. Wilson and W. A. P. Smith, Eds. BMVA Press, September 2016, pp. 91.1–91.12.

[158] A. Ghodrati, M. Pedersoli, and T. Tuytelaars, "Is 2d information enough for viewpoint estimation?" in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[159] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1510–1519.

[160] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[161] X. Zhou, A. Karpur, L. Luo, and Q. Huang, "Starmap for category-agnostic keypoint and viewpoint estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 318–334.

[162] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2686–2694.

[163] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *NIPS*, 1993, p. 737–744.

[164] Y. Xiao, Y. Du, and R. Marlet, "Posecontrast: Class-agnostic object viewpoint estimation in the wild with pose-aware contrastive learning," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 74–84.

[165] C. Redondo-Cabrera, R. J. López-Sastre, Y. Xiang, T. Tuytelaars, and S. Savarese, "Pose estimation errors, the ultimate diagnosis," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*. Springer, 2016, pp. 118–134.

[166] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[167] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 7074–7082.

[168] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," *Advances in neural information processing systems*, vol. 31, 2018.

[169] C. Beckham and C. Pal, "Unimodal probability distributions for deep ordinal classification," in *ICML*, D. Precup and Y. W. Teh, Eds., vol. 70, Aug 2017, pp. 411–419.

[170] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[171] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[172] C. Redondo-Cabrera, R. J. López-Sastre, Y. Xiang, T. Tuytelaars, and S. Savarese, "Pose estimation errors, the ultimate diagnosis," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*. Springer, 2016, pp. 118–134.

[173] S. Bianco, R. Cadène, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[174] S. Magistri, F. Sambo, D. C. De Andrade, F. SCHOEN, M. Simoncini, L. Bravi, S. CAPRASECCA, L. Kubin, and L. Taccari, "Systems and methods for utilizing a deep learning model to determine vehicle viewpoint estimations," Dec. 20 2022, uS Patent 11,532,096.

[175] T. Bianconcini, L. Sarti, L. Taccari, F. Sambo, F. Schoen, E. Civitelli, and S. Magistri, "Systems and methods for utilizing machine learning for vehicle detection of adverse conditions," Mar. 16 2023, uS Patent App. 17/447,510.