Universitat Autònoma
de Barcelona

Lifelong Learning of Neural Networks: Detecting Novelty and Adapting to New Domains without Forgetting

A dissertation submitted by **Marc Masana** to Universitat Autònoma de Barcelona in fulfilment of the degree of **Doctor of Philosophy** in the Dept. Ciències de la Computació.

Bellaterra, November 4, 2020

Directors	**Dr. Joost van de Weijer** Centre de Visió per Computador Universitat Autònoma de Barcelona **Dr. Andrew D. Bagdanov** Media Integration and Communication Center University of Florence
Thesis committee	**Dr. Joan Serrà** Applied AI Dolby Laboratories **Dr. Bogdan Raducanu** Centre de Visió per Computador Universitat Autònoma de Barcelona **Dr. German I. Parisi** Department of Informatics Universität Hamburg
International evaluators	**Dr. Adrian Popescu** Department for Ambient Intelligence and Interactive Systems CEA-List Institute **Dr. Vincenzo Lomonaco** Department of Computer Science and Engineering University of Bologna

This document was typeset by the author using LaTeX 2ε.

ISBN: 978-84-121011-9-5

Printed by Ediciones Gráficas Rey, S.L.

Als meus pares

Abstract

Computer vision has gone through considerable changes in the last decade as neural networks have come into common use. As available computational capabilities have grown, neural networks have achieved breakthroughs in many computer vision tasks, and have even surpassed human performance in others. With accuracy being so high, focus has shifted to other issues and challenges. One research direction that saw a notable increase in interest is on lifelong learning systems. Such systems should be capable of efficiently performing tasks, identifying and learning new ones, and should moreover be able to deploy smaller versions of themselves which are experts on specific tasks. In this thesis, we contribute to research on lifelong learning and address the compression and adaptation of networks to small target domains, the incremental learning of networks faced with a variety of tasks, and finally the detection of out-of-distribution samples at inference time.

We explore how knowledge can be transferred from large pretrained models to more task-specific networks capable of running on smaller devices by extracting the most relevant information based on activation statistics. Using a pretrained model provides more robust representations and a more stable initialization when learning a smaller task, which leads to higher performance and is known as domain adaptation. However, those models are too large for certain applications that need to be deployed on devices with limited memory and computational capacity. In this thesis we show that, after performing domain adaptation, some learned activations barely contribute to the predictions of the model. Therefore, we propose to apply network compression based on low-rank matrix decomposition using the activation statistics. This results in a significant reduction of the model size and the computational cost.

Like human intelligence, machine intelligence aims to have the ability to learn and remember knowledge. However, when a trained neural network is presented with learning a new task, it ends up forgetting previous ones. This is known as catastrophic forgetting and its avoidance is studied in continual learning. The work presented in this thesis extensively surveys continual learning techniques (both when knowing the task-ID at test time or not) and presents an approach to avoid catastrophic forgetting in sequential task learning scenarios. Our technique is based on using ternary masks in order to update a network to new tasks, reusing the knowledge of previous ones while

not forgetting anything about them. In contrast to earlier work, our masks are applied to the activations of each layer instead of the weights. This considerably reduces the number of mask parameters to be added for each new task; with more than three orders of magnitude for most networks. Furthermore, the analysis on a wide range of work on incremental learning without access to the task-ID, provides insight on current state-of-the-art approaches that focus on avoiding catastrophic forgetting by using regularization, rehearsal of previous tasks from a small memory, or compensating the task-recency bias.

We also consider the problem of out-of-distribution detection. Neural networks trained with a cross-entropy loss force the outputs of the model to tend toward a one-hot encoded vector. This leads to models being too overly confident when presented with images or classes that were not present in the training distribution. The capacity of a system to be aware of the boundaries of the learned tasks and identify anomalies or classes which have not been learned yet is key to lifelong learning and autonomous systems. In this thesis, we present a metric learning approach to out-of-distribution detection that learns the task at hand on an embedding space.

Keywords: *computer vision, machine learning, neural network compression, out-of-distribution detection, continual learning*

Resum

La visió per computador ha experimentat canvis considerables en l'última dècada, ja que les xarxes neuronals han passat a ser d'ús comú. A mesura que les capacitats computacionals disponibles han crescut, les xarxes neuronals han aconseguit avenços en moltes tasques de visió per computador i fins i tot han superat el rendiment humà en altres. Donada l'elevada precisió d'aquestes xarxes, l'atenció s'ha desplaçat cap a altres problemes i reptes. Un camp de recerca que ha experimentat un notable augment de l'interès és la dels sistemes d'aprenentatge continuat. Aquests sistemes haurien de ser capaços de realitzar tasques de manera eficient, identificar-ne i aprendre'n de noves, a més de ser capaços de desplegar versions més compactes d'ells mateixos que siguin experts en tasques específiques. En aquesta tesi, contribuïm a la investigació sobre l'aprenentatge continuat i abordem la compressió i adaptació de xarxes a dominis més petits, l'aprenentatge incremental de xarxes enfrontades a diverses tasques i, finalment, la detecció d'anomalies i novetats en temps d'inferència.

Explorem com es pot transferir el coneixement des de grans models pre-entrenats a xarxes amb tasques més específiques, capaces d'executar-se en dispositius més limitats extraient la informació més rellevant. L'ús d'un model pre-entrenat proporciona representacions més robustes i una inicialització més estable quan s'aprèn una tasca més específica, cosa que comporta un rendiment més alt i es coneix com a adaptació de domini. Tanmateix, aquests models són massa grans per a determinades aplicacions que cal desplegar en dispositius amb poca memòria i poca capacitat de càlcul. En aquesta tesi demostrem que, després de realitzar l'adaptació de domini, algunes activacions apreses amb prou feines contribueixen a les prediccions del model. Per tant, proposem aplicar la compressió de xarxa basada en la descomposició de matrius de baix rang mitjançant les estadístiques de les activacions. Això es tradueix en una reducció significativa de la mida del model i del cost computacional.

Igual que la intel·ligència humana, el *machine learning* pretén tenir la capacitat d'aprendre i recordar el coneixement. Tot i això, quan una xarxa neuronal ja entrenada aprèn una nova tasca, s'acaba oblidant de les anteriors. Això es coneix com a oblit catastròfic i s'estudia la seva prevenció en l'aprenentatge continu. El treball presentat en aquesta tesi estudia àmpliament les tècniques d'aprenentatge continu (tant quan es coneix o no l'identificador de tasca) com una aproximació per evitar oblits catastròfics

en escenaris d'aprenentatge seqüencial. La nostra tècnica es basa en l'ús de màscares ternàries per tal d'actualitzar una xarxa a tasques noves, reutilitzant el coneixement d'altres anteriors sense oblidar res d'elles. A diferència dels treballs anteriors, les nostres màscares s'apliquen a les activacions de cada capa en lloc dels pesos. Això redueix considerablement el nombre de paràmetres que s'afegiran per a cada nova tasca; amb més de tres ordres de magnitud per a la majoria de xarxes. A més, analitzem l'estat de l'art en aprenentatge incremental sense accés a l'identificador de tasca. Això proporciona informació sobre les direccions de recerca actuals que se centren a evitar l'oblit catastròfic mitjançant la regularització, l'assaig de tasques anteriors des d'una petita memòria externa o compensant el biaix de la tasca més recent.

També considerem el problema de la detecció d'anomalies. Les xarxes neuronals entrenades amb una funció de cost basada en entropia creuada obliguen les sortides del model a tendir cap a un vector codificat de sortida única. Això fa que els models tinguin massa confiança quan intenten predir imatges o classes que no estaven presents a la distribució original. La capacitat d'un sistema per ser conscient dels límits de les tasques apreses i identificar anomalies o classes que encara no s'han après és clau per a l'aprenentatge continu i els sistemes autònoms. En aquesta tesi, presentem un enfocament d'aprenentatge mètric per a la detecció d'anomalies que aprèn la tasca en un espai mètric.

Paraules clau: *visió per computador, aprenentatge automàtic, compressió de xarxes neuronals, detecció d'anomalies, aprenentatge continu*

Resumen

La visión por computador ha experimentado cambios considerables en la última década a medida que las redes neuronales se han vuelto de uso común. Debido a que las capacidades computacionales disponibles han ido aumentando, las redes neuronales han logrado avances en muchas tareas de visión por computador e incluso han superado el rendimiento humano en otras. Dado que la precisión es tan alta, el enfoque se ha desplazado a otros problemas y desafíos. Una dirección de investigación que ha experimentado un aumento notable en interés son los sistemas de aprendizaje continuado. Estos sistemas deben ser capaces de realizar tareas de manera eficiente, identificar y aprender otras nuevas y, además, deben poder implementar versiones más compactas de sí mismos que sean expertos en tareas específicas. En esta tesis, contribuimos a la investigación sobre el aprendizaje continuado y abordamos la compresión y adaptación de redes a pequeños dominios, el aprendizaje incremental de redes ante una variedad de tareas y, finalmente, la detección de anomalías y novedades durante la inferencia.

Exploramos cómo se puede transferir el conocimiento de grandes modelos pre-entrenados a redes con tareas más específicas capaces de ejecutarse en dispositivos más pequeños, extrayendo la información más relevante basada en estadísticas de las activaciones. El uso de un modelo pre-entrenado proporciona representaciones más robustas y una inicialización más estable al aprender una tarea más pequeña, lo que conduce a un mayor rendimiento y se conoce como adaptación de dominio. Sin embargo, esos modelos son demasiado grandes para ciertas aplicaciones que deben implementarse en dispositivos con memoria y capacidad computacional limitadas. En esta tesis mostramos que, después de realizar la adaptación de dominio, algunas activaciones aprendidas apenas contribuyen a las predicciones del modelo. Por lo tanto, proponemos aplicar compresión de redes basada en la descomposición matricial de bajo rango utilizando las estadísticas de las activaciones. Esto da como resultado una reducción significativa del tamaño del modelo y el coste computacional.

Al igual que la inteligencia humana, el *machine learning* tiene como objetivo tener la capacidad de aprender y recordar conocimientos. Sin embargo, cuando una red neuronal ya entrenada aprende una nueva tarea, termina olvidando las anteriores. Esto se conoce como olvido catastrófico y su prevención se estudia en el aprendizaje continuo. El trabajo presentado en esta tesis analiza ampliamente las técnicas de aprendizaje

continuo (tanto cuando se conoce el ID de la tarea durante inferencia como cuando no) y presenta un enfoque para evitar el olvido catastrófico en escenarios de aprendizaje secuencial de tareas. Nuestra técnica se basa en utilizar máscaras ternarias cuando la red tiene que aprender nuevas tareas, reutilizando los conocimientos de las anteriores sin olvidar nada de ellas. A diferencia otros trabajos, nuestras máscaras se aplican a las activaciones de cada capa en lugar de a los pesos. Esto reduce considerablemente el número de parámetros que se agregarán para cada nueva tarea; con más de tres órdenes de magnitud para la mayoría de las redes. Además, el análisis de una amplia gama de trabajos sobre aprendizaje incremental sin acceso a la identificación de la tarea, proporciona información sobre los enfoques actuales del estado del arte que se centran en evitar el olvido catastrófico mediante el uso de la regularización, el ensayo de tareas anteriores con memorias externas, o compensando el sesgo hacia la tarea más reciente.

También consideramos el problema de la detección de anomalías. Las redes neuronales entrenadas con una función de coste basada en entropía cruzada obligan a las salidas del modelo a tender hacia un vector de salida única. Esto hace que los modelos tengan demasiada confianza cuando se les presentan imágenes o clases que no estaban presentes en la distribución del entrenamiento. La capacidad de un sistema para conocer los límites de las tareas aprendidas e identificar anomalías o clases que aún no se han aprendido es clave para el aprendizaje continuado y los sistemas autónomos. En esta tesis, presentamos un enfoque de aprendizaje con métricas para la detección de anomalías que aprende la tarea en un espacio métrico.

Palabras clave: *visión por computador, aprendizaje automático, compresión de redes neuronales, detección de anomalías, aprendizaje continuo*

Acknowledgements

Working towards obtaining a Ph.D has not been an easy task and has been one of the most challenging experiences of my life. This Ph.D would not have been possible without the help, contribution, support and love of many distinct individuals. In the following lines, I would like to express my deepest appreciation to all those who have assisted me during this long journey. Shall this be a humble tribute to all those who have contributed to this work.

Let me start with those academically close, i.e., my supervisors, without their support this thesis would have never seen the light. Joost, we have been working together for more than 8 years, your support and supervision have helped me develop and grow as a researcher during several stages of my academic journey. Thanks for all the discussions, ramblings with pen and paper, long nights before deadlines and your confidence in me. I am specially thankful for the freedom given and for allowing me to feel like a useful addition to the group that you have been building. Andy, your discussions and guidance have also helped me improve my research. Thanks for your patience and advice which have greatly helped me improve my programming and writing skills.

I also thank Prof. Tinne Tuytelaars who welcomed me into her research lab and gave me the opportunity to share knowledge with her group and participate in the survey project with Rahaf and Matthias. The discussions and new ideas shared helped shape the continual learning parts of this thesis. I also extend my gratitude to those who I have collaborated with and that have introduced me to new research fields. Jorge, Antonio, Loïc, Ozan, Gert and Ursula, thanks for your support, trust and fun times. I also extend my thanks to the Generalitat de Catalunya and the FI grant program for their financial support during the development of this thesis.

Pursuing a Ph.D cannot be done without sharing the good and tough times. My deepest gratitude to my colleagues at the Computer Vision Center with whom I shared many moments, and in special to the LAMP group. I appreciate the insight and discussions from Bogdan, Mikhail, Ramón, Luis, Abel, Bartłomiej and David. My sincere thanks to Yaxing, Victoria, Olaia, Fei, Albin, Chenshen, Aymen, Aitor, Lichao, Oguz, Kai, Javad, Carola, Shun, Minghan and Shiqi. Biel, Mikel, Laura, Lu and Xialei

thanks for your boundless confidence in me, for all the cooking, travelling, relaxed discussions and cozy afternoons having tea and playing board games. Your friendship has been the best of this stage. Biel, without you lunch breaks will never be the same. Lu, Xialei and Yaxing, thanks for teaching me so much. I would also want to extend my thanks to the administration staff for all the help and sympathy provided over the years. To Mari Carmen, Claire, Ana, Montse, Eva, Silvia, Gigi, Mireia, Kevin, Alexandra, Meritxell and Núria, thank you very much.

To the friends who stay close to my heart throughout the years and geographical distances. To my Pollancres, thanks for the unconditional support, let's keep expanding this second family. To Alba, for being my lighthouse when I'm lost at what to do. Mirko and Silvia, thanks for spreading mindfulness, good vibes and stories worth telling. Jorge, my gratitude for all the conversations and friendly banter, for the patience and understanding. To Ines, Fanny, Betty, Bobby, Simona, and all the troupe, thanks for making me feel like a true Zuagrasta. From the cold lands, thanks to Sanne for always spreading positivity. To Antonio for putting up with me at home when working at weird times. To Cris and Marta for always making time to have a coffee. To Ana, Emi, Albert, Antonio, Ricard, Amaia, Sarah, Dena, Arjun, Fran, Xim, Federico, Maxim, Tom, Benjamin, Amal, Kai-li, Matthew and Medea, thanks for all the moments and laughs.

I leave my most inner circle to the end. This thesis is dedicated to my parents, Joan and Mary, who have always given me their love and countless tupperwares of food to survive. Thanks for everything you've done for me, for encouraging me to pursue my dreams, and to my mom for being a warrior and fighting through adversities to be here today to see my thesis finished. Without your support I would have never had a chance to get to this point. Thanks to all of my family, Montse, Rosa Maria, Diego, Tere, Antonino, Dolly, Toni, Farida, Fadoua, Halima, Diego, Isaac, Carolina, Silvia and Cristina for making celebrations feel warmer, pointless discussions memorable and forgetting about my problems easier. My respects to my grandpa and tiet Antonio, I hope you would be proud, I miss you. To Sabine, Günther and Lisa, thanks for your support and for your patience when Cori and me ramble about research stuff. Finally, my biggest love-thanks and endless gratitude to Cori, who stood by me at every step of the journey, comforted me through my struggles, listened to my complaints, celebrated my achievements and never lost faith in me.

Contents

1 Introduction

We live in the epoch of big data. Especially visual data is playing a rapidly growing role in our lives. Every day large amounts of images and videos are uploaded to social media. Also, machines store and analyze petabytes of visual data for industrial, health and robotics applications. The field of computer vision studies methods to unlock the potential of this data with the goal to extract the information available in it. Helped by the ongoing revolution in artificial intelligence, this has resulted in dramatic improvements over the last decade.

Traditionally, hand-crafted algorithms have dominated computer vision. Hand-crafting refers here to the process in which computer vision scientists design the various parts of an algorithm, typically designing the features (e.g. SIFT) and classifier (e.g. an SVM) separately [42, 99]. The hand-crafted approach has been superseded by deep learning algorithms which allow for the end-to-end training of both feature extractors and classifiers. As a consequence, deep learning has replaced a wide variety of features that were designed for different computer vision applications. The ability of deep learning to jointly optimize the feature extractor and classifier has resulted in large performance improvements throughout the field of computer vision, including a large variety of classical problems such as object detection, scene classification or image classification [48, 78, 121, 189].

The theory of deep learning was actually developed already in the previous century. In 1998, a gradient-based learning technique was proposed consisting of a multilayer neural network trained with a backpropagation algorithm [80]. However, computational limitations at the time and the availability of labelled data were not sufficient to make the method suitable for complex computer vision problems. It was not until 2012 that technology caught up and deep features significantly improved state-of-the-art results for object recognition [78]. In the field of object recognition, fine-tuning of convolutional neural networks trained on large datasets made it possible to improve state-of-the-art performance also for small datasets like PASCAL VOC [121]. Excellent results were achieved on image classification, scene recognition and object detection [48, 121, 189]. Later, several network architectures trained on the ImageNet dataset were introduced [60, 78, 157]. The increase in dataset size and availability of data made possible the training of large networks on large-scale datasets, and thus the learning of rich representations that could later be exploited via feature extraction or transfer learning.

Deep learning initially focused on training single networks from large datasets in a single training session. However, at first deep neural networks did not perform very well on small domains: the millions of weights typically present in deep neural networks would overfit to the few available samples. Very soon it was found that one of the main advantages of deep learning is its ability to transfer knowledge to other domains. In this case, rather than training a neural network from scratch, the weights are initialized with the weights of a network trained on a large dataset – called the source dataset. Next, the weights are fine-tuned on the new labelled data – called the target dataset. This process allows the knowledge learned on a large domain to be transferred to new, often smaller, domains. Originally this was investigated mainly for supervised models [11], but later also for generative models [173].

However powerful training on massive labeled datasets or transfer learning may be, they still have several restrictions. Typically, networks are trained to be optimal in a closed world, meaning that they have to perform well on data from a known distribution. However, in the real-world this assumption can be often too restrictive. It is therefore believed important to enhance algorithms with lifelong learning capability which allows them to operate on an open world, i.e. learning new tasks while not forgetting previous tasks, adapting to new domains, joining knowledge from multiple models, detecting data which is not within the training data distribution, etc.

Therefore, in this thesis we explore some desirable characteristics for computer vision systems that should scale properly. When trying to learn large corpus of knowledge that expands or changes over time, it is useful to have a large system that contains all the knowledge acquired so far. However, when deploying the system for specific tasks or devices with limited computational processing, the knowledge specific to the task should be easily accessible without the interference of other non-related knowledge. When new data becomes available, the system should be able to integrate the knowledge without forgetting previously learned tasks. Furthermore, it is desirable that the learning of a new task would benefit from previously learned related knowledge to make the learning process easier and further understand the connections between those tasks to exploit them later on. Finally, as the system expands, it should be aware of the extent of its knowledge and the tasks that is capable of performing. This way, we avoid having a system that would claim high confidence in tasks that it has never seen. In summary, the characteristics that we aim to address in this thesis are:

- network compression: the ability to compress the system to a specific task, reducing the computational cost of deployment by removing unuseful information (chapter 2).
- continual learning: the ability to learn new tasks without forgetting any of the previously learned knowledge, while scaling gracefully (chapter 3 and 4).

- out-of-distribution detection: the ability to detect when an input is outside of the learned distributions, knowing the boundaries of the acquired knowledge (chapter 5).

In the following sections, we introduce each of those characteristics and identify the objectives we will pursue in this thesis.

1.1 Network compression

An advantage of the increase on large datasets availability is that there exist several models from different architectures pretrained on those datasets. Deep neural networks trained on those large datasets can be easily transferred to new domains with far fewer labeled examples by a process called fine-tuning. This has the advantage that representations learned in the large source domain can be exploited on smaller target domains [121]. However, networks designed to be optimal for the source task are often prohibitively large for the target task. Network compression aims to solve that issue by allowing for transfer to a reduced version of the network (see Fig. 1.1).

Network compression has received a great deal of attention recently due to the wide usage of mobile devices, which have limited computational capacities. Several approaches have addressed the problem of reducing computational resources by modifying the internal representation of each layer taking into account the in-

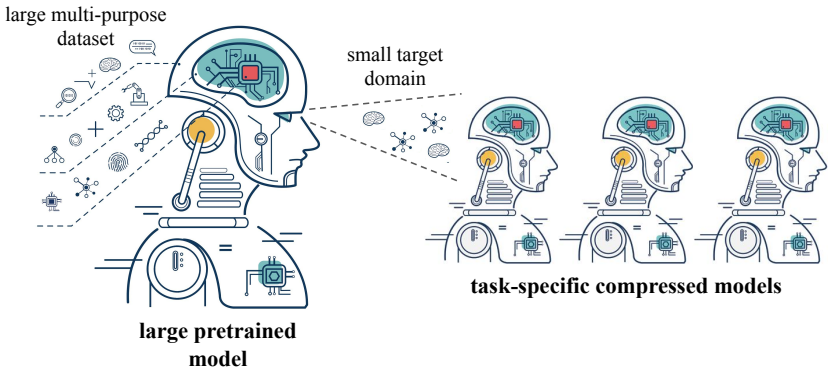

Figure 1.1 – Large networks trained on large datasets can be compressed into task-specific models that can be deployed to smaller devices with lower computational memory and cost.

herent redundancy of its parameters. Common approaches exploit linear structures within convolutional layers and approach each convolutional kernel using low-rank kernels [9, 37, 69]. The main idea relies on the fact that performing a convolution with the original kernels is equivalent to convolving with a set of base filters followed by a linear combination of their output. In this thesis, we study the use of activation statistics to compress models for a domain transfer classification task. When going from the source domain to the target domain, some filters might be focusing on aspects that were present in the source domain, but are irrelevant or non-existent in the target domain. Therefore, with that information we can reduce the number of parameters of the neural network and compress the model without performance loss. Based on this analysis we propose the following research objective:

> *Objective 1*: Pretrained networks contain large amounts of knowledge but are often too large to be used on certain devices. We aim to study compression in the context of domain adaptation, where knowledge unrelated to the target task can be dropped in exchange for computation efficiency.

More specifically, we address the compression of networks after domain transfer. We focus on compression algorithms based on low-rank matrix decomposition. Existing methods base compression solely on learned network weights and ignore the statistics of network activations. We show that domain transfer leads to large shifts in network activations and that it is desirable to take this into account when compressing. We demonstrate that considering activation statistics when compressing weights leads to a rank-constrained regression problem with a closed-form solution. Because our method takes into account the target domain, it can more optimally remove the redundancy in the weights. Experiments show that our proposed method significantly outperforms existing low-rank compression techniques. With our approach, the *fc6* layer of VGG-19 can be compressed 4x more than using truncated SVD alone – with only a minor or no loss in accuracy. When applied to domain-transferred networks it allows for compression down to only 5-20% of the original number of parameters with only a minor drop in performance.

If we create computer vision systems capable of extending their knowledge, those will be able to perform a large number of tasks and will grow over time. However, applications for specialized on specific tasks will always be necessary, and will probably need to be deployed efficiently. Therefore, it becomes useful to have large systems capable of doing large amounts of tasks and exploit the connections between them, while being able to compress specific knowledge into an easily deployable network for a specific task.

1.2 Continual learning

As more larger datasets become available and computational costs are reduced, models capable of solving larger tasks become also available. However, training a model each time that a new task is required to be learned might become unfeasible. Older data might be unavailable, new data might not be able to be stored due to privacy issues, or the frequency at which the system needs to be updated cannot support training a new model with all data frequently enough. A solution to these problems can be found in continual learning (also known as lifelong learning, incremental learning or neverending learning). The objective of continual learning is to be able to learn new tasks without the need to access data from previously learned tasks, except for maybe a small buffer of data.

When neural networks learn a new task, if no special measures are used, the new knowledge will be prioritized over the older knowledge, usually causing to forget the latter. This issue is commonly referred to as *catastrophic forgetting* [50, 74, 111] (see Fig. 1.2). As observed in humans presented with a new task to learn, new knowledge is acquired or integrated by leveraging knowledge from previous ones (forward transfer) as well as integrating the newly acquired knowledge into previous tasks (backward transfer) [44]. This makes for two desirable properties of continual learning systems. Continual learning methods must balance retaining knowledge from previous tasks while learning new knowledge for the current task. This problem is called the *stability-plasticity dilemma* [113]. However, it usually is difficult to estimate how much forgetting will happen when learning a new task even when having a stability-plasticity trade-off. Based on this we propose another research objective:

> *Objective 2*: For many practical applications which aim for continual learning, allowing for a variable amount of forgetting is undesirable. Therefore, we aim to design a learning system that can adapt to new tasks, while using previous knowledge, but without any forgetting.

In this thesis, we propose an approach without any forgetting to continual learning, where at inference the task-label is known. By using ternary masks on the activations we can upgrade a model to new tasks, reusing knowledge from previous tasks while not forgetting anything about them. Using masks prevents both catastrophic forgetting and backward transfer. We argue, and show experimentally, that avoiding the former largely compensates for the lack of the latter, which is rarely observed in practice. In contrast to earlier works, our masks are applied to the features (activations) of each layer instead of the weights. This considerably reduces the number of mask parameters to be added for each new task; with more than three orders of magnitude for most networks. The encoding of the ternary masks into two bits per feature creates very

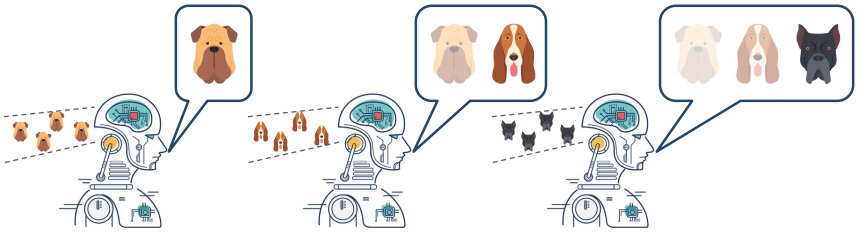

Figure 1.2 – Illustration of catastrophic forgetting of previous learned knowledge when learning new classes that have not been seen for a while are forgotten (represented as slowly fading away).

little overhead to the network, avoiding scalability issues. Our masks do not permit any changes to features which are used by previous tasks. As this may be too restrictive to allow learning of new tasks, we add task-specific feature normalization. This way, already learned features can adapt to the current task without changing the behavior of these features for previous tasks. Extensive experiments on several fine-grained datasets and ImageNet show that our method outperforms current state-of-the-art while reducing memory overhead in comparison to weight-based approaches.

In both Chapters 2 and 3 we take advantage of the memory reduction that comes from applying our proposed methods on the activations instead of the weights. In the first chapter, we use the information on the activations to reduce the amount of weights needed to perform a task. While in the second chapter we apply masks on the activations instead of weights to significantly reduce the memory overhead. This will show the benefits of activation-based methods over weight-based methods.

Although having no forgetting is desirable, in some cases it is not possible, specially when the task is not known at inference time. Because of that, class-incremental learning has been gathering more attention in the continual learning community. Class-incremental focuses on scenarios where a sequence of tasks is learned one at a time. However, unlike task-incremental learning, the task-ID is not known at inference time. This leads to our third research objective:

> *Objective 3*: As some systems become more dynamical and data becomes accessible in streams, systems should be able to learn while mitigating catastrophic forgetting. This desire has resulted in a wave of recent works, however a comprehensive comparison of their performance is currently lacking. Therefore, a survey of current state-of-the-art on incremental learning is needed.

For future learning systems continual learning is desirable, because it allows for: efficient resource usage by eliminating the need to retrain from scratch at the arrival of new data; reduced memory usage by preventing or limiting the amount of data required to be stored – also important when privacy limitations are imposed; and learning that more closely resembles human learning. The main challenge for incremental learning is catastrophic forgetting, which refers to the precipitous drop in performance on previously learned tasks after learning a new one. Incremental learning of deep neural networks has seen explosive growth in recent years. Initial work focused on task incremental learning, where a task-ID is provided at inference time. Recently we have seen a shift towards class-incremental learning where the learner must classify at inference time between all classes seen in previous tasks without recourse to a task-ID. In this thesis, we provide a complete survey of existing methods for incremental learning, and in particular we perform an extensive experimental evaluation on twelve class-incremental methods. We consider several new experimental scenarios, including a comparison of class-incremental methods on multiple large-scale datasets, investigation into small and large domain shifts, and comparison on various network architectures.

1.3 Out-of-distribution detection

Over time, computer vision systems should be able to accumulate a broad range of knowledge. However, it is important that the system is aware of the boundaries of such knowledge (see Fig. 1.3). In recent works [61, 82, 88], it was shown that current cross-entropy-based losses tend to promote too high confidence on predictions of data outside of the learned knowledge.

When neural networks process images which do not resemble the distribution seen during training, so called out-of-distribution images, they often make wrong predictions, and do so too confidently. The capability to detect out-of-distribution images is therefore crucial for many real-world applications. We divide out-of-distribution detection between novelty detection (images of classes which are not in the training set but are related to those), and anomaly detection (images with classes which are unrelated to the training set). By related we mean they contain the same type of objects, like digits in MNIST [80] and SVHN [117]. Most existing work has focused on anomaly detection, and has addressed this problem considering networks trained with the cross-entropy loss. Differently from them, we propose to use metric learning which does not have the drawback of the softmax layer (inherent to cross-entropy methods), which forces the network to divide its prediction power over the learned classes.

Objective 4: we aim to apply metric learning to the out-of-distribution detection problem, thus evading the problems introduced by using a cross-entropy loss. Metric learning can naturally be extended to new classes whereas classification networks are forced to produce an answer in terms of known classes.

To address this objective, we present an approach which consists on learning an embedding that can measure out-of-distribution samples using an Euclidean distance. The proposed loss learns an embedding which forms clusters out of the in-distribution classes using a contrastive loss [53]. When presented with out-of-distribution samples, they are not enforced to form a separate cluster but to separate from all in-distribution clusters instead. This forces the distance between out-of-distribution samples and the in-distribution classes to be larger and easier to detect. The use of metric learning instead of a cross-entropy loss and corresponding softmax scores is shown to be beneficial for more consistent out-of-distribution detection. Furthermore, by creating the embedding space, we allow to better distinguish between novel and anomalous samples and show that research should focus on the more challenging problem of novelty detection. The detection of new unseen classes can be easily used in continual learning pipelines to detect which tasks should be learned next. Finally, to illustrate that we present a new benchmark dataset which is closer to a real-world classification problem and in which we compare to the state-of-the-art.

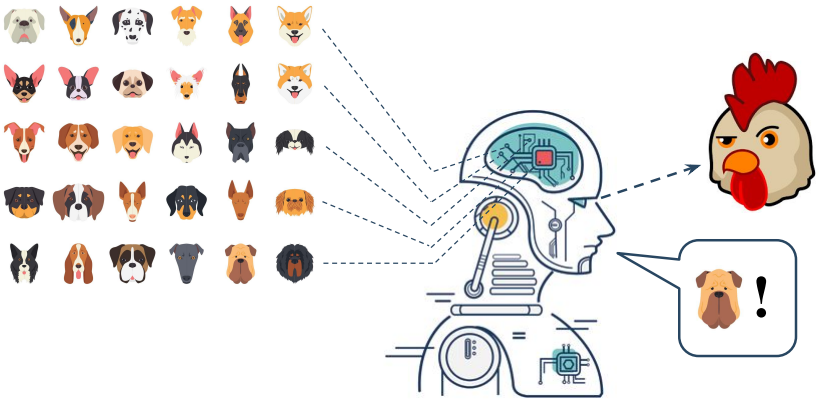

Figure 1.3 – When presented with a new sample outside of the learned distribution, neural networks will predict a class from the learned distribution with great confidence. Out-of-distribution detection proposes algorithms to address this problem.

2 Domain-adaptive deep network compression*

2.1 Introduction

One of the important factors in the success of deep learning for computer vision is the ease with which features, pretrained on large datasets such as Imagenet [36, 143] and Places2 [189], can be transferred to other computer vision domains. These new domains often have far fewer labeled samples available but which, due to the high correlation which exists between visual data in general, can exploit an already learned representation trained on large datasets. The most popular method to transfer the representations is by means of fine-tuning, where the network is initialized with the pretrained network weights, after which they are further adjusted on the smaller target domain [121]. These fine-tuned networks, which have the same size as the originally trained network, can then be applied to the task of the target domain (see Fig. 2.1). However, one must question whether a target domain task requires such a large network and whether the resulting network is not highly redundant.

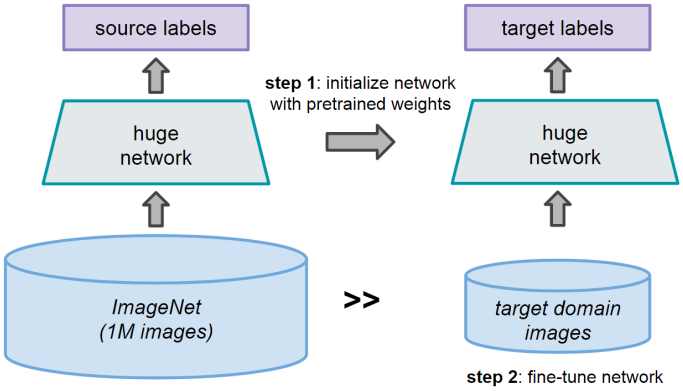

Figure 2.1 – Domain transfer main pipeline. Could we compress the network without losing accuracy on the target domain?

*This chapter is based on a publication in the IEEE International Conference in Computer Vision (ICCV), 2017 [110].

A drawback of Convolutional Neural Networks (CNNs) is that they generally require large amounts of memory and computational power (often provided by GPU). As a result they are less suitable for small devices, like cell phones, where requirements for energy efficiency limit CPU, GPU, and memory usage. This observation has motivated much research into network compression. Approaches include methods based on weight quantization [54, 133], weight removal from fully-connected [190] or convolutional layers [10], compact representations of convolutional layers through tensor decompositions [9, 37, 69], as well as training of thinner networks from predictions of a larger *teacher* network [63, 141].

One efficient method for the compression of fully-connected layers is based on applying singular value decomposition (SVD) to the weight matrix [37, 179, 190]. Compression is achieved by removing columns and rows related to the least significant singular values. Then, the original layer is replaced by two layers which have fewer parameters than the original layer. The method has been successfully applied to increase efficiency in detection networks like Fast R-CNN [46]. In these networks the truncated SVD approach is applied to the *fc6* and *fc7* layers, and the authors showed that with only a small drop in performance these layers can be compressed to 25% of their original sizes. In the original paper [37, 179, 190] the compression is always applied on the source domain, and no analysis of its efficiency for domain transfer exists.

In this chapter we investigate network compression in the context of domain transfer from a network pretrained on a large dataset to a smaller dataset representing the target domain. To the best of our knowledge we are the first to consider network compression within the context of domain transfer, even though this is one of the most common settings for the application of deep networks. Our approach is based on weight matrix decomposition that takes into account the activation statistics of the original network on the target domain training set[†]. We first adapt a pretrained network with fine-tuning to a new target domain and then proceed to compress this network. We argue that, because the statistics of the activations of network layers change from the source to the target domain, it is beneficial to take this shift into account. Most current compression methods do not consider activation statistics and base compression solely on the values in the weight matrices [37, 54, 81, 133, 179]. We show how the activation statistics can be exploited and that the resulting minimization can be written as a rank-constrained regression problem for which there exists a closed-form solution. We call our approach Domain Adaptive Low Rank (DALR) compression, since it is a low-rank approximation technique that takes into account the shift in activation statistics that occurs when transferring to other domains. As an additional contribution,

[†]With activation statistics we refer to their direct usage during compression, but we do not explicitly model the statistical distribution.

we show that the original SVD algorithm can be improved by compensating the bias for the activations.

The chapter is organized as follows. In the next section we discuss work from the literature related to network compression. In section 2.3 we discuss in detail the motivation behind our network compression approach, and in section 2.4 we show how network compression can be formulated as a rank constraint regression problem. In section 2.5 we report on a range of compression experiments performed on standard benchmarks. Finally, we conclude with a discussion of our contribution in section 2.6.

2.2 Related work

Network compression has received a great deal of attention recently. In this section we briefly review some of the works from the literature relevant to our approach.

Network pruning. A straightforward way to reduce the memory footprint of a neural network is by removing unimportant parameters. This process can be conducted while training [10, 56, 81, 190], or by analyzing the influence of each parameter once the network has been trained [89]. For instance, in [190], the authors use tensor low-rank constraints to (iteratively) reduce the number of parameters of fully-connected layers.

Computationally efficient layer representations. Several approaches have addressed the problem of reducing computational resources by modifying the internal representation of each layer taking into account the inherent redundancy of its parameters. Common approaches exploit linear structures within convolutional layers and approach each convolutional kernel using low-rank kernels [9, 37, 69]. The main idea relies on the fact that performing a convolution with the original kernels is equivalent to convolving with a set of base filters followed by a linear combination of their output. In [75], the authors propose two network layers that are based on dictionary learning to perform sparse representation learning, directly within the network. In general, these approaches show significant reduction in the computational needs with a minimum drop of performance.

Parameter quantization. Previous works mentioned above on efficient representations focus on modifying the representation of a complete layer. Parameter quantization is slightly different as it aims at finding efficient representations of each parameter individually (i.e., representing each parameter with fewer bits). A common practice to minimize the memory footprint of the network and reduce the computational cost during inference consists of training using 32-bits to represent the parameters while performing inference more efficiently using 16-bits without significantly affecting the performance. More aggressive quantization processes have been also analyzed in [49] where the authors propose an approach to directly thresholding values at 0 resulting

in a decrease of the top-1 performance on ImageNet by less tan 10%. More recently, several works have adopted quantization schemes into the training process [133]. For instance, in [133], the authors propose an approach to train a network directly using binary weights resulting in very efficient networks with a very limited accuracy.

In [54] the authors propose an approach to combine pruning, quantization and coding to reduce the computational complexity of the network. The authors first analyze the relevance of each parameter pruning the irrelevant ones and then, after fine-tuning the pruned network, the remaining parameters are quantized. Results show a significant reduction in the number of parameters (up to 35x) without affecting the performance of the network.

Network distillation. These approaches aim at mimicking a complicated model using a simpler one. The key idea consists of training an ensemble of large networks and then use their combined output to train a simpler model [19]. Several approaches have built on this idea to train the network based on the soft output of the ensemble [62], or to train the network mimicking the behavior not only of the last layer but also of intermediate ones [141].

All these methods for pruning, quantization, or compression in general, have shown results for reducing the footprint of networks and for reducing its computational complexity. However, they are usually applied to the same target domain as the one used for training the original network. In contrast, in this chapter we investigate network compression in the context of domain transfer. That is, compressing a network that has been trained on a generic large dataset in order to reduce its computational complexity when used in a different target domain using a smaller dataset. In this context, the most related work is the approach presented in [51] exploring non-negative matrix factorization to identify an interesting set of variables for domain transfer. However, the work in [51] does not consider network compression and focuses on unsupervised tasks.

2.3 Motivation

Training networks for a specific task starting from a network pretrained on a large, generic dataset has become very common practice, and to the best of our knowledge we are the first to consider compression of these types of domain-adapted networks. We investigate the compression of fully-connected layers by means of matrix decomposition. The basic principle is to decompose a weight matrix into two matrices which contain fewer parameters than the original weight matrix. This decomposition can then be applied to the network by replacing the original layer with two new layers (see Fig. 2.2). An existing approach is based on truncated Singular Value Decomposition [37, 179]. The decomposition of that method is determined solely by the weight

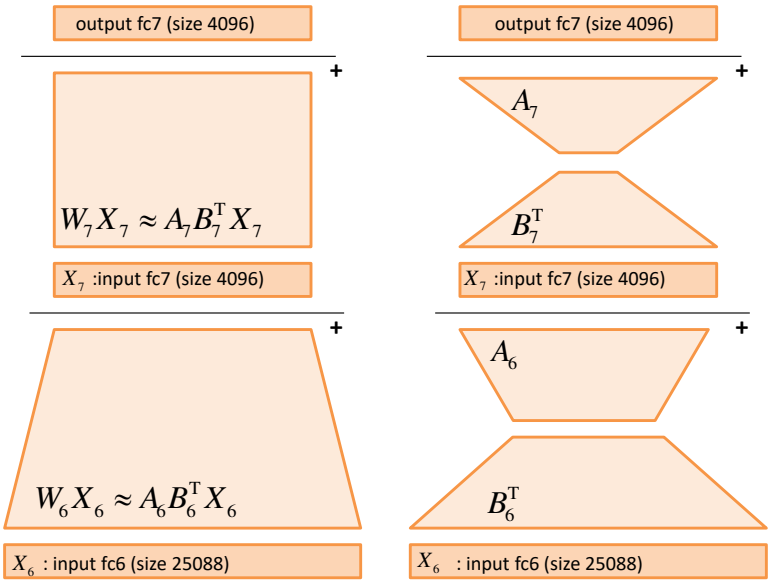

Figure 2.2 – Example of compressing last two layers of the VGG-16 network (*fc6*, *fc7*). The original weight matrix is approximated by two matrices. The main novelty in this chapter is that we consider the input X of each layer when compressing the corresponding weight matrix W. This is especially relevant when doing domain transfer from pretrained networks where activation statistics can be significantly skewed in the target domain.

matrix and ignores the statistics of layer activations in the new domain.

To gain some insight into this shifting of the activation distributions in deep networks when changing domain, we take a closer look at the inputs to the two fully-connected layers *fc6* and *fc7* (which are the output of *pool5* and *fc6*, respectively), of the VGG-19 network [157]. We analyze the *activation rate* of neurons which is the fraction of images in which a neuron has non-zero response. A value of 0.3 means that the neuron is activated in 30% of the images in the dataset. In Fig. 2.3 we show the activation rates of the VGG-19 network on ImageNet, on the CUB-200-2011 Birds dataset [170], on the Oxford 102 Flowers dataset [120] and on the Stanford 40 actions dataset [180].

The first thing to observe is that the activation rate is fairly constant across all the input dimensions (i.e. activation rate of neurons in the previous layer) when computed on the ImageNet dataset (i.e. source domain). Apparently the network has optimized

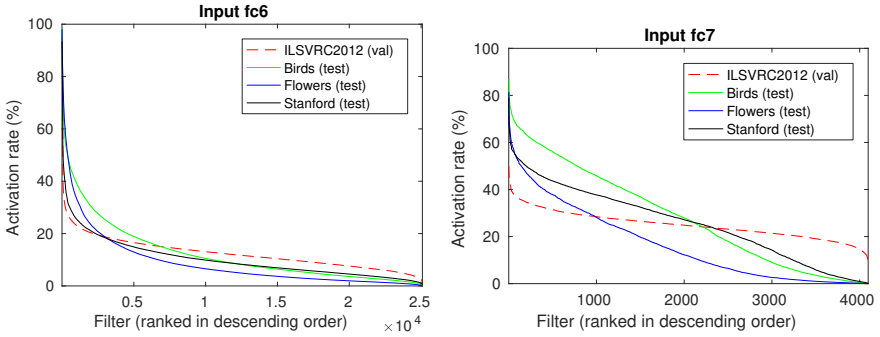

Figure 2.3 – Activation rates (ranked in decreasing order) of the input to (top) *fc6*, and (bottom) *fc7* for the Birds, Flowers and Stanford datasets in the VGG-19 trained on ILSVRC2012. The dimensions of the inputs are $7 \times 7 \times 512 = 25,088$ (output of *pool5*) and 4,096 (output of *fc6*), respectively. Note the more uniform distribution for ILSVRC2012 (no domain change).

its efficiency by learning representations which have a similar frequency of occurrence in the ImageNet dataset. However, if we look at the activation rates in the three target domains we see that the distributions are significantly skewed: a fraction of neurons is much more frequent than the rest, and the activation rates are lower than in the source domain. This is especially clear for the input to *fc7* where activation rates vary significantly. If we consider the percentage of input dimensions which accumulates 50% of the activations (which is the point where the area under the curve to the left is equal to the area under the curve to the right), we see a clear shift from ImageNet with 41.38% to 19.51% in Flowers, 24.93% in Birds and 29.44% in Actions (and from 32.29% to 14.61%, 19.13% and 25% for *fc6*, respectively). This clearly shows that there exists a significant change in the relative importance of neurons from previous layers, optimized for a source domain, when applied on new domains. Given this significant shift, we believe that it is important to take these activation statistics into account when compressing network layers after domain transfer. Keeping lower weights connected to high activation neurons can lead to more efficient compression rather than only focusing on the value of the weights themselves as is done by current compression methods.

2.4 Compression by matrix decomposition

We start by introducing some notation. Consider a single fully-connected layer, with input and bias $x, b \in \mathbb{R}^n$, output $y \in \mathbb{R}^m$ and layer weights $W \in \mathbb{R}^{m \times n}$, related according to:

$$y = Wx + b, \tag{2.1}$$

or when considering a set of p inputs to the layer:

$$Y = WX + b1_p^\mathsf{T}, \tag{2.2}$$

where 1_p is a vector with ones of size $p \times 1$, $Y \in \mathbb{R}^{m \times p}$ is the set of outputs, and $X \in \mathbb{R}^{n \times p}$ is the set of p inputs to the layer.

Several compression works have focused on compressing W into $\hat{W}$ so that $||W - \hat{W}||_F$ is minimized [37, 81, 179]. The novelty of our work is that we also consider the inputs X. As a consequence we will focus on compressing W into $\hat{W}$ in such a way that $||Y - \hat{Y}||_F$ is minimal.

2.4.1 Truncated SVD and Bias Compensation (BC)

One approach to compression is to apply SVD such that $W = USV^\mathsf{T}$ where $U \in \mathbb{R}^{m \times m}$, $S \in \mathbb{R}^{m \times n}$, $V \in \mathbb{R}^{n \times n}$ [81, 179]. The layer weights W can be approximated by keeping the k most significant singular vectors, $\hat{W} = \hat{U}\hat{S}\hat{V}^\mathsf{T}$ where $\hat{U} \in \mathbb{R}^{m \times k}$, $\hat{S} \in \mathbb{R}^{k \times k}$, $\hat{V} \in \mathbb{R}^{n \times k}$. Compression is obtained by replacing the original layer by two new ones: the first with weights $\hat{S}\hat{V}^\mathsf{T}$ and the second with weights $\hat{U}$. Note it is crucial that the two new layers contain fewer parameters than the original network (i.e. that $nm > (n+m)k$).

In this truncated SVD approach the bias term b of the original network is added to the second layer. We propose an alternative bias term which takes into account the inputs X. We define $W = \hat{W} + \breve{W}$, where $\breve{W}$ is the residual which is lost due to the approximation. We want to find the new bias that minimizes $||Y - \hat{Y}||_F$ given inputs X. Accordingly:

$$||Y - \hat{Y}||_F = ||WX + b1_p^\mathsf{T} - (\hat{W}X + \hat{b}1_p^\mathsf{T})||_F = ||\hat{b}1_p^\mathsf{T} - (b1_p^\mathsf{T} + \breve{W}X)||_F. \tag{2.3}$$

The bias which minimizes this is then:

$$\hat{b} = b + \breve{W}X1_p = b + \breve{W}\bar{x} \tag{2.4}$$

where $\bar{x} = X1_p$ is the mean input response. Note that if X were zero centered, then $\bar{x}$ would be zero and the optimal bias $\hat{b} = b$. However, since X is typically taken right after a ReLU layer, this is generally not the case and SVD can introduce a systematic bias in the output which in our case can be compensated by using Eq. 2.4.

2.4.2 Domain Adaptive Low Rank (DALR) matrix decomposition

In the previous subsection we considered the inputs to improve the reconstruction error by compensating for the shift in the bias. Here, we also take into account the inputs for the computation of the matrix decomposition. In contrast, the SVD decomposition does not take them into account. Especially in the case of domain transfer, where the statistics of the activations can significantly differ between source and target domain, decomposition of the weight matrix should consider this additional information. A network trained on ImageNet can be highly redundant when applied to for example a *flower* dataset; in this case most features important for man-made objects will be redundant.

The incorporation of input X is done by minimizing $||Y - \hat{Y}||_F$. We want to decompose the layer with weights W into two layers according to:

$$W \approx \hat{W} = AB^{\mathsf{T}}, \tag{2.5}$$

where $\hat{W} \in R^{m \times n}$, $A \in R^{m \times k}$ and $B \in R^{n \times k}$ again chosen such that $m \times n > (m+n) \times k$. We want the decomposition which minimizes:

$$\min_{A,B} ||Y - \hat{Y}||_F = \min_{A,B} ||WX - AB^{\mathsf{T}}X||_F, \tag{2.6}$$

where we have set $\hat{b} = b$ and subsequently removed it from the equation. Eq. 2.6 is a rank constrained regression problem which can be written as:

$$\begin{aligned} &\arg\min_{C} ||Z - CX||_F^2 + \lambda ||C||_F^2 \\ &\text{s.t. } \operatorname{rank}(C) \leq k, \end{aligned} \tag{2.7}$$

where $C = AB^{\mathsf{T}}$ and $Z = WX$, and we have added a ridge penalty which ensures that C is well-behaved even when X is highly co-linear. We can rewrite Eq 2.7 as:

$$\begin{aligned} &\arg\min_{C} ||Z^* - CX^*||_F^2 \\ &\text{s.t. } \operatorname{rank}(C) \leq k, \end{aligned} \tag{2.8}$$

where we use

$$X^*_{n\times(p+n)} = \begin{pmatrix} X & \sqrt{\lambda}I \end{pmatrix}, \quad \text{and} \tag{2.9}$$

$$Z^*_{m\times(p+n)} = \begin{pmatrix} Z & 0 \end{pmatrix}. \tag{2.10}$$

In Ashin [116] the authors show that there is a closed form solution for such minimization problems based on the SVD of Z. Applying SVD we obtain $Z = USV^\mathsf{T}$. Then the matrices A and B in Eq. 2.5 which minimize Eq. 2.8 are:

$$\begin{aligned} A &= \hat{U} \\ B &= \hat{U}^\mathsf{T} Z X^\mathsf{T} \left(XX^\mathsf{T} + \lambda I\right)^{-1} \end{aligned} \tag{2.11}$$

where $\hat{U} \in R^{m\times k}$ consists of the first k columns of U.

Network compression is obtained by replacing the layer weights W by two layers with weights B and A, just as in the truncated SVD approach. The first layer has no biases and the original biases b are added to the second layer. Again we could apply Eq. 2.4 to compensate the bias for the difference between W and $\hat{W}$. However, this was not found to further improve the results.

2.4.3 Reconstruction error analysis

We discussed three different approaches to compressing a weight matrix W. They lead to the following approximate outputs $\hat{Y}$:

$$\begin{aligned} \text{SVD}: \hat{Y} &= \hat{U}\hat{S}\hat{V}^\mathsf{T} X + b \\ \text{SVD + BC}: \hat{Y} &= \hat{U}\hat{S}\hat{V}^\mathsf{T} X + \hat{b} \\ \text{DALR}: \hat{Y} &= AB^\mathsf{T} X + b \end{aligned} \tag{2.12}$$

To analyze the ability of each method to approximate the original output Y we perform an analysis of the reconstruction error given by:

$$\varepsilon = ||Y - \hat{Y}||_F. \tag{2.13}$$

We compare the reconstruction errors on the CUB-200-2011 Birds and the Oxford 102 Flowers datasets. The reconstruction error is evaluated on the test set, whereas the inputs and outputs of the layers are extracted from the training set for computing the matrix approximations $\hat{Y}$. We provide results for *fc6* and *fc7*, the two fully-connected layers of the VGG-19 network.

In Figure 2.4 we show the results of this analysis. We see that bias compensation provides a drop in error with respect to SVD for layer *fc6*, however the gain is

insignificant for layer *fc7*. Our DALR method obtains lower errors for both layers on both datasets for most of the compression settings. This shows the importance of taking activation statistics into account during compression.

2.5 Experimental results

Here we report on a range of experiments to quantify the effectiveness of our network compression strategy. Although experiments are done using the MatConvNet library [169], we facilitate code in both matlab and python (Tensorflow [1]).[‡]

2.5.1 Datasets

We evaluate our DALR approach to network compression on a number of standard datasets for image recognition and object detection.

CUB-200-2011 Birds: consists of 11,788 images (5,994 train) of 200 different bird species [170]. Each image is annotated with bounding box, part location (head and body), and attribute labels. Part location and attributes are not used in the proposed experiments. However, bounding boxes are used when fine-tuning the model from VGG-19 pretrained on ImageNet in order to provide some data augmentation for the existing images.

Oxford 102 Flowers: consists of 8,189 images (2,040 train+val) of 102 species of flowers common in the United Kingdom [120]. Classes are not equally represented across the dataset, with the number of samples ranging from 40 to 258 per class.

Stanford 40 Actions: consists of 9,532 images (4,000 for training) of 40 categories that depict different human actions [180]. Classes are equally represented on the training set and all samples contain at least one human performing the corresponding action.

PASCAL 2007: consists of approximately 10,000 images (5,011 train+val) containing one or more instances of 20 classes [40]. The dataset contains 24,640 annotated objects, with the training and validation sets having a mean of 2.52 objects per image, and the test set a mean of 2.43 objects per image. This dataset is used to evaluate our approach for object detection.

ImageNet: consists of more than 14 million images of thousands of object classes. The dataset is organized using the nouns from the WordNet hierarchy, with each class having an average of about 500 images. We use CNNs pretrained on the 1,000-class ILSVRC subset.

[‡]Code available at: https://mmasana.github.io/DALR

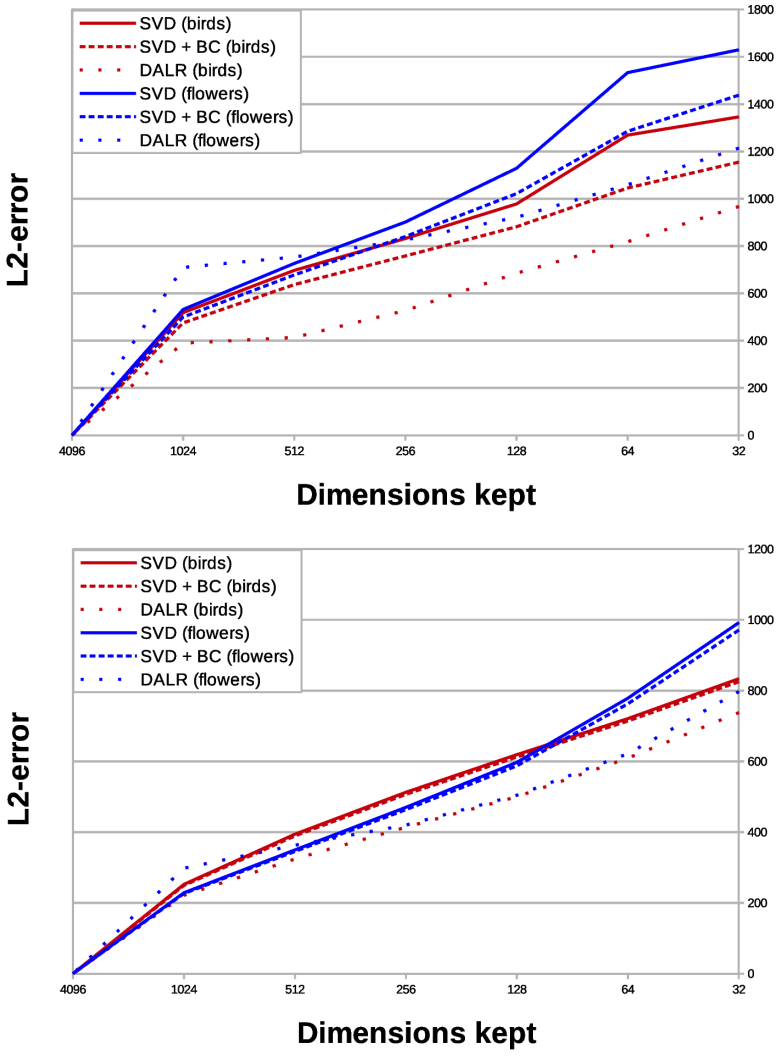

Figure 2.4 – Reconstruction error as a function of dimensions kept k for (top) *fc6* and (bottom) *fc7* layers on CUB-200-2011 Birds and Oxford 102 Flowers depending on the degree of compression.

dim kept params	32 0.91%	64 1.82%	128 3.64%	256 7.27%	512 14.54%	1024 29.08%
SVD	79.44	50.82	36.44	34.80	34.40	**34.18**
SVD + BC	73.41	46.54	36.21	34.82	34.33	34.22
DALR	**66.43**	**44.50**	**36.06**	**34.63**	**34.28**	34.20

Table 2.1 – Top-1 error rate results on ImageNet when compressing *fc6* in the source domain. We report the dimensions k kept in the layer and the percentage of parameters compressed. The uncompressed top-1 error rate is 34.24%.

2.5.2 Compression on source domain

Before performing experiments on domain adaptation, we applied Bias Compensation and DALR compression on the source domain and compare it to the truncated SVD method as baseline. We used the original VGG-19 network trained on the 1,000 ImageNet classes. Since ImageNet is a very large dataset, we randomly selected 50 images from each class to build our training set for DALR and truncated SVD compression. Then, we extracted the activations of this training set for the *fc6* and *fc7* layers to compress the network using DALR at different rates. The compressed networks were evaluated on the ImageNet validation set.

Results are provided in Tables 2.1 and 2.2. Results on *fc6* show a slight performance increase for the most restricting compression settings ($k = 32, 64, 128$). However the gain is relatively small. On *fc7* the results are even closer and the best results are obtained with bias compensation. Even though the proposed compression methods outperform standard SVD the gain is small when done on the source domain. This is most probably due to the fact that the inputs have relatively uniform activation rates and considering them does not significantly change the matrix decomposition (see also the discussion in Section 2.4.3). In general these results suggest that compressing a model without changing the domain can be done effectively with decompositions of filter weights (e.g. truncated SVD), and does not benefit significantly from the additional information coming from the inputs, in contrast to when there is a domain change involved (see the following sections).

2.5.3 Image recognition

Here we use the Birds, Flowers and Actions datasets to evaluate our compression strategies. We apply the Bias Compensation and the DALR compression techniques to fine-tuned VGG-19 models. For all image recognition experiments we used the publicly available MatConvNet library [169].

dim kept	32	64	128	256	512	1024
params	1.56%	3.13%	6.25%	12.5%	25%	50%
SVD	57.07	40.68	35.50	34.63	34.40	34.35
SVD + BC	**55.57**	**39.75**	**35.14**	**34.51**	**34.35**	**34.30**
DALR	56.32	40.25	35.26	34.54	34.40	34.33

Table 2.2 – Top-1 error rate results on ImageNet when compressing *fc7* in the source domain. The uncompressed top-1 error rate is 34.24%.

Fine-tuning: The VGG-19 [157] is trained on ImageNet. Since this network excelled on the ImageNet Large-Scale Visual Recognition Challenge in 2014 (ILSVRC-2014), it is a strong candidate as a pretrained CNN source for the transfer learning. Very deep CNNs are commonly used as pretrained CNNs in order to initialize the network parameters before fine-tuning. For each dataset, a fine-tuned version of VGG-19 was trained using only the training set. Although initialized with the VGG-19 weights, layers *fc6* and *fc7* are given a 0.1 multiplier to the network's learning rate. The number of outputs of the *fc8* (last) layer is changed to fit the number of classes in the dataset. All the convolutional layers are frozen and use the VGG-19 weights.

Evaluation metrics: All results for image recognition are reported in terms of classification accuracy. The compression rate of fully-connected layers is the percentage of the number of parameters of the compressed layer with respect to the original number of parameters.

Baseline performance: We first apply truncated SVD to the *fc6* and *fc7* weight matrices. In the original VGG-19 and fine-tuned models, W_{fc6} has 25088×4096 parameters and W_{fc7} has 4096×4096 parameters. Applying truncated SVD results in a decomposition of each weight matrix into the two smaller matrices. If we keep the k largest singular vectors, those two matrices will change to $(25088 + 4096)k$ and $(4096 + 4096)k$ parameters for *fc6* and *fc7* respectively. Since SVD does not take into account activations, and there is no compression method to our knowledge that uses activations in order to reduce the number of parameters in the weight matrices, we also show results for activation-based pruning. The pruning strategy consists of removing the rows or columns of the weight matrices which are less active for that specific dataset, following our earlier work in [109].

Results: Tables 2.3 to 2.8 show the performance of compressing the *fc6* and *fc7* layers using SVD and pruning baselines, as well as the proposed Bias Compensation and DALR techniques. Results confirm the tendency observed in the analysis of the L2-error reconstruction curves in Figure 2.4. DALR compression has a better

dim kept params	32 0.91%	64 1.82%	128 3.64%	256 7.27%	512 14.54%	1024 29.08%
SVD	16.83	36.47	51.74	54.47	54.85	55.25
SVD + BC	27.91	46.00	53.50	54.83	55.21	55.44
Pruning (mean)	4.30	8.06	12.57	25.82	37.25	50.41
Pruning (max)	4.06	7.01	15.26	25.27	36.69	48.95
DALR	**48.81**	**54.51**	**55.78**	**55.85**	**55.71**	**55.82**

Table 2.3 – Birds fc6 compression - original accuracy: 55.73%.

dim kept params	32 1.56%	64 3.13%	128 6.25%	256 12.5%	512 25%	1024 50%
SVD	26.86	44.13	52.19	54.30	54.80	55.13
SVD + BC	28.72	47.07	53.62	54.45	55.02	55.30
Pruning (mean)	2.49	3.45	9.15	18.31	34.05	47.24
Pruning (max)	6.46	9.60	13.84	22.63	33.28	45.67
DALR	**51.21**	**54.16**	**55.21**	**55.59**	**55.71**	**55.85**

Table 2.4 – Birds fc7 compression - original accuracy: 55.73%.

performance than the other methods at the same compression rates on both *fc6* and *fc7* for Birds, Flowers and Actions. In all our experiments DALR provides a slight boost in performance even when compressing to 25% of the original parameters. Bias compensation slightly improves the original SVD method on both layers except on Flowers for *fc7*. Since the *fc6* layer has more parameters, it is the layer that allows for more compression at a lower loss in performance. The advantages of DALR are especially clear for that layer, and for a typical setting where one would accept a loss of accuracy of around one percent, truncated SVD must retain between 4x and 8x the number of parameters compared to DALR to maintain the same level of performance. Finally, both pruning methods are consistently outperformed by compression methods, probably due to the effect pruning has on subsequent layers (*fc7* and *fc8*).

In the previous experiment we evaluated the layer compression separately for *fc6* and *fc7*. To get a better understanding of the potential joint compression, we perform a compression experiment with DALR on both layers simultaneously. In order to find suitable compression pairs for both layers at the same time, we implemented an iterative solution. At each step, we slightly increase the compression on both layers.

dim kept params	32 0.91%	64 1.82%	128 3.64%	256 7.27%	512 14.54%	1024 29.08%
SVD	14.00	47.37	59.90	75.07	77.72	78.61
SVD + BC	29.55	57.93	71.96	75.91	78.00	78.63
Pruning (mean)	1.42	4.91	19.74	49.07	71.23	77.64
Pruning (max)	1.81	4.96	10.36	33.99	60.14	74.78
DALR	**72.17**	**76.42**	**77.95**	**78.22**	**78.94**	**78.94**

Table 2.5 – Flowers fc6 compression - original accuracy: 78.84%.

dim kept params	32 1.56%	64 3.13%	128 6.25%	256 12.5%	512 25%	1024 50%
SVD	58.14	70.30	75.10	77.02	78.01	78.58
SVD + BC	57.07	70.14	75.53	77.15	78.00	78.50
Pruning (mean)	19.13	25.81	37.26	55.80	67.54	75.33
Pruning (max)	8.18	18.91	27.06	42.95	62.69	72.63
DALR	**72.32**	**76.55**	**77.79**	**78.48**	**78.86**	**78.86**

Table 2.6 – Flowers fc7 compression - original accuracy: 78.84%.

dim kept params	32 0.91%	64 1.82%	128 3.64%	256 7.27%	512 14.54%	1024 29.08%
SVD	38.18	55.59	66.59	67.97	68.38	68.94
SVD + BC	46.76	60.14	66.96	68.17	68.40	69.00
Pruning (mean)	3.98	7.27	13.70	32.14	52.02	62.46
Pruning (max)	5.44	13.12	26.72	46.11	53.40	63.16
DALR	**64.70**	**68.33**	**69.31**	**69.65**	**69.54**	**69.52**

Table 2.7 – Actions fc6 compression - original accuracy: 68.73%.

dim kept params	32 1.56%	64 3.13%	128 6.25%	256 12.5%	512 25%	1024 50%
SVD	50.98	62.27	66.72	68.08	68.44	68.60
SVD + BC	53.62	62.74	67.32	68.37	68.67	68.82
Pruning (mean)	12.46	17.57	25.23	34.20	51.05	61.19
Pruning (max)	14.84	20.37	22.90	36.35	49.30	59.33
DALR	**63.87**	**67.46**	**68.44**	**68.75**	**68.78**	**68.78**

Table 2.8 – Actions fc7 compression - original accuracy: 68.73%.

Then, both options are evaluated on the validation set, and the compression rate with better performance is applied to the network and used on the next iteration. When both steps in compression exceed a defined drop in performance (here set to a 1% accuracy drop), the iterative process stops and the compressed network is evaluated on the test set. Results are shown in Figure 2.5. This implementation tends to compress *fc6* more because it has more room for compression than *fc7*, as seen also in the experiments reported in Tables 2.3 to 2.8. The results show that we can compress the parameters of the fully-connected layers to as few as 14.88% for Flowers, as few as 6.81% for Birds, and as few as 29.85% for Actions while maintaining close to optimal performance.

2.5.4 Object detection

One of the most successful approaches to object detection is RCNN [47] (including its Fast [46] and Faster [137] variants). This approach is also an example of the effectiveness of the fine-tuning approach to domain transfer, and also of the importance of network compression for efficient detection. The authors of [46] analyzed the timings of forward layer computation and found that 45% of all computation time was spent in *fc6* and *fc7*. They then applied truncated SVD to compress these layers to 25% of their original size. This compression however came with a small drop in performance of 0.3 mAP in detection on PASCAL 2007.

For comparison, we have also run SVD with bias compensation and our compression approach based on low-rank matrix decomposition. Results are presented in Table 2.9. Here we apply compression at varying rates to *fc6* (which contains significantly more parameters), and compress *fc7* to 256 pairs of basis vectors (which is the same number used in [46]). What we see here is that at the same compression rate for *fc6* (1,024) proposed in [137], our low-rank compression approach does not impact performance and performs equal to the uncompressed network. When we

	aeroplane	bicycle	bird	boat	bottle	bus	car
No Compression	75.9	77.4	65.3	53.9	38.0	76.8	78.2
SVD @ 1024	74.4	77.6	65.9	54.9	38.4	76.7	78.2
SVD + BC @ 1024	73.9	77.8	65.9	55.0	38.5	76.7	78.3
DALR @ 1024	74.6	77.7	66.4	53.7	37.6	77.0	78.1
SVD @ 768	74.2	77.5	65.8	53.5	37.8	76.6	78.3
SVD + BC @ 768	74.4	77.5	65.6	53.8	38.0	77.3	78.2
DALR @ 768	74.2	77.5	66.7	53.8	37.8	77.4	78.1

	cat	chair	cow	diningtable	dog	horse	motorbike
No Compression	80.9	40.6	74.0	67.2	79.4	82.4	74.9
SVD @ 1024	81.6	40.0	73.0	65.9	78.9	81.9	75.7
SVD + BC @ 1024	81.5	40.1	72.6	66.4	79.1	82.2	75.7
DALR @ 1024	81.7	40.5	73.3	67.4	79.7	81.9	74.8
SVD @ 768	82.3	39.8	72.5	66.1	79.4	81.5	74.2
SVD + BC @ 768	82.2	39.7	72.5	66.3	79.7	82.1	75.4
DALR @ 768	82.0	40.1	72.7	66.2	79.0	81.7	75.5

	person	pottedplant	sheep	sofa	train	tvmonitor	mAP
No Compression	66.2	33.4	66.0	67.3	73.3	67.1	**66.9**
SVD @ 1024	65.6	33.7	65.3	67.3	72.4	65.7	**66.6**
SVD + BC @ 1024	65.7	33.9	65.5	67.5	72.1	66.1	**66.7**
DALR @ 1024	65.9	34.0	65.9	66.9	73.7	66.3	**66.9**
SVD @ 768	65.6	33.8	64.7	67.9	71.7	66.3	**66.5**
SVD + BC @ 768	65.7	34.0	65.1	68.1	71.7	66.0	**66.7**
DALR @ 768	66.1	33.9	65.8	66.7	73.2	66.0	**66.7**

Table 2.9 – Compression and bias compensation on Fast-RCNN on PASCAL 2007.

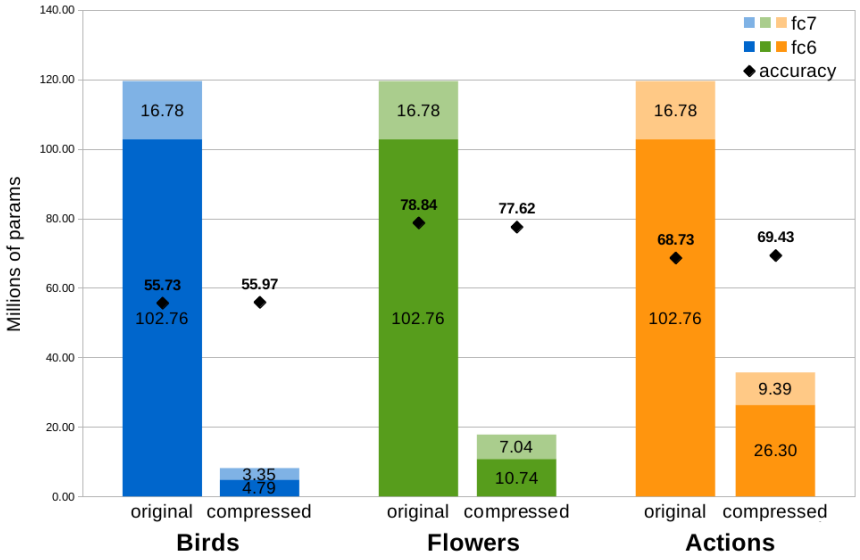

Figure 2.5 – Reduction in number of parameters for both *fc6* and *fc7*.

increase the compression rate of *fc6* (768) we see a drop of 0.4 mAP for standard SVD and only half of that for both SVD with bias compensation and DALR.

2.6 Conclusions

We proposed a compression method for domain-adapted networks. Networks which are designed to be optimal on a large source domain are often overdimensioned for the target domain. We demonstrated that networks trained on a specific domain tend to have neurons with relatively flat activations rates, indicating that almost all neurons are equally important in the network. However, after transferring to a target domain, activation rates tend to be skewed. This motivated us to consider activation statistics in the compression process. We show that compression which takes activations into account can be formulated as a rank-constrained regression problem which has a closed-form solution. As an additional contribution we show how to compensate the bias for the matrix approximation done by SVD. This is consistently shown to obtain improved results over standard SVD.

Experiments show that DALR not only removes redundancy in the weights, but also balances better the parameter budget by keeping useful domain-specific parameters

while removing unnecessary source-domain ones, thus achieving higher accuracy with fewer parameters, in contrast to truncated SVD, which is blind to the target domain. On further experiments in image recognition and object detection, the DALR method significantly outperforms existing low-rank compression techniques. With our approach, the *fc6* layer of VGG-19 can be compressed 4x more than using truncated SVD alone – with only minor or no loss in accuracy.

The Bias Compensation and DALR techniques were applied to fully-connected layers. To show the effectiveness of those methods we applied them to standard networks with large fully-connected layers. On more recent networks, like ResNets [60], most of the computation has moved to convolutional layers, and the impact of the proposed method would be restricted to the last layer. However, VGG-like networks are very much used in current architectures [12, 136, 189]. Extending the proposed compression method to convolutional layers is an important research question to address in future works.

This chapter shows that domain transferred networks can be significantly compressed. The amount of compression seems to correlate with the similarity [11, 181] between the source and target domain when we compare it to the ordering proposed in [11] (see Table II). According to this ordering, the similarity with respect to ImageNet in descending order is image classification (PASCAL), fine-grained recognition (Birds, Flowers) and compositional (Actions). We found that higher compression rates can be applied in target domains further away from the source domain.

3 Continual learning without any forgetting*

3.1 Introduction

Fine-tuning has been established as the most common method to use when learning a new task on top of an already learned one. This works well if you no longer require the system to perform the previous task. However, in many real-world situations one is interested in learning consecutive tasks, all of which, in the end, the system should be able to perform all. This is the setting studied in lifelong learning, also referred to as sequential, incremental or continual learning. In this setting, the popular approach of fine-tuning suffers from catastrophic forgetting [44, 50, 86, 111, 113]: all network capabilities are used for learning the new task, which leads to forgetting of the previous ones.

A popular strategy to avoid this is to use weight-importance loss proxies or regularizers [4, 74, 83]. These approaches compute an importance score for each of the weight parameters of the model based on previous tasks and use this to decide which weights can be modified for the current one. A drawback of these methods is that they need to store an extra variable (the importance score) for each weight. This leads to an overhead of a float per weight parameter, i.e. double the number of parameters which have to be stored. Other methods work with a binary mask to select part of the model for each task [101, 102]. This leads to an overhead of one bit per task per weight parameter. Finally, some methods directly make a copy of the network [144] or rely on the storage of exemplars [26, 97, 135, 138], which again increases memory consumption and renders these methods unsuitable when privacy requirements forbid storing of data.

In this chapter, we advocate computing a mask at the level of features† (activations) instead of at the level of the weights. We need the mask to be ternary, i.e. adding a third state for allowing features to be used during the forward pass while being masked during the backward pass. That allows the model to reuse the representations from previous tasks without modifying them and introducing forgetting. This drastically reduces the number of extra parameters that must be stored. As an example, the

*This chapter is based on a submission to a journal, 2020 [107].

†With features or activations we refer to the inputs of a layer that are also the output of a previous layer. The input images can be considered features for the network to perform the task at hand, but here we only use the term features for the activations between layers.

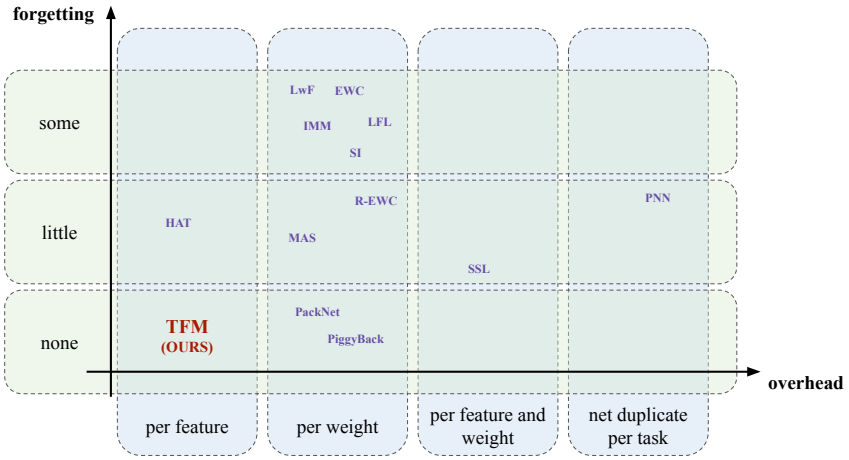

Figure 3.1 – Task-incremental learning comparison of methods depending on how much forgetting they allow and how much overhead they need.

popular Alexnet architecture [78] has around 60 million weights, while having fewer than 10k features. One earlier method that builds on this idea is HAT [153], which stores an attention value for each feature for each task. Recently, SSL [8] also brings attention to the activation neurons by promoting sparsity with different losses inspired by lateral inhibition in the mammalian brain. Those two recent works stress the importance of focusing on the features instead of the weights, not only because of the reduction in memory overhead, but also because they allow for better performance and less forgetting. However, both methods still allow some forgetting as new tasks are learned.

Over time, forgetting typically increases with the number of tasks [4, 74, 83, 97, 135, 153]. However, for many practical systems, it is undesirable that the accuracy of the system deteriorate over time as the system learns new tasks. Moreover, under these settings, the user typically has no control on the amount of forgetting, i.e. there are no guarantees on the performance of the system after new tasks have been added. Even worse: to the user, it is unknown how much the system has actually forgotten or how well the system still performs on older tasks.

For these reasons, some works have studied systems which perform continual learning without any forgetting at all. Currently, apart from the methods that make copies of the network for each task [144], mask-based approaches are the only ones that guarantee no-forgetting [101, 102]. Indeed, all methods that allow backward propagation into the parameters of previously learned tasks have no control on the

amount of forgetting. Not updating the weights used for previous tasks using a binary mask prevents any forgetting. In the case of recent approaches PackNet [102] and PiggyBack [101], this is enforced by binary-masking weights or learning masks that will be binarized after the task is learned. However, both these non-forgetting methods mask weights and therefore have a larger memory overhead than methods which would put the masks on the features instead. Another drawback of [101] is that it requires a backbone network as a starting point.

Compared to previous work, we propose a method for continual learning which does not suffer from any forgetting while introducing minimal overhead (see Fig. 3.1). Due to the nature of our proposed mask-based approach, we will only focus on evaluation of task-aware experimental setups. Instead of applying masks to all weights in the network, we propose to move the masks to the feature level, thereby significantly reducing the memory overhead. Our initial method requires only a 2-bit mask value for each activation for each task. In addition, we introduce a task-dependent normalization on the features. This allows adjustment of previously learned features to be of more optimal use for later tasks, without changing the performance or weights assigned to previous tasks. This introduces an additional memory overhead of two floats per activation per task. Nevertheless, this method still has a significantly lower memory overhead than any method which stores additional parameters per weight [4, 74, 83, 90, 101, 102, 185].

The remainder of the chapter is organized as follows. In the next section we review existing continual learning methods. In Section 3.3, we discuss masks on the features and introduce the proposed method Ternary Feature Masks (TFM). Then, we present the experimental setup and analysis of the results in Section 3.4 and conclude in Section 3.5.

3.2 Related work

Lifelong learning in the proposed continual learning setup has been addressed in multiple prior works. A large number of these use regularization-based techniques to reduce catastrophic forgetting without having to store raw input. They can be divided into two main families. Distillation approaches use teacher-student setups that aim at preserving the output of the teacher model on the new data distribution [72, 87, 131, 144, 188]. The output for each of the previous tasks is encoded using targets, exemplars or other representations and constrained when learning the new task. Model-based approaches [4, 8, 24, 74, 83, 90, 185] focus on overcoming catastrophic forgetting by defining the importance of the weights in the network. By penalizing changes to important weights, that have a big impact on the network output or performance, the loss protects previous information.

Learning without Forgetting (LwF) proposes to use the knowledge distillation loss [87] to preserve the performance of previous tasks. However, if the data distribution of the new task is very different from the previous tasks, performance drops drastically [6]. In order to solve this, [135] (iCaRL) stores a subset of each task's data as exemplars; while [131] (EBLL) solves the issue by learning undercomplete autoencoders for each task. Less Forgetting Learning (LFL) is also similar to LwF, preserving the previous tasks' performance by penalizing changes on the shared representation [72]. This approach is based on the assumption that the task-specific decision boundaries should not change, and freezes the last layer instead of mitigating the change with a loss. Expert Gate [6] (EGate) learns a model for each task and an autoencoder gate which will choose the model to be used. Recently, Deep Model Consolidation (DMC) proposes to learn new classes separately and then learn a final model with double distillation and extra unlabelled data [188]. However, most of these methods need a pre-processing step before each task. Furthermore, a main issue is also the scalability when learning many tasks, since the described methods have to store data, autoencoders, or larger models for each new task.

When learning a new task, most model-based approaches apply a smooth penalty for changing weights, proportional to their importance for previous tasks [4, 74, 83, 90, 185]. One of the main issues for these methods is that, depending on the task relatedness and the capacity of the network, they might over or under-estimate the importance of those weights. The main difference among these methods is how the importance of weights is calculated. In Elastic Weight Consolidation (EWC) an approximation of the diagonal of the Fisher Information Matrix (FIM) is used [74]. Rotated EWC proposes a rotation of the weight space to get a better approximation of the FIM [90]. In Incremental Moment Matching (IMM) the moments of the posterior distribution are matched incrementally [83]. They assume a diagonal covariance matrix which means that there is no correlation among the parameters, and selectively balance between two weights using variance information. Synaptic Intelligence (SI) computes the importance weights in an online fashion by storing how much the loss would change for each parameter over the training [185]. However, this method is prone to under-estimating when using pretrained networks, while over-estimating by relying too much on the gradient descent of some batches. Memory Aware Synapses (MAS) computes the weight importance in an online unsupervised way, connecting their approach with Hebbian learning [4]. It promotes the importance of those weights that even with small changes have a big effect on the network predictions. Finally, those approaches that learn an importance parameter for each weight can be combined with Selfless Sequential Learning (SSL) which imposes sparsity or local neuron inhibition on the neuron activations with a loss term [8].

Other works use different underlying methods. Progressive Neural Networks (PNN) add lateral connections at each layer of the network to a duplicate of that

Family	Method	Revisit data	Require Backbone net	Easily expandable	Overhead	Forgetting	Forward transfer	Features or weights
Baseline	Finetune	No	No	Yes	None	Yes	Little	neither
	Joint	Yes	No	Yes	None	No	Little	neither
	Freeze	No	Yes	Yes	None	No	Backbone only	neither
Distillation	LwF [87]	No	No	No	1 float *pp*	Some	Yes	weights
	LFL [72]	No	No	No	1 float *pp*	Some	Yes	weights
	PNN [144]	No	No	Yes	duplicate *pt*	Some	Yes	weights
	P&C [151]	No	No	No	extra network	Some	Little	weights
Model-based	EWC [74]	No	No	No	1 float *pp*	Some	Yes	weights
	R-EWC [90]	No	No	No	1 float *pp*	Some	Yes	weights
	IMM [83]	No	No	No	1 float *pp pt*	Some	Yes	weights
	SI [185]	No	No	No	1 float *pp*	Some	Yes	weights
	MAS [4]	No	No	No	1 float *pp*	Some	Yes	weights
	SSL [8]	No	No	No	1 float *pfp*	Some	Yes	both
Mask-based	PackNet [102]	No	No	No	1 int *pp pt*	No	Yes	weights
	PiggyBack [101]	No	Yes	No	1 bit *pp pt*	No	Backbone only	weights
	HAT [153]	No	No	No	1 float *pf pt*	Some	Yes	features
	TFM w/o FN (Ours)	No	No	Yes	2 bits *pf pt*	No	Yes	features
	TFM (Ours)	No	No	Yes	2 bits + 2 floats *pf pt*	No	Yes	features

Table 3.1 – Summary of related work characteristics. *pf*: per feature, *pp*: per parameter, *pt*: per task, *pfp*: per feature and parameter.

layer [144]. Then, the new column learns the new task while the old one keeps the weights fixed, meaning that resources are duplicated each time a task is added. This approach leads to non-forgetting while making the knowledge of previous tasks available during the learning of a new one through distillation. However, as each new task adds a column with the corresponding connections, the overhead scales considerably with the number of tasks. Progress and Compress (P&C) expands the idea of PNN with the use of EWC but keeping the number of parameters constant [151]. They propose a two-component setup with a knowledge base and an active column that follows a similar setup as PNN with lateral connections. Learn to Grow (LtG) proposes a two-part approach with a neural structure optimization component and a learning component which fine-tunes the parameters [85]. The neural structure component allows for each layer to reuse existing weights, adapt them or grow the network. In the worst case scenario, layers have to be added which makes the growth linear in the number of tasks. The fine-tuning component does parameter optimization and can be fixed or use an existing approach such as EWC to avoid catastrophic forgetting.

Apart from the above mentioned families, some recent works use masks to directly influence or completely remove forgetting for each parameter. We refer to this family of approaches as mask-based. These approaches give better control on the flow of gradients through the network and have the benefit of reducing or removing catastrophic forgetting more than the previously mentioned alternatives, at the cost of depleting network capacity faster as new tasks are learned. PathNet uses evolutionary strategies to learn selective routing through the weights [43]. However, it is not end-to-end differentiable and computationally very expensive. PackNet trains the network with available weights, then prunes the less relevant weights and retrains with a smaller subset of them [102]. Those weights are then not available for further learning of new tasks, which quickly reduces the capacity of the network. This results in lower number of parameters being free and performance dropping quickly when considering long sequences of tasks. Piggyback proposes to use a pretrained network as a backbone and then uses binary masks on the weights to create different sub-networks for each task [101]. The main drawback with that approach is the backbone network itself, which is crucial to being able to learn each task on top of it and cannot have a too different distribution from them. In the case of PackNet, the use of a backbone network is not needed but recommended since it becomes more difficult to learn tasks from scratch (especially with larger networks) than it is from a network fine-tuned on a large-scale dataset. In these last two methods, although the binary mask for each parameter has almost no overhead for the network usage, storing a mask for each task does not scale very well after a certain number of tasks. Finally, Hard Attention to the Task (HAT) proposes a hard attention mechanism on the features after each layer [153]. The attention embeddings are non-binary and are learned together with each task and conditioned by the previous tasks' attentions. Because of the annealing of the

slope of the sigmoid used on the embeddings, [153] also define a different backward propagation through their attention mechanism with gradient compensation. This approach offers plasticity to the embeddings in order to learn them, but also allows the possibility to forget previous tasks during the backpropagation step. A no-forgetting idea is discussed in the appendices of their manuscript with a note on binary masks, connecting the removal of plasticity to the *inhibitory synapses* idea [112].

In our proposed approach we take the latter side of that balance, using rigid masks that reduce plasticity but also ensure non-forgetting of previous tasks. Our approach also focuses on a natural expansion of the capacity of the network, which is not addressed in HAT and most of the previous related work. Our proposed approach uses masks on the features of the network to have a better control over which weights can be modified while learning new tasks. At the same time, the mask being ternary allows weights fixed for previous tasks to be used on new tasks without modifying those weights. This masking strategy allows the network to not forget anything from previous tasks and reduce the computational overhead in comparison to masking the weights.

We show in Table 3.1 a comprehensive overview of some of the characteristics that we consider most relevant to the experimental setup we propose. Our proposed method is unique in that it combines being expandable, having low overhead, and having no forgetting. All other methods have to choose only one of those three characteristics, if any. Finally, we want to mention that there have been some methods proposed that consider a memory budget [135], or an online setup [98], and methods with exemplars where each sample is used only once [26, 97, 138]. However, those setups are significantly different from the sequential setup we propose and therefore we do not compare to these. The same is true of Learn to Grow, which belongs to the optimal architecture search setup and can be mixed or extended with the above mentioned approaches.

3.3 Learning without any forgetting

Here we propose our method for task-aware continual learning, which is designed to learn new tasks without any forgetting of previously learned ones. As discussed in the introduction, in order to enforce non-forgetting of previous tasks, the use of masks that create rigid states is efficient. Freezing the weights once learned allows keeping the knowledge fixed without the possibility of forgetting. Works which have addressed this problem have focused on *weight-masks* where an additional parameter is learned for each weight in the network [101, 102]. From a network overhead point of view, we argue that it is, however, better to work with *feature-masks* which learn an additional parameter for each feature in the network. In Table 3.2 we compare the number of

Network	#weights	#features
LeNet [80]	59,956	226
AlexNet [78]	54,547,712	9,344
VGGNet [157]	119,579,904	10,880
ResNet-50 [60]	19,330,304	22,720

Table 3.2 – Difference between weights and features for different common network architectures. Last fully-connected is not taken into account since it depends on the number of classes.

weights and features in several popular networks. The table clearly shows that the overhead is significantly lower: on average weight-masks are a quadratic factor bigger than feature-masks.

First, we will discuss binary masks and how those can easily encode the parts that we want to learn and the parts that we want to fix. Afterwards, we will explore what happens when we want to learn more than one task and how binary masks need to be extended to ternary masks to make room for a new state. Finally, we will explore the use of feature normalization to allow for less rigid learning of new tasks.

3.3.1 Binary feature masks

Using binary feature masks on neural networks means that the masked neuron will have one of two states (0 or 1). When the masks are directly multiplied by the neuron activations, it will either use the corresponding filters or not (same for the backward pass, which will either be applied or not). Then, for each task we have a binary mask with the neurons that can be used or not. Since we do not allow for any forgetting, those masks will have to be disjoint. In Fig. 3.2 we show an example with two tasks where each is only allowed to use neurons not used by the other. A large number of connections are completely unused, making the two sub-networks totally separable from one another.

Consider a fully-connected layer (the theory can easily be extended to convolutional layers). The output of the layer is $y = Wx$ where $y \in \mathbb{R}^{p\times 1}$, $x \in \mathbb{R}^{q\times 1}$ and $W \in \mathbb{R}^{p\times q}$. The binary feature mask for the forward pass is defined as follows:

$$y = (Wx) \odot m^{t,l}, \tag{3.1}$$

where $m^{t,l} \in \mathbb{R}^{p\times 1}$ refers to the mask for task t at layer l and $\odot$ is an element-wise multiplication. Masks from different tasks are forced to select different features

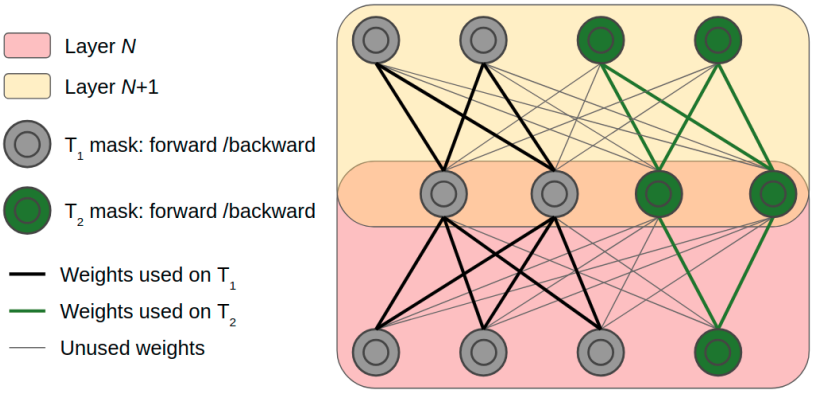

Figure 3.2 – Binary masks encode two states: used or unused. In this case, neurons in grey are learnable for task 1 but neurons in green are not, and the opposite is true for task 2. All grey weights are unused by both tasks.

$(\left(m^{s,l}\right)^{\mathrm{T}} \cdot m^{t,l} = 0 \; \forall \, s \neq t)$). The backward pass for training task t is defined as:

$$\frac{\partial \mathscr{L}}{\partial W_{ij}} = (m_i^{t,l} \wedge m_j^{t,l-1}) x_j \frac{\partial \mathscr{L}}{\partial y_i}, \tag{3.2}$$

where $\wedge$ is the AND logical operator and there are only non-zero gradients for those weights which join in a feature which is masked for task t.

This setup allows the associated weights to a neuron to be either used-and-learnable, or neither. If used, they will contribute forward to the next layers (which is good, as it promotes forward transfer, i.e. sharing of knowledge from previous tasks). Yet at the same time this also implies that it will be possible to modify them (which is bad, as it introduces catastrophic forgetting on the previous tasks). With only binary masks, you cannot have one without the other. Alternatively, one could also define two separate binary states: “used" and ‘learnable". This has been used for a long time in deep learning by freezing weights [121]. Freezing weights is a mask-based way of switching on and off the learning of a layer. In this case, in both states the layer would contribute to the outcome of the network, but the update of the weights would only be done on those layers that are not masked. Here, we further explore this idea. We advocate that, in a sequential setup where the capacity of the network might increase when learning new tasks, the best way to mask the neurons is by having three states: “used", “learnable" and “unused". This can be achieved by using ternary masks on the activations.

3.3.2 Ternary Feature Masks (TFM)

Being able to use the connections between the neurons of the previous tasks and the neurons of the newly added task is important to reuse learned information and reduce the capacity that must be added. By using a ternary mask we can define three states:

- **forward only**: the features are used during the forward pass so that the learned information from previous tasks is used; but the backward pass step is removed in order to keep the weights and prevent forgetting. This state is used on the features from previous tasks.
- **normal**: forward and backward passes are applied as usual in order to learn the task at hand. This state is used on the new features created by the expansion of the network.
- **masked**: neither forward nor backward passes are allowed, the features do not contribute to the network inference and the weights associated to it are frozen. This state is used at test time only when evaluating an old task after a new task is added. When extending the capacity of the network, the new features will not be used when doing inference on the previous tasks since those did not exist at the moment of their training.

Similar as in the case of the binary mask we assign features to tasks with a mask $m^{t,l}$ (with l the corresponding layer). Again overlap in the selected features is not allowed. However, different than before, we now define a second mask $n^{t,l}$ per task t which is defined as:

$$n_i^{t,l} = \begin{cases} 1, & \text{if } \exists\, s \leq t : m_i^{s,l} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

The forward and backward pass are now given by:

$$y = (Wx) \odot n^{t,l} \tag{3.4}$$

and

$$\frac{\partial \mathscr{L}}{\partial W_{ij}} = (n_j^{t,l-1} n_i^{t,l} - n_j^{t-1,l-1} n_i^{t-1,l}) x_j \frac{\partial \mathscr{L}}{\partial y_i}, \tag{3.5}$$

respectively. During the forward pass, features which were selected by previously learned tasks can be used in the current task. During the backward pass, we make sure that all new weights can be updated while forcing the existing ones from previous tasks to remain the same. In Fig. 3.3 we show an example with two tasks, where

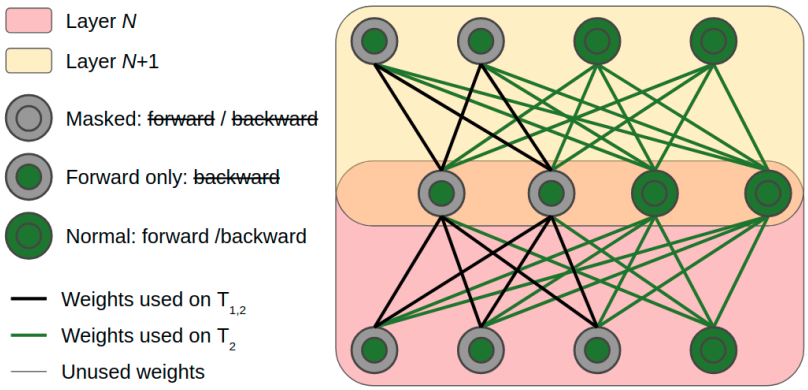

Figure 3.3 – Ternary masks encode three states: masked (frozen), forward only or normal (forward and backward). Compared to Fig. 3.2, all unused connections can now be learned without forgetting previous knowledge.

adding features to the layer allows for more connections to be used than in the binary case. The part of the mask corresponding to $n_j^{t,l-1} n_i^{t,l}$ corresponds to all available connections at task t. In a similar way, the part of the mask corresponding to $n_j^{t-1,l-1} n_i^{t-1,l}$ corresponds to all available connections at task $t-1$. Subtracting both terms allows us to mask the connections that contain the already learned content and apply backpropagation only on the new connections.

Note that this definition also allows to use the same forward and backward pass in case we would want to re-train one of the previous tasks. However, since we do not contemplate this option for our proposed setup, we can simplify equation 3.5 to non-revisiting task incremental learning. In this case, the forward pass remains the same as in equation 3.4, and the backward pass can be rewritten as:

$$\frac{\partial \mathscr{L}}{\partial W_{ij}} = (m_i^{t,l} \vee m_j^{t,l-1}) x_j \frac{\partial \mathscr{L}}{\partial y_i}, \tag{3.6}$$

where $\vee$ is the OR logical operator which makes the mask active when either operands are active.

Since $n_i^{t,l}$ can never be 0 if one of the current or previous $m_i^{1..t,l}$ is 1, both masks $m^{t,l}$ and $n^{t,l}$ can be combined in a single ternary mask. This is because weights associated to a feature that is not used in the forward pass are never updated. With this ternary mask, the states are associated as follows: when $m_i^{t,l} = 1$ and $n_i^{t,l} = 1$ the neuron is used and learnable (normal state), when $m_i^{t,l} = 0$ and $n_i^{t,l} = 1$ the neuron is used and

contributes to the forward pass but the associated weights are not updated (forward only state), and finally when $m_i^{t,l}=0$ and $n_i^{t,l}=0$ the neuron is unused, not taking part in the inference or the update of the network (masked state). In our implementation we assign these three states to a ternary mask as 2, 1 and 0 respectively[‡].

Allowing previously learned parameters to be used in the forward pass, but only updating network parameters assigned to the current task in the backward pass, is also applied in Packnet [102] and HAT [153]. However, in contrast to us, Packnet has the masks on the weights and not on the features. HAT applies a soft activation mask, which permits forgetting of previous tasks. We further differ from these methods by the task-specific feature normalization (discussed in the next section) which is a crucial ingredient of our method, and which not only allows exploitation of previously learned features, but also their adaptation to the current task. This is not possible in Packnet or HAT.

3.3.3 Task-specific Feature Normalization (FN)

Since binary or ternary masks freeze related weights learned on previous tasks, those weights have no room for flexibility to small changes in features. This means that even when being very similar to the ones needed for a new task, they will tend to learn a similar version of those filters with shifted or scaled operators. This phenomenon is similar to the one observed when learning several styles for style transfer networks. A way of reusing learned filters in a more efficient way but still keeping the non-forgetting property would be to use a similar approach to *conditional instance normalization* [39], which consists in transforming a set of features x into a normalized version $\hat{x}$ depending on the task.

Let $x_{l,1..I}$ be the features of layer l, and γ_t, β_t the learnable parameters for each feature of each layer given a fixed task t. We define the task-specific feature normalization of $x_{l,i}$ as:

$$\hat{x}_{l,i} = \gamma_{t,l,i}\, x_{l,i} + \beta_{t,l,i}\,, \tag{3.7}$$

where we apply a conditional normalization on the task without applying an instance normalization on the mean and standard deviation across the spatial dimensions. These parameters allow slight adjustments to the new tasks in learned activations without modifying existing parameters (thus no forgetting happens) and with little overhead to the network capacity since the γ and β parameters are for each feature and not for all weights.

[‡]Code available at: https://github.com/mmasana/TernaryFeatureMasks

3.3.4 Growing ternary feature masks

One of the core characteristics of our proposed method is that it can easily grow and expand the capacity of the network as is required. Methods that allow the weights to be modified, with the addition of more tasks will eventually modify them, regardless of which approach is chosen to define the importance weights. Capacity is limited, and when the network is close to running out of it for learning the new task, the trade-off will allow for forgetting to take place. It is at this point that most methods are not easily expanded to accommodate more weights and grow the network before too much forgetting happens. This is because most approaches cannot predict the performance drop on previous tasks before learning the new one. With our approach, we enforce this to be a core part of the system. This provides the capacity to be able to not forget at a very small cost, while allowing the network to grow in parameters only when necessary.

Given a network with L layers, any layer with the corresponding $y_{l,1..I}$ learned features can be expanded if those learned features are not enough to represent the new task. When expanding a layer by N new features, the output of the layer grows to $y_{l,1..I+N}$. That affects only the newly added forward mask values:

$$n_j^{t,l} = \begin{cases} 1, & \text{for current task, if } 1 \leq j \leq I+N \\ 0, & \text{for previous tasks, if } I < j \leq I+N, \end{cases} \tag{3.8}$$

so that all features can be seen while learning the new task but ignored by previous tasks. And then the backward mask:

$$m_j^{t,l} = \begin{cases} 0, & \text{for current task, if } 1 \leq j \leq I \\ 1, & \text{for current task, if } I < j \leq I+N \\ 0, & \text{for previous tasks, if } I < j \leq I+N, \end{cases} \tag{3.9}$$

so that it only affects the new connections without modifying previous knowledge.

Training small tasks on large networks at the beginning of a continual learning setup usually leads to overfitting or too much repetition of filters. Feature usage on the new tasks look very unbalanced in comparison to learning larger tasks [110] (see Chapter 2). We believe that learning tasks in an efficient capacity and growing as more is needed is a much better approach to avoid overfitting. This observation is backed by the better results some other approaches have when pruning and retraining on smaller sub-networks than when directly pruning or learning in larger sub-networks [102, 144].

Algorithm 1 : Growing Ternary Feature Masks

Input: ternary mask m for each task and layer
Input: s number of features to add per layer
Require: Network with L layers
Require: Tasks $1,\ldots,T$, current task $t > 1$
Loop for each layer
for $l = 1,\ldots,L$
  old_size ← current_output_size(l)
  new_size ← old_size + s^l
  # For each previous task
  for $k = 1,\ldots,t-1$
    $m^{k,l}$[old_size:new_size] ← 0
  end for
  # For the current task
  $m^{t,l}$[0:old_size] ← 1
  $m^{t,l}$[old_size:new_size] ← 2
end for

3.3.5 Ternary mask implementation

The formulation of our proposed method in Sec. 3.3.2 states that we can combine both masks m^t and n^t into a ternary mask. In order to make the implementation easier and more efficient, this is done by creating a ternary mask that is set to be state 2 for all features at task 1. This means that the first task will work as a normal network that allows for learning the task at hand as if we were using fine-tuning. Then, when moving to task 2, we will grow the network and add new features. The masks for task 1 associated with the new features will be set to state 0, therefore these are not used when evaluating nor are they learnable for task 1. The masks for task 2 will then be created by setting the previous existing features to state 1 and the new added features to state 2. This would allow the features with state 1 to be used during the forward pass for mask $n^{t=2}$, while the features with state 2 will contribute to both forward pass for mask $n^{t=2}$ and backward pass for mask $m^{t=2}$ (using Eq. 3.6). This process is explained in Algorithm 1.

3.3.6 A note on choosing expansion rates

The continual learning philosophy states that data from previous tasks should not be used when learning new ones; only data of the current task can be used in each training session. It is also common in machine learning to use a part of the training set as

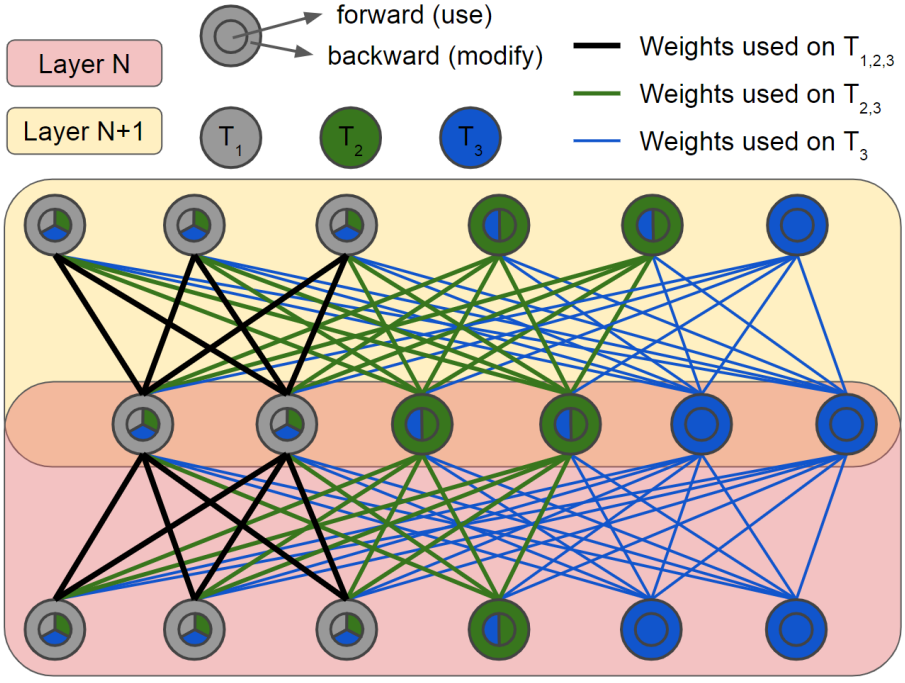

Figure 3.4 – Network growth with ternary feature masks over three tasks.

validation in order to choose the best hyper-parameters. Therefore, when learning each task, a validation set of that specific task at hand can be used to train the network avoiding over-fitting, but no other data can be used (neither from test nor from other previous or future tasks). It is important to state that we strictly comply to these rules in the experiments we propose.

As explained in Section 3.3.4, our proposed approach can be expanded as needed in order to learn the new tasks without having to change the connections from previous tasks. However, this flexibility of choosing how many features will be added to each layer can easily become a rabbit-hole of architecture optimization. Because of that, we decide to propose a simple setup for how we apply our proposed approach to be comparable to the other state-of-the-art. We take the maximum layer size for all approaches to be the same as the VGGnet or AlexNet architectures for the experiments in the following Section 3.4. This way, all approaches will have a similar number of parameters.

The differences between Figs. 3.2 and 3.3 are further explained in Fig. 3.4 During

the first task, the only used features are those that have a grey-border in them. This means that for the two shown layers, task 1 is learned by using 8 features (with 12 connections). Once task 2 arrives, we fix the grey-border features and expand the network with the green-border ones. The new task then uses the existing network and expands it with 5 features (with 24 new connections). The masks for the green-border features are set to unused for masks of task 1, while they are learnable for task 2. Finally, as task 3 comes, 5 features are also added (with 36 new connections). Masks corresponding to the new features are set to unused for tasks 1 and 2, while set to learnable for task 3. This way of expanding the network also shows that as we learn more tasks, more knowledge is available from previous ones. This opens the possibility of having to add less and less features over time since the addition of each feature creates more connections to learn. In practice, one can imagine that when learning new tasks that are very similar to previous ones, no new features will have to be added and the current network knowledge and a specific head for the task will be enough. Further research and analysis on the specific details for each layer expansion and architecture is left for future work.

3.4 Experimental results

In this section we report on a range of experiments to quantify the effectiveness of our proposed approach and compare with other state-of-the-art methods and baselines.

3.4.1 Experimental setup

Datasets. We evaluate our approaches on a larger, lower resolution dataset (Tiny ImageNet ILSVRC2012 [36]), on a large-scale dataset (ImageNet [143]) and some fine-grained classification datasets: Oxford 102 Flowers [120], CUB-200-2011 Birds [170] and Stanford Actions [180]. Statistics over those datasets are summarized in Table 3.3. For all experiments we take a fixed random set of 10% of images for validation. The validation set is equally distributed among the number of classes and fixed for each experiment to ensure the fairness of the comparison. Since the test set is not labelled for ImageNet ILSVRC2012, we use the validation set for test instead.

Tiny ImageNet is a 64×64×3 resized version of ImageNet with 200 of the classes. ImageNet uses 256×256×3 inputs that use random cropping to 224×224×3 during training for data augmentation. In the case of Birds we resize the bounding box annotations of the objects to 224×224×3 for all splits. We do the same for Actions but without using the bounding box annotations. For Flowers we resize to 256×256×3 and also do data augmentation by random cropping 224×224×3 patches during training and using the central crop for evaluation. In all experiments we perform random

horizontal flips during training for data augmentation.

We decide to not do experiments on permuted MNIST since they have been shown to not allow a fair comparison between different approaches [83]. The MNIST data contains a too large amount of zeros on each input, which leads to an easy identification of the important weights that can be frozen so that they do not overlap with the other tasks. Furthermore, the MNIST data might be too simple to represent more realistic scenarios. We do not consider other low resolution datasets than Tiny ImageNet for the same reason.

The semantically similar splits used for the experiments with Tiny ImageNet are grouped as described in Table 3.4.

Network architectures. For Tiny ImageNet we use VGG-16, which has been shown to provide high performance [157]. Since Tiny ImageNet is low resolution, the last max-pool layer and the last three convolutional layers from the feature extractor are removed. For ImageNet and the fine-grained datasets we use AlexNet [78]. No pretrained weights are used and the models are trained from scratch using only samples from the training set. All approaches except ours have access to the full network from the beginning. However, our proposed TFM starts with a network that is smaller than the proposed ones at each layer (reduced number of output filters). Then, it grows as explained in Sec. 3.3.4 as more features are added every time a new task is learned. We limit the growth of the network to the total size of the one used by all other approaches. Therefore, at the end of learning all tasks, all approaches will have had access to the same amount of network capacity besides the extra parameters or regularizers that each method requires (see Caption in Table 3.1).

Training details. All experiments are trained using backpropagation with plain Stochastic Gradient Descent following a similar setup to HAT [153]. With a batch size of 64, learning rate starts at 0.05, decaying by a factor of 3 when 5 consecutive epochs have no improvement on the validation loss, until either the learning rate is reduced below 10^{-4} or 200 epochs have passed. Data splits, task sequence, data loader

Dataset	#Train	#Eval	#Classes
Tiny ImageNet [36]	100,000	10,000	200
ImageNet ILSVRC2012 [143]	1,280,861	50,000	1000
Oxford Flowers [120]	2,040	6,149	102
CUB-200-2011 Birds [170]	5,994	5,794	200
Stanford Actions [180]	4,000	5,532	40

Table 3.3 – Summary of datasets used.

Task	Semantic group	Classes
1	Animals (flying & insects)	scorpion, black widow, tarantula, spider web, centipede, trilobite, grasshopper, stick insect, cockroach, mantis, ladybug, dragonfly, monarch butterfly, sulphur butterfly, fly, bee, goose, black stork, king penguin, albatross.
2	Artifacts (smaller)	abacus, binoculars, candle, chain, chest, compass, dumbbell, nail, hourglass, lampshade, pill bottle, computer keyboard, acorn, plunger, syringe, teddy bear, torch, comic book, remote control, umbrella.
3	Music, Sport and Kitchen	basketball, punching bag, rugby ball, scoreboard, stopwatch, volleyball, CD player, drumstick, iPod, oboe, organ, refrigerator, cask, plate, wooden spoon, teapot, frying pan, beaker, bucket, dining table.
4	Animals (land)	brown bear, red panda, koala, pig, ox, bison, bighorn sheep, gazelle, dromedary, elephant, orangutan, chimpanzee, baboon, cougar, lion, fire salamander, bullfrog, tailed frog, alligator, boa constrictor.
5	Artifacts (bigger)	altar, maypole, bannister, flagpole, fountain, parking meter, pay-phone, pole, cash machine, birdhouse, bathtub, rocking chair, potter's wheel, sewing machine, space heater, turnstile, memorial tablet, desk, vestment, reel.
6	Food	water jug, wok, guacamole, ice cream, lollipop, pretzel, cauliflower, bell pepper, mashed potato, mushroom, orange, lemon, pomegranate, banana, meat loaf, pizza, potpie, espresso, soda bottle, beer bottle.
7	Animals (water & pets)	dugong, goldfish, jellyfish, brain coral, American lobster, spiny lobster, sea slug, sea cucumber, guinea pig, snail, slug, poodle, Chihuahua, Yorkshire terrier, golden retriever, Labrador retriever, German shepherd, tabby cat, Persian cat, Egyptian cat.
8	Clothes and wearables	academic gown, poncho, apron, backpack, bikini, bow tie, cardigan, fur coat, gasmask, kimono, military uniform, miniskirt, neck brace, Christmas stocking, sandal, snorkel, sock, sombrero, sunglasses, swimming trunks.
9	Transport	bullet train, station wagon, freight car, go-kart, rickshaw, limousine, lifeboat, moving van, police van, school bus, convertible, trolleybus, missile, crane, sports car, tractor, gondola, broom, cannon, mower.
10	Buildings and scenes	barbershop, barn, lighthouse, butcher shop, candy store, water tower, triumphal arch, suspension bridge, steel arch bridge, viaduct, thatched roof, cliff dwelling, dam, obelisk, picket fence, cliff, coral reef, lakeside, seacoast, alp.

Table 3.4 – Semantically similar splits for tiny ImageNet.

shuffle and network initialization are fixed for all approaches given a seed. Following the results in [34], we use dropout with $p = 0.5$.

Baselines. Fine-tuning has no extra hyperparameters and simply uses the cross-entropy loss to learn each task as it comes, without using data from previous tasks nor avoiding catastrophic forgetting. Joint training breaks the no-revisiting data rule and learns with data from the current task as well as all the previous tasks, serving as an upper-bound to compare all approaches. Finally, we propose to use Freezing as a baseline where we learn the first task and then freeze all layers except the head for the remaining tasks.

Hyperparameters. Distillation and model-based approaches use hyperparameters to control the trade-off between forgetting and intransigence on the knowledge of previous tasks. On top of that, LwF has a temperature scaling hyperparameter for the cross-entropy loss. From the mask-based models, HAT has a trade-off hyperparameter too and a maximum for the sigmoid gate steepness. PackNet has a prune percentage of the layers.

TFM has a growth percentage which is equal for all layers. This percentage is set on the validation set according to the following protocol. For each new task, several growth percentages are evaluated without the knowledge of previous or future tasks. Rather than choosing the growth rate with best performance, we pick the lowest growth rate which obtains a performance within a margin of the best performance (we set the margin to be 1.5% for Tiny ImageNet and 0.1% for fine-grained). For ImageNet this scheme would be computationally demanding and we use a fixed growth schedule, starting from 55% of the weights for the first task and add 5% for all remaining tasks. Then, we fix that hyperparameter and we run the final experiment on train and evaluate on the test set.

3.4.2 Fine-grained datasets

A common setup to evaluate continual learning over a number of learning sessions are disjoint splits (tasks) inside the same classification dataset. However, some approaches report results on only two tasks [43, 72, 83], which becomes too similar to transfer learning setups and does not allow to evaluate the true potential of continual learning. Because of that, we choose to evaluate our proposed approach and its variations on more than two tasks. It should be noted that we start training from scratch resulting in lower scores than reported by papers which train from a pretrained network. However, because of the large number of classes in ImageNet (including a subset of Birds) we consider training from scratch provides a more natural setting for continual learning.

We compare our proposed method (TFM) and an ablation version of it without the task-specific feature normalization (TFM w/o FN) with the mask-based approaches

(HAT, PackNet), a well-known model-based approach (EWC) and the baselines (Fine-tuning, Freezing, Joint) on three fine-grained datasets (Flowers, Birds, Actions). As explained earlier, to make the comparison fair our approach will learn the first task on a smaller network and then grow and learn the next tasks with a maximum growth the size of the network other approaches use.

As shown in Table 3.5, our proposed approach outperforms the other approaches for the three datasets. For these datasets only on Flowers a considerable performance gain is observed when adding task-specific feature normalization. Only PackNet manages to obtain competitive results. However, on both Birds and Flowers TFM does significantly better, while having a much lower memory overhead than PackNet (0.2Mb versus 27.3Mb respectively). It is also interesting how well Freezing works as a non-forgetting baseline.

3.4.3 Task-similarity effects on Tiny ImageNet

Next we experiment on several ten-task splits of Tiny ImageNet. We compare our proposed approach (TFM) with two distillation methods with low overhead (LFL [72], LwF [87]), two of the best known model-based methods (EWC [74], IMM [83]) and two of the most recent mask-based methods (HAT [153], PackNet [102]). We also compare the approaches to three baselines (Fine-tuning, Freezing, Joint). The setup uses the VGGnet introduced in Sec. 3.4.1. The model is trained from scratch on 10 tasks with the same number of classes. We propose to evaluate those approaches under the same conditions on a random Tiny ImageNet partition (see Table 3.6) and on a semantically similar partition (see Table 3.7).

In the case of the random splits (see Table 3.6), most methods have quite good results on the last tasks with minor to no forgetting. LFL, IMM and EWC provide some improvement over Fine-tuning. LwF has a very good performance due to tasks being quite similar. All mask-based models have a very similar performance, with PackNet having the better performance.

In the semantically similar splits (see Table 3.7), which has a more different distribution for each task than the random case, some approaches have difficulties avoiding catastrophic forgetting as the sequence gets longer. It is interesting to see, that the good results of LwF on the random split are not repeated when we have semantic splits. As observed before, LwF fails when there exists large changes in the distributions of the features between the tasks [6]. Mask-based models outperform all other approaches again, with TFM having the better performance.

Both Tables 3.6 and 3.7 show accuracy and forgetting of each task after training the 10 tasks, and thus having learned the 200 Tiny ImageNet classes. Results show that mask-based approaches achieve better overall performance than other approaches on both splits, getting close to the joint training baseline. Freezing the network after

Oxford 102 Flowers					
Method	Task 1	Task 2	Task 3	Task 4	Avg.
Fine-tuning	10.0 (-20.3)	5.1 (-17.1)	6.7 (-13.6)	17.3 (0.0)	9.8
Freezing	30.3 (0.0)	39.8 (0.0)	32.0 (0.0)	33.1 (0.0)	33.8
Joint	54.6 (+24.3)	58.9 (+11.5)	57.7 (+4.5)	47.0 (0.0)	54.6
EWC [74]	12.1 (-18.2)	11.6 (-38.1)	9.3 (-24.4)	25.8 (0.0)	14.7
HAT [153]	17.2 (-12.7)	19.3 (-28.5)	28.6 (+1.4)	31.6 (0.0)	24.2
PackNet [102]	32.0 (0.0)	53.7 (0.0)	43.6 (0.0)	37.9 (0.0)	41.8
TFM w/o FN	36.4 (0.0)	54.1 (0.0)	38.6 (0.0)	39.0 (0.0)	42.0
TFM	36.4 (0.0)	53.8 (0.0)	45.5 (0.0)	37.6 (0.0)	**43.3**

CUBS 200 Birds					
Method	Task 1	Task 2	Task 3	Task 4	Avg.
Fine-tuning	7.4 (-30.2)	2.6 (-30.0)	29.7 (-3.4)	43.1 (0.0)	20.7
Freezing	37.6 (0.0)	35.1 (0.0)	35.4 (0.0)	38.4 (0.0)	36.6
Joint	48.7 (+11.1)	52.1 (+6.0)	50.7 (+1.5)	51.9 (0.0)	50.8
EWC [74]	16.2 (-21.4)	19.0 (-21.2)	24.2 (-14.0)	41.7 (0.0)	25.3
HAT [153]	18.7 (-1.8)	19.4 (-0.4)	28.5 (-0.6)	31.2 (0.0)	24.4
PackNet [102]	35.3 (0.0)	42.8 (0.0)	44.4 (0.0)	45.9 (0.0)	42.1
TFM w/o FN	42.9 (0.0)	44.1 (0.0)	48.3 (0.0)	49.1 (0.0)	46.1
TFM	42.9 (0.0)	43.1 (0.0)	49.9 (0.0)	48.8 (0.0)	**46.2**

Stanford 40 Actions					
Method	Task 1	Task 2	Task 3	Task 4	Avg.
Fine-tuning	24.4 (-10.5)	26.5 (-7.7)	17.6 (-16.8)	28.9 (0.0)	24.4
Freezing	34.9 (0.0)	29.4 (0.0)	30.1 (0.0)	30.5 (0.0)	31.2
Joint	45.7 (+10.8)	40.3 (+4.8)	43.2 (-1.1)	40.2 (0.0)	42.4
EWC [74]	24.2 (-10.7)	28.2 (-2.0)	25.2 (-5.6)	34.3 (0.0)	28.0
HAT [153]	25.7 (-1.0)	25.5 (-2.7)	30.1 (-2.1)	34.4 (0.0)	28.9
PackNet [102]	32.5 (0.0)	32.9 (0.0)	36.7 (0.0)	34.3 (0.0)	34.1
TFM w/o FN	35.3 (0.0)	38.3 (0.0)	39.2 (0.0)	38.0 (0.0)	37.7
TFM	35.3 (0.0)	37.2 (0.0)	42.0 (0.0)	37.2 (0.0)	**38.0**

Table 3.5 – Comparison with the state-of-the-art. Accuracy after learning 4 tasks on AlexNet from scratch. Number between brackets indicates forgetting.

Tiny ImageNet - classes randomly split											
Approach	Task 1 (1-20)	Task 2 (21-40)	Task 3 (41-60)	Task 4 (61-80)	Task 5 81-100)	Task 6 (101-120)	Task 7 (121-140)	Task 8 (141-160)	Task 9 (161-180)	Task 10 (181-200)	Avg. all
Fine-tuning	38.1 (-13.6)	36.0 (-13.7)	43.2 (-16.0)	44.1 (-18.6)	45.5 (-12.6)	54.5 (-13.6)	50.3 (-15.7)	50.5 (-13.4)	51.0 (-13.1)	61.2 (0.0)	47.4
Freezing	51.7 (0.0)	36.4 (0.0)	39.5 (0.0)	41.7 (0.0)	42.9 (0.0)	46.2 (0.0)	45.7 (0.0)	41.1 (0.0)	41.2 (0.0)	40.9 (0.0)	42.7
Joint	58.6 (+6.9)	53.9 (+8.3)	59.1 (+3.7)	61.8 (+7.9)	57.7 (+2.9)	66.0 (+2.6)	64.0 (+3.1)	60.2 (+5.9)	57.9 (+1.0)	53.8 (0.0)	59.3
LfL [72]	32.4 (-18.9)	35.4 (-17.0)	43.4 (-15.7)	44.1 (-20.2)	45.0 (-15.0)	55.9 (-14.5)	49.4 (-16.1)	51.1 (-12.4)	58.6 (-8.0)	61.4 (0.0)	47.7
LwF [87]	45.1 (-6.6)	45.5 (-2.2)	53.5 (-4.6)	57.6 (-2.6)	56.2 (0.0)	65.7 (+0.4)	63.5 (-0.3)	58.4 (-1.9)	59.6 (-0.3)	58.5 (0.0)	56.4
IMM [83]	50.6 (-1.1)	38.5 (+0.3)	44.7 (-0.1)	49.2 (+0.3)	47.5 (+1.1)	51.9 (-1.4)	53.7 (-0.6)	47.7 (-0.4)	50.0 (-2.2)	48.7 (0.0)	48.3
EWC [74]	33.9 (-17.4)	35.4 (-14.4)	43.6 (-15.4)	46.7 (-15.9)	49.5 (-9.1)	52.5 (-15.8)	47.8 (-20.0)	50.2 (-13.8)	56.6 (-9.9)	61.4 (0.0)	47.8
HAT [153]	46.8 (-0.2)	49.1 (+0.8)	55.8 (+0.2)	58.0 (-0.2)	53.7 (+0.3)	61.0 (+0.1)	58.7 (0.0)	54.0 (-0.1)	54.6 (-0.1)	50.3 (0.0)	54.2
PackNet [102]	52.5 (0.0)	49.7 (0.0)	56.5 (0.0)	59.8 (0.0)	55.0 (0.0)	64.7 (0.0)	61.7 (0.0)	55.9 (0.0)	55.2 (0.0)	52.5 (0.0)	**56.4**
TFM w/o FN	49.6 (0.0)	47.2 (0.0)	54.8 (0.0)	58.2 (0.0)	55.0 (0.0)	64.0 (0.0)	59.3 (0.0)	53.6 (0.0)	55.5 (0.0)	51.9 (0.0)	54.9
TFM	48.2 (0.0)	47.7 (0.0)	56.7 (0.0)	58.2 (0.0)	54.8 (0.0)	62.2 (0.0)	61.5 (0.0)	57.3 (0.0)	58.5 (0.0)	54.8 (0.0)	56.0

Table 3.6 – Comparison with the state-of-the-art. Tiny ImageNet on VGGnet from scratch. Accuracy of each task after learning all tasks. Number between brackets indicates forgetting. Classes are randomly split and fixed for all approaches.

Tiny ImageNet - classes semantically split											
Approach	Task 1 fly anim.	Task 2 small artif.	Task 3 hobbies	Task 4 land anim.	Task 5 big artif.	Task 6 food	Task 7 pets/aquatic	Task 8 wearables	Task 9 transport	Task 10 scenes	Avg. all
Fine-tuning	17.1 (-34.2)	19.7 (-17.7)	20.9 (-24.5)	16.7 (-30.5)	20.8 (-28.7)	29.2 (-22.0)	30.7 (-21.0)	25.2 (-17.8)	40.2 (-18.9)	59.9 (0.0)	28.0
Freezing	51.3 (0.0)	28.5 (0.0)	27.2 (0.0)	29.6 (0.0)	29.0 (0.0)	35.0 (0.0)	31.7 (0.0)	23.9 (0.0)	37.0 (0.0)	34.7 (0.0)	32.8
Joint	55.0 (+3.7)	41.9 (+7.9)	46.2 (+6.3)	44.9 (+4.9)	44.7 (+7.1)	49.0 (+4.8)	46.6 (+4.9)	36.4 (+4.7)	51.2 (+5.7)	51.1 (0.0)	46.7
LfL [72]	17.2 (-34.1)	18.4 (-21.0)	21.5 (-24.0)	18.7 (-30.5)	20.2 (-28.6)	27.4 (-23.9)	28.4 (-22.3)	26.0 (-18.4)	41.2 (-17.6)	59.1 (0.0)	27.8
LwF [87]	34.0 (-13.9)	18.4 (-14.5)	32.6 (-0.8)	36.5 (-5.6)	40.1 (-0.5)	43.1 (-2.5)	41.8 (-1.3)	32.7 (-1.1)	50.3 (-0.5)	48.1 (0.0)	37.8
IMM [83]	42.3 (-9.0)	28.8 (+0.1)	26.5 (-3.1)	30.7 (-3.6)	32.5 (-3.1)	28.8 (-13.0)	35.4 (-6.2)	27.3 (-3.7)	43.6 (-4.9)	42.7 (0.0)	33.9
EWC [74]	20.2 (-31.1)	18.5 (-19.3)	20.2 (-26.3)	20.9 (-28.9)	24.7 (-22.8)	25.5 (-27.5)	28.7 (-23.4)	23.0 (-19.6)	39.8 (-20.2)	56.8 (0.0)	27.8
HAT [153]	44.6 (+0.4)	34.8 (+0.2)	40.8 (-0.1)	45.4 (+0.4)	40.8 (-2.5)	49.8 (0.0)	44.9 (-0.2)	33.1 (-1.7)	51.9 (0.0)	53.8 (0.0)	44.0
PackNet [102]	47.0 (0.0)	35.7 (0.0)	42.7 (0.0)	48.6 (0.0)	45.8 (0.0)	48.1 (0.0)	45.9 (0.0)	38.3 (0.0)	51.2 (0.0)	49.1 (0.0)	45.2
TFM w/o FN	46.4 (0.0)	34.7 (0.0)	38.8 (0.0)	44.1 (0.0)	42.0 (0.0)	48.3 (0.0)	46.5 (0.0)	35.7 (0.0)	52.0 (0.0)	54.8 (0.0)	44.3
TFM	46.4 (0.0)	37.2 (0.0)	40.4 (0.0)	44.1 (0.0)	44.2 (0.0)	48.2 (0.0)	46.4 (0.0)	37.5 (0.0)	53.5 (0.0)	54.7 (0.0)	**45.3**

Table 3.7 – Comparison with the state-of-the-art. Tiny ImageNet on VGGnet from scratch. Accuracy of each task after learning all tasks. Number between brackets indicates forgetting. Classes are split by semantic closeness and fixed for all approaches.

Tiny ImageNet - larger first task											
Approach	Task 1 (1-110)	Task 2 (111-120)	Task 3 (121-130)	Task 4 (131-140)	Task 5 (141-150)	Task 6 (151-160)	Task 7 (161-170)	Task 8 (171-180)	Task 9 (181-190)	Task 10 (191-200)	Avg. (111+)
Finetuning	18.4 (-33.2)	39.6 (-34.4)	56.6 (-21.0)	58.0 (-23.8)	44.6 (-32.8)	63.0 (-21.8)	51.0 (-27.2)	51.4 (-19.0)	70.4 (-14.0)	80.0 (0.0)	57.2
Freezing	51.6 (0.0)	68.6 (0.0)	70.2 (0.0)	77.9 (0.0)	68.1 (0.0)	78.8 (0.0)	72.9 (0.0)	64.2 (0.0)	76.7 (0.0)	70.2 (0.0)	69.9
LfL [72]	16.7 (-33.9)	58.3 (-6.6)	59.5 (-2.9)	64.6 (-2.2)	58.2 (-1.4)	64.7 (-0.6)	63.4 (+0.5)	54.4 (+0.1)	61.0 (0.0)	56.5 (0.0)	60.0
LwF [87]	10.8 (-40.0)	29.5 (-43.5)	44.6 (-30.4)	61.2 (-21.2)	55.5 (-15.1)	73.3 (-8.8)	71.7 (-3.5)	62.3 (-1.8)	77.8 (-2.2)	74.3 (0.0)	61.1
IMM [83]	26.1 (-26.3)	50.4 (-13.8)	59.5 (-19.3)	61.6 (22.8)	54.0 (-21.8)	63.2 (-22.5)	58.2 (-19.9)	56.0 (-13.9)	75.0 (-6.7)	79.9 (0.0)	62.0
EWC [74]	51.8 (-0.7)	24.8 (-0.1)	43.3 (-1.2)	61.8 (-0.5)	55.5 (-0.3)	70.8 (-0.7)	67.6 (-0.1)	53.8 (-0.6)	70.1 (-1.2)	61.7 (0.0)	56.6
HAT [153]	46.1 (0.0)	60.1 (-0.1)	68.2 (+0.2)	73.2 (+0.1)	63.2 (+0.1)	76.2 (+0.1)	67.4 (0.0)	58.1 (-0.1)	73.2 (0.0)	59.1 (0.0)	66.5
PackNet [102]	47.6 (0.0)	74.0 (0.0)	74.2 (0.0)	79.0 (0.0)	65.2 (0.0)	76.2 (0.0)	69.4 (0.0)	61.4 (0.0)	73.4 (0.0)	64.8 (0.0)	70.8
TFM w/o FN	49.6 (0.0)	69.8 (0.0)	71.1 (0.0)	79.8 (0.0)	68.4 (0.0)	78.4 (0.0)	72.9 (0.0)	64.8 (0.0)	75.9 (0.0)	70.2 (0.0)	72.4
TFM	49.9 (0.0)	70.4 (0.0)	71.4 (0.0)	80.8 (0.0)	70.5 (0.0)	79.4 (0.0)	73.9 (0.0)	64.4 (0.0)	76.5 (0.0)	72.1 (0.0)	**73.3**

Table 3.8 – Comparison with the state-of-the-art. Tiny ImageNet on VGGnet from scratch. Accuracy of each task after learning all tasks. Numbers between brackets indicates forgetting. Average on the smaller tasks 2 to 10.

ImageNet - classes randomly split											
Approach	Task 1 (1-100)	Task 2 (101-200)	Task 3 (201-300)	Task 4 (301-400)	Task 5 (401-500)	Task 6 (501-600)	Task 7 (601-700)	Task 8 (701-800)	Task 9 (801-900)	Task 10 (901-1000)	Avg. all
Finetuning	25.8 (-43.0)	32.2 (-36.2)	31.4 (-35.3)	37.8 (-27.7)	39.1 (-27.7)	43.7 (-25.7)	46.0 (-22.8)	50.0 (-16.5)	53.4 (-12.1)	63.7 (0.0)	42.3
Freezing	68.8 (0.0)	53.5 (0.0)	52.0 (0.0)	51.2 (0.0)	51.3 (0.0)	53.9 (0.0)	52.2 (0.0)	53.9 (0.0)	51.7 (0.0)	51.2 (0.0)	54.0
LwF [87]	27.6 (-41.2)	37.2 (-19.9)	42.0 (-22.6)	44.4 (-20.9)	50.5 (-14.1)	56.6 (-11.3)	57.9 (-9.1)	61.2 (-5.0)	62.0 (-1.3)	62.7 (0.0)	50.2
IMM [83]	68.5 (-0.3)	53.6 (0.0)	52.1 (0.0)	51.7 (-0.1)	52.5 (+0.3)	55.5 (+0.2)	54.7 (+0.1)	53.5 (0.0)	54.2 (+0.1)	51.8 (0.0)	54.8
EWC [74]	21.8 (-47.0)	26.5 (-41.7)	29.5 (-36.5)	32.9 (-32.6)	35.6 (-30.9)	40.4 (-28.1)	40.0 (-26.2)	44.7 (-20.7)	47.8 (-16.2)	61.1 (0.0)	38.0
PackNet [102]	67.5 (0.0)	65.8 (0.0)	62.2 (0.0)	58.4 (0.0)	58.6 (0.0)	58.7 (0.0)	56.0 (0.0)	56.5 (0.0)	54.1 (0.0)	53.6 (0.0)	59.1
TFM (Ours)	63.6 (0.0)	62.2 (0.0)	60.1 (0.0)	61.6 (0.0)	62.6 (0.0)	64.5 (0.0)	64.0 (0.0)	63.7 (0.0)	63.0 (0.0)	59.9 (0.0)	**62.5**

Table 3.9 – Comparison with the state-of-the-art. ImageNet on AlexNet from scratch. Accuracy of each task after learning all tasks. Number between brackets indicates forgetting.

the first task and learning only the head for the remaining tasks works better in the semantically similar splits than in the random splits. Furthermore, in Table 3.7, the Freezing baseline offers better results than LFL, and is better or competitive enough with the model-based approaches.

3.4.4 Effect of starting-task size on Tiny ImageNet

We propose an experimental setup where the first task of Tiny ImageNet uses 110 classes (55%) while the remaining 9 tasks use 10 classes (5%) each. This allows most of the methods to start with a rich representation after learning the first task. In this setup, comparing existing methods with the Freezing baseline is interesting. In Table 3.8 we show the results of this scenario. Note that in the last column we average only over the smaller tasks (T2-T10). EWC shows little forgetting, keeping knowledge while learning new tasks. However, keeping the first task from forgetting causes the model to become too rigid and learn the rest of the tasks with more difficulty and having a lower overall performance. Trying to lower the trade-off hyperparameter shows a stronger forgetting of the first tasks and causes severe catastrophic forgetting. Distillation approaches try to keep representations the same as new tasks are learned. However, small changes in the weights cause forgetting later into the sequence. HAT works fine, but with a limited capacity to make changes, ends up not learning the new tasks as easily. Freezing the network after the first task seems to be one of the best options in this setup, since the rich representation of the first 110 classes is a good starting point to learn the rest of the tasks with a simple classifier. We therefore advocate that the Freezing baseline should always be included in continual learning comparisons since it often provides a much harder baseline than Fine-tuning. Only PackNet and TFM are able to improve over that baseline even if they start from a smaller capacity, with TFM having the best results.

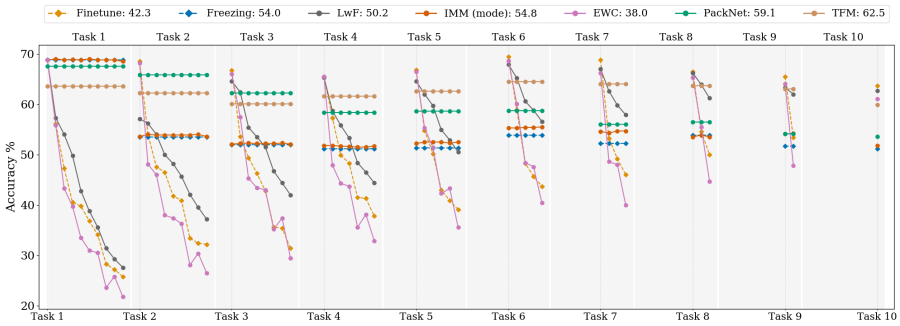

Figure 3.5 – Per task accuracy for ImageNet on AlexNet from scratch.

3.4.5 ImageNet

Most of the compared task-aware approaches have not been evaluated using a large-scale dataset such as ImageNet. We therefore compare our proposed method (TFM) with some of those state-of-the-art approaches. In Table 3.9 we can see that TFM outperforms all the other approaches at the end of learning ImageNet split into 10 tasks of random classes. An evolution of the results for each task is shown in Fig. 3.5. LwF does well when learning each new task with the help of the representation of the previous tasks. However, as more tasks are included, the older tasks start forgetting more. IMM (mode) has the opposite effect, it focuses on intransigence and tries to keep the knowledge of the older tasks, running out of capacity for the newer tasks. This allows for the approach to not forget much and even have a small backward transfer, but at the cost of performing worse with newer tasks. EWC has one of the worse performances, possibly due to the difficulty of having a good approximation of the FIM when there is so many classes per task. Both PackNet and TFM have a good overall performance with non-forgetting, and rely on the amount of capacity of the network more than the other approaches. As shown in Fig. 3.6, PackNet has a better performance during the first three tasks, taking advantage of the compression power of the pruning and fine-tuning. However, as the remaining capacity of the network gets smaller, TFM is capable of growing at a more scalable pace, getting a better performance on the remaining seven tasks and achieving the best results overall.

3.4.6 Comparison of memory usage

Previous experiments show that mask-based approaches are better at overcoming catastrophic forgetting on task-aware settings. However, unlike most distillation and

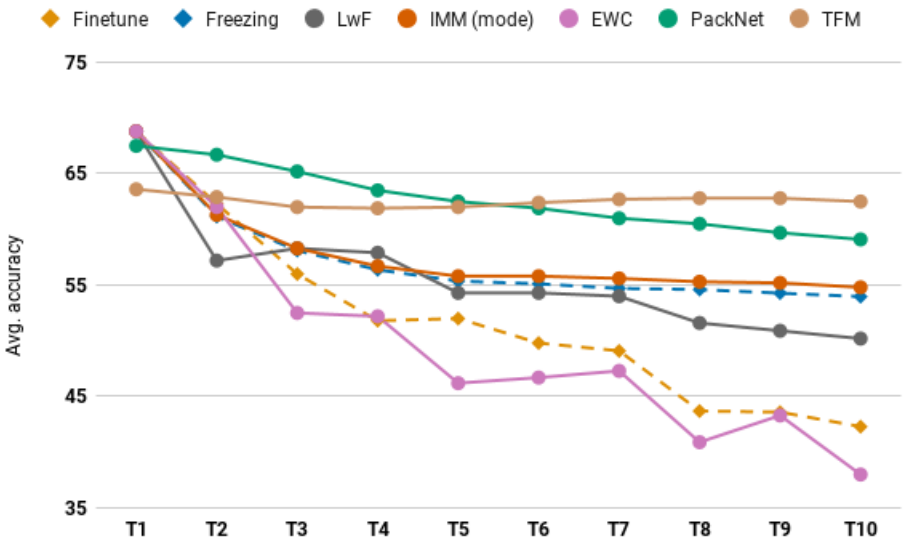

Figure 3.6 – Average accuracy progress after learning each task from ImageNet on AlexNet from scratch.

model-based approaches, they make use of some overhead memory during inference. In Fig. 3.7 we visualize (in log scale) the absolute memory overhead used by the mask-based approaches on the ImageNet experiment in Sec 3.4.5. Considering that the network used is around 220Mb, we can observe that the approaches that focus on using embeddings or masks on the features (HAT, TFM) have a negligible overhead in comparison to the weight-masked approaches (PackNet). It should also be noticed that mask-based approaches keep the same overhead during training, while distillation and model-based approaches usually duplicate the network size at least. In conclusion, our method has similar memory usage as HAT, however, we outperform this method on all proposed experiments, and our method is significantly more memory efficient than PackNet whose performance we either match or outperform.

3.5 Conclusions

For many practical applications, it is important that network accuracy on tasks does not deteriorate when learning new tasks. Therefore, in this chapter, we propose a new method for continual learning which does not suffer from any forgetting. Differently than previous methods which apply masks to the weights, we propose

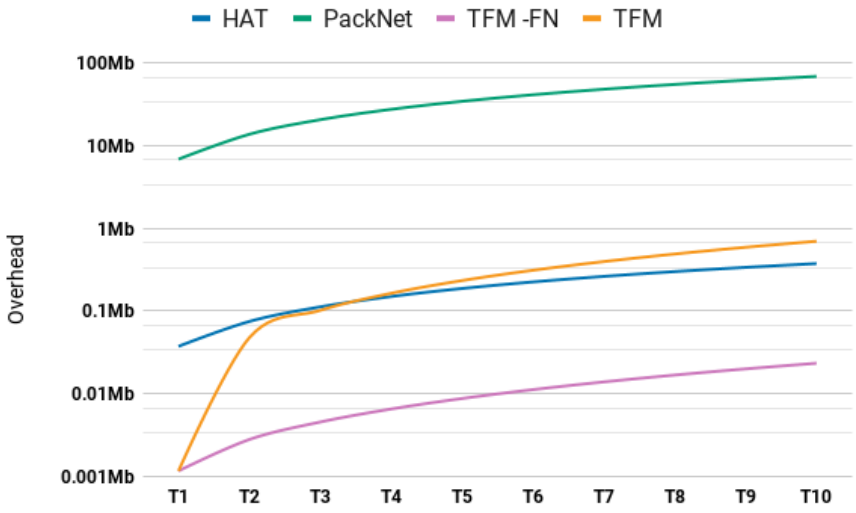

Figure 3.7 – Log scale overhead growth for ImageNet on AlexNet.

to move the mask to the features (activations). This greatly reduces the number of extra parameters which are added per task and reduces the overhead of the network which other approaches incur. In addition, we propose to apply a task-specific feature normalization of features, which allows adjusting previously learned features to new tasks. In ablation experiments this was found to improve results of the ternary feature masks. Furthermore, when compared to a wide range of other continual learning techniques, our method consistently outperforms those on a variety of datasets.

4 Class-incremental learning*

4.1 Introduction

Incremental learning, also often referred to as *continual* or *lifelong learning*, aims to develop artificially intelligent systems that can continuously learn to address new tasks from new data while preserving knowledge learned from previously learned tasks [135, 164]. In most incremental learning (IL) scenarios, tasks are presented to a learner in a sequence of delineated *training sessions* during which only data from a single task is available. After each training session, the learner should be capable of performing all previously seen tasks on unseen data. The biological inspiration for this learning model is clear, as it reflects how humans acquire and integrate new knowledge: when presented with new tasks to learn, we leverage knowledge from previous ones and integrate newly learned knowledge into previous tasks [44].

This contrasts markedly with the prevailing supervised learning paradigm in which labeled data for all tasks is jointly available during a single training session of a deep network. Incremental learners only have access to data from a single task at a time while being evaluated on all learned tasks so far. The main challenge in incremental learning is to learn from data from the current task in a way that prevents forgetting of previously learned tasks. The naive approach of fine-tuning, so fruitfully applied to domain transfer problems, suffers from the lack of data from previous tasks and the resulting classifier is unable to classify data from them. This drastic drop in performance on previously learned tasks is a phenomenon known as *catastrophic forgetting* [50, 74, 111]. Incremental learning aims to prevent catastrophic forgetting, while at the same time avoiding the problem of *intransigence* which inhibits adaptation to new tasks [24].

In addition to the biological motivations for study, there are numerous practical advantages to incremental learning. Incremental learners optimize computational resources better than classical supervised learning. In the classical supervised learning approach, when presented with new tasks to incorporate into the scope of an artificially intelligent system, the best (often only) option is to jointly retrain over all tasks, previous and new. However, in many scenarios data from previous tasks may not be available due to privacy considerations.

*This chapter is based on a submission to a journal, 2020 [105].

We adopt the viewpoint on continual learning first proposed along with the iCaRL approach [135] and the terminology used in [167]. In incremental learning the training is divided into a sequence of tasks, and in any training session the learner has only access to the data of the current task (optionally, some methods can consider a small amount of stored data from previous tasks). Most early methods for incremental learning considered the scenario, known as *task-incremental learning* (task-IL), in which the algorithm has access to a task-ID at inference time. This has the clear advantage that methods do not have to discriminate between classes coming from different tasks. More recently, methods have started addressing the more difficult scenario of *class-incremental learning* (class-IL),[†] where the learner does not have access to the task-ID at inference time, and therefore must be able to distinguish between all classes from all tasks. In the last few years a wide variety of methods for class-incremental learning have been proposed, and we believe the time is ripe to provide a broad overview and experimental comparison of them in one place.

In this chapter we set out to identify the main challenges for class-IL, and we organize the proposed solutions in three main categories: *regularization-based* solutions that aim to minimize the impact of learning new tasks on the weights that are important for previous tasks; *exemplar-based* solutions that store a limited set of exemplars to prevent forgetting of previous tasks; and solutions that directly address the problem of *task-recency bias*, a phenomenon occurring in class-IL methods that refers to the bias towards recently-learned tasks. In addition to an overview of progress in class-IL in recent years, we also provide an extensive experimental evaluation of existing methods. We evaluate several of the more popular regularization methods (often proposed for task-IL), and extend them with exemplars for a more fair comparison to recently developed methods. In addition, we perform extensive experiments comparing twelve methods on several scenarios and also evaluate class-IL methods on a new, more challenging multi-dataset setting. Finally, we are the first to compare these methods on a wide range of network architectures. We provide code for our extensible class-IL evaluation framework that ensures reproducibility of all results (see Sec. 4.5.1).

This chapter is organized as follows. In Sec. 4.2 we describe related work, focusing mainly on the IL literature not included in the experiments. In Sec. 4.3 we define class-IL and the main challenges which need to be addressed. In Sec. 4.4 we start by defining the scope of methods we consider for our experimental evaluation based on a list of desiderata. Then we introduce the main approaches that have been proposed for class-IL. In Sec. 4.5, we outline our experimental setup and follow with an extensive experimental evaluation in Sec. 4.6. In Sec. 4.7 we discuss several emerging trends in class-IL and then finish with conclusions in Sec. 4.8.

[†]We do not refer to the scenario where each task only contains a single class, but consider adding a group of classes for each task.

4.2 Related work

In this section we broadly review related work from the literature on incremental learning. Detailed discussion of class-incremental methods specifically covered by this chapter is deferred until Sec. 4.4.

Existing surveys. The problem of catastrophic forgetting has been acknowledged for many years. Already in the eighties, McCloskey and Cohen [111] showed that while human subjects suffered from gradual forgetting of previously learned tasks, algorithms trained with backpropagation suffered from catastrophic forgetting. Radcliff [134] confirmed this finding on a wider range of tasks trained with backpropagation. An excellent review on early approaches to mitigating catastrophic forgetting is by French [44]. This review also discusses how the brain prevents catastrophic forgetting and lays out possible solutions for neural network design. With the resurgence of deep learning from around 2011 [78], after breakthroughs in hardware (GPUs) and availability of large labeled datasets (like ImageNet [36]), the problem of catastrophic forgetting quickly gained renewed attention [50, 74]. This led to a surge of work in incremental learning, continual learning and lifelong learning.

This surge of new research has also resulted in recent surveys on the subject. Parisi et al. [124] provide an extensive survey on lifelong learning. This review is especially valuable because of its in-depth discussion of how biological systems address lifelong learning. They thoroughly discuss biologically-motivated solutions, such as structural plasticity, memory replay, curriculum and transfer learning. Another review [84] focuses on continual learning for robotics, and puts special effort into unifying evaluation methodologies between continual learning for robotics and non-robotics applications, with the aim of increasing cross-domain progress in continual learning. These reviews, however, do not provide an experimental performance evaluation of existing methods in the field.

Some recent surveys do include evaluation of methods. Pfulb and Gepperth [127] propose a training and evaluation paradigm for task-IL methods. Their evaluations are limited to two tasks. Lomonaco and Maltoni [94] evaluate several strategies on weight initialization: random initialization, starting from a pretrained network and starting from a pretrained but only learning the classifier. De Lange et al. [34] perform an extensive survey of task-IL with an experimental evaluation, including an analysis of model capacity, weight decay, and dropout regularization within context of task-IL. In addition, they propose a framework for continually determining the stability-plasticity trade-off of the learner – which we also apply in our evaluation. Existing surveys focus on task-IL, and to the best of our knowledge there is no survey which categorizes and broadly evaluates class-IL approaches. Given the increased attention to class-IL in recent years, we think such a survey is timely.

Task-incremental learning. As discussed in [34], most task-IL methods can be grouped into families of techniques having similar characteristics. Most regularization-based and replay-based methods can be applied to both task-IL and class-IL. Those are described more in depth in Sec. 4.4. Parameter isolation methods are usually applied to task-IL problems since they become computationally expensive or under-perform without access to the task-ID.

Mask-based methods, as seen in Chapter 3, reduce or completely eliminate catastrophic forgetting by applying masks to each parameter or to each layer's representations. However, by learning useful paths for each task in the network structure, the simultaneous evaluation of all learned tasks is not possible. This forces several forward passes with different masks, which makes such methods very effective for task-aware evaluation, but impractical for task-agnostic settings [34, 107]. Piggyback learns masks on network weights while training a backbone [101]. PackNet learns weights and then prunes them to generate masks [102]. HAT [153] applies attention masks on layer activations to penalize modifications to those that are important for a specific task. TFM [107], our work detailed in Chapter 3, applies ternary masks on the activations during training so that weights can be reused but not modified if they relate to already learned tasks. DAN [142] combines existing filters to learn filters for new tasks. Finally, PathNet [43] learns selective routing through the weights using evolutionary strategies.

Architecture growing methods dynamically increase the capacity of the network to reduce catastrophic forgetting. They rely on promoting a more intransigent model capable of maintaining previous task knowledge, while extending that model in order to learn new tasks. This makes some of these methods impractical when the task-ID is not known, or adds too many parameters to the network which makes them unfeasible for large numbers of tasks. EG [6] duplicates the model for each new task in order completely eliminate forgetting. PNN [144] duplicates each layer and adds lateral connections between duplicates for each task. Old weights are fixed, allowing access to that information while learning the new task. However, at each task the networks grows quadratically. To solve that issue, P&C [151] proposes duplicating the network only once to keep the number of parameters fixed, and use Elastic Weight Consolidation (EWC [74]) to mitigate forgetting. RCL [178] adaptively expands each layer of the network and uses an RNN controller to determine the number of filters to add for each task. In LtG [85], the authors propose an architecture search approach. A network structure optimization component allows each individual layer to be reused, extended or duplicated, while a learning component fine-tunes parameters.

Online learning. Online methods are based on streaming frameworks where learners are allowed to observe each example only once instead of iterating over a set of examples in a training session [125]. Instances are non-i.i.d and usually have temporal

correlation. Lopez-Paz [97] establishes definitions and evaluation methods for this setting and describes GEM, which uses a per-task exemplar memory to constrain gradients so that the approximated loss from previous tasks is not increased. A-GEM [26] improves on GEM in efficiency by constraining based on the average of gradients from previous class exemplars. However, the authors of [27] show that simply training on the memorized exemplars, similar to the well-established technique in reinforcement learning [115], outperforms previous results. GSS [7] performs gradient-based exemplar selection based on the GEM and A-GEM procedure to allow training without knowing the task boundaries. MIR [5] trains on the exemplar memory by selecting exemplars that will have a bigger loss increase after each training step. In [140] the memory is used to store discrete latent embeddings from a Variational Autoencoder that allows generation of previous task data for training. MER [139] combines experience replay with a modification of the meta-learning method Reptile [119] to select replay samples which minimize forgetting.

Variational continual learning. Variational continual learning is based on the Bayesian inference framework. VCL [118] proposes to merge online and Monte Carlo variational inference for neural networks yielding variational continual learning. It is general and applicable to both discriminative and generative deep models. VGL [41] introduces Variational Generative Replay, a variational inference generalization of Deep Generative Replay (DGR), which is complementary to VCL. UCL [3] proposes uncertainty-regularized continual learning based on a standard Bayesian online learning framework. It gives a fresh interpretation of the Kullback-Leibler (KL) divergence term of the variational lower bound for the Gaussian mean-field approximation case. FBCL [29] proposes to use Natural Gradients and Stein Gradients to better estimate posterior distributions over the parameters and to construct coresets using approximated posteriors. IUVCL [161] proposes a new best-practice approach to mean-field variational Bayesian neural networks. These methods normally consider only the task-aware setting. BGD [186] updates the posterior in closed form and that does not require a task-ID.

Pseudo-rehearsal methods. In order to avoid storing exemplars and privacy issues inherent in *exemplar rehearsal*, some methods learn to generate examples from previous tasks. DGR [154] generates those synthetic samples using an unconditional GAN. An auxiliary classifier is needed to assign ground truth labels to each generated sample. An improved version is proposed in MeRGAN [175], where a label-conditional GAN and replay alignment are used. DGM [122] combines the advantages of conditional GANs and synaptic plasticity using neural masking. A dynamic network expansion mechanism is introduced to ensure sufficient model capacity. Lifelong GAN [187] extends image generation without catastrophic forgetting from label-conditional to image-conditional GANs. As an alternative to exemplar rehearsal, some methods

perform *feature replay* [73, 177], which need a fixed backbone network to provide good representations.

Incremental Learning beyond image classification. Shmelkov et al. [155] propose to learn object detectors incrementally. They use Fast-RCNN [46] as the network and propose distillation losses on both bounding box regression and classification outputs. Additionally, they choose to distill the region proposal with the lowest background scores, which filters out most background proposals. Hao et al. [55] extend Faster-RCNN [137] with knowledge distillation. Similarly, Michieli et al. [114] propose to distill both on the output logits and on intermediate features for incremental semantic segmentation. Recently, Cermelli et al. [23] model the background by revisiting distillation-based methods and the conventional cross entropy loss. Specifically, previous classes are seen as background for the current task and current classes are seen as background for distillation. Incremental semantic segmentation has also been applied to remote sensing [163] and medical data [123].

Catastrophic forgetting has been mainly studied in feed-forward neural networks. Only recently the impact of catastrophic forgetting in recurrent LSTM networks was studied [146]. In this work, they observe that catastrophic forgetting is even more notable in recurrent networks than feed-forward networks. This is because recurrent networks amplify small changes of the weights. To address catastrophic forgetting an expansion layer technique for RNNs was proposed in [32]. A Net2Net technique [28] was combined with gradient episodic memory in [158]. In addition, they propose a benchmark of tasks for training and evaluating models for learning sequential problems. Finally, preventing forgetting for the task of captioning was studied in [35].

4.3 Class-incremental learning

In this section, we define the specifics of the class-incremental learning setup we consider and discuss the main challenges that class-IL methods must address.

4.3.1 General class-incremental learning setup

Our investigation focuses on class-incremental learning scenarios in which the algorithm must learn a sequence of tasks. By *task*, we refer to a set of classes disjoint from classes in other (previous or future) tasks. In each *training session* the learner only has access to data from a single task. We optionally consider a small memory that can be used to store some exemplars from previous tasks. Tasks consist of a number of classes, and learners are allowed to process the training data of the current task multiple times during the training session. We do not consider the online learning setting used in some papers [97] in which each data sample is only seen once.

More formally, an incremental learning problem $\mathscr{T}$ consists of a sequence of n tasks:

$$\mathscr{T} = [(C^1, D^1), (C^2, D^2), \ldots, (C^n, D^n)], \tag{4.1}$$

where each task t is represented by a set of classes $C^t = \{c_1^t, c_2^t ..., c_{n^t}^t\}$ and training data D^t. We use N^t to represent the total number of classes in all tasks up to and including task t: $N^t = \sum_{i=1}^{t} |C^i|$. We consider class-incremental classification problems in which $D^t = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_{m^t}, \mathbf{y}_{m^t})\}$, where $\mathbf{x}$ are input features for a training sample, and $\mathbf{y} \in \{0,1\}^{N^t}$ is a one-hot ground truth label vector corresponding to $\mathbf{x}_i$. During training for task t, the learner only has access to D^t, and the tasks do not overlap in classes (i.e. $C^i \cap C^j = \emptyset$ if $i \neq j$).

Our incremental learners are deep networks parameterized by weights θ and we use $\mathbf{o}(\mathbf{x}) = h(\mathbf{x}; \theta)$ to indicate the output logits of the network on input $\mathbf{x}$. We further split the neural network in a feature extractor f with weights ϕ and linear classifier g with weights V according to $\mathbf{o}(\mathbf{x}) = g(f(\mathbf{x}; \phi); V)$. We use $\hat{\mathbf{y}} = \sigma(h(\mathbf{x}; \theta))$ to identify the network predictions, where σ indicates the softmax function. After training on task t, we evaluate the performance of the network on all classes $\bigcup_{i=1}^{t} C^i$. This contrasts with task-IL where the task-ID t is known and evaluation is only over task C^t at inference time.

Most class-IL classifiers are trained with a cross-entropy loss. When training only on data from the current task t, we can consider two cross-entropy variants. We can consider a cross-entropy over *all* classes up to the current task:

$$\mathscr{L}_c(\mathbf{x}, \mathbf{y}; \theta^t) = \sum_{k=1}^{N^t} y_k \log \frac{\exp(\mathbf{o}_k)}{\sum_{i=1}^{N^t} \exp(\mathbf{o}_i)}. \tag{4.2}$$

Note that in this case, since the softmax normalization is performed over *all* previously seen classes from all previous tasks, errors during training are backpropagated from all outputs – including those which do not correspond to classes belonging to the current task.

Instead, we can consider only network outputs for the classes belonging to the current task t and define the following cross-entropy loss:

$$\mathscr{L}_{c^*}(\mathbf{x}, \mathbf{y}; \theta^t) = \sum_{k=1}^{|C^t|} y_{N^{t-1}+k} \log \frac{\exp(\mathbf{o}_{N^{t-1}+k})}{\sum_{i=1}^{|C^t|} \exp(\mathbf{o}_{N^{t-1}+i})} \tag{4.3}$$

This loss only considers the softmax-normalized predictions for classes from the *current* task. As a consequence, errors are backpropagated only from the probabilities

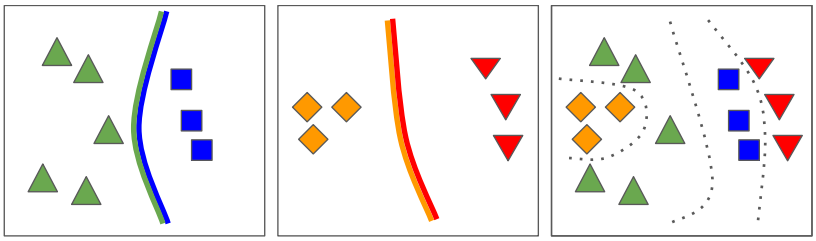

Figure 4.1 – A network trained continually to discriminate between task 1 (left) and task 2 (middle) is unlikely to have learned features to discriminate between the four classes (right). We call this problem *inter-task confusion*.

related to these classes from task t.

When using exemplars representing data from previous tasks, it is natural to apply Eq. 4.2 which considers the estimated probabilities on both previous and new classes. However, in the results section we will show that when no exemplars are used, Eq. 4.3 results in significantly less catastrophic forgetting. Interestingly, fine-tuning with Eq. 4.3 leads to a much stronger baseline than fine-tuning with Eq. 4.2, which is generally reported in literature.

4.3.2 Challenges of class-incremental learning

The fundamental obstacles to effective class-incremental learning are conceptually simple, but in practice very challenging to overcome. These challenges originate from the sequential training of tasks and the requirement that at any moment the learner must be able to classify all classes from all previously learned tasks. Incremental learning methods must balance retaining knowledge from previous tasks while learning new knowledge for the current task. This problem is called the *stability-plasticity dilemma* [113]. A naive approach to class-IL which focuses solely on learning the new task will suffer from *catastrophic forgetting*: a drastic drop in the performance on previous tasks [50, 111]. Preventing catastrophic forgetting leads to a second important problem of class-IL, that of *intransigence*: the resistance to learn new tasks [24]. There are several causes of catastrophic forgetting in class-incremental learners:

- **Weight drift**: While learning new tasks, the network weights relevant to *old* tasks are updated to minimize a loss on the *new* task. As a result, performance on previous tasks suffers – often dramatically.

- **Activation drift**: Closely related to weight drift, changing weights result in changes to activations, and consequently in changes to the network output. Focusing on activations rather than on weights can be less restrictive since this allows weights to change as long as they result in minimal changes in layer activations.
- **Inter-task confusion**: in class-IL the objective is to discriminate all classes from all tasks. However, since classes are never jointly trained the network weights cannot optimally discriminate all classes (see Fig. 4.1). This holds for all layers in the network.
- **Task-recency bias**: Separately learned tasks might have incomparable classifier outputs. Typically, the most dominant task bias is towards more recent task classes. This effect is clearly observed in confusion matrices which illustrate the tendency to miss-classify inputs as belonging to the most recently seen task (see Fig. 4.2).

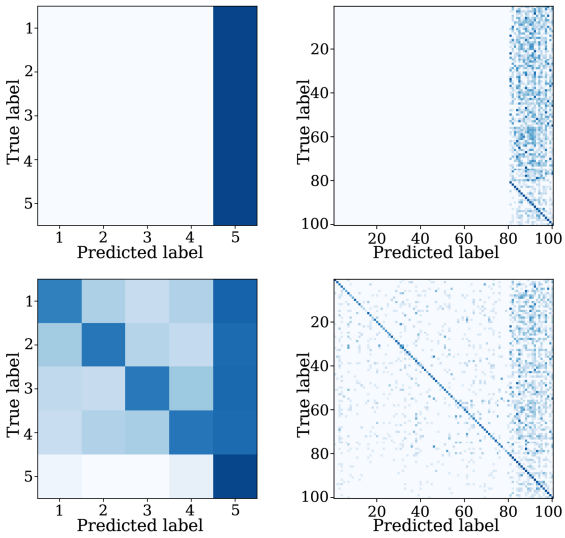

Figure 4.2 – Examples of task and class confusion matrices for fine-tuning (top row) and fine-tuning with 2,000 exemplars (bottom row) on CIFAR-100. Note the large bias towards the classes of the last task for fine-tuning. By exploiting exemplars, the resulting classifier is clearly less biased.

The first two sources of forgetting are related to network drift and have been broadly considered in task-IL literature. Regularization-based methods either focus on preventing the drift of important weights [4, 24, 74, 185] or the drift of activations [72, 87].

The last two points are specific to class-incremental learners since they have no access to a task-ID at inference time. Much of the research in class-IL has focused on reducing task imbalance [13, 66, 176], which addresses the task-recency bias. To prevent inter-task confusion and learn representations which are optimal to discriminate between all classes, rehearsal [22,135] or pseudo-rehearsal [154,175,177] is commonly used.

4.4 Approaches

In this section, we describe several approaches to address the above mentioned challenges of class-incremental learning. We divide them into three main categories: regularization-based methods, rehearsal-based methods, and bias-correction methods. First, we discuss the scope of this chapter by articulating and motivating the properties that class-incremental learning methods should have for consideration in our experimental evaluation.

4.4.1 Scope of our experimental evaluation

The literature on IL is vast and growing, and several definitions and interpretations of class-IL have been proposed in recent years [34, 135, 151, 167]. In order to narrow the scope of this survey to a broad group of usefully comparable methods, we consider class-IL methods that are:

1. **Incremental**: methods that are trainable from a stream of data drawn from a non-stationary distribution.

2. **Task transferable**: class-incremental learners capable of exploiting knowledge from previous classes to improve learning of new ones (forward transfer), as well as exploiting new data to improve performance on previous tasks (backward transfer).

3. **Task-agnostic**: incremental learners able to predict classes from all previously learned tasks in a *task-agnostic* way (i.e. without recourse to a task oracle providing a subset possible classes).

4. **Offline**: methods in which data is presented in *training sessions* whose data is *i.i.d* and can be processed multiple times before moving on to the next task.

5. **Fixed network architecture**: methods using a fixed architecture for all tasks, without adding significant amount of parameters to the architecture for new tasks.

6. **Tabula rasa**: incremental learners trained from scratch which do not require pretraining on large labeled datasets. This property eliminates potential biases introduced by the class distributions seen during pretraining and any exploits derivable from that knowledge.

7. **Mature**: methods applicable to complex image classification problems.

Properties 1 and 2 are intrinsic characteristics of IL methods, and property 3 distinguishes class-incremental from task-incremental learning. While properties 4–7 are characteristics that we use to select methods for our evaluation.

Finally, we consider one additional property:

8. **Exemplar-free**: methods not requiring storage of image data from previous tasks. This is an important characteristic of methods which should be privacy-preserving (optional).

Our evaluation considers methods requiring image exemplars as well as those which do not and can therefore be applied in systems having stricter privacy considerations. Methods not requiring any data storage are seeing increased attention in a world where data privacy and security are fundamental for many users and are under increased legislative control.

4.4.2 Regularization approaches

Several approaches use regularization terms together with the classification loss in order to mitigate catastrophic forgetting. Some regularize on the weights and estimate an importance metric for each parameter in the network [4, 24, 74, 83, 90, 185], while others focus on the importance of remembering feature representations [72, 87, 131, 156, 188]. Most of these approaches have been developed within the context of task-IL and have been reviewed by other works [34]. Because of their importance also for class-IL, we discuss them briefly. Regularization of feature representations in particular is widely used in class-IL. Finally, we will describe several regularization techniques developed recently specifically for class-IL.

Weight regularization. The first class of approaches focuses on preventing weight drift determined to be relevant for previous tasks. They do so by estimating a prior importance of each parameter in the network (which are assumed to be independent) after learning each task. When training on new tasks, the importance of each parameter is used to penalize changes to them. That is, in addition to the cross-entropy

classification loss, an additional loss is introduced:

$$\mathscr{L}_{\text{reg}}(\theta^t) = \frac{1}{2}\sum_{i=1}^{|\theta^{t-1}|}\Omega_i(\theta_i^{t-1}-\theta_i^t)^2, \tag{4.4}$$

where θ_i^t is weight i of the network currently being trained, θ_i^{t-1} is the value of this parameter at the end of training on task $t-1$, $|\theta^{t-1}|$ is the number of weights in the network, and Ω_i contains importance values for each network weight.

Kirkpatrick et al. [74] proposed *Elastic Weight Consolidation* (EWC) in which Ω_i is calculated as diagonal approximation of the empirical Fisher Information Matrix. However, this captures the importance of the model at the minimum after each task is learned, while ignoring the influence of those parameters along the learning trajectory in weight space. In [90], they improve EWC by rotating the parameter space to one that provides a better approximation of the Fisher Information Matrix. However, the model has to be extended with fixed parameters during training, which does not increase the capacity of the network but incurs in a computational and memory cost.

In contrast, [185] proposed the *Path Integral* approach (PathInt), that accumulates the changes in each parameter online along the entire learning trajectory. As noted by the authors, batch updates to the weights might lead to overestimating the importance, while starting from pretrained models might lead to underestimating it. To address this, *Memory Aware Synapses* (MAS) [4] also proposes to calculate Ω_i online by accumulating the sensitivity of the learned function (the magnitude of the gradient). In [24], the *Riemanian Walk* (RWalk) algorithm is proposed: both Fisher Information Matrix approximation and online path integral are fused to calculate the importance for each parameter. In addition, RWalk uses exemplars to further improve results.

Data regularization. The second class of regularization-based approaches aims to prevent activation drift and is based on knowledge distillation [19, 62] which was originally designed to learn a more compact student network from a larger teacher network. Li et al. [87] proposed to use the technique to keep the representations of previous data from drifting too much while learning new tasks. Their method, called *Learning without Forgetting* (LwF) applies the following loss:

$$\mathscr{L}_{dis}\left(\mathbf{x};\theta^t\right) = \sum_{k=1}^{N^{t-1}}\pi_k^{t-1}(\mathbf{x})\log\pi_k^t(\mathbf{x}), \tag{4.5}$$

where $\pi_k(\mathbf{x})$ are temperature-scaled logits of the network:

$$\pi_k(\mathbf{x}) = \frac{e^{\mathbf{o}_k(\mathbf{x})/T}}{\sum_{l=1}^{N^{t-1}} e^{\mathbf{o}_l(\mathbf{x})/T}}, \tag{4.6}$$

and $\mathbf{o}(\mathbf{x})$ is the output of the network before the softmax is applied, and T is the temperature scaling parameter. We use $\pi^{t\text{-}1}$ to refer to the predictions of the network after training task t-1. The temperature scaling was introduced in [62] to help with the problem of having the probability of the correct class too high. Many methods use $T=2$ [22, 66, 87, 176] and we also apply that in the experiments. It is important to note that when using exemplars the distillation loss is typically also applied to the exemplars of previous classes [22, 66, 135, 176].

A very similar approach, called *less-forgetting learning* (LFL), was proposed by Jung et al. [72]. LFL preserves previous knowledge by freezing the last layer and penalizing differences between the activations before the classifier layer. However, since this can introduce larger issues when the domain shift is too large, other approaches introduced modifications to deal with it. Encoder-based lifelong learning [131] extends LwF by optimizing an undercomplete autoencoder which projects features to a manifold with fewer dimensions. One autoencoder is learned per task, which makes the growth linear, although the autoencoders are small compared to the total model size.

The learning without forgetting loss in Eq. 4.5 was originally proposed for a task-IL setting. However, it has since been a key ingredient of many class-IL methods [22, 66, 93, 135, 176, 188]. Some works have observed that the loss works especially well when the domain shift between tasks is small (as is typically the case for class-IL), however, when domain shifts are large its efficacy drops significantly [6].

Recent developments in regularization. Several new regularization techniques have been proposed in recent work on class-IL. Zagoruyko and Komodakis [184] proposed to use the attention of the teacher network to guide the student network. *Learning without Memorizing* (LwM) [38] applies this technique to class-IL. The main idea is that the attention used by the network trained on previous tasks should not change while training the new task. Features contributing to the decision of a certain class label are expected to remain the same. This is enforced by the attention distillation loss:

$$\mathscr{L}_{AD}\left(\mathbf{x};\theta^{t}\right) = \left\| \frac{Q^{t-1}(\mathbf{x})}{\|Q^{t-1}(\mathbf{x})\|_2} - \frac{Q^{t}(\mathbf{x})}{\|Q^{t}(\mathbf{x})\|_2} \right\|_1, \tag{4.7}$$

where the attention map Q is given by:

$$Q^{t}(\mathbf{x}) = \text{Grad-CAM}\left(\mathbf{x},\theta^{t},c\right) \tag{4.8}$$

$$Q^{t\text{-}1}(\mathbf{x}) = \text{Grad-CAM}\left(\mathbf{x},\theta^{t\text{-}1},c\right) \tag{4.9}$$

and is generated with the Grad-CAM algorithm [152]. Grad-CAM computes the gradient with respect to a target class c to produce a coarse localization map indicating

the image regions which contributed to the prediction. Here we cannot use the target class label, because this label did not exist when training the previous model θ^{t-1}. Instead, the authors propose to use the previous class predicted with highest probability to compute the attention maps: $c = \text{argmax}\, \text{h}\left(\mathbf{x}; \theta^{\text{t}-1}\right)$.

Another recent method building upon LwF is *Domain Model Consolidation* (DMC) [188]. It is based on the observation that there exists an asymmetry between previous and new classes when training: new classes have explicit and strong supervision, whereas supervision for old classes is weaker and communicated by means of knowledge distillation. To remove this asymmetry, they propose to apply a *double distillation* loss on the model $\theta^{t\text{-}1}$ trained on previous classes and a newly trained model θ^t for the new classes (allowing this model to forget previous tasks):

$$\mathscr{L}_{DD}(\mathbf{u};\theta) \;=\; \frac{1}{N^t}\sum_{k=1}^{N^t}\left(\mathbf{o}_k(\mathbf{u}) - \mathring{\mathbf{o}}_k(\mathbf{u})\right)^2, \tag{4.10}$$

where $\mathring{\mathbf{o}}_k$ are normalized logits:

$$\mathring{\mathbf{o}}_k(\mathbf{u}) \;=\; \begin{cases} \mathbf{o}_k^{t\text{-}1}(\mathbf{u}) - \dfrac{1}{N^{t\text{-}1}}\displaystyle\sum_{l=1}^{N^{t\text{-}1}} \mathbf{o}_l^{t\text{-}1}(\mathbf{u}) & \text{if } 1 \le k \le N^{t\text{-}1} \\ \mathbf{o}_k^{t}(\mathbf{u}) - \dfrac{1}{N^{t}}\displaystyle\sum_{l=1}^{N^{t}} \mathbf{o}_l^{t}(\mathbf{u}) & \text{if } N^{t\text{-}1} < k \le N^{t}. \end{cases} \tag{4.11}$$

Here $\mathbf{o}^{t\text{-}1}(\mathbf{u})$ refers to the logits from the network trained on previous tasks, and $\mathbf{o}^t(\mathbf{u})$ the ones trained on the the new classes. Because the algorithm does not have access to data of previous tasks, they propose to use auxiliary data $\mathbf{u}$, which can be any unlabeled data from a similar domain.

Finally, the *less-forget constraint* proposed by Hou et al. [66] in their method is a variant of LwF. Instead of regularizing network predictions, they propose to regularize on the cosine similarity between the L2-normalized logits of the previous and current network:

$$\mathscr{L}_{lf}(\mathbf{x};\theta) = 1 - \frac{\langle \mathbf{o}^{t\text{-}1}(\mathbf{x}), \mathbf{o}^t(\mathbf{x})\rangle}{||\mathbf{o}^{t\text{-}1}(\mathbf{x})||_2 ||\mathbf{o}^t(\mathbf{x})||_2}, \tag{4.12}$$

where $\langle\cdot,\cdot\rangle$ is the inner product between vectors. This regularization is less sensitive to task imbalance because the comparison is between normalized vectors. The authors show that this loss reduces bias towards new classes.

4.4.3 Rehearsal approaches

Rehearsal methods keep a small number of exemplars [22, 135, 176] (exemplar rehearsal), or generate synthetic images [122, 154] or features [73, 177] (pseudo-rehearsal). By replaying the stored or generated data from previous tasks rehearsal methods aim to prevent the forgetting of previous tasks. Most rehearsal methods combine the usage of exemplars to tackle the inter-task confusion with approaches that deal with other causes of catastrophic forgetting. The usage of exemplar rehearsal for class-IL was first proposed in *Incremental Classifier and Representation Learning* (iCaRL) [135]. This technique has since been applied in the majority of class-IL methods. In this section, we focus on the choices which need to be taken when applying exemplars.

Memory types. Exemplar memory must be extended at the end of a training session after the model has already been adapted to the new task. If the memory has a fixed maximum size across all tasks (fixed memory), some exemplars must first be removed to make space for new ones. This ensures that the memory capacity stays the same and the capacity is fixed. The more tasks and classes that are learned, the less representation each class has for rehearsal. After learning a certain amount of tasks, the memory could be expanded to better accommodate the new distributions. However, previously removed samples will be lost, thus the decision of when to expand is an important one. If the memory is allowed to grow (growing memory), then only new samples from the current task need to be added. This enforces the classes to have a stable representation during rehearsal across all tasks, at the cost of having a linear increase of memory, which might not be suitable in some applications. In both cases, the number of exemplars per class is enforced to be the same to ensure an equal representation of all classes.

Sampling strategies. The simplest way to select exemplars to add to the memory is by randomly sampling from the available data (random), which has been shown to be very effective without much computational cost [24, 135].

Inspired by [174], iCaRL proposes to select exemplars based on their corresponding feature space representations (herding). Representations are extracted for all samples and the mean for each class is calculated. The method iteratively selects exemplars for each of the classes. At each step, an exemplar is selected so that, when added to the exemplars of its class, the resulting exemplar mean is closest to the real class mean. The order in which exemplars are added is important, and taken into account when some need to be removed. Although this iterative selection procedure usually outperforms random, it increases computational cost.

In RWalk [24], two other sampling strategies are proposed. The first one calculates the entropy of the softmax outputs and selects exemplars with higher entropy (entropy). This enforces selection of samples that have more distributed scores across all classes.

Similarly, the second one selects exemplars based on how close they are to the decision boundary (distance), assuming that the feature space and the decision boundaries do not change too much. For a given sample $(\mathbf{x}_i, \mathbf{y}_i)$, the pseudo-distance to the decision boundary is calculated by $f(\mathbf{x}_i;\phi)^T V_{\mathbf{y}_i}$, meaning that the smaller the distance, the closer to the decision boundary.

For these sampling strategies (except for random), the order which exemplars are chosen is recorded following a decreasing order of importance. If a fixed memory is used and some memory must be freed to make space for new exemplars, the exemplars with the lower importance are the ones removed first.

Task balancing. When applying rehearsal during the training of a new task, the weight of the new classes compared to the previous ones is defined by the trade-off between the two parts of the loss, as well as the number of samples from each class at each training step. Most approaches sample the training batches from the joint pool between new data and rehearsal exemplars [24, 66, 135, 176]. This means that batches are clearly over-represented by new samples and rely on the trade-off between the cross-entropy loss and the other losses that prevent forgetting. In contrast [22] proposes having a more balanced training where batches are equally distributed between new and previous classes. This seems to have quite beneficial effects in compensating for the task imbalance during training.

Combining rehearsal and data regularization. Several methods [22, 66, 135, 176] use the distillation loss from Learning without Forgetting [87] to deal with the activation drift in combination with exemplars. However, Beloudah and Popescu [13] do the important observation that this distillation term actually hurts performance when using exemplars. We will confirm this in our results, however, we will show that in some scenarios a combination of weight regularization and exemplar rehearsal can be beneficial.

4.4.4 Bias-correction approaches

Bias-correction methods aim to address the problem of task-recency bias, which refers to the tendency of incrementally learned network to be biased towards classes in the most recently learned task. This is mainly caused by the fact that, at the end of training, the network has seen many examples of the classes in the last task but none (or very few in case of rehearsal) from earlier tasks. One direct consequence of this, as observed by Hou et al. [66], is that the classifier norm is larger for new classes than for the previous ones and that the classifier bias favors the more recent classes. This effect is shown in Fig. 4.3, where the lower biases and reduced norm of the classifier make less likely for the network to select any of the previous classes. In this section, we discuss several approaches to address this problem.

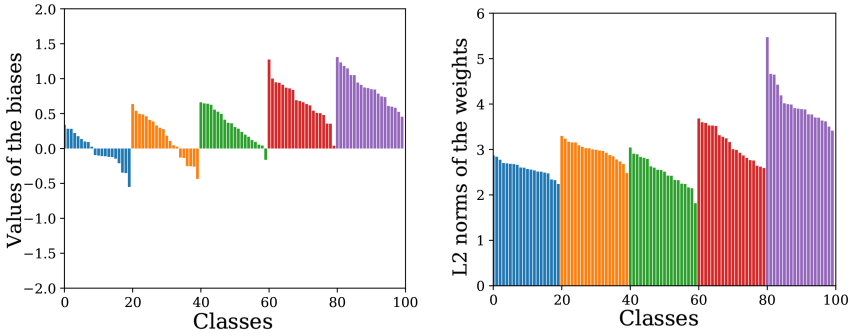

Figure 4.3 – Bias and weight analysis for iCaRL with 2,000 exemplars on CIFAR-100 on ResNet-32. We show the ordered biases and norm of the last classification layer of the network for different tasks. Note how the bias and the norm of the weights are higher for the last tasks. This results in a *task-recency bias*.

The earlier mentioned iCaRL method [135] combines exemplars and Learning without Forgetting, using a classifier layer and cross-entropy loss during training. To prevent the task-recency bias, they do not use the classifier at inference. Instead they compute the class mean of the exemplars in the feature representation, and then apply a nearest exemplar-mean for classification. Since this process is independent of the weights and biases of the final layer, the method was shown to be much less prone to the task-recency bias.

One simple yet effective approach to prevent task-recency bias has been proposed by Castro et al. [22] in their method *End-to-End Incremental Learning* (EEIL). They suggest introducing an additional stage, called *balanced training*, at the end of each training session. In this phase an equal number of exemplars from all classes is used for a limited number of iterations. To avoid forgetting the new classes, they introduce a distillation loss on the classification layer only for the classes from the current task. Balanced training could come at the cost of overfitting to the exemplars that have been stored, when these do not completely represent the distribution.

Another simple and effective approach to preventing task-recency bias was proposed by Wu et al. [176], who call their method *Bias Correction* (BiC). They add an additional layer dedicated to correcting task bias to the network. A training session is divided into two stages. During the first stage they train the new task with the cross-entropy loss and the distillation loss (see Eq. 4.5). Then they use a split of a very small part of the training data to serve as a validation set during the second phase.

They propose to learn a linear transformation on top of the logits, $\mathbf{o}_k$, to compensate for the task-recency bias. The transformed logits are given by:

$$\mathbf{q}_k = \alpha_s \mathbf{o}_k + \beta_s, \; c_k \in C^s \tag{4.13}$$

where α_s and β_s are the parameters which compensate for the bias in task s. For each task there are only two parameters which are shared for all classes in that task (initialized to $\alpha_1 = 1$ and $\beta_1 = 0$). In the second phase, all the parameters in the network are frozen, except for the parameters of the current task α_t and β_t. These are optimized with a standard softmax on the transformed logits $\mathbf{q}_k$ using the set-aside validation set. Finally, they only apply a weight decay on β parameters and not on the α parameters.

As mentioned earlier, task-recency bias was also observed by Hou et al. [66]. In their method *Learning a Unified Classifier Incrementally via Rebalancing* (LUCIR), they propose to replace the standard softmax layer σ with a cosine normalization layer according to:

$$\mathscr{L}_{cos}(\mathbf{x}; \theta^t) = \sum_{k=1}^{N^t} y_k \log \frac{\exp(\eta \langle \frac{f(\mathbf{x})}{||f(\mathbf{x})||}, \frac{V_k}{||V_k||} \rangle)}{\sum_{i=1}^{N^t} \exp(\eta \langle \frac{f(\mathbf{x})}{||f(\mathbf{x})||}, \frac{V_i}{||V_i||} \rangle)} \tag{4.14}$$

where $f(\mathbf{x})$ are the feature extractor outputs, $\langle \cdot, \cdot \rangle$ is the inner product, V_k are the classifier weights (also called class embedding) related to class k, and η is a learnable parameter which controls the peakiness of the probability distribution.

Hou et al. [66] also address the problem of inter-task confusion. To prevent new classes from occupying a similar location as classes from previous tasks, they apply the *margin ranking loss*. This loss pushes the current embedding away from the embeddings of the K most similar classes according to:

$$\mathscr{L}_{mr}(\mathbf{x}) = \sum_{k=1}^{K} \max\Big(m - \langle \tfrac{f(\mathbf{x})}{||f(\mathbf{x})||}, \tfrac{V_y}{||V_y||} \rangle + \langle \tfrac{f(\mathbf{x})}{||f(\mathbf{x})||}, \tfrac{V_k}{||V_k||} \rangle, 0\Big) \tag{4.15}$$

where $\hat{V}_y$ refers to the ground truth class embedding of $\mathbf{x}$, $\hat{V}_k$ refer to the embedding of the closest classes, and m is the margin.

Finally, another approach that addresses task-recency bias was proposed by Belouadah and Popescu [13] with their method called *Class-IL with dual memory* (IL2M). Their method is similar to BiC [176] in the sense that they propose to rectify the network predictions. However, where BiC learns to rectify the predictions by adding an additional layer, IL2M rectifies based on the saved certainty statistics of predictions of classes from previous tasks. Defining $m = \arg\max \hat{\mathbf{y}}(\mathbf{x})$, they compute the rectified

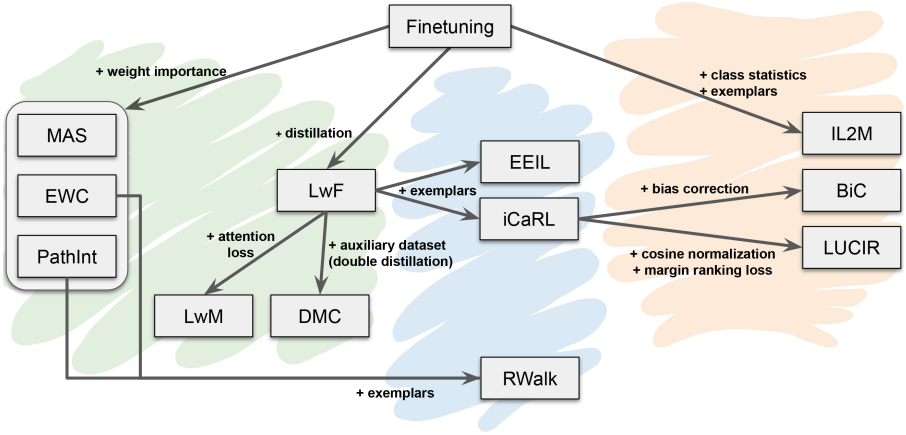

Figure 4.4 – Relation between class-incremental methods.

predictions of the previous classes k as:

$$\hat{y}_k^r(\mathbf{x}) = \begin{cases} \hat{y}_k(\mathbf{x}) \times \frac{\overline{y}_k^p}{\overline{y}_k^t} \times \frac{\overline{y}^t}{\overline{y}^p} & \text{if } m \in C^t \\ \hat{y}_k(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{4.16}$$

Here $\overline{y}_k^p$ (superscript p refers to past) is the mean of the predictions $\hat{y}_k$ for all images of class c_k after training the task in which class c_k is first learned ($c_k \in C^p$), and $\overline{y}^p$ is the mean of the predictions for all classes in that task. Both $\overline{y}_k^p$ and $\overline{y}^p$ are stored directly after their corresponding training session. $\overline{y}_k^t$ is the mean of the predictions $\hat{y}_k$ for all images of class c_k after training the new task (this is computed based on the exemplars). Similarly, $\overline{y}^t$ is the mean of the predictions for all classes in the new task. As can be seen the rectification is only applied when the predicted class is a new class ($m \in C^t$). If the predicted class is an old class, the authors argue that no rectification is required since the prediction does not suffer from task-imbalance.

4.4.5 Relation between class-incremental methods

In previous sections we discussed the main approaches to mitigating catastrophic forgetting by incremental learning methods. We summarize the relations between the discussed methods in Fig. 4.4 starting from the naive fine-tuning approach. In the diagram we show all methods which we compare in Sec. 4.6. The diagram

distinguishes between methods using exemplars to retain knowledge (blue, orange) and exemplar-free methods (green).

Most notably, the huge impact of Learning without Forgetting (LwF) [87] upon the whole field of class-incremental learning is clear. However, we expect that with the recent findings of [13], which show that when combined with exemplars fine-tuning can outperform LwF, could somewhat lessen its continuing influence. Weight regularization methods [4, 74, 185], applied frequently in the task-IL setting, are significantly less used for class-IL. They can also be trivially extended with exemplars and we include results of this in our experimental evaluation. Finally, Fig. 4.4 also shows the influence of iCaRL [135] in the development of more recent methods [66, 176].

4.5 Experimental setup

In this section, we explain the experimental setup and how we evaluate the approaches. We also introduce the baselines and the experimental scenarios used to gather the results presented in Sec. 4.6.

4.5.1 Code framework

In order to make a fair comparison between the different approaches, we implemented a versatile and extensible framework. Datasets are split into the same partitions and data is queued in the same order at the start of each task. All library calls related to randomness are synchronized and set to the same seed so that the initial conditions for all methods are the same. Data from previous tasks (excluding exemplar memory) is not available during training, thus requiring selection of any stability-plasticity-based trade-off before a training session of a task is completed (see also Sec. 4.5.6). All methods were implemented using the original author implementations (when available) which we adapted to work under the different experimental setups proposed below (i.e. different datasets and networks).

The current version of the code includes implementations of several baselines and the following methods: EWC, MAS, PathInt, RWalk, LwM, DMC, LwF, iCaRL, EEIL, BiC, LUCIR, and IL2M. The framework includes extending most exemplar-free methods with the functionality of exemplars. The framework facilitates using these methods with a wide variety of network architectures, and allows to run the various experimental scenarios we perform in this chapter. As such, our framework contributes to the wider availability and comparability of existing methods, which will facilitate future research and comparisons of class-IL methods[‡].

[‡]Code available at: https://github.com/mmasana/FACIL

Dataset	#Train	#Eval	#Classes
CIFAR-100 [77]	50,000	10,000	100
Oxford Flowers [120]	2,040	6,149	102
MIT Indoor Scenes [129]	5,360	1,340	67
CUB-200-2011 Birds [170]	5,994	5,794	200
Stanford Cars [76]	8,144	8,041	196
FGVC Aircraft [100]	6,667	3,333	100
Stanford Actions [180]	4,000	5,532	40
VGGFace2 [21]	491,746	50,000	1,000
ImageNet ILSVRC2012 [143]	1,280,861	50,000	1,000

Table 4.1 – Summary of datasets used.

4.5.2 Datasets

We study the effects of CL methods for image classification on nine different datasets whose statistics are summarized in Table 4.1. First, we compare the three main categories of approaches described in Sec. 4.4 on the CIFAR-100 dataset [77]. It contains 32×32 colour images for 100 classes, with 600 samples for each class divided into 500 for training and 100 for testing. For data augmentation, a padding of 4 is added to each side, and crops of 32×32 are randomly selected during training and the center cropped is used during testing. Input normalization and random horizontal flipping are also performed.

Next, we use a variety of fine-grained classification datasets: Oxford Flowers [120], MIT Indoor Scenes [129], CUB-200-2011 Birds [170], Stanford Cars [76], FGVC Aircraft [100], and Stanford Actions [180]. These provide higher resolution and allow studying the effects on larger domain shifts when used as different tasks. To study the effects of the different approaches on smaller domain shifts we use the VGGFace2 dataset [21]. Since the original dataset has no standard splits for our setting, we keep the 1,000 classes that have the most samples and split the data following the setup from [13]. This means that this dataset is not totally balanced, but at least all used classes have a large enough pool of samples.

Finally, the ImageNet dataset [143] is used as a more realistic and large-scale scenario. It consists of 1,000 diverse object classes with different numbers of samples per class. Since this dataset takes time and needs a lot of resources, we also use the reduced ImageNet-Subset, which contains the first 100 classes from ImageNet as in [135].

In order to apply a patience learning rate schedule and an hyperparameter selection framework, an additional separate split of 10% from training is assigned to validation

to those datasets that do not provide one. For all datasets except CIFAR-100, images are resized to 256×256 with random crops of 224×224 for training and center crops for testing. Input normalization and random horizontal flipping are also performed.

4.5.3 Network architectures

As suggested in [60], ResNet-32 and ResNet-18 are commonly used in the literature for CIFAR-100 and datasets with larger resolution (input sizes of around 224×224×3), respectively. However, it is not so common to see other networks being used in class-IL. In this work, we also perform experiments with different networks and evaluate the effects they have on performance. Specifically, we use AlexNet [78], ResNet-18 [60], VGG-11 [157], GoogleNet [162] and MobileNets [67, 145]. We use different networks on different scenarios and make a wider comparison on ImageNet subset in Sec. 4.6.6). All experiments done on CIFAR-100 are trained on ResNet-32 from scratch.

We have selected the networks to represent a wide variety of network architectures commonly used in deep learning, allowing us to compare them within a continual learning setting. We have chosen AlexNet and VGG-11 as architectures which start with a number of initial convolutional layers followed by several fully connected layers. ResNets have achieved superior performance in many different computer vision tasks, and we therefore consider ResNet-18. We have also included GoogleNet which uses skip-connection and 1×1 convolutions are used as a dimension reduction module to reduce computation. We are also interested to evaluate incremental learning on compact networks. We have therefore selected MobileNet, which, to better trade off latency and accuracy, propose to replace standard convolution layers by depthwise separable convolutions. This makes them suitable to run on mobile devices.

4.5.4 Metrics

In incremental learning, $a_{t,k} \in [0,1]$ denotes the accuracy of task k after learning task t ($k \leq t$), which provides fine-grained information about the incremental process. In order to compare the overall incremental learning process, the *average accuracy* is defined as $A_t = \frac{1}{t}\sum_{i=1}^{t} a_{t,i}$ at task t. This measure is commonly used to compare performances of different methods with a single value. It has to be noted that when tasks have different number of classes, a weighted version needs to be used. Moreover, though *average accuracy* can be easily compared since it is a single value, it can also hide many insights. The more tasks that are learned, the more information that can be hidden. To address this, additional metrics focusing on selected key aspects of IL such as *forgetting* and *intransigence* [24] are needed.

Forgetting estimates how much the model forgot about previous task k at current task t and is defined as $f_{t,k} = \max_{i\in\{1,\dots,t-1\}} a_{i,k} - a_{t,k}$. As with accuracy, this measure

can be averaged over all tasks learned so far: $F_t = \frac{1}{t-1}\sum_{i=1}^{t-1} f_i^t$. The lower the F_t is, the less forgetting is happening during incremental learning. In a similar way, *intransigence* quantifies a model's inability to learn a new task. Both can be considered complementary measures that help understand the stability-plasticity dilemma. The borderline case of a model that is never trained after first task will have no forgetting at all, but will be unable to learn new tasks. *Intransigence* for the t-th task is calculated as $I_t = a_t^* - a_{t,t}$, where a_t^* is the accuracy of a reference model for task t trained jointly on all data. The lower the $I_t \in [-1, 1]$, the better the model for that task.

To gain more insight about the classifier performance, a *confusion matrix* can be used, which gives information of miss-classification between each pair of classes. Despite the fact that it is not a single-value metric, it is often used to summarize the behavior of a classifier across many incremental tasks.

4.5.5 Baselines

Training with only a cross-entropy loss (see Eq. 4.2) is the default Fine-tuning (FT) baseline common in most IL works. This learns each task incrementally while not using any data or knowledge from previous tasks and is often used to illustrate the severity of catastrophic forgetting. However, when moving to a class-IL scenario where all previous and new classes are evaluated, other fine-tuning variants can be considered. We might not update the weights corresponding to the outputs of previous classes (FT+), which avoids the slow forgetting due to not having samples for those classes (see Eq. 4.3). As seen in Table 4.2, this simple modification has an impact on the baseline performance. Since previous classes will not be seen, freezing the weights associated with them avoids biased modifications based only on new data. Furthermore, in the proposed scenarios approaches usually make use of an exemplar memory, which helps improve overall performance and avoid catastrophic forgetting by replaying previously seen classes. Therefore, as an additional baseline we also consider extending FT with the same exemplar memory as exemplar-based approaches (FT-E). The result of this is quite clearly more beneficial than the other FT baselines, and makes the baseline more comparable with approaches using the same memory.

In the case of Freezing (FZ), the baseline is also simple: we freeze all layers except the last one (corresponding to the classification layer or head of the network) after the first task is learned. Similarly to FT, we can also make the simple modification of not updating the weights directly responsible for the previous classes outputs (FZ+). This extends freezing to that specific group of weights which we know will not receive a gradient from previous class samples. As seen in Table 4.2, this leads to a more robust baseline. However, if we add exemplars (FZ-E) the performance decreases. We have also observed that, when starting from a larger or more diverse first task, freezing can achieve much better performance since the learned representations before freezing are

	T2	T3	T4	T5	T6	T7	T8	T9	T10
FT	33.9	27.9	19.1	17.7	12.2	11.6	10.2	9.0	7.9
FT+	39.7	32.4	25.5	20.7	16.4	14.3	12.7	11.0	9.7
FT-E (fixed)	59.4	55.2	46.6	49.1	45.9	42.3	38.2	39.5	36.5
FT-E (grow)	48.0	42.5	33.1	36.5	35.8	31.8	33.5	31.6	32.0
FZ	24.1	18.4	12.8	12.7	9.2	8.2	7.8	6.3	5.3
FZ+	40.5	31.1	24.4	24.0	21.1	18.9	17.2	16.1	14.8
FZ-E (fixed)	44.9	36.3	22.9	21.7	18.0	17.4	13.6	13.2	9.3
FZ-E (grow)	37.1	29.7	19.5	19.1	14.7	15.5	12.3	12.5	9.4

Table 4.2 – Average accuracy for different baseline variants on CIFAR-100 (10/10). E denotes using 2,000 exemplars (fixed memory) or 20 exemplars per class (grow) selected with herding. All baselines start with 75.3 accuracy after task 1.

more robust.

Finally, we also use as an upper bound the joint training over all seen data (Joint). In order to have this baseline comparable over all learned tasks, we perform incremental joint training which uses all seen data at each task, starting from the model learned for the previous one. This baseline gives us an upper bound reference for all learned tasks.

4.5.6 Hyperparameter selection

For a fair comparison of IL methods, two main issues with non-IL evaluation need to be addressed. The first one is that choosing the best hyperparameters for the sequence of tasks after those are learned is not a realistic scenario in that information from future tasks is used. A better comparison under an IL setting is to search for the best hyperparameters as the tasks are learned with the information at hand for each of them. Second, it makes the comparison very specific to the scenario, and in particular to the end of the specific sequence of tasks. It provides a less robust evaluation of the results over the rest of tasks, which means that other task sequence lengths are not taken into account. We feel that a broader evaluation of CL methods should include results over all tasks as if each of them were the last one for hyperparameter selection purposes.

In order to provide this more robust evaluation, we use the Continual Hyperparameter Framework (CHF) proposed in [34]. This framework assumes that at each task, only the data for that task is available, as in a real scenario. For each task, a *Maximal Plasticity Search* phase is used with Fine-tuning, and a *Stability Decay* phase is used with the corresponding method. This allows to establish a reference performance first

and find the best stability-plasticity trade-off second [34].

General hyperparameters: the CHF is used for the stability-plasticity trade-off hyperparameters that are associated to intransigence and forgetting when learning a new task. It first performs a learning rate (LR) search with Fine-tuning on the new task. This corresponds to the *Maximal Plasticity Search* phase.

The LR search is limited to {5e-1, 1e-1, 5e-2} on the first task since all experiments are trained from scratch. For the remaining tasks, the LR search is limited to the three values immediately lower than the one chosen for the first task from this set: {1e-1, 5e-2, 1e-2, 5e-3, 1e-3}. We use a patience scheme as a LR scheduler where the patience is fixed to 10, the LR factor to 3 (LR is divided by it each time the patience is exhausted), and the stopping criteria is either having a LR below 1e-4 or if 200 epochs have passed (100 for VGGFace2 and ImageNet). We also do gradient clipping at 10,000, which is mostly negligible for most training sessions except the first one. We use SGD with momentum set to 0.9 and weight decay fixed to 0.0002. Batch size is 128 for most experiments except 32 for fine-grained datasets and 256 for ImageNet and VGGFace2. All code is implemented using Pytorch.

Once the shared hyperparameters are searched, the best ones are fixed and the accuracy for the first phase is stored as a reference. The hyperparameter directly related to the stability-plasticity trade-off is set to a high value which represents a heavy intransigence to learn the new task, close to freezing the network so that knowledge is preserved. At each search step, the performance is evaluated on the current task and compared to the reference accuracy from the *Maximal Plasticity Search* phase. If the method accuracy is above the 80% of the reference accuracy, we keep the model and trade-off as the ones for that task. If the accuracy is below the threshold, the trade-off is reduced in half and the search continues. As the trade-off advances through the search, it becomes less intransigence and slowly converges towards higher forgetting, which ultimately would correspond to the Fine-tuning of the previous phase. This corresponds to the *Stability Decay* phase.

Specific hyperparameters: the hyperparameters that have no direct correspondence with the intransigence-forgetting duality are set to the recommended values for each of the methods. The methods have the following implementations:

- **LwF**: we implement the $\mathscr{L}_{dis}$ distillation loss following Eqs. 5-6, and fix the temperature scaling parameter to $T = 2$ as proposed in the original work (and used in most of the literature). This loss is combined with the $\mathscr{L}_c$ cross-entropy loss from Eqs. 2-3 with a trade-off that is chosen using the CHF and starts with a value of 10. In our implementation we choose to duplicate the older model for training to evaluate the representations (instead of saving them at the end of the previous session) to benefit from the data augmentation. That older model can be removed after the training session to avoid overhead storage.

- **EWC**: the fusion of the old and new importance weights is done with α = 0.5 to avoid the storage of the importance weights for each task. The Fisher Information Matrix is calculated by using all samples from the current task based on the predicted class. The loss introduced in Eq. 4 is combined with the $\mathscr{L}_c$ cross-entropy loss with a trade-off chosen using the CHF and with a starting value of 10,000.

- **PathInt**: we fix the damping parameter to 0.1 as proposed in the original work. As in LwF and EWC, the trade-off between the quadratic surrogate loss and the cross-entropy loss is chosen using the CHF with a starting value of 1.

- **MAS**: we implement MAS in the same way as EWC, with α = 0.5 and the same Fisher Information Matrix setting. The trade-off between the importance weights penalty and the cross-entropy loss is chosen using the CHF and a starting value of 400.

- **RWalk**: since it is a fusion of EWC and PathInt, the same parameters α = 0.5, Fisher Information Matrix setting and damping = 0.1 are fixed.The starting value for the CHF on the trade-off between their proposed objective loss and the cross-entropy loss is 10.

- **DMC**: we implement the $\mathscr{L}_{DD}$ double distillation loss from Eqs. 10-11. We set the auxiliary dataset batch size to 128, and the student is neither initialized from the previous tasks or new task models but random, as proposed in the original work.

- **LwM**: We combine the cross-entropy loss with the distillation loss and $\mathscr{L}_{AD}$ attention distillation using the β and γ trade-offs respectively. The β trade-off is the one that balances the stability-plasticity dilemma and we chose it using the CHF with a starting value of 2. The γ trade-off is fixed to 1 since it does not directly affect the stability-plasticity dilemma. Since there is no mention in the original work on which are the better values to balance the three losses, that last value was chosen after a separate test with values $\gamma \in (0, 2]$ and fixed for all scenarios in Section 6.

- **iCaRL**: we implement the five algorithms that comprise iCaRL. The distillation loss is combined with the cross-entropy loss during the training sessions and chosen using the CHF with a starting value of 4. However, during evaluation, the NME is used instead of the softmax outputs.

- **EEIL**: we implement EEIL with the balanced and unbalanced training phases. The unbalanced phase uses the hyperparameters shared across all methods.

However, for the balanced phase the LR is reduced by 10 and the number of training epochs to 40. As with LwF, $T = 2$ and the trade-off is chosen using the CHF starting at 10. However, we apply a slight modification to the original work by not using the addition of noise to the gradients. Our preliminary results with this method showed that it was consistently detrimental to performance, which provided a worse representation of the capabilities of the method.

- **BiC**: the distillation stage is implemented the same as LwF, as in the original paper, with $T = 2$. However, the trade-off between distillation and cross-entropy losses is not chosen using the CHF. The authors already propose to set it to $\frac{n}{n+m}$, where n is the number of previous classes, and m is the number of new classes, and we keep that decision. On the bias correction stage, also following the original work, we fix the percentage of validation split used from the total amount of exemplar memory to be 10%.

- **LUCIR**: for this method we make two changes on the architecture of the model. First, we replace the classifier layer by a cosine normalization layer following Eq. 14; and second we remove the ReLU from the penultimate layer to allow features to take both positive and negative values. However, since this procedure is only presented in the original work for ResNet models, we do not extend it to other architectures. The original code used a technique called imprint weights during the initialization of the classifier. However, since it was not mentioned in the original paper, and preliminary experiments showed no significant difference, we decided to not include it in our implementation.

  The cross-entropy loss is combined with the $\mathscr{L}_{lf}$ less-forget constraint from Eq. 12 and the $\mathscr{L}_{mr}$ margin ranking loss from Eq. 15. The number of new class embeddings chosen as hard negatives and the margin threshold are fixed to $K = 2$ and $m = 0.5$ as in the original work. The margin ranking loss is combined with the cross-entropy loss in a one-to-one ratio, while the less-forget constraint is chosen using the CHF with a starting value of 10, as is the trade-off related to the stability-plasticity dilemma.

- **IL2M**: since it only stores some statistics on the classes and applies them after the training is done in the same way as Fine-tuning, there is no hyperparameter to tune for this method.

Finally, the Fine-tuning, Freezing and Joint training baselines have no hyperparameters associated to them, reducing the Continual Hyperparameter Framework to only performing the learning rate search for each task before doing the final training.

4.5.7 Experimental scenarios

To make the following results section easier to read, we define a few experimental scenarios here. We denote a dataset with B tasks and A classes on the first task as (A/B). For example, a CIFAR-100 (10/10) experiment refers to splitting the dataset into 10 tasks with the first task having 10 classes. This corresponds to an equal split among tasks and classes, making for the total amount of 100 total classes. Another setting that we use is CIFAR-100 (50/11), which means that the first task has 50 classes, and the remaining 50 classes are divided into 10 tasks of 5 classes each. Among others, these are the two main proposed settings for evaluating the different approaches and their characteristics on simpler scenarios, before moving into larger and more realistic ones.

4.6 Experimental results

In this section, we discuss different aspects of continual learning approaches on CIFAR-100 being the default dataset for comparisons and we extend the experiments to larger datasets, such as VGGFace2, different fine-grained datasets and ImageNet. It includes analysis on different regularization methods with and without exemplars. Then we demonstrate how bias-correction methods perform through both task and class confusion matrices. We analyze how different aspects of exemplar usage affect performance, such as memory properties and sampling strategies. Additionally, we evaluate different methods on two popular scenarios whether the first task starts with half of classes. We also demonstrate how domain shift can result in different performance for continual learning. Next, we compare the most prominent methods on a wide range of network architectures. Then we experiment different methods on the large scale dataset ImageNet.

4.6.1 On regularization methods

Most of the regularization approaches have been proposed for a task-IL setting where the task-ID is known at inference time [72, 74, 83, 87, 131]. Since regularization is applied to weights or representations, they can be easily extended to a class-IL setting without much or any modification. This makes for a more challenging problem, and several more recent regularization methods already show results for class-IL [24, 38, 188]. Similarly to the baselines in Sec. 4.5.5, when not using exemplars, methods can freeze the weights of the final layer associated with previous classes to improve performance based on the assumption that only data from new classes is used during a training session. This helps the problem of vanishing weights from learned classes and the task-recency bias, especially when using weight decay.

	avg. acc. after	FT	LwF	EWC	PathInt	MAS	RWalk	DMC	LwM
No exemplars (task-IL)	task 2	55.9	73.2	61.6	62.0	64.8	63.2	71.5	73.6
	task 5	47.3	73.8	59.8	60.4	62.8	57.4	72.7	75.2
	task 10	37.3	65.8	56.4	56.6	53.9	46.6	67.0	69.1
No exemplars (Class-IL)	task 2	30.8	56.1	39.5	36.2	42.3	45.4	57.9	53.9
	task 5	13.1	40.9	24.0	21.8	24.4	23.9	42.2	37.3
	task 10	7.8	29.7	12.3	12.2	11.5	12.9	26.7	20.4
2,000 exemplars fixed memory (Class-IL)	task 2	59.4	59.9	56.8	56.0	56.2	56.5	-	64.7
	task 5	49.1	44.8	39.3	30.7	31.2	46.4	-	52.9
	task 10	36.5	31.8	25.8	10.5	19.1	31.4	-	38.1
20 exemplars per class growing memory (Class-IL)	task 2	48.0	51.4	43.2	42.0	41.4	41.3	-	52.1
	task 5	36.5	33.1	30.6	25.2	22.9	27.2	-	39.7
	task 10	32.0	27.4	23.9	15.8	16.8	18.9	-	33.5

Table 4.3 – Average accuracy for regularization-based methods on CIFAR-100 (10/10) on ResNet-32 trained from scratch.

In Table 4.3 we compare regularization-based methods for both task-IL and class-IL. Three methods that apply data regularization (LwF, DMC, LwM) and three weight regularization methods (EWC, PathInt, MAS) are compared on CIFAR-100 (10/10). The ten tasks are learned sequentially, and each method and setting shows average accuracy at the second, fifth and final tasks to illustrate different sequence lengths. We start by comparing the regularization methods without using exemplars. Results clearly show a significant drop in performance due to the lack of the task-ID, especially after 5 and 10 tasks. LwF obtains better results than weight-based regularization methods, which might explain why distillation has been the dominant approach for most rehearsal methods [22, 66, 135, 176].

We also expand the regularization methods with exemplars to see how it affects their performance. Note that these methods are originally proposed without exemplars, except for RWalk. In Table 4.3 we include results with a fixed memory of 2,000 exemplars, and with a growing memory of 20 exemplars per class. When using a fixed memory of exemplars, all methods improve after each task. However, that is not true in all cases for the growing memory. The reduced number of exemplars available when learning the first tasks in comparison to a fixed memory has some impact on the results. In this case, LwF outperforms EWC, PathInt and MAS, while having a similar performance than RWalk for fixed memory. Note how RWalk without exemplars does not show much improvement over other weight-based regularization methods, but that changes when a fixed memory is used. One of the most interesting results of this experiment is that LwM obtains the best results in all cases when combined with exemplars, even though the method was originally not proposed with exemplars. Furthermore, FT-E performs the second best in this scenarios, in front of LwF, as also

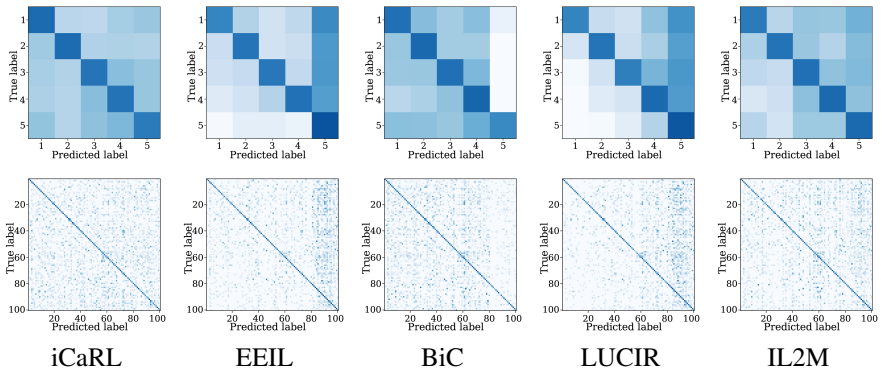

Figure 4.5 – Task (top) and class (bottom) confusion matrices. CIFAR-100 (20/5) with 2,000 exemplars selected with herding.

noticed in [13]. It should be noted that in some of the next experiments we find that weight regularization and exemplars can actually achieve good results.

Finally, DMC uses a large memory based on an auxiliary dataset (300 classes from ImageNet-32, as described in [188]), we provide task-IL and class-IL results while using said extra memory, and no exemplars from the learned classes are stored. The method provides privacy-preserving properties at the cost of some performance. However, we found that in these experiments the gain obtained by distillation from an additional dataset is rather small.

Given these results, in the following experiments we will mainly compare to the best performing regularization methods, namely LwF, LwM and EWC.

4.6.2 On bias-correction

As seen in Fig. 4.2, there exists a clear bias towards recent tasks. Here we evaluate the success of class-IL methods to address the task-recency bias. To allow for a better visualization, we use a CIFAR-100 (20/5) split with ResNet-32 trained from scratch and a fixed memory of 2,000 exemplars. In the text, we will also give in brackets the average accuracy after the last task for all methods we considered.

We show the task and class confusion matrices for different bias-correction approaches in Fig. 4.2 and Fig. 4.5. The FT-E baseline, despite having improved performance due to the use of rehearsal strategies (40.9), still has a clear task-recency bias. iCaRL clearly benefits from using the NME classifier, removing most task-recency bias, although at the cost of having slightly worse performance (43.5) than

the other approaches. EEIL ignores the task-recency bias during training of new tasks, however at the end of each training session it performs balanced training based only on the exemplars. This method obtains good performance (47.6), as balanced training calibrates all outputs from previous classes and thus removes a large part of the task-recency bias. BiC does a very good job at avoiding the bias while maintaining a good performance (45.7). It is clear that the newer tasks have less inter-task classification errors. However, it seems like the small pool of samples used for learning the α and β parameters (see Eq. 4.13) leads to having the opposite effect, and BiC appears to over-compensate toward previous tasks. LUCIR shows a more gradual task-recency bias while maintaining good performance (47.3). This could be related to the change in experimental scenario. LUCIR was shown to work better when having a larger first task followed by some smaller ones. In the more challenging setup used here their bias-correction struggles to obtain good results. Finally, IL2M clearly overcomes task-recency bias while improving on iCaRL (45.6). The class confusion matrix looks similar or better than iCaRL, but the task confusion matrix seems to point towards more inter-task miss-classifications.

These results show that the two methods that have better performance (EEIL, LUCIR) still suffer from task-recency bias, while approaches that have a better solution for it (iCaRL, BiC, IL2M) still have a margin for performance improvement. This leaves room for future work to better combine or create new approaches that can both have better overall performance while simultaneously addressing the bias-correction issue.

4.6.3 On exemplar usage

After establishing that most methods can benefit from exemplars or use them as their main tool to avoid catastrophic forgetting, we study the effects of different characteristics related to them. The number of exemplars to store is limited by the type and amount of memory available, and exemplars are selected at the end of each training session following a sampling strategy.

On memory size: We first analyze how the number of exemplars per class affects performance as we expand the exemplar memory. In Figure 4.6 we compare several rehearsal methods with different numbers of exemplars per class in a growing memory. As expected, in almost all cases performance increases as more exemplars are added. LUCIR and iCaRL always perform equal to or better than FT+ and FZ+. When using few exemplars per class, the weights of the last layer can be modified by large gradients coming from new classes while very little to no variability of gradients comes from previous ones. We found that the freezing of the last layer weights as used in FT+ provides a larger advantage than is obtained with only a few exemplars (see results

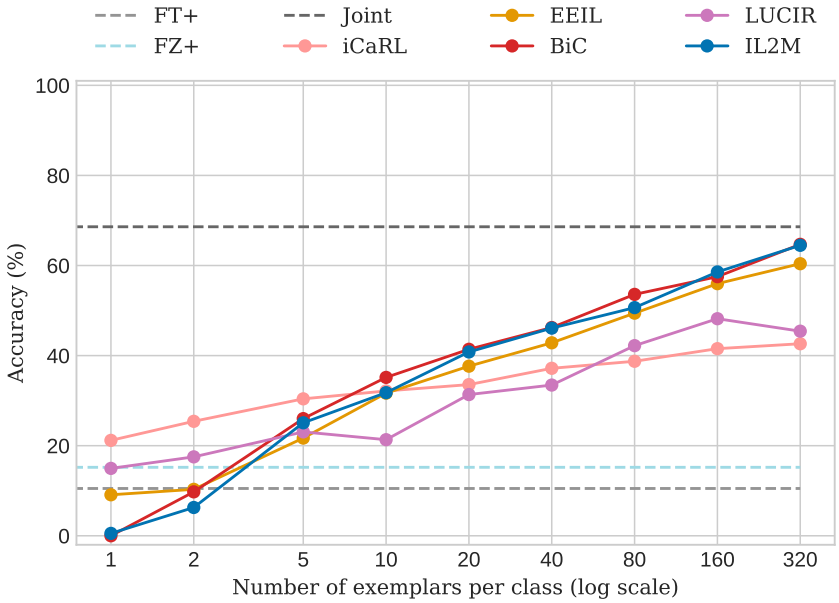

Figure 4.6 – Results for CIFAR-100 (10/10) on ResNet-32 trained from scratch with different exemplar memory sizes.

with fewer than five exemplars for EEIL, BiC, and IL2M).

Adding more samples becomes more costly after 20 exemplars per class in comparison to the gain in performance obtained. As an example, expanding the memory from 10 to 20 samples per class on BiC yields a 6.2 point gain in average accuracy. Expanding from 20 to 40 yields a 4.8 point gain at the cost of doubling the memory size. For the other methods, these gains are similar or worse. Although starting with better performance spot with fewer exemplars per class, iCaRL has a slight slope which makes the cost of expanding the memory less beneficial. LUCIR follows with a similar curve, and both seem to be further away from Joint training (upper bound), probably due to the differences in how the classification layer is defined (NME and cosine normalization, respectively). Finally, BiC, IL2M and EEIL are quite close to Joint training when using a third of the data as memory (160 out of 500 maximum samples per class). To maintain a realistic memory budget, and given the lower performance gains from increasing said memory, we fix growing memories to use 20 exemplars per class.

avg. acc. after	sampling strategy	FT-E	LwF-E	EWC-E	EEIL	BiC
task 2	random	**67.3**	60.8	55.2	62.9	62.2
	herding	59.4	**61.9**	56.8	**67.2**	**62.4**
	entropy	57.9	57.8	56.4	57.7	64.2
	distance	55.5	56.5	51.0	56.0	61.9
	inv-entropy	57.6	57.6	56.4	62.6	61.7
	inv-distance	55.8	54.6	**57.4**	61.7	59.9
task 5	random	**51.3**	**48.6**	39.6	**54.7**	53.4
	herding	49.1	47.8	**41.7**	53.4	**54.9**
	entropy	38.7	36.3	33.3	45.3	43.6
	distance	41.1	38.1	27.7	45.5	42.7
	inv-entropy	40.6	41.0	36.6	47.4	45.8
	inv-distance	38.9	39.5	36.6	45.5	44.4
task 10	random	**37.1**	30.5	26.1	40.8	39.9
	herding	36.5	**30.9**	**26.8**	**42.1**	**42.9**
	entropy	21.8	18.8	14.4	28.7	29.3
	distance	20.4	16.9	10.6	27.4	25.3
	inv-entropy	29.0	25.1	22.9	31.3	34.7
	inv-distance	27.1	24.2	23.1	31.3	35.8

Table 4.4 – CIFAR-100 (10/10) for different sampling strategies with fixed memory of 2,000 exemplars on ResNet-32.

On sampling strategies: As introduced in Sec. 4.4.3, for rehearsal approaches there are different strategies to select which exemplars to keep. In Table 4.4 we compare the FT-E baseline, the two most common regularization-based methods (LwF-E, EWC-E), and two of the latest bias-correction methods (EEIL, BiC). We use the four different sampling strategies introduced in Sec. 4.4.3: random, herding (mean of features), entropy-based, and plane distance-based. We also add a variation of the last two which chooses the samples furthest away from the task boundaries to observe the effect of choosing the least confusing samples instead. We denote these as *inv-entropy* and *inv-distance*. These methods and strategies are evaluated under our two main proposed scenarios: CIFAR-100 (10/10) and (50/11).

Results on the CIFAR-100 (10/10) scenario show a clear preference across all approaches for the herding sampling strategy, except for FT-E which prefers random.

acc. after	sampling strategy	FT-E	LwF-E	EWC-E	EEIL	BiC
task 2	random	42.4	49.0	**47.2**	44.5	55.5
	herding	**48.0**	**51.7**	45.1	**47.9**	53.5
	entropy	39.6	43.6	38.6	38.4	46.1
	distance	36.0	44.0	33.3	37.4	43.6
	inv-entropy	41.4	44.5	45.5	43.3	**55.6**
	inv-distance	44.3	48.2	43.9	40.3	47.9
task 5	random	**38.5**	34.2	30.4	**41.3**	43.2
	herding	36.5	**36.6**	**34.1**	40.8	**44.6**
	entropy	27.3	24.4	20.2	28.2	31.4
	distance	25.1	25.2	20.0	27.6	31.2
	inv-entropy	34.5	32.4	30.0	35.9	41.6
	inv-distance	33.1	32.5	30.0	37.0	38.3
task 10	random	**32.5**	26.0	22.7	37.3	36.1
	herding	32.0	**26.3**	**23.6**	**38.8**	**39.1**
	entropy	16.1	14.8	10.7	23.0	25.9
	distance	17.1	13.5	8.5	23.0	22.7
	inv-entropy	28.7	22.2	21.8	30.1	32.8
	inv-distance	29.2	23.3	20.6	27.1	35.4

Table 4.5 – CIFAR-100 (50/11) with different sampling strategies and fixed memory of 20 exemplars per class on ResNet-32 trained from scratch.

The second best strategy in some cases, and generally close to herding, is random. Both these strategies clearly outperform the others when evaluating after 5 and 10 tasks in both scenarios. When only evaluating after two tasks for the (10/10) scenario, the gap between them is much smaller, probably due to the large number of exemplars available at that point (2,000). It is notable that for shorter task sequences, entropy- and distance-based perform similar to the proposed inverse versions. However, for larger sequences of tasks, the inverse versions perform better. This could be due to samples further away from the boundaries (and closer to the class centers) becoming more relevant when the number of exemplars per class becomes smaller.

The better performance achieved by using herding in comparison to other sampling strategies is also very clear in the CIFAR-100 (50/11) scenario. As seen in Table 4.5, for longer task sequences herding has a clear benefit over the other sampling strategies

when using class-incremental learning methods. In the case of shorter sequences, similar to transfer learning, performance does not seem to specifically favour any sampling strategy.

On different starting scenarios: We explore two scenarios with different numbers of classes in the starting task. The first one compares a large number methods on CIFAR-100 (10/10), with classes equally split across all tasks. For the second scenario, we compare the same methods on CIFAR-100 (50/11) which is similar to having the first task being a pretrained starting point with more classes and a richer feature representation before the subsequent 10 smaller tasks are learned. Both those scenarios are further extended and presented under the two described types of memory: fixed (2,000 total exemplars) and growing (20 exemplars per class), with herding as the sampling strategy. DMC uses an external dataset (reduced ImageNet-32 [188]) that is already larger than the memories, so no exemplars are stored. All other methods which were not originally proposed with exemplars have been adapted to use them and show better performance overall than their original versions.

In Figure 4.7, BiC, EEIL and IL2M achieve the best results after learning 10 tasks with both fixed and growing memories. They are followed by iCaRL, LwM-E, and then LUCIR. LUCIR and BiC have different starting points on task 1 since they do not have the same initial conditions as the other approaches (LUCIR uses cosine linear layers, while BiC uses fewer data during training because it stores some for the later training of the bias-correction parameters). It is quite clear that the approaches that tackle task-recency bias have an overall better performance than the others. Furthermore, as already noted by [13], FT-E achieves competitive performance similar to the lowest performance of that family. In general, most methods seem to suffer less catastrophic forgetting when using a fixed memory that allows storing more exemplars during early tasks. For some approaches, the difference is quite considerable after learning 5 tasks and slightly better after the full 10-task sequence.

In Figure 4.8 the results are quite different. In general, starting from a larger number of classes makes all methods perform better, probably due to anchoring to the first task making the features already more diverse. This is specially noticeable in the case of FZ-E, which benefits significantly from freezing after a much more representative first task. This shows the importance of comparing to this baseline when doing experiments with pretrained models or a very strong first task. In this scenario, LUCIR and iCaRL have a much better performance with a fixed memory, followed by MAS-E, RWalk and BiC.

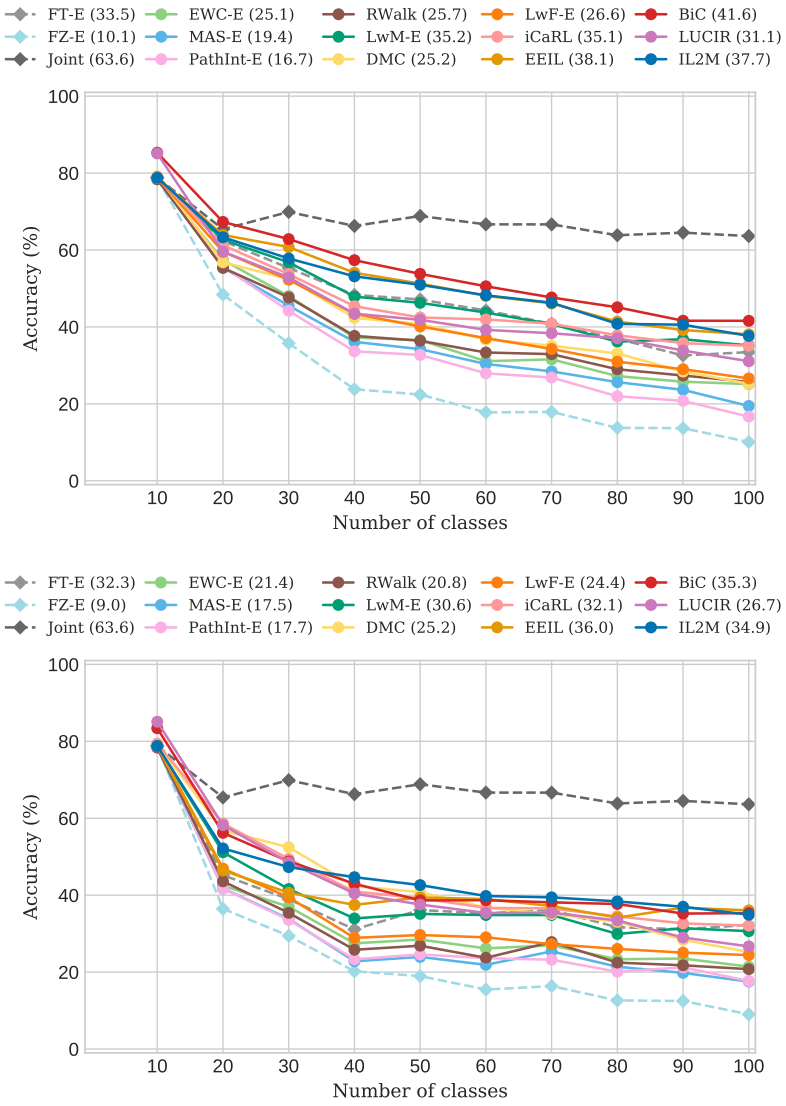

Figure 4.7 – CIFAR-100 (10/10) with 2,000 exemplar fixed memory (top), and 20 exemplars per class growing memory (bottom).

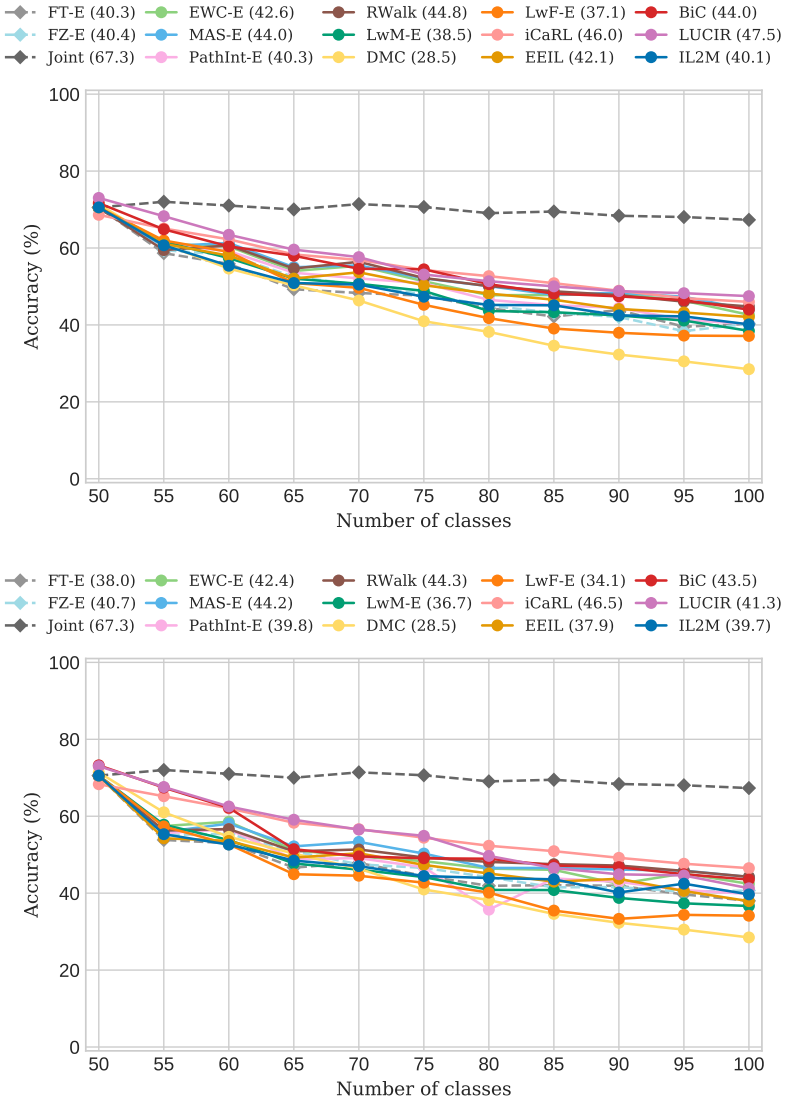

Figure 4.8 – CIFAR-100 (50/11) with 2,000 exemplar fixed memory (top), and 20 exemplars per class growing memory (bottom).

4.6.4 On semantic tasks

The popularity of iCaRL and the interest in comparing with it makes it quite common to utilize the random class ordering for experiments based on CIFAR-100 [77]. The authors of iCaRL use a random order of classes which is fixed in the iCaRL code by setting the random seed to 1993 just before shuffling the classes. However, this gives very little insight on class orderings which make use of the coarse labels from that dataset to group classes into sharing similar semantic concepts. This was explored in our earlier work for the tinyImageNet (Stanford, CS231N [165]) dataset in [34, 107], showing that some methods report different results based on different semantics-based class orderings. In [34], the iNaturalist [168] dataset is split into tasks according to supercategories and are ordered using a relatedness measure. Having tasks with different semantic distributions and learning tasks in different orders is interesting for real-world applications where subsequent tasks are based on correlated data instead of fully random. In another of our recent works, [108] brings attention to the learning variability between using different class orderings when learning a sequence of tasks incrementally.

In joint training, specific features in the network can be learned that focus on differentiating two classes that are otherwise easily confused. However, in an IL setting those discriminative features become more difficult to learn or can be modified afterwards, especially when the classes belong to different tasks. Thus, the difficulty of the task can be perceived differently in each scenario. Depending on the method, this issue may be handled differently and therefore lead to more catastrophic forgetting. This setting is different from the one proposed in Curriculum Learning [15], since the objective here is not to find the best order to learn tasks efficiently, but rather to analyze incremental learning settings (in which the order is not known in advance) and analyze the robustness of methods under different task orderings.

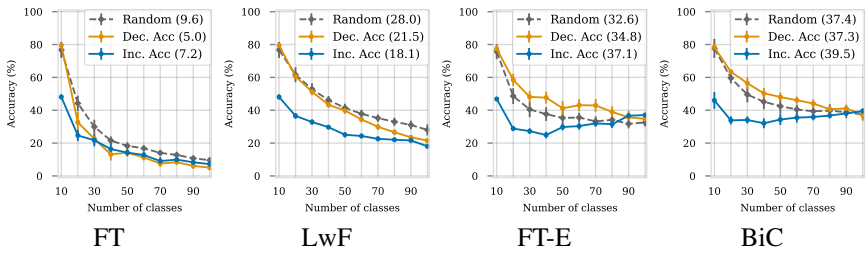

Figure 4.9 – Class ordering results for CIFAR-100 on ResNet-32 trained from scratch. For FT-E and BiC, 20 exemplars per class are sampled using herding. Error bars indicate standard deviation over six runs.

In order to investigate robustness to class orderings, we use the 20 coarse-grained labels provided in the CIFAR-100 dataset to arrive at semantically similar groups of classes. Then, we order these groups based on their classification difficulty. To assess performance we trained a dedicated model with all CIFAR-100 data in a single training session and use this model accuracy as a proxy value for classification difficulty. Finally, we order them from easier to harder (Dec. Acc.) and the other way around (Inc. Acc.). Results are presented in Fig. 4.9 for two methods without exemplars (FT+, LwF), and two methods with exemplars (FT-E, BiC). Performance can be significantly lower when using a semantics-based ordering compared to random one. In the examplar-free cases, special care of the used task ordering should be taken as the final performance after learning all classes can have quite some variability as seen in the LwF case. However, the variation with respect to the orderings is mitigated by the use of exemplars. Therefore, evaluating methods which use exemplars with randomized task orderings often suffices.

4.6.5 On domain shift effects

Up to this point all experiments were performed on a dataset with a small input size and a wide variety of classes from a similar distribution. In this experiment, we study the effects of using tasks which have different degrees of domain shifts between them and whose images also have higher resolution.

Smaller domain shift: We first conduct experiments on very small domain shifts between different classes and tasks, as is the case for VGGFace2 [21]. We divide the 1,000 classes equally into 25 tasks of 40 classes, store 5,000 exemplars in a fixed memory and train ResNet-18 from scratch. In Fig. 4.10 we see that LUCIR, BiC and IL2M perform the best among all methods. In particular, LUCIR achieves 73.0 average accuracy after 25 tasks, which is relatively high compared to previous experiments on CIFAR-100, which indicates that this approach might be more indicated for smaller domain shifts. Surprisingly, FT-E performs only 4.2 points lower than LUCIR and above all the other remaining approaches except for EWC-E, which also performs well with small domain shifts between tasks. EEIL shows competitive performance on the first 13 tasks, but starts to decline for the remaining ones. On the other hand, iCaRL has a larger drop in performance during early tasks, maintains the performance quite well afterwards, and ends up with similar results as EEIL and LwM-E. Of the regularization-based methods, EWC-E is superior to both LwF-E and LwM-E. FZ+ has better performance when starting from a larger first task (due to more robust feature representations), which we assumed would translate into a good performance when having small domain shifts between tasks and classes. However, the initial frozen representations are not discriminative enough to generalize to new classes.

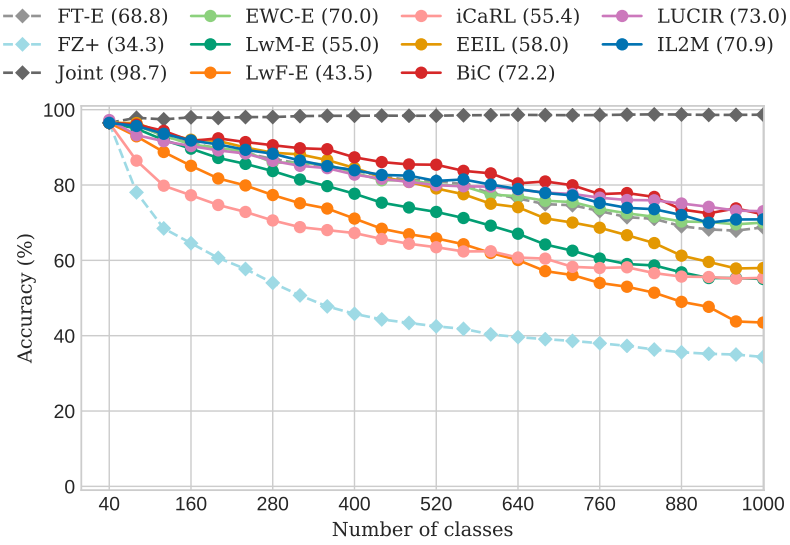

Figure 4.10 – Small domain shifts on VGGFace2 (40/25) on ResNet-18 trained from scratch and 5,000 exemplar fixed memory.

Larger domain shift: We are the first to compare class-IL methods to incrementally learn classes from various datasets. As a consequence tasks have large domain shifts and different number of classes. We use six fine-grained datasets (Flowers, Scenes, Birds, Cars, Aircraft and Actions) learned sequentially on ResNet-18 from scratch with a growing memory of 5 exemplars per class. The number of classes varies among the tasks (see Table 4.1), but the classes inside each of them are closely related. In Fig. 4.11 we see that most approaches have a similar performance, much unlike previous experiments. It is noticeable that bias-correction methods do not have a clear advantage compared to other approaches. It seems that when the domain shift between tasks is large, inter-task confusion becomes the major cause for catastrophic forgetting. Solving the task-recency bias provides a lower performance advantage than in other scenarios and only improves the outputs of the corresponding task. The forgetting which is caused by the large weight and activation drift originated from the large domain shifts seems to dominate in this scenario. The fact that no method clearly outperforms the FT-E baseline shows that scenarios with large domain shifts, where catastrophic forgetting is caused by inter-task confusion, are still an important direction of study since most proposed methods focus on weight drift

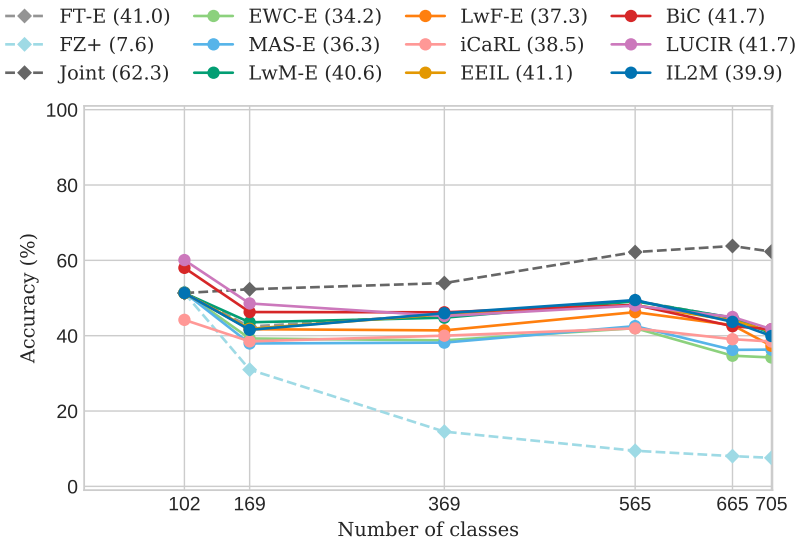

Figure 4.11 – Large domain shifts with multiple fine-grained datasets (Flowers, Scenes, Birds, Cars, Aircraft, Actions).

(EWC-E, MAS-E), activation drift (LwF-E, LwM-E, iCaRL, EEIL, BiC, LUCIR) or task-recency bias (iCaRL, BiC, LUCIR, IL2M).

Another interesting effect we visualize in Fig. 4.11 is the behaviour when learning Actions. The other datasets have a very clear focus on color and shape features to discriminate between their classes. However, for Actions, context is very relevant to identify which action the human is portraying in the image. Some features from the Scenes dataset can be helpful to identify an indoor or outdoor action, but in general this dataset is less related to the others. And we see that already Joint training lowers a bit the average accuracy when learning this task, as do most of the methods. Only EWC-E and MAS-E maintain or improve when learning that task, raising the question whether weight regularization-based methods have an advantage in these scenarios. However, it is also easier to maintain the accuracy if it was lower to start with.

On "interspersed" domains: We propose another scenario that is not explored in class-IL: revisiting learned distributions to learn new classes. We propose to learn four fine-grained datasets split into four tasks of ten classes each for a total of 16 tasks. A group consists of four tasks, one from each dataset in this order: Flowers, Birds, Actions, Aircraft. The experiment consists of four group repetitions, where

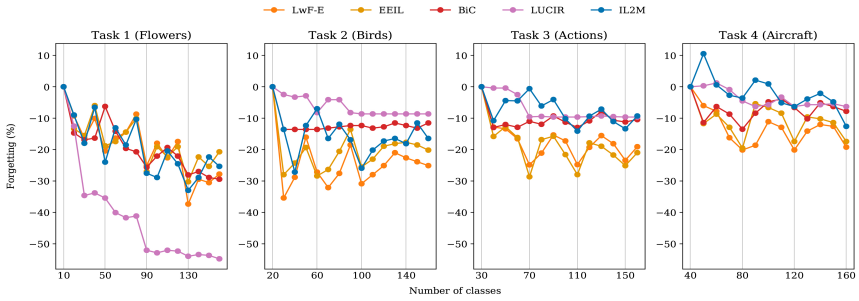

Figure 4.12 – Forgetting when revisiting old domains with new classes from different fine-grained datasets on AlexNet.

each group contains different classes (for a total of 160). This allows us to analyze how class-IL methods perform when similar tasks re-appear after learning different tasks. We refer to this scenario as "interspersed" domains since classes from each domain are distributed across tasks.

Results of forgetting on the first group during the whole sequence are presented in Fig. 4.12. We clearly see a difference between LUCIR and other methods. LUCIR suffers quite a large loss on the first task at the beginning of the sequence and after the second group is learned, never recovering any performance for that task. However, LUCIR shows very little forgetting for the remaining tasks in the sequence. This seems to be related to the preference of LUCIR to have a larger first task with more diverse feature representations, as also observed in earlier experiments. For the remaining methods, the first task has a lot of variation with a general decaying trend. BiC has an initial drop right after learning each of the other tasks, but manages to prevent further forgetting, though with some variability on the first Aircraft task. LwF-E and EEIL have a more cyclic pattern of forgetting and recovering. Forgetting is more pronounced when the task being learned is of the same dataset as the current one, and seems to slightly recover when learning less similar tasks. Finally, the forgetting of IL2M shows a lot of variation, which might be related to the lack of a distillation loss keeping new representations closer to previous ones.

4.6.6 On network architectures

We compare the four most competitive methods over a range of different network architectures in Table 4.6. An interesting observation is that for different networks, the performance rankings of the methods can change completely. For instance, in

architectures which do not use skip connections, iCaRL performs the best when using AlexNet and VGG-11. On the other hand, BiC performs worse without skip connections, but performs the best with architectures that have them (ResNet-18, MobileNet and GoogleNet). BiC exhibits the least forgetting among all methods, even having positive forgetting which indicates that performance improves on some tasks after learning them. However, this result comes at the expense of having slightly lower performance for each task just after learning them. IL2M is more consistent compared to other methods using different networks, never having the best nor the worst performance. Networks without skip connections seem to reduce forgetting for iCaRL and IL2M. EEIL suffers more forgetting compared to other methods across different networks.

ResNet-18 obtains the best result among all networks with BiC. Note that in most of the literature, ResNet-18 is used as the default network for this scenario and similar ones. However, as shown above, it seems that methods benefit from architectures differently. Another interesting observation is that MobileNet, which has the lowest number of operations and can run on devices with limited capacity, has very competitive results compared to the other networks. These results show that existing IL approaches can be applied to different architectures with comparable results to the scenarios presented in the literature.

4.6.7 On large-scale scenarios

Finally, we compare different methods using ResNet-18 on ImageNet (40/25) with a growing memory of 20 exemplars per class. Figure 4.13 shows that BiC and iCaRL achieve the best performance with 32.4% and 30.2% average accuracy after 25 tasks, respectively. Surprisingly, EWC-E and FT-E outperform some methods, such as EEIL and LUCIR, in this setting. Note that in other settings, EEIL and LUCIR often perform better than EWC-E and FT-E. LwF-E and LwM-E obtain worst results compared to how they previously performed. We note that BiC, iCaRL, IL2M and LUCIR avoid a larger initial drop in performance during the first four tasks compared to other methods and continue learning without major drops in performance except for LUCIR. Of the rest of the methods, EWC-E, FT-E and EEIL seem to stabilize after the initial drop and show less forgetting as new tasks are added. RWalk, LwF-E and LwM-E continue having a larger drop in performance after task four, which only RWalk slightly recovers from. In this scenarios with a larger number of classes and more variability, methods which can easily handle early tasks will perform better afterwards. On the second half of the sequence, most approaches have the same stable behaviour since the network has learned a robust representation from the initial tasks.

		task 2	task 5	task 9	A_{10}
AlexNet 60m params 2012	iCaRL	39.6 (-23.2)	30.0 (-8.4)	33.0 (-5.2)	**38.8**
	EEIL	27.4 (-55.0)	25.2 (-49.0)	22.6 (-49.4)	35.6
	BiC	30.6 (-31.8)	26.4 (+14.0)	21.2 (+16.8)	34.4
	IL2M	27.4 (-52.4)	21.6 (-41.2)	44.0 (-25.2)	35.2
VGG-11 133m params 2014	iCaRL	32.4 (-30.0)	34.0 (-24.8)	42.6 (-8.2)	**43.2**
	EEIL	29.6 (-56.0)	29.0 (-50.4)	32.8 (-45.6)	40.9
	BiC	32.4 (-33.8)	19.6 (+3.4)	31.0 (-3.2)	32.1
	IL2M	27.8 (-58.2)	31.0 (-19.6)	54.0 (-17.4)	42.2
GoogleNet 6.8m params 2014	iCaRL	35.0 (-30.0)	29.2 (-24.0)	43.6 (-12.2)	43.7
	EEIL	18.2 (-68.4)	26.0 (-49.2)	31.8 (-45.0)	36.1
	BiC	27.2 (-51.2)	39.8 (-14.2)	49.0 (-4.4)	**44.5**
	IL2M	23.6 (-59.0)	23.0 (-36.6)	40.0 (-36.0)	38.2
ResNet-18 11m params 2015	iCaRL	38.4 (-31.8)	29.6 (-21.8)	43.8 (-9.8)	43.6
	EEIL	26.0 (-59.4)	26.8 (-52.8)	28.2 (-48.8)	36.6
	BiC	31.2 (-48.6)	41.0 (+0.4)	49.4 (+4.4)	**45.6**
	IL2M	26.2 (-60.8)	24.0 (-47.8)	35.0 (-44.4)	37.2
MobileNet 4.2m params 2017	iCaRL	38.4 (-33.4)	33.6 (-21.6)	40.2 (-23.8)	43.5
	EEIL	21.2 (-68.4)	29.0 (-52.4)	25.4 (-54.8)	37.4
	BiC	39.4 (-44.2)	41.2 (-11.4)	45.2 (-14.0)	**44.7**
	IL2M	35.0 (-46.6)	24.2 (-24.2)	42.6 (-30.0)	42.1

Table 4.6 – ImageNet-Subset-100 (10/10) with different networks trained from scratch. Task accuracy when the task was learned and forgetting after learning all classes (between brackets). Final column reports the average accuracy after 10 tasks.

4.7 Emerging trends in class-IL learning

Here we briefly discuss some recent developments in class-IL we think will play an important role in the coming years.

Exemplar learning. Recently, an exciting new direction has emerged that parametrizes exemplars and optimizes them to prevent forgetting [25, 93]. This enables much more efficient use of available storage. Liu et al. [93] propose Mnemonics Training, a method that trains the parametrized exemplars in a two-phase procedure. During the first phase, the model is optimized on the new data and exemplars of previous tasks while preventing forgetting with a distillation loss. In the second phase, the exemplars

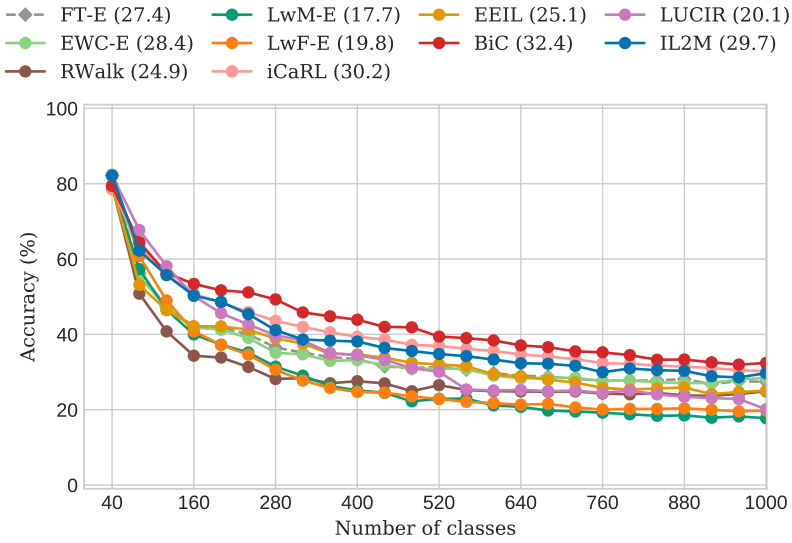

Figure 4.13 – ImageNet (40/25) on ResNet-18 with growing memory of 20 exemplars per class and herding sampling.

are optimized to prevent the forgetting when evaluated on the current task data (i.e. when finetuning on these exemplars the increase of the cross-entropy loss on the current data is minimal). The proposed method is combined with weight transfer [126] to reduce the number of parameters that are learned for each task. Chaudry et al. [25] generalize the theory to a streaming setting, where the learning of the exemplars does not require multiple loops over the data for every task. They propose learning a single anchor exemplar that prevents forgetting on future tasks. Because there is no access to future data, preventing forgetting on future tasks is approximated by preventing it on previous tasks (represented by exemplars).

Both methods [25,93] show that they can outperform random and herding strategies. Optimizing the available storage by computing more efficient exemplars is expected to be one of the main future research directions, and more efficient use of limited storage in IL systems in general is expected to attract more research in the coming years.

Feature rehearsal. Pseudo-rehearsal is a good alternative to storing exemplars [122, 154, 175]. It learns a separate network that generates images of previous tasks. However, current state-of-the-art image generation methods still struggle to realistically generate complex image data, and therefore this approach has been mainly applied

to relatively simple datasets and is known to obtain unsatisfying results on complex ones. To address this problem, some recent works have proposed to perform *feature* replay instead of image replay [68, 92, 177], where instead a generator is trained to generate features at some hidden layer of the network. In this way rehearsal can also be applied to complex datasets. Another closely related line of research is based on the observation that storing feature exemplars is much more compact than storing images [58]. Moving away from image replay towards different variants of feature replay is expected to gain traction.

Explicit task classification. The majority of class-IL methods incrementally learn a classifier over all classes up until those in the current task. Another approach is to learn one classifier head per task that only distinguishes between the classes within the task, and another classifier that predicts the task label. This would allow to extend any task-IL method to a class-IL method. An early version of this idea was proposed by Aljundi et al. [6] where gating autoencoders are used to predict the task label. A recent work extends a mask-based method (similar to [153]) with an explicit task classifier [2]. It is yet unclear why and when explicit task classification is expected to outperform learning one joined classifier. Further study and comparison between these two fundamentally different strategies to the problem of class-IL is needed. A recent work of Rajasegaran et al. [130] convincingly shows that for the plausible scenario where images at inference time are processed in batches with the same task-ID, this knowledge can significantly improve the quality of the explicit task classifier.

Self- and unsupervised incremental learning. Being able to incrementally learn representations from an unsupervised data stream is a desirable feature in any learning system. This direction applied to class-IL has received relatively little attention to date. Rao et al. [132] propose a method that performs explicit task classification and fits a mixture of Gaussians on the learned representations. They also explore scenarios with smooth transitions from one task to another. Still in its infancy, more research on unsupervised incremental learning is expected in coming years. In addition, leveraging the power of self-supervised representation learning [71] is only little explored within the context of IL, and is expected to gain interest.

Meta-learning. Meta-learning aims to learn new tasks leveraging information accrued while solving related tasks [148]. Riemer et al. [139] show that such a method can learn parameters that reduce interference of future gradients and improves transfer based on future gradients. Javed and White [70] explicitly learn a representation for continual learning that avoids interference and promotes future learning. These initial works have shown the potential of meta-learning on small datasets. However, we expect these techniques to be further developed in the coming years, and will start to obtain results on more complex datasets like the ones considered in our evaluation.

4.8 Conclusions

We performed an extensive survey of class-incremental learning. We organized the proposed approaches along three main lines: regularization, rehearsal, and bias-correction. In addition, we provided extensive experiments in which we compare twelve methods on a wide range of incremental learning scenarios. Here we briefly enumerate the main conclusions from these experiments:

- When comparing exemplar-free methods, LwF obtains the best results (see Table 4.3). Among the other regularization methods, data regularization (LwM) obtains superior results compared to weight regularization (EWC and MAS). Exemplar-free methods can currently not compete with exemplar rehearsal methods, and given the more restrictive setting in which they operate, we advocate comparing them separately.
- When combining LwF with exemplars, we confirm the results in [13] showing that the added regularization does not improve results and the baseline method of finetuning with exemplars performs better (see Table 4.3). However, using LwM for data regularization does perform better than the baseline.
- We found that in several scenarios weight regularization outperforms the baseline FT-E significantly (see Figs. 4.8 and 4.10), showing that the IL community choice of data regularization with LwF (see Fig. 4.4) instead of weight regularization should be reconsidered.
- Herding is a more robust exemplar sampling method than random for larger sequences of tasks, but is not better than others for short sequences (see Table 4.4).
- Methods that explicitly address the task-recency bias obtain better performance for class-IL (see Figs. 4.7, 4.8, 4.10, 4.11, 4.13): we found that BiC obtains state-of-the-art on several experiments (notably on ImageNet). IL2M obtains consistent good performance on most datasets. Also, iCaRL and EEIL obtain good performance on several datasets, but fail to outperform the baseline FT-E on others. Methods like LUCIR require a good starting representation – for example in the scenario with the larger first task or smaller domain shifts, LUCIR can be state-of-the-art.
- Current methods have mainly presented results on datasets with small domain shifts (typically random class orderings from a single dataset). When considering large domain shifts none of the methods significantly outperform the baseline FT-E (see Fig. 4.11). Large domain shift scenarios have been considered for task-IL, but our results show that they require new techniques to obtain satisfactory results in class-IL settings.

- We are the first to compare class-IL methods on a wide range of network architectures, showing that current class-IL works on variety of networks. Results show that most are sensitive to architecture and rankings change depending on the network used. It is quite clear that using a network with skip connections favors some methods, while their absence favors others.

5 Metric learning for novelty and anomaly detection*

5.1 Introduction

Deep neural networks have obtained excellent performance for many applications. However, one of their known shortcomings is that they can be overly confident when presented with images (and classes) not present in the training set. Therefore, a desirable property of these systems is the capacity to not produce an answer if an input sample belongs to an unknown class, that is, a class for which it has not been trained. The field of research dedicated to this goal is called out-of-distribution detection [61,82,88]. Performing out-of-distribution detection is important not only to avoid classification errors but also as the first step towards lifelong learning systems [30]. Such systems would detect out-of-distribution samples in order to later update the model accordingly [74,90].

The problem of out-of-distribution detection has also been called one-class classification, novelty and anomaly detection [128]. More recently, associated to deep neural network classifiers, some works refer to it as open-set recognition [14]. In this chapter, we distinguish two cases of out-of-distribution which we believe are quite different: we propose to term as *novelty* an image from a class different from those contained in a dataset used for training, but that bears some resemblance to them (for instance because it shows the same kind of object from untrained points of view). This is a very important problem in many computer vision applications. For example, imagine a system that classifies traffic signs on-board a car and takes automatic decisions accordingly. It can happen that it finds a class of local traffic signs which was not included in the training set, and this must be detected to avoid taking wrong decisions. We reserve the word *anomaly* for completely unrelated samples, like different type of objects, images from another unrelated dataset, or background patches in the case of traffic sign classification. This is also relevant from the point of view of commercial applications. In fact, most previous works focus on anomaly detection. Novelty detection remains rather unexplored. To the best of our knowledge only [150] and [88] perform some intra-dataset out-of-distribution detection experiments. The three previous works closest to ours [61, 82, 88], revolve around one idea: given a discriminative neural network model, use the output probabilities to take the decision

*This chapter is based on a publication in the British Machine Vision Conference (BMVC), 2018 [106].

of seen/unseen class. These networks are optimized to distinguish between the classes present in the training set, and are not required to explicitly model the marginal data distribution. As a consequence, at testing time the system cannot assess the probability of the presented data, complicating the assessment of novel cases.

Here we explore a completely different approach: to learn an embedding where one can use Euclidean distance as a measure of "out-of-distributioness". We propose a loss that learns an embedding where samples from the same in-distribution class form clusters well separated from the space of other in-distribution classes *and* also from out-of-distribution samples. The contributions to the problem of out-of-distribution detection presented in this chapter are the following. First, the use of metric learning for out-of-distribution detection, instead of doing it on the basis of the cross-entropy loss and corresponding softmax scores. Second, we distinguish between novelty and anomaly detection and argue that research should focus on the more challenging problem of novelty detection. Third, we obtain comparable or better results than state-of-the-art in both anomaly and novelty detection. Finally, in addition to the experiments with benchmark datasets in order to compare with previous works, we also address a real-world classification problem, traffic sign recognition, for which we obtain good detection *and* accuracy results.

5.2 Related work

This chapter is related to anomaly detection in its different meanings. Also to open-set recognition, as one of the most important applications of out-of-distribution detection. And finally to metric learning, the basis of our approach. In the following we briefly review the most related works in each of these areas. Out-of-distribution detection should not be confused with another desirable property of machine learning systems, namely the reject option, which is, the ability to decide to not classify an input if the confidence on any of the labels is too low (see for example [45] and references therein). The difference is that in the latter case it is assumed that the sample does belong to some class present during training.

Anomaly and novelty detection. Also known as out-of-distribution detection, it aims at identifying inputs that are completely different from or not present in the original data distribution used for training [128]. In [18], novelty detection is performed by learning a distance in an embedding. It proposes a Kernel Null Foley-Sammon transform that aims at projecting all the samples of each in-distribution class into a single point in a certain space. Consequently, novelty detection can be performed by thresholding the distance of a test sample to the nearest of the collapsed class representations. However, they employ handcrafted features, thus optimizing only the transform parameters and not the representation, as in the currently prevailing

paradigm of deep learning.

Although Deep Neural Networks (DNNs) have been established as state-of-the-art on many computer vision classification and detection tasks, overconfidence in the probability score of such networks is a common problem. DNNs capable of detecting many objects with high accuracy can still be fooled by predicting new, never-seen objects with high confidence. This problem can be defined by the ability of the network to decide if a new test sample is in-distribution (i.e. from a class or from the data used to train the classifier) or is instead out-of-distribution.

In [61] the authors showed that DNNs trained on MNIST [80] images can frequently produce high confidence guesses (+90%) on random noise images. They propose a baseline for evaluation of out-of-distribution detection methods and show that there is room for future research to improve that baseline. Their baseline assumes that out-of-distribution samples will have a more distributed confidence among the different classes than an in-distribution sample. Recently, in [88] the authors propose ODIN, a simple method applied to DNNs that uses a softmax layer for classification and does not need the network to be retrained. The key idea is to use temperature scaling and input pre-processing, which consists on introducing small perturbations in the direction of the gradients for the input images.

The authors of [82] diverged from the other threshold-based methods by proposing a new training method. They add two loss terms that force the out-of-distribution samples to be less confident and improve the in-distribution samples respectively. In both these works, trained DNNs follow a typical softmax cross-entropy classification loss, where each dimension on the output embedding is assigned to measure the correlation with a specific class from that task. Other than previous work which focuses on networks trained with the cross-entropy, our work studies out-of-distribution for networks which are optimized for metric learning. These networks do not have the normalization problem which is introduced by the softmax layer, and are therefore expected to provide better estimates of out-of-distribution data. One final work is worth mentioning in the context of DNNs. In [150] the authors propose to discern between seen and unseen classes through the dimensions of certain layer activations which have extreme values. They achieve good accuracy on ImageNet but only when the number of selected classes is very small.

Open Set Recognition. This shares with out-of-distribution detection the goal of discriminating samples from two different distributions. But it places the emphasis on how to apply it to improve classifier capabilities so that it can still perform well when the input may contain samples not belonging to any class in the training set. One of the first works is [147], which formalized the problem as one of (open) risk minimization in the context of large margin classifiers, producing what they called a one-versus-set Support Vector Machine. More recently, a method to adapt deep neural networks to

handle open set recognition has been proposed in [14]. The key idea is to replace the conventional softmax layer in a network by a so called openmax layer. It takes the N activations (where N is the number of classes) of the penultimate layer of the network and estimates the probability for each training class, like in softmax, plus that of not being a sample of the training data. This later is done by fitting a Weilbull density function to the distance between the mean activation value for each class and those of the training samples. We see thus that distance between last layer activations or features plays a key role. This is coincident with our method, only that features in their case are learned through a loss function similar to cross-entropy whereas we explicitly will learn a distance such that in-distribution samples cluster around one center per class and out-of-distribution samples are pushed away from all these centers.

Metric Learning. Several computer vision tasks such as retrieval, matching, verification, even multi-class classification, share the need of being able to measure the similarity between pairs of images. Deriving such a measure from data samples is known as metric learning [79]. Two often cited seminal works on this subject through neural networks are [31, 53], where the Siamese architecture was proposed for this purpose. Differently from classification networks, the goal is to measure how similar two instances are in terms of the Euclidean distance, rather than learn a representation amenable for classification. Another popular architecture is the triplet network [64]. For both of them many authors have realized that mining the samples of the training set in order to find *difficult* or challenging pairs or triplets is important in order to converge faster or to better minima [149, 159, 160]. Like them, we have also resorted to a mining strategy in order to obtain good results on the task of out-of-distribution detection.

5.3 Metric learning for out-of-distribution detection

Most recent works on out-of-distribution detection are based on supervised training of neural networks, which optimizes the cross-entropy loss. In these cases, the network output has a direct correspondence with the solution of the task, namely a probability for each class. However, the representation of the output vector is forced to always sum to one. This means that when the network is shown an input which is not part of the training distribution, it will still give probabilities to the nearest classes so that they sum up to one. This phenomena has led to the known problem of neural networks being too overconfident about content that they have never seen [61].

Several works have focused on improving the accuracy of the confidence estimate of methods based on the cross-entropy, adapting them in such a way that they yield lower confidences for out-of-distribution samples [61, 82, 88]. We hypothesize that the problem of the overconfident network predictions is inherent to the use of the

cross-entropy loss, and therefore propose to study another class of network objectives, namely those used for metric learning. In metric learning methods, we minimize an objective which encourages images with the same label to be close and images with different labels to be at least some margin apart in an embedding space. These networks do not apply a softmax layer, and therefore are not forced to divide images which are out-of-distribution from the known classes.

5.3.1 Metric learning

For applications such as image retrieval, images are represented by an embedding in some feature space. Images can be ordered (or classified) according to the distance to other images in that embedding space. It has been shown that using metric learning methods to improve the embeddings can significantly improve their performance [52]. The theory of metric learning was extended to deep neural networks by Chopra et al. [31]. They proposed to pass images through two parallel network branches which share the weights (also called a Siamese network). A loss considers both embeddings, and adapts the embedding in such a way that similar classes are close and dissimilar classes are far in that embedding space.

Traditionally these networks have been trained with contrastive loss [53], which is formulated as:

$$L(x_1, x_2, y; W) = \frac{1}{2}\left(1-y\right) D_w^2 + \frac{1}{2} y \left(\max\left(0, m - D_w\right)\right)^2, \tag{5.1}$$

where $D_w = ||f_W(x_1) - f_W(x_2)||_2$ is the distance between the embeddings of images x_1 and x_2 computed by network f_W with weights W. The label $y = 0$ indicates that the two images are from the same class, and $y = 1$ is used for images from different classes. The loss therefore minimizes the distance between images of the same class, and increases the distance of images of different classes until this distance surpasses the margin m (see Fig. 5.1). Several other losses have been proposed for Siamese networks [64, 149, 160, 171, 172] but in this chapter we will evaluate results with the contrastive loss to provide a simple baseline on which to improve.

5.3.2 Out-of-Distribution Mining (ODM)

In the previous section, we considered that during training only examples of in-distribution data are provided. However, some methods consider the availability of some out-of-distribution data during training [82]. This is often a realistic assumption since it is relatively easy to obtain data from other datasets or create out-of-distribution examples, such as samples generated with Gaussian noise. However, it has to be noted that the out-of-distribution data is used unlabeled, and is of a different distribution

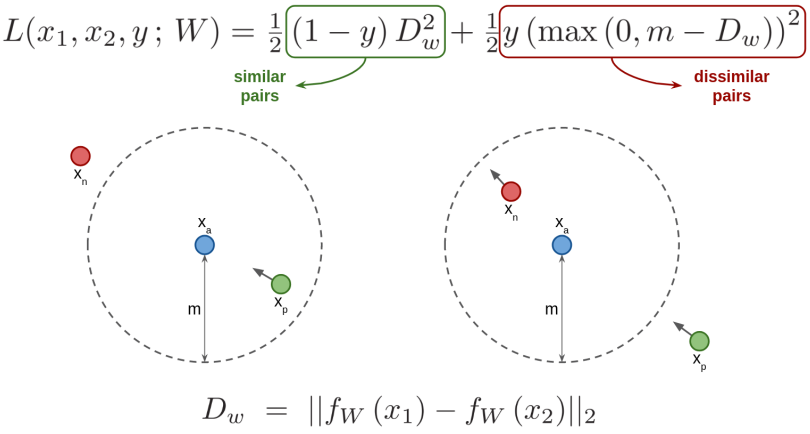

Figure 5.1 – Visualization of the contrastive loss [53]. Pairs are formed by x_a and x_p if the pairs are of the same class ($y = 0$), and x_a and x_n if the pairs are from different classes.

from the out-of-distribution used at testing. The objective is to help the network be less confident about what it does not know. Therefore, noise or even unlabeled data can be used to strengthen the knowledge boundaries of the network.

We propose to adapt the contrastive loss to incorporate the out-of-distribution data:

$$L(x_1, x_2, y; W) = \frac{1}{2}(1-y)\,z D_w^2 + \frac{1}{2}yz\,(\max(0, m - D_w))^2, \tag{5.2}$$

where we have introduced a label z which is zero when both images are out-of-distribution and one otherwise. This loss is similar to Eq. 5.1, but with the difference that in case of a pair of images where one is an out-of-distribution image ($z = 1$, $y = 1$) they are encouraged to be at least m distance apart. Note that we do not enforce the out-of-distribution images to be close, since when $z = 0$ the pair does not contribute to the loss. It is important to make sure that there are no pairs of out-of-distribution samples so that they are not treated as a single new class and forced to be grouped into a single cluster. A summary of the mining strategy is shown in Fig. 5.2.

In practice, we have not implemented a two-branches Siamese network but followed recent works [91, 166] which devise a more efficient approach to minimize losses traditionally computed with Siamese networks. The idea is to sample a mini-batch of images which we forward through a single branch until the embedding layer.

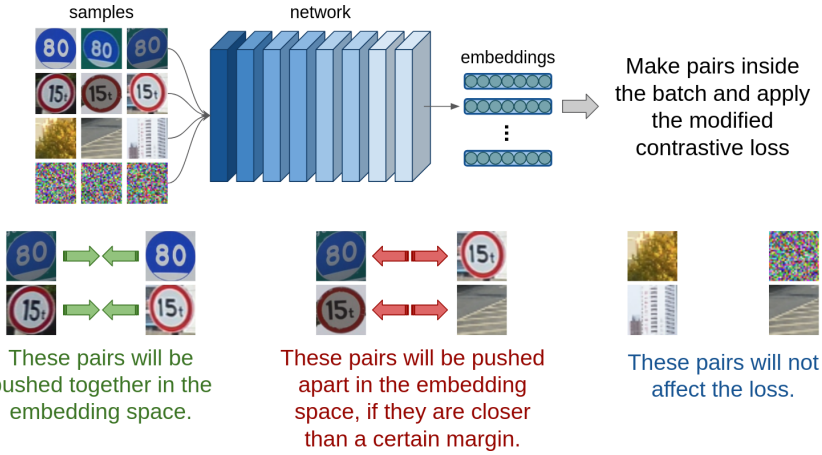

Figure 5.2 – Out-of-Distribution Mining pipeline. Pairs containing at least an in-distribution are learned following the contrastive loss. However, pairs consisting of two out-distribution samples do not contribute to the loss since we do not want them to form clusters in the output space.

We then sample pairs from them in the loss layer and backpropagate the gradient. This allows the network to be defined with only one copy of the weights instead of having two branches with shared weights. At the same time, computing the pairs after the embedding also allows the use of any subgroup of possible pairs among all the images from the minibatch. When computing the pairs we make sure that pairs of out-of-distribution samples are not used. As a result z will never be 0 and we can in practice directly apply Eq. 5.1 instead of Eq. 5.2.

5.3.3 Anomaly and novelty detection

In this chapter we distinguish between two categories of out-of-distribution data:

> **Novelty:** samples that share some common space with the trained distribution, which are usually concepts or classes which the network could include when expanding its knowledge. If you train a network specialized in different dog breeds, an example would be a new dog breed that was not in the training set. Furthermore, if the classes are more complex, some novelty out-of-distribution could be new viewpoints or modifications of an existing learned class.

> **Anomaly:** samples that are not related to the training distribution. In this category we could include background images, Gaussian noise, or classes unrelated to the training distribution (i.e. SVHN would be a meaningful anomaly for CIFAR-10). Since anomalies are further from the in-distribution than novelties, these are expected to be easier to detect.

To further illustrate the difference between novelties and anomalies consider the following experiment. We train a LeNet network on the classes 2, 6 and 7 from the MNIST dataset [80] under the same setup for both cross-entropy (CE) and contrastive (ML) losses. We also train it with our proposed method which introduces out-of-distribution mining during training (ODM). We use classes 0, 3, 4, and 8 as those seen out-of-distribution samples during training. Then, we visualize the embeddings for different out-of-distribution cases from closer to further resemblance to the train set : 1) similar numbers 5, 9 and 1 as novelty, 2) SVHN [117] and CIFAR-10 [77] as anomalies with a meaning, and 3) the simpler Gaussian noise anomalies.

In Figure 5.3 we show the 3-dimensional output embedding spaces for CE, ML and ODM in rows 1, 2 and 3 respectively. As expected, the CE space is bounded inside the shown triangle, since the three dimensions of the output (the number of classes) have to always sum up to 1. For SVHN, CE correctly assigns low confidence for all classes. However, for CIFAR-10, Gaussian noise and Novelty it increasingly is more confident about the probability of an out-of-distribution image to be classified as an in-distribution one. In the case of ML, all anomalies seem to be more separated from the in-distributions for each class, and only the Novelty is still too close to the cluster centers. With the introduction of out-of-distribution samples during training, ODM shows how out-of-distribution images are kept away from the in-distribution, allowing the network to be confident about what it is capable of classifying and what not.

We also provide some quantitative performance results of the comparison on the MNIST dataset. In this case we applied a 5-dimensional embedding space for ML so the representation is rich enough to make the discrimination between in-dist and out-dist. For CE, as it is fixed to the number of classes, the embedding space is 3-dimensional. In Table 5.1 we see that ML performs better than CE in all cases. ODM almost solves the novelty problem while keeping a similar performance on anomalies as ML. It is noticeable that CE struggles a bit more with Gaussian noise than the other anomalies. In this case, CE still produces highly confident predictions for some of the noise images.

In conclusion, this experiment shows that there is a difference between novel and anomaly out-of-distribution samples for both cross-entropy and metric learning approaches, stressing that these must be approached differently. Furthermore, the overconfidence of the cross-entropy methods is more clear on novelty detection cases, and among the anomaly cases, the Gaussian noise seems to be the one with more

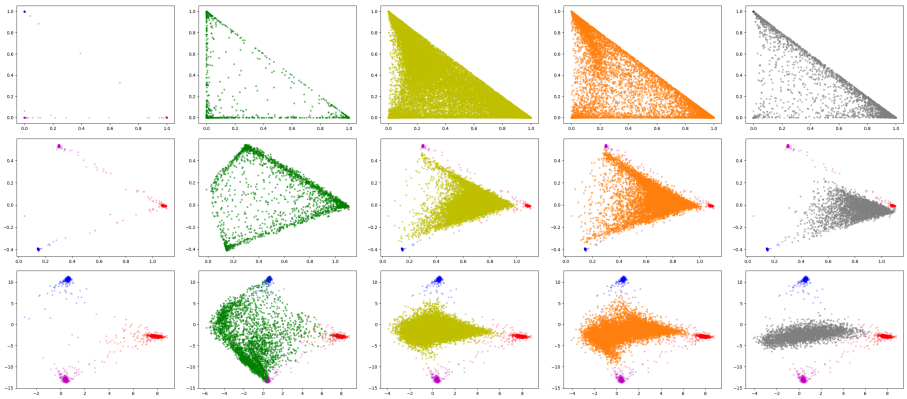

Figure 5.3 – Embedding spaces for CE, ML and ODM (rows respectively) being tested on in-dist 2, 6, 7 of MNIST (red, blue, purple), and out-dist 5, 9, 1 of MNIST (green), SVHN (yellow), CIFAR-10 (orange), and Gaussian noise (grey).

overconfident cases. In those cases, a metric learning approach presents more benefits for out-of-distribution detection. It allows the output embedding space to be more representative of the learned classes around the class centers, and naturally has the ability to give low scores to unseen data. Finally, when some out-of-distribution samples are shown during training, the network is more capable of adapting the embedding space to be more separable against anomaly data.

5.4 Results

To assess the performance of our proposed method, we first compare with existing state-of-the-art out-of-distribution detection methods on the SVHN [117] and CIFAR-10 [77] datasets using VGGnet [162] and evaluated with the metrics explained in Sec. 5.4.1[†]. Furthermore, as a more application-based benchmark, we propose to compare cross-entropy based strategies and metric learning strategies on the Tsinghua dataset [191] of traffic signs. In this second set of experiments we use our own implementation of the metrics defined in [88].

[†]Code available at: https://mmasana.github.io/OoD_Mining

Method	In-dist accuracy	Out-dist	FPR at 95% TPR	Detection Error	AUROC	AUPR-in	AUPR-out
CE	99.70	Novelty	33.76	19.38	92.33	92.73	92.29
		Gaussian noise	0.70	2.85	98.85	99.21	98.14
		SVHN	0.23	2.60	99.48	98.64	99.91
		CIFAR-10	2.86	3.93	98.96	98.02	99.57
Ours - ML	99.54	Novelty	21.05	13.03	94.48	94.02	94.46
		Gaussian noise	0.00	1.95	98.54	99.21	95.15
		SVHN	0.00	1.74	98.88	98.76	99.61
		CIFAR-10	0.01	2.36	98.87	98.93	99.12
Ours - ODM	99.64	Novelty	0.16	1.67	99.95	99.94	99.96
		Gaussian noise	0.00	1.76	99.14	99.46	97.66
		SVHN	0.00	0.96	99.65	99.41	99.89
		CIFAR-10	0.00	1.31	99.54	99.45	99.68

Table 5.1 – Quantitative comparison between cross-entropy and metric learning based methods training on LeNet for MNIST – 2, 6, 7 (In-dist), 0, 3, 4 and 8 (Seen Out-dist) and 5, 9, 1 (Unseen Out-dist Novelty).

5.4.1 Out-of-distribution detection metrics

In out-of-distribution detection, comparing different detector approaches cannot be done by measuring only accuracy. The question we want to answer is if a given test sample is from a different distribution than that of the training data. The detector will use information from the classifier or embedding space, but the prediction is whether that processed sample is part of the in-distribution or the out-distribution. To measure this, we adopt the metrics proposed in [88]:

- **FPR at 95% TPR** is the corresponding False Positive Rate (FPR=FP/(FP+TN)) when the True Positive Rate (TPR=TP/(TP+FN)) is at 95%. It can be interpreted as the misclassification probability of a negative (out-distribution) sample to be predicted as a positive (in-distribution) sample; and
- **Detection Error** measures the probability of misclassifying a sample when the TPR is at 95%. Assuming that a sample has equal probability of being positive or negative in the test, it is defined as $0.5(1-\text{TPR})+0.5\text{FPR}$,

where TP, FP, TN, FN correspond to true positives, false positives, true negatives and false negatives respectively. These two metrics were also changed to **TNR at 95% TPR** and **Detection Accuracy** in [82], which can be calculated with $1-x$ from the two metrics above, respectively. We use the latter metrics only when comparing to other state-of-the-art methods. This is also done because the implementation in both [82, 88]

allows for using a TPR which is not at 95% in some cases, meaning that the Detection Error can go below 2.5 since TPR is not fixed to 0.95.

In order to avoid the biases between the likelihood of an in-distribution sample to being more frequent than an out-of-distribution one, we need threshold independent metrics that measure the trade-off between false negatives and false positives. We adopt the following performance metrics proposed in [61]:

- **AUROC** is the Area Under the Receiver Operating Characteristic proposed in [33]. It measures the relation between between TPR and FPR interpreted as the probability of a positive sample being assigned a higher score than a negative sample.
- **AUPR** is the Area Under the Precision-Recall curve proposed in [104]. It measures the relationship between precision (TP/(TP+FP)) and recall (TP/(TP+FN)) and is more robust when positive and negative classes have different base rates. For this metric we provide both AUPR-in and AUPR-out when treating in-distribution and out-distribution samples as positive, respectively.

5.4.2 Comparison with state-of-the-art

We compare our method with two very recent state-of-the-art methods. One of them uses a confidence classifier and an adversarial generator (CC-AG) [82] and like ours uses out-of-distribution images during training. The second method is ODIN [88] which does not consider out-of-distribution images during training. In [82] they compare CC-AG with ODIN [88], and show that they can perform much better in the novelty case but similar for the anomaly cases.

We train each SVHN and CIFAR-10 as the in-distribution datasets while using the other dataset as the seen out-of-distribution during training. We train using VGGnet, just like [82], with a contrastive loss of margin 10 and a 25% of (in-dist, out-dist) pairs every two batches. Following the experiments of [82], we test the resulting networks on the in-distribution test set for classification, and TinyImageNet [36], LSUN [182] and Gaussian noise for out-of-distribution detection. For evaluation we use the proposed metrics from their implementation, namely: true negative rate (TNR) when true positive rate (TPR) is at 95%, detection accuracy, area under the receiver operating characteristic curve (AUROC) and both area under the precision-recall curve for in-distribution (AUPR-in) and out-distribution (AUPR-out).

Table 5.2 shows the results. For SVHN as the in-distribution results are as expected, with ODIN having lower results due to not using any out-of-distribution during training, and both CC-AG and ODM having near perfect performance. In the case of CIFAR-10 being the in-distribution, the same pattern is repeated for the seen distribution from SVHN. However, for the unseen out-of-distributions, CC-AG achieves the lower

In-dist classification	Out-dist	TNR at 95% TPR	Detection Accuracy	AUROC	AUPR-in	AUPR-out
SVHN 93.8/**94.2**/68.7	CIFAR-10*	47.4/**99.9**/99.8	78.6/**99.9**/99.8	62.6/**99.9**/99.5	71.6/**99.9**/99.7	91.2/99.4/**99.9**
	Tiny	49.0/**100.0**/99.0	79.6/**100.0**/99.1	64.6/**100.0**/99.0	72.7/**100.0**/96.5	91.6/99.4/**99.8**
	LSUN	46.3/**100.0**/99.4	78.2/**100.0**/99.5	61.8/**100.0**/99.3	71.1/**100.0**/97.8	90.8/99.4/**99.8**
	Gaussian	56.1/**100.0**/**100.0**	83.4/**100.0**/**100.0**	72.0/**100.0**/**100.0**	77.2/**100.0**/**100.0**	92.8/99.4/**100.0**
CIFAR-10 80.1/**80.6**/54.0	SVHN*	13.7/**99.8**/**99.8**	66.6/**99.8**/99.7	46.6/**99.9**/**99.9**	61.4/**99.9**/**99.9**	73.5/99.8/**100.0**
	Tiny	13.6/10.1/**17.1**	62.6/58.9/**66.9**	39.6/31.8/**66.2**	58.3/55.3/**60.3**	**71.0**/66.1/68.2
	LSUN	14.0/10.8/**19.6**	63.2/60.2/**70.9**	40.7/34.8/**68.4**	58.7/56.4/**59.5**	**71.5**/68.0/70.7
	Gaussian	2.8/**3.5**/3.0	50.0/50.0/**64.2**	10.2/14.1/**49.8**	48.1/49.4/**64.1**	39.9/**47.0**/46.7

Table 5.2 – Comparison with the state-of-the-art. All metrics show the methods as ODIN/CC-AG/ODM, red indicates worst performance, bold indicates best, * for seen distribution.

performance on both TinyImageNet and LSUN datasets, and ODIN the lower for Gaussian noise. Although not always achieving the best performance, ODM is able to compete with the best cases, and is never the worst performer. Gaussian noise seems to be the most difficult case on CIFAR-10, which is a more complex dataset than SVHN. For ODIN, as it is only based on cross-entropy, it becomes to overconfident. In the case of CC-AG and ODM, the low results might be related to Gaussian noise being too different from the out-distribution seen during training.

Finally, it is important to note that metric learning has a lower classification accuracy for in-distribution. This has already been observed in [65], where features learned by classification networks with typical softmax layers are compared with metric learning based features on several benchmark datasets. For good classification results our metric learning network should be combined with those of a network trained with cross-entropy. One could also consider a network with two heads, where after some initial shared layers a cross-entropy branch and a metric learning branch are trained in a multi-task setting.

5.4.3 Tsinghua traffic sign dataset

We evaluate our method on a real application: traffic sign recognition in the presence of unseen traffic signs (novelty) and not-a-traffic-sign detection (anomaly). We compare our proposed method ODM against ODIN [88], as a cross-entropy based method, on the Tsinghua dataset [191]. We divide traffic sign classes into three disjoint partitions: the in-distribution classes, seen out-of-distribution images used for training, and unseen out-of-distribution images used for testing on out-of-distribution detection.

Restricted splits: Since Tsinghua contains some very similar traffic sign classes which would rarely be learned without each other (i.e. all speed limits, all turning

Figure 5.4 – In-distribution (top-left), seen (top-right) and unseen(bottom-left) out-of-distribution partition classes, and background patches (bottom-right) from the proposed Tsinghua split.

arrows, ...), we group those that are too similar in order to build a more reasonable and natural split than just a random one. For the same reason, we also discard classes with less than 10 images as they introduce errors. Therefore, we generate a random split which applies by the mentioned restrictions (see Fig. 5.4), by taking a 50-20-30% split of the classes for the in-distribution, seen out-distribution and unseen out-distribution respectively.

Regarding anomalies, we consider Gaussian noise, but also background patches from the same Tsinghua dataset images. Those patches are generated randomly from the central area of the original full frames to avoid an unbalanced ratio of ground and sky images, which can be semantically richer and more challenging. In a real traffic sign detector application, where detected possible traffic signs are fed to a classifier, this kind of anomalies are more realistic and account for possible detection errors more than Gaussian noise. The global performance of the system can be improved by avoiding that those anomalies reach the classifier and produce an overconfident error.

For this experiment, we learn a 32-dimensional embedding space, training a WRN-28-10 model [183] with an Adam optimizer at learning rate 0.0001 for 10,000 steps. The same training parameters are used for ODIN since they provided the best results on the validation set. Table 5.3 shows the results of the comparison between ODIN,

Method	In-dist accuracy	Out-dist	FPR at 95% TPR	Detection error	AUROC	AUPR-in	AUPR-out
ODIN	98.29	Tsinghua (unseen)	8.74	6.87	97.82	96.19	98.92
		Background (unseen)	22.42	13.71	96.43	92.13	98.48
		Noise (unseen)	0.23	2.61	98.59	98.40	98.76
Ours - ML	98.93	Tsinghua (unseen)	5.23	5.11	98.77	97.38	99.45
		Background (unseen)	0.25	2.62	99.35	99.03	99.64
		Noise (unseen)	0.07	2.53	99.51	99.25	99.72
Ours - ODM	98.96	**Tsinghua** (seen)	4.38	4.70	99.01	98.01	99.63
		Background (unseen)	0.17	2.60	99.28	98.81	99.67
		Noise (unseen)	0.00	2.51	99.69	99.51	99.73
Ours - ODM	98.57	Tsinghua (unseen)	8.65	6.82	97.84	94.40	98.57
		Background (seen)	0.01	2.50	99.99	99.94	99.99
		Noise (unseen)	0.00	2.50	100.00	99.97	99.99
Ours - ODM	99.00	Tsinghua (unseen)	5.72	5.36	98.50	97.09	99.30
		Background (unseen)	1.51	3.25	98.53	97.97	99.20
		Noise (seen)	0.00	2.50	100.00	99.93	99.99

Table 5.3 – Comparison between ODIN and our proposed learning strategies on a WRN-28-10 architecture when using novelty, anomaly (background patches and Gaussian noise) as seen out-of-distribution data as well as not seen out-of-distribution.

ML and ODM for both seen novelty and anomaly cases. Note that our implementation of the Detection Error metric is fixed to use the FPR at a TPR of 95%, making a value of 2.50 the one of a perfect detector (see Sec. 5.4.1).

In terms of in-distribution classification accuracy, both methods are equivalent. However, the comparison of plain metric learning (Ours-ML) with ODIN shows that learning an embedding can be more suitable for out-of-distribution detection of both novelty and anomalies. Introducing out-of-distribution samples during training slightly improves all cases. Using anomalies as seen out-of-distribution during training helps the detection of the same kind of anomaly as expected since anomalies will be forced to be further away from the in-distribution in the embedding space. However, in some cases, it can damage novelty detection, which would not be guaranteed to be pushed away from the learned classes.

Random splits: Alternatively to the Tsinghua split generated with the restrictions introduced earlier, we also perform the comparison in a set of 10 random splits without applying any restriction to the partition classes. We still discard the classes with fewer than 10 images per class. Table 5.4 shows the average performance for this set of splits with their respective standard deviation. Since the split of the classes is random, this leads to highly similar or mirrored classes to be separated into in-distribution and out-distribution, creating situations that are very difficult to predict correctly. For

Method	In-dist accuracy	Out-dist	FPR at 95% TPR	Detection error	AUROC	AUPR-in	AUPR-out
ODIN	99.29±0.05	Tsinghua (unseen)	20.85±2.28	12.92±1.14	93.50±1.05	93.78±1.93	92.41±0.73
		Background (unseen)	8.39±6.34	6.70±3.17	98.06±1.26	97.02±3.15	98.79±0.60
		Noise (unseen)	0.03±0.43	2.53±0.85	99.67±0.34	99.60±0.39	99.74±0.41
Ours - ML	99.16±0.16	Tsinghua (unseen)	21.05±3.25	13.03±1.62	94.18±0.92	94.42±1.12	92.75±1.08
		Background (unseen)	1.91±1.02	3.45±0.51	99.14±0.32	98.79±0.35	99.40±0.22
		Noise (unseen)	0.30±0.96	2.65±0.48	99.27±0.36	99.09±0.40	99.43±0.35
Ours - ODM	99.13±0.22	**Tsinghua** (seen)	16.29±4.53	10.65±2.26	96.27±0.86	96.78±0.93	95.11±1.15
		Background (unseen)	0.39±1.63	2.71±0.31	99.50±0.27	99.30±0.31	99.66±0.20
		Noise (unseen)	0.01±1.39	2.51±0.70	99.59±0.54	99.51±0.60	99.69±0.43
Ours - ODM	99.09±0.18	Tsinghua (unseen)	20.36±3.63	12.68±1.81	93.47±1.55	93.58±2.10	92.00±1.74
		Background (seen)	0.01±0.03	2.51±0.01	99.97±0.02	99.92±0.03	99.98±0.01
		Noise (unseen)	0.00±0.00	2.50±0.01	99.99±0.03	99.97±0.05	99.99±0.01
Ours - ODM	99.02±2.42	Tsinghua (unseen)	20.87±1.63	12.93±0.81	93.65±1.05	94.01±1.48	92.33±0.89
		Background (unseen)	0.97±1.19	2.99±0.60	99.14±0.19	98.90±0.23	99.39±0.19
		Noise (seen)	0.00±0.00	2.50±0.01	100.00±0.00	99.98±0.01	99.99±1.85

Table 5.4 – Comparison between ODIN and our proposed learning strategies on a WRN-28-10 architecture, when using novelty, anomaly (background patches and Gaussian noise) as seen out-of-distribution data as well as not seen out-of-distribution. The experiments are performed on a set of 10 random splits and the metrics provided are the mean of the metrics on the individual splits ± its standard deviation.

instance, detecting that a turn-left traffic sign is part of the in-distribution while the turn-right traffic sign is part of the out-distribution, is very difficult in many cases. Therefore, the results from the random splits have a much lower performance, specially for the novelty case.

When comparing the metric learning based methods, ODM improves over ML for the test set that has been seen as out-distribution during training. In general, using novelty data as out-distribution makes an improvement over said test set, as well as for background and noise. However, when using background images to push the out-of-distribution further from the in-distribution class clusters in the embedding space, novelty is almost unaffected. The same happens when noise is used as out-distribution during training. This could be explained by those cases improving the embedding space for data that is initially not so far away from the in-distribution class clusters. This would change the embedding space to push further the anomalies, but would leave the novelty classes, originally much closer to the clusters, almost at the same location.

When introducing out-of-distribution samples, the behaviour on the random splits is the same as for the restricted splits: while introducing novelty helps the detection

on all cases, introducing anomaly helps the detection of the same kind of anomaly.

Embedding visualization: Figure 5.5 shows the embeddings for ODM (with novelty as seen out-of-distribution) and ML after applying PCA. When using ML, the novelties are not forced away from the in-distribution clusters so they share the embedding space in between those same in-distribution clusters. In the case of ODM, the out-of-distribution clusters are more clearly separated from the in-distribution ones.

5.5 Conclusions

In this chapter, we propose a metric learning approach to improve out-of-distribution detection which performs comparable or better than the state-of-the-art. We show that metric learning provides a better output embedding space for detecting data outside the learned distribution than cross-entropy softmax based models. This opens an opportunity to further research on how this embedding space should be learned, with restrictions that could further improve the field. The presented results suggest that out-of-distribution data might not all be seen as a single type of anomaly, but instead a continuous representation between novel and anomalous data. In that spectrum, anomaly detection is the easier task, giving more focus on the difficulty of novelty detection. Finally, we also propose a new benchmark for out-of-distribution detection on the Tsinghua dataset, as a more realistic scenario for novelty detection.

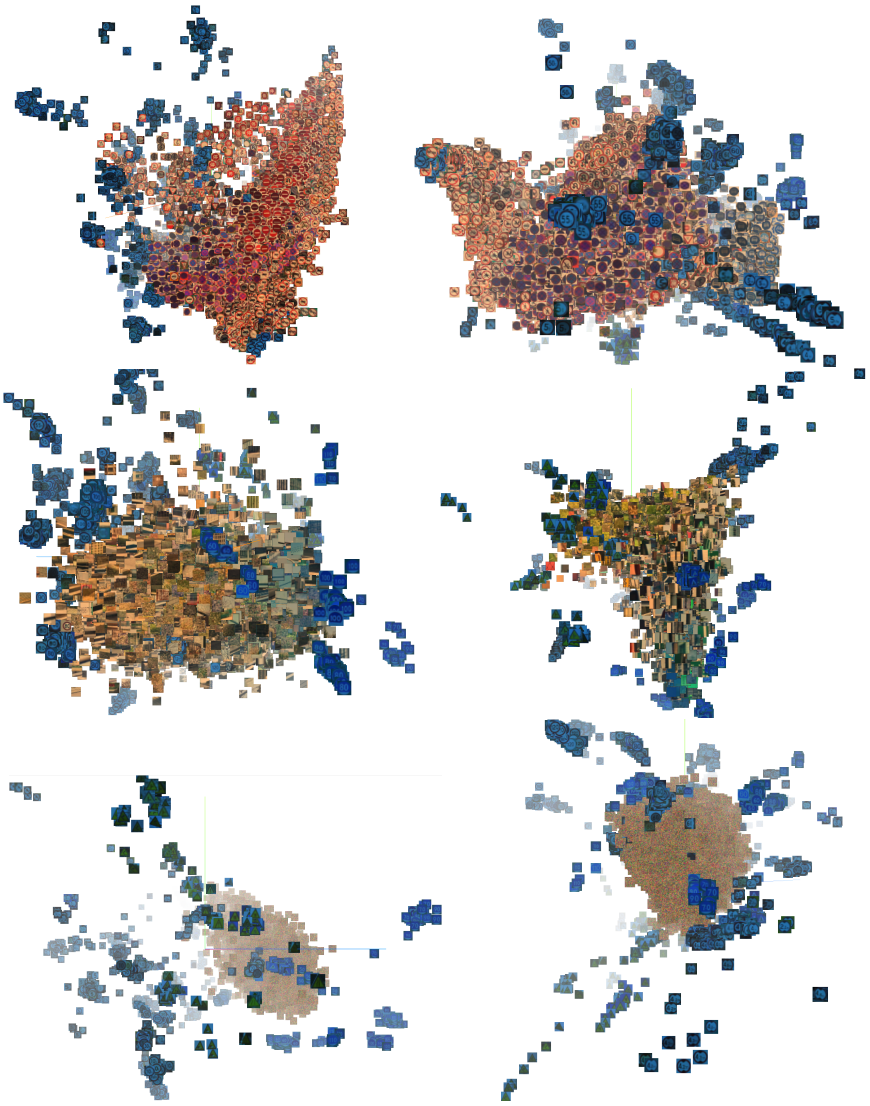

Figure 5.5 – Embedding spaces after PCA for ODM (left) and ML (right) tested for in-dist (blue shaded) and out-dist (yellow shaded). Results are for Tsinghua (first row), background patches (second row) and Gaussian noise (third row).

6 Conclusions

This thesis aims to develop theory which allows learning of neural networks beyond the standard setting where all training data is jointly available and testing data can be assumed to come from the same distribution as the training data. We therefore address the compression and adaptation of networks to small target domains, the incremental learning of networks faced with a sequence of tasks, and finally the detection of out-of-distribution samples at inference time. First we summarize the main contributions of the thesis, and then we discuss possible directions for future work.

6.1 Summary of contributions

Chapter 1 presents four research objectives on which this thesis focuses. All of these objectives are related to scalability problems and here we summarize each and discuss the solutions presented in this thesis.

> *Objective 1*: Pretrained networks contain large amounts of knowledge but are often too large to be used on certain devices. We aim to study compression in the context of domain adaptation, where knowledge unrelated to the target task can be dropped in exchange for computational efficiency.

In Chapter 2 we propose a compression method for domain-adapted networks from a large dataset as source to a smaller, fine-grained dataset as target. We demonstrate that networks trained on a specific domain tend to have neurons with relatively flat activation rates, indicating that almost all neurons are equally important in the network. However, after transferring to a target domain, activation rates tend to be skewed. We show that compression which takes activations into account can be formulated as a rank-constrained regression problem which has a closed-form solution. As an additional contribution we show how to compensate for bias introduced by the matrix approximation done by SVD, which consistently obtains improved results. Furthermore, the experimental results support that higher compression rates can be applied to target domains further away from the source domain.

> *Objective 2*: For many practical applications of continual learning, allowing for a variable amount of forgetting is undesirable. Therefore, we

> aim to design a learning system that can adapt to new tasks, while using previous knowledge, but without any forgetting.

In Chapter 3 we propose a new method for continual learning which does not suffer from any forgetting. This is achieved by relating each task with a ternary mask at each layer that identifies the activations that are used for each task. By applying the masks on the activations, we greatly reduce the number of extra parameters added per task and the overhead of the network in which other methods incur. In addition, we propose to apply a task-specific feature normalization of features, which allows adjusting previously learned features to new tasks. Ablation experiments show that improves results of the proposed ternary feature masks. When compared to a wide range of other continual learning techniques in the task-incremental learning setting, our method consistently outperforms those on a variety of datasets. Finally, our experimental section is one of the first to compare different approaches under different class orderings. We explore how incremental learning is affected by learning tasks with a random or semantic task split, as well as starting from a larger task which provides a more robust representation.

> *Objective 3*: As some systems become more dynamical and data becomes accessible in streams, systems should be able to learn while mitigating catastrophic forgetting. This desire has resulted in a wave of recent works, however a comprehensive comparison of their performance is currently lacking. Therefore, we survey the current state-of-the-art on class-incremental learning.

In Chapter 4 we perform an extensive survey on class-incremental learning. We study the related work, identify the main challenges and organize existing approaches into three families: regularization, rehearsal and bias-correction. We provide experiments on twelve methods on a wide range of incremental learning scenarios, some well-known ones and some new ones. We show that current exemplar-free methods can not compete with exemplar rehearsal methods, and that the baseline of fine-tuning with exemplars performs better than most of them. We compare the exemplar sampling methods and agree with most of the community that random sampling is usually a good enough option, but herding sampling provides a more robust performance for larger tasks. We point out that methods which tackle the task-recency bias obtain better performance. However, most presented results in the related works are on experimental setups with small domain shifts, while we show that large domain shift scenarios might require new techniques to obtain satisfactory results. Finally, we are the first to present a comparison of a wide range of network architectures, showing that current class-incremental learning methods benefit differently from them.

Objective 4: we aim to apply metric learning to the out-of-distribution detection problem, thus evading the problems introduced by using a cross-entropy loss. Metric learning can naturally be extended to new classes whereas classification networks are forced to produce an answer in terms of known classes.

In Chapter 5 we propose a metric learning approach to improve out-of-distribution detection which performs comparable or better than the cross-entropy-based methods. We show that metric learning provides a better output embedding space to detect data outside the learned distribution than cross-entropy softmax-based models. The presented results suggest that out-of-distribution data might not all be seen as a single type of anomaly, but instead a continuous representation between novel and anomalous data. In that spectrum, anomaly detection is the easier task, giving more focus on the difficulty of novelty detection. Finally, we also propose a new benchmark for out-of-distribution detection based on the Tsinghua dataset, as a more realistic scenario for novelty detection.

Software packages: the code supporting most of our work has been made available publicly, contributing to the machine learning and computer vision communities for comparison and further research.

Applied research: some of the work done in this thesis has been used in more applied fields such as medical imaging, allowing for collaboration with other research groups and the publication of other manuscripts. For example, the results on network compression from Chapter 2 where successfully transferred to polyp detection in colonoscopy images [16, 17]. In that field, when deploying trained models, the hardware has memory limitations which can be circumvented by applying the domain-adaptive network compression presented in this thesis.

6.2 Future research directions

Network compression has become a mature, independent research field. However, its principle focus is on the compression of single networks into smaller ones while maintaining network performance. We think that one of the future challenges for lifelong learning is rather the compression and fusion of sets of networks coming from a variety of domains. When faced with new domains, the learning system should decide which networks to deploy to the new domain. Developing theory and measures for this selection is still an open research area. After adapting to the new domain, the network can be either discarded, added or fused to the set of networks that will be used to employ to future domains. The tools to make these decisions are also still in their infancy stage and we think this to be an exciting future research direction.

Many continual learning methods rely on the storage of exemplar images from previous tasks. The storing of images, however, is not optimal. Therefore, recent works have focused on the combination of compression and continual learning [20, 59]. This has lead to Online Continual Compression, a problem where storage of a representative dataset and simultaneous compression of the model are performed from a non i.i.d data stream. Furthermore, with the increase in new available data from data streams, storing large datasets to train models will become less appealing since it does not capture the slight changes in the data distribution. Online learning techniques can solve this issue, at the cost of some performance limitations, which we expect to be addressed in future research.

Another exciting research direction is evaluation of continual learning in real-world applications. A recent dataset which has propelled research in this direction is the CORe50 [95], which since its introduction has been used in many works [57, 96, 103, 124]. This dataset considers a practical setting which is common in robotics: objects are presented one by one to a robot, which has to incrementally learn them, while not forgetting previously learned objects. Methods are allowed to use a small buffer of images from previously seen objects. An additional challenge of the dataset is that it considers distinct sessions, each with different background and lighting. We are especially interested in verifying if the conclusions of Chapter 4 generalize to this dataset.

When learning new tasks in continual learning, the next batch of classes to learn is defined by a human. However, in scenarios like the one presented above, it can be undesirable to have a human analyze the data that has been learned and design which data to learn next. With the use of out-of-distribution, unknown classes can be identified separately from the known tasks. This opens the option to learn from the out-of-distribution and to analyze that data in order to automatize or better choose which tasks need to be added next. This automation would allow us to close the loop of continual learning, where some systems would be able to independently learn new tasks while identifying the next potential tasks that would better fit into the models.

Summary of published works

1. **Marc Masana**, Joost van de Weijer and Andrew D. Bagdanov. “On-the-fly network pruning for object detection". In International Conference on Learning Representations (ICLR), 2016.

2. Ozan Caglayan, Walid Aransa, Yaxing Wang, **Marc Masana**, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault and Joost Van de Weijer. “Does multimodality help human and machine for translation and image captioning?". Publication accepted at WMT 2016 after winning the Multimodal Machine Translation challenge (WMT), 2016.

3. Esteve Cervantes, Long Long Yu, Andrew D. Bagdanov, **Marc Masana**, Joost van de Weijer. “Hierarchical part detection with deep neural networks". In IEEE International Conference on Image Processing (ICIP), 2016.

4. Ozan Caglayan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, **Marc Masana**, Luis Herranz and Joost Van de Weijer. “Lium-cvc submissions for wmt17 multimodal translation task". Publication accepted at WMT 2017 after winning the Multimodal Machine Translation challenge (WMT), 2017.

5. **Marc Masana**, Joost van de Weijer, Luis Herranz, Andrew D. Bagdanov and Jose M. Alvarez. “Domain-adaptive deep network compression". In International Conference on Computer Vision (ICCV), 2017.

6. Xialei Liu*, **Marc Masana***, Luis Herranz, Joost van de Weijer, Antonio M. Lopez and Andrew D. Bagdanov. “Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting". In IEEE International Conference on Pattern Recognition (ICPR), 2018.

7. Jorge Bernal, Aymeric Histace, **Marc Masana**, Quentin Angermann, Cristina Sánchez-Montes, Cristina Rodriguez, Maroua Hammami, Ana Garcia-Rodriguez, Henry Córdova, Olivier Romain, Gloria Fernández-Esparrach, Xavier Dray and Javier Sanchez. “Polyp Detection Benchmark in Colonoscopy Videos using GTCreator: a Flexible Annotation Tool for Image Datasets". In International Journal of Computer Assisted Radiology and Surgery (IJCARS) 2018.

8. **Marc Masana**, Idoia Ruiz, Joan Serrat, Joost van de Weijer and Antonio M. Lopez. “Metric learning for novelty and anomaly detection". In British Machine Vision Conference (BMVC), 2018.

9. Aymen Azaza, Joost van de Weijer, Ali Douik and **Marc Masana**. “Context Proposals for Saliency Detection". In Computer Vision and Image Understanding (CVIU), 2018.

10. Ozan Caglayan, Adrien Bardet, Fethi Bougares, Loïc Barrault, Kai Wang, **Marc Masana**, Luis Herranz and Joost van de Weijer. “Lium-cvc submissions for wmt18 multimodal translation task". Publication accepted at WMT 2018 after winning the Multimodal Machine Translation challenge (WMT), 2018.

11. Jorge Bernal, Aymeric Histace, **Marc Masana**, Quentin Angermann, Cristina Sánchez-Montes, Cristina Rodríguez de Miguel, Maroua Hammami, Ana García-Rodríguez, Henry Córdova, Olivier Romain, Gloria Fernández-Esparrach, Xavier Dray and Javier Sánchez. “GTCreator: a flexible annotation tool for image-based datasets". In International Journal of Computer Assisted Radiology and Surgery (IJCARS), 2019.

12. Corina Kräuter, Ursula Reiter, Albrecht Schmidt, **Marc Masana**, Rudolf Stollberger, Michael Fuchsjäger and Gert Reiter. “Objective extraction of the temporal evolution of the mitral valve vortex ring from 4D flow MRI". In International Society for Magnetic Resonance in Medicine (ISMRM), 2019.

13. Corina Kräuter, Ursula Reiter, Clemens Reiter, Volha Nizhnikava, **Marc Masana**, Albrecht Schmidt, Michael Fuchsjäger, Rudolf Stollberger and Gert Reiter. “Automated mitral valve vortex ring extraction from 4D-flow MRI". In Magnetic Resonance in Medicine (MRM), 2020.

14. Aymen Azaza, Joost van de Weijer, Ali Douik, Javad Zolfaghari and **Marc Masana**. “Saliency from High-Level Semantic Image Features". In SN Computer Science, 2020.

15. **Marc Masana**, Bartłomiej Twardowski and Joost van de Weijer. “On Class Orderings for Incremental Learning". In Continual Learning Workshop at International Conference on Machine Learning (ICML), 2020.

16. David Berga, **Marc Masana** and Joost Van de Weijer. “Disentanglement of Color and Shape Representations for Continual Learning". In Continual Learning Workshop at International Conference on Machine Learning (ICML), 2020.

17. **Marc Masana**, Tinne Tuytelaars and Joost van de Weijer. “Ternary Feature Masks: sequential learning without any forgetting". Under review, 2020.

18. **Marc Masana**, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov and Joost van de Weijer. “Class-incremental learning: survey and performance evaluation". Under review, 2020.

19. Matthias De Lange, Rahaf Aljundi, **Marc Masana**, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh and Tinne Tuytelaars. “Continual learning: A comparative study on how to defy forgetting in classification tasks". In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021.

Summary of published code

1. **DALR**, implementation of the method presented in “Domain-adaptive deep network compression" ICCV 2017, https://mmasana.github.io/DALR.

2. **R-EWC**, implementation of the method presented in “Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting" ICPR 2018, https://github.com/xialeiliu/RotateNetworks.

3. **OoD**, implementation of the method presented in “Metric learning for novelty and anomaly detection" BMVC 2018, https://mmasana.github.io/OoD_Mining.

4. **Class-Orderings**, implementation of the method presented in “On Class Orderings for Incremental Learning" ICML Workshop 2020, https://github.com/mmasana/Class-Orderings.

5. **TFM**, implementation of the method presented in “Ternary Feature Masks: sequential learning without any forgetting" Under review 2020, https://github.com/mmasana/TernaryFeatureMasks.

6. **FACIL**, implementation of the method presented in “Class-incremental learning: survey and performance evaluation" Under review 2020, https://github.com/mmasana/FACIL.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

[5] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[6] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11816–11825, 2019.

[8] Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[9] Jose M. Alvarez and Lars Petersson. Decomposeme: Simplifying convnets for end-to-end learning. *arXiv preprint arXiv:1606.05426*, 2016.

[10] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2270–2278, 2016.

[11] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(9):1790–1802, 2016.

[12] Aayush Bansal, Xinlei Chen, Bryan Russell, Abhinav Gupta Ramanan, et al. Pixelnet: Representation of the pixels, by the pixels, and for the pixels. *arXiv preprint arXiv:1702.06506*, 2017.

[13] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[14] A. Bendale and T. E. Boult. Towards open set deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, 2016.

[15] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, 2009.

[16] Jorge Bernal, Aymeric Histace, Marc Masana, Quentin Angermann, Cristina Sánchez-Montes, Cristina Rodríguez de Miguel, Maroua Hammami, Ana García-Rodríguez, Henry Córdova, Olivier Romain, et al. Gtcreator: a flexible annotation tool for image-based datasets. *International Journal of Computer Assisted Radiology and Surgery*, 14(2):191–201, 2019.

[17] Jorge Bernal, Aymeric Histace, Marc Masana, Quentin Angermann, Cristina Sánchez-Montes, Cristina Rodriguez, Maroua Hammami, Ana Garcia-Rodriguez, Henry Córdova, Olivier Romain, et al. Polyp detection benchmark in colonoscopy videos using gtcreator: A novel fully configurable tool for easy and fast annotation of image databases. In *International Congress and Exhibition on Computer Assisted Radiology and Surgery*, 2018.

[18] Paul Bodesheim, Alexander Freytag, Erik Rodner, Michael Kemmler, and Joachim Denzler. Kernel null space methods for novelty detection. In *IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 3374–3381, 2013.

[19] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 535–541, 2006.

[20] Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. Online learned continual compression with adaptive quantization modules. In *International Conference on Machine Learning (ICML)*, 2020.

[21] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face & Gesture Recognition*, 2018.

[22] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *European Conference on Computer Vision (ECCV)*, 2018.

[23] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[24] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision (ECCV)*, 2018.

[25] Arslan Chaudhry, Albert Gordo, Puneet K Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. *arXiv preprint arXiv:2002.08165*, 2020.

[26] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations (ICLR)*, 2019.

[27] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. In *International Conference on Machine Learning (ICML)*, 2019.

[28] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations (ICLR)*, 2016.

[29] Yu Chen, Tom Diethe, and Neil Lawrence. Facilitating bayesian continual learning by natural gradients and stein gradients. In *NeurIPS Continual Learning Workshop*, 2019.

[30] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(3):1–145, 2016.

[31] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–546, 2005.

[32] Robert Coop and Itamar Arel. Mitigation of catastrophic forgetting in recurrent neural networks using a fixed expansion layer. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2013.

[33] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *International Conference on Machine Learning (ICML)*, pages 233–240. ACM, 2006.

[34] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.

[35] Riccardo Del Chiaro, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost Van de Weijer. Ratt: Recurrent attention to transient tasks for continual image captioning. In *ICML LifelongML Workshop*, 2020.

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[37] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[38] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *International Conference on Learning Representations (ICLR)*, 2017.

[40] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[41] Sebastian Farquhar and Yarin Gal. A unifying bayesian view of continual learning. In *NeurIPS Deep Learning Workshop*, 2019.

[42] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645, 2009.

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

[44] Robert M French. Catastrophic forgetting in connectionist networks. In *Trends in cognitive sciences*, volume 3, pages 128–135. Elsevier, 1999.

[45] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4885–4894, 2017.

[46] Ross Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[47] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[48] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 437–446, 2015.

[49] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[50] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[51] Liangke Gui and Louis-Philippe Morency. Learning and transferring deep convnet representations with group-sparse factorization. *Springer*, 3, 2015.

[52] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 498–505. IEEE, 2009.

[53] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006.

[54] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[55] Yu Hao, Yanwei Fu, Yu-Gang Jiang, and Qi Tian. An end-to-end architecture for class-incremental object detection with knowledge distillation. In *International Conference on Multimedia and Expo*, 2019.

[56] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks (ICNN)*, pages 293–299, 1993.

[57] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *International Conference on Robotics and Automation (ICRA)*, pages 9769–9776. IEEE, 2019.

[58] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision (ECCV)*, 2019.

[59] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision (ECCV)*, 2020.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[61] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[62] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[63] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[64] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[65] Shota Horiguchi, Daiki Ikami, and Kiyoharu Aizawa. Significance of softmax-based features in comparison to distance metric learning-based features. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 42(5):1279–1285, 2019.

[66] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[67] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[68] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *European Conference on Computer Vision (ECCV)*, 2020.

[69] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *BMVA British Machine Vision Conference (BMVC)*, 2014.

[70] Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1820–1830, 2019.

[71] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020.

[72] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

[73] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[74] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, pages 3521–3526, 2017.

[75] Parker Koch and Jason J Corso. Sparse factorization layers for neural networks with limited supervision. *arXiv preprint arXiv:1612.04468*, 2016.

[76] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE International Conference on Computer Vision Workshops*, 2013.

[77] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[79] Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

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

[81] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NIPS)*, pages 598–605, 1990.

[82] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations (ICLR)*, 2018.

[83] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4655–4665, 2017.

[84] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020.

[85] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning (ICML)*, 2019.

[86] Xuhong Li, Yves Grandvalet, and Franck Davoine. A baseline regularization scheme for transfer learning with convolutional neural networks. *Pattern Recognition*, 98:107049, 2020.

[87] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(12):2935–2947, 2017.

[88] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[89] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 806–814, 2015.

[90] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *International Conference on Pattern Recognition (ICPR)*, 2018.

[91] Xialei Liu, Joost van de Weijer, and Andrew D Bagdanov. Rankiqa: Learning from rankings for no-reference image quality assessment. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[92] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[93] Yaoyao Liu, An-An Liu, Yuting Su, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[94] Vincenzo Lomonaco and Davide Maltoni. Comparing incremental learning strategies for convolutional neural networks. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 175–184. Springer, 2016.

[95] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78, pages 17–26, 2017.

[96] Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Fine-grained continual learning. *arxiv preprint arXiv:1907.03799*, 2019.

[97] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[98] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[99] David G Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157. IEEE, 1999.

[100] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[101] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *European Conference on Computer Vision (ECCV)*, 2018.

[102] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[103] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.

[104] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[105] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation. *arXiv preprint arXiv:2010.15277*, 2020.

[106] Marc Masana, Idoia Ruiz, Joan Serrat, Joost van de Weijer, and Antonio M Lopez. Metric learning for novelty and anomaly detection. In *BMVA British Machine Vision Conference (BMVC)*, 2018.

[107] Marc Masana, Tinne Tuytelaars, and Joost van de Weijer. Ternary feature masks: continual learning without any forgetting. *arXiv preprint arXiv:2001.08714*, 2020.

[108] Marc Masana, Bartłomiej Twardowski, and Joost Van de Weijer. On class orderings for incremental learning. In *ICML Workshop on Continual Learning*, 2020.

[109] Marc Masana, Joost van de Weijer, and Andrew D Bagdanov. On-the-fly network pruning for object detection. *International Conference on Learning Representations Workshop*, 2016.

[110] Marc Masana, Joost van de Weijer, Luis Herranz, Andrew D Bagdanov, and Jose M Alvarez. Domain-adaptive deep network compression. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[111] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[112] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[113] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504, 2013.

[114] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation. In *IEEE International Conference on Computer Vision Workshops*, 2019.

[115] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013.

[116] Ashin Mukherjee. *Topics on Reduced Rank Methods for Multivariate Regression*. PhD thesis, The University of Michigan, 2013.

[117] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[118] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[119] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[120] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing (ICCVGIP)*, pages 722–729. IEEE, 2008.

[121] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724, 2014.

[122] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jähnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[123] Firat Ozdemir, Philipp Fuernstahl, and Orcun Goksel. Learn the new, keep the old: Extending pretrained models with new anatomy and images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2018.

[124] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[125] German I Parisi and Vincenzo Lomonaco. Online continual learning on sequences. In *Recent Trends in Learning From Data*, pages 197–221. Springer, 2020.

[126] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[127] Benedikt Pfülb and Alexander Gepperth. A comprehensive, application-oriented study of catastrophic forgetting in dnns. In *International Conference on Learning Representations (ICLR)*, 2019.

[128] Marco A. F. Pimentel, David A. Clifton, Lei A. Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[129] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[130] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. iTAML: An incremental task-agnostic meta-learning approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[131] Amal Rannen, Rahaf Aljundi, and Matthew B Blaschko Tinne Tuytelaars. Encoder based lifelong learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1320–1328, 2017.

[132] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[133] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[134] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[135] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2001–2010, 2017.

[136] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[137] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[138] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations (ICLR)*, 2018.

[139] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations (ICLR)*, 2019.

[140] Matthew Riemer, Tim Klinger, Djallel Bouneffouf, and Michele Franceschini. Scalable recollections for continual lifelong learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[141] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015.

[142] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2018.

[143] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

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

[145] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[146] Monika Schak and Alexander Gepperth. A study on catastrophic forgetting in deep lstm networks. In *International Conference on Artificial Neural Networks*, pages 714–728. Springer, 2019.

[147] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(7):1757–1772, July 2013.

[148] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[149] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[150] Alexander Schultheiss, Christoph Käding, Alexander Freytag, and Joachim Denzler. Finding the unknown: Novelty detection with extreme value signatures of deep neural activations. In Volker Roth and Thomas Vetter, editors, *German Conference on Pattern Recognition (GCPR)*, pages 226–238. Springer, 2017.

[151] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*, 2018.

[152] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[153] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, 2018.

[154] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2990–2999, 2017.

[155] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[156] Daniel L Silver and Robert E Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 90–101. Springer, 2002.

[157] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[158] Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. Toward training recurrent neural networks for lifelong learning. *Neural computation*, 32(1):1–35, 2020.

[159] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[160] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012, 2016.

[161] Siddharth Swaroop, Cuong V Nguyen, Thang D Bui, and Richard E Turner. Improving and understanding variational continual learning. *arXiv preprint arXiv:1905.02099*, 2019.

[162] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[163] Onur Tasar, Yuliya Tarabalka, and Pierre Alliez. Incremental learning for semantic segmentation of large-scale remote sensing data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(9):3524–3537, 2019.

[164] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems (NIPS)*, 1996.

[165] Stanford University. Tiny imagenet challenge, course cs231n, 2017.

[166] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4170–4178, 2016.

[167] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. In *NeurIPS Continual Learning Workshop*, 2018.

[168] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[169] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACM Multimedia*, pages 689–692, 2015.

[170] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

[171] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[172] Jiang Wang, Thomas Leung, Chuck Rosenberg, Jinbin Wang, James Philbin, Bo Chen, Ying Wu, et al. Learning fine-grained image similarity with deep ranking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[173] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *European Conference on Computer Vision (ECCV)*, pages 218–234, 2018.

[174] Max Welling. Herding dynamical weights to learn. In *International Conference on Machine Learning (ICML)*, 2009.

[175] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay GANs: learning to generate images from new categories without forgetting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[176] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[177] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[178] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 899–908, 2018.

[179] Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, pages 2365–2369, 2013.

[180] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[181] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014.

[182] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[183] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVA British Machine Vision Conference (BMVC)*, 2016.

[184] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations (ICLR)*, 2017.

[185] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, pages 3987–3995, 2017.

[186] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018.

[187] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong gan: Continual learning for conditional image generation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[188] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[189] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[190] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision (ECCV)*, pages 662–677. Springer, 2016.

[191] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2110–2118, 2016.