Class Incremental Continual Learning in Deep Neural Networks

Apprentissage Continu Classe par Classe pour les Réseaux de Neurones Profonds

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 575, Electrical, Optical, Bio-physics and Engineering (EOBE)
Spécialité de doctorat: Electronique et Optoélectronique, Nano- et Microtechnologies

Unité de recherche: Université Paris-Saclay, CNRS, Centre de Nanosciences et de Nanotechnologies, 91120, Palaiseau, France
Référent: Faculté des sciences d'Orsay

Thèse présentée et soutenue à Palaiseau, le 30/06/2021, par

Guillaume HOCQUET

Composition du jury

Direction de la thèse

Thèse de doctorat

NNT: 2021UPAST070

À tous ceux qui m'ont inspiré à poursuivre une carrière scientifique

Acknowledgment

Poursuivre une thèse en informatique était un projet de longue date et qui comptait beaucoup pour moi. Je tiens ainsi à remercier chaleureusement tous ceux qui ont contribué à sa réalisation et à son succès. Ces trois ans et demi passés au sein du CEA Nano Innov et du laboratoire C2N du CNRS m'ont énormément apporté tant sur le plan scientifique que sur le plan humain.

Merci à mon directeur de thèse Damien QUERLIOZ et à mon encadrant Olivier BICHLER pour m'avoir fait confiance pour mener à bien ce travail de recherche; pour m'avoir fait bénéficier d'une grande liberté dans le choix de mes méthodes de travail, de mes pistes de recherche et de mon organisation; pour leur soutien au moment de la rédaction d'articles et de ce manuscrit.

Merci aux membres de mon jury, Pierre BESSIERE, Vincent GRIPON, Jacques-Olivier KLEIN, Martial MERMILLOD et Teodora PETRISOR, qui ont examiné en détail mes travaux et montré leur appréciation de ma démarche dans leurs questions (qui ont duré près de deux heures !).

Merci à mes amis et collègues du CEA, Alix, Amir, Aurore, David, Gabriel, Inna, Jason, Johannes, Pierre-Guillaume, Quentin, Thibault, Thibaut et Vincent (x2), pour la bonne ambiance quotidienne et pour les soirées.

Merci à mes amis et collègues du CNRS, Atreya, Axel, Clément, Damir, Kamel, Liza, Marie, Maryam, Maxence, Thibaut et Tifenn, pour m'avoir accompagné tout au long de cette aventure.

Merci à mes amis Antoine, Antonin, Céline, Clément, Emmanuel, Florent, Lou, Mathieu, Paul-Marie et William, qui se sont intéressés à mes travaux et m'ont posé plein de questions pour essayer de comprendre.

Enfin, merci à ma famille pour leur soutien.

Résumé substantiel en français

L'apprentissage continu, parfois aussi appelé apprentissage incrémental, désigne une procédure d'apprentissage qui permet d'entraîner un modèle sur une séquence de tâches, et de s'assurer que ce modèle est capable d'obtenir de bonnes performances sur chacune de ces tâches après chaque phase d'apprentissage. La difficulté est que lors de l'apprentissage d'une nouvelle tâche, on suppose que le modèle n'a pas accès aux données des tâches précédentes.

Une liste de critères a été proposée par la communauté scientifique pour illustrer le comportement d'une méthode idéale d'apprentissage continu :

- Une consommation de mémoire limitée : Puisque l'un des principaux objectifs de l'apprentissage continu est d'éviter le stockage d'une grande quantité de données, il est légitime de s'attendre à ce qu'un algorithme d'apprentissage continu puisse limiter son impact sur la mémoire lors de l'apprentissage d'une nouvelle tâche. Cela peut se faire en fixant une limite au stockage des données ou à l'expansion du réseau.

- Pas d'identification de tâche : L'algorithme doit pouvoir s'adapter quelque soit la façon dont il reçoit les données d'entraînement, notamment dans le cas où on ne l'informe pas que les données fournies correspondent à une nouvelle tâche.

- Transfert vers l'avant : Le processus d'apprentissage sur de nouvelles tâches doit bénéficier des connaissances acquises précédemment pour être plus performant qu'un simple apprentissage sur chaque tâche indépendamment.

- Transfert vers l'arrière : On peut également s'attendre à ce que l'acquisition de connaissances avec les tâches suivantes améliore les performances sur les tâches précédentes.

- Indépendance de la taille de la tâche : L'algorithme doit être capable de fonctionner dans diverses conditions, en particulier lorsque le nombre de classes dans une tâche peut être aussi faible qu'une seule classe par tâche.

La conception d'un algorithme capable de répondre à toutes les exigences précédentes est un défi ardu et sort du cadre de notre étude. Nous nous concentrerons essentiellement sur le critère de la consommation de mémoire, puisque notre motivation est de produire un algorithme qui pourrait être utilisé sur des dispositifs embarqués, et sur le critère de la taille de la tâche pour nous assurer que notre algorithme soit capable de traiter des tâches composées d'une seule classe. Notre approche sera donc spécialement dédiée aux problèmes d'apprentissage continu classe par classe.

Après avoir passé en revue les principales approches d'apprentissage continu de l'état de l'art, nous avons remarqué que les modèles basés sur le calcul de distance présentaient des caractéristiques prometteuses pour relever les défis de l'apprentissage continu classe par classe. Dans cette thèse, nous allons nous appuyer sur cette approche pour développer des modèles plus performants. Ainsi, nous proposons de nous appuyer sur un extracteur de caractéristiques fixe lors de l'apprentissage de nouvelles tâches. De cette façon, il nous suffit de développer une stratégie pour exploiter, le mieux possible, les caractéristiques extraites par ce premier modèle.

La principale limite des modèles classiques basés sur le calcul de distance est qu'ils reposent sur des mesures de distances basiques telles que la distance euclidienne ou la similarité cosinus. Dans ce travail, nous proposons plutôt de construire une séquence de modèles, un pour chaque classe. L'objectif de chaque modèle sera de calculer la probabilité qu'un échantillon appartienne à la distribution de ses données d'apprentissage. En particulier, l'entraînement de chaque modèle reposera uniquement sur les données d'une seule classe.

Pour ce faire, nous avons besoin d'un modèle capable de construire une estimation de la fonction de densité de probabilité des données pour chaque classe. Grâce à cette estimation, nous serons en mesure de prédire à quelle classe appartient un échantillon donné en identifiant la classe ayant la probabilité la plus élevée. La fonction qui mesure l'adéquation d'un modèle statistique à un échantillon de données est la vraisemblance. Afin de calculer cette fonction, il faut s'appuyer sur des modèles génératifs :

- Les modèles basés sur l'énergie ont été étudiés depuis longtemps, mais ces approches sont difficiles à passer à l'échelle sur des ensembles de données complexes.

- Les réseaux adversariaux génératifs sont une architecture largement adoptée pour générer des images réalistes, mais ils ne reposent pas sur l'estimation de la vraisemblance.

- Les autoencodeurs variationnels constituent un moyen simple et efficace de générer des données, mais ils ne fournissent qu'une approximation de la valeur des variables latentes.

- Les modèles autorégressifs permettent de calculer des densités de probabilité mais ont un coût de calcul élevé car ils nécessitent plusieurs itérations pour un échantillon.

- Les réseaux de neurones inversibles sont un modèle récemment développé qui permettent de calculer efficacement la valeur exacte de la vraisemblance.

Par conséquent, les réseaux de neurones inversibles semblent être un choix favorable pour notre objectif. Il existe plusieurs façons de les implémenter. Le développement de ces méthodes est un domaine de recherche très actif encore aujourd'hui.

Dans nos premières expériences, nous proposons d'entraîner pour chaque classe un réseau de neurones inversible. À l'inférence, nous exécutons chaque réseau sur un échantillon et celui qui présente la probabilité la plus élevée détermine la classe de l'échantillon. Les méthodes pour lesquelles plusieurs modèles sont en concurrence pour identifier la classe d'un échantillon sont connues sous le nom de stratégies "One-versus-All". C'est pourquoi nous avons nommé notre système OvA-INN pour One-versus-All Invertible Neural Networks.

La mise à jour d'un modèle complet avec de nouvelles données peut être difficile avec un accès limité aux données précédentes. Le modèle peut avoir des performances réduites sur les classes précédentes car la procédure de mise à jour ne peut pas s'appuyer sur toutes les connaissances des données précédentes. Si cela n'est pas fait correctement, le modèle peut oublier complètement comment traiter les classes précédentes, c'est ce qu'on appelle l'oubli catastrophique. Mais dans notre cas, nous évitons ce problème en entraînant des modèles indépendants. Une fois qu'un modèle est entraîné sur ses données, il ne sera plus mis à jour par la suite.

Nous avons reproduit les protocoles expérimentaux utilisés par les approches d'apprentissage continu de l'état de l'art en apprenant à reconnaître des chiffres du jeu de données MNIST (10 classes) et des images du jeu de données CIFAR-100 (100 classes). Nos résultats expérimentaux ont montré que notre méthode était très efficace par rapport à ses concurrents, en particulier les méthodes basées sur le réapprentissage avec stockage partiel des données précédentes (comme iCaRL, GEM et SupportNet) et les méthodes basées sur le pseudo réentraînement avec des données générées (comme PGMA et DGR).

Dans nos expériences suivantes, nous étudions le coût en mémoire pour le stockage des réseaux de neurones inversibles. Nous proposons une nouvelle stratégie pour améliorer l'efficacité de leur stockage en mémoire. L'idée est de factoriser les matrices des paramètres du modèle. Au lieu d'être indépendantes les unes des autres, elles reposent sur une représentation commune qui peut être mise à jour avec chaque nouvelle classe apprise. Cela nécessite d'ajouter une contrainte dans la fonction de coût pour éviter une dérive des paramètres précédemment appris.

Nos résultats expérimentaux montrent un meilleur compromis entre la précision et la consommation de mémoire pour l'apprentissage continu de CIFAR-100 par rapport aux autres approches. Notre approche est même plus efficace en mémoire qu'un simple modèle de plus proches voisins consistant à stocker la sortie moyenne de chaque classe.

Dans nos dernières expériences, nous proposons d'étudier l'impact de l'extracteur de caractéristiques pour l'apprentissage continu. Jusqu'à présent, nos travaux se sont en effet appuyés sur un modèle pré-entraîné pour extraire les caractéristiques pertinentes à utiliser pour l'apprentissage continu. Alors que la plupart des travaux dans la littérature sont consacrés à la conception de nouvelles stratégies d'entraînement pour mettre à jour un extracteur de caractéristiques, nous nous intéressons davantage à comprendre ce qui fait un bon extracteur de caractéristiques, sans qu'il soit nécessaire de le mettre à jour par la suite. En d'autres termes, la communauté cherche généralement des réponses à la question "Comment entraîner un réseau dans ces conditions ?". De notre côté, nous pensons qu'il est également essentiel de se demander : "Quel type de réseaux est compatible avec ce style d'apprentissage ?".

Les architectures neuronales typiques peuvent être vues comme un empilement de transformations qui synthétisent des concepts de plus en plus abstraits à partir des données afin de répondre à un problème (classification, segmentation, régression,...). Les dernières couches contiennent toutes les informations nécessaires à la réalisation d'une prédiction. C'est pourquoi il est courant d'utiliser un modèle pré-entraîné pour répondre à un autre problème, à condition que les données du nouveau problème soient suffisamment similaires aux données du problème précédent. C'est une approche pertinente pour les tâches d'apprentissage traditionnelles, lorsque toutes les données sont disponibles dès le départ. Les réseaux neuronaux sont entraînés à découvrir des motifs pertinents dans les données et à rejeter toute information ne permettant pas de répondre au problème posé. Cependant, cette stratégie semble limitée dans le cas de l'apprentissage continu, où un bon modèle doit conserver autant d'informations que possible sur les données, car ces informations peuvent devenir utiles pour une tâche future. Par exemple, si seule la détection de forme est nécessaire à un modèle pour classer un ensemble d'images, il n'apprendra aucune caractéristique liée à la couleur. Ainsi, si nous devions introduire de nouvelles classes dans cette tâche de classification, et que ces classes nécessitaient des caractéristiques de couleur pour les distinguer des anciennes classes, les caractéristiques précédemment apprises ne seraient pas pertinentes pour cette nouvelle tâche.

Pour rendre un extracteur de caractéristiques aussi efficace que possible pour l'apprentissage continu, nous proposons de concevoir un modèle qui rejette le moins d'informations possible. C'est également une propriété des réseaux neuronaux inversables. Cependant, on ne peut pas simplement utiliser un réseau de neurones inversible comme extracteur de caractéristiques, car il aurait un coût élevé en mémoire. L'espace des caractéristiques doit être réduit en empilant des couches de réseaux neuronaux inversibles les unes sur les autres et en réduisant l'espace des caractéristiques entre chacune d'elles. Ce compromis permet de garder l'information utile accessible tout en ayant un impact limité sur la mémoire.

Nos résultats montrent que le pré-entraînement de l'extracteur de caractéristiques que nous proposons sur les 50 premières classes de CIFAR-100, puis l'apprentissage des 50 classes suivantes avec OvA-INN, conduit à de meilleures performances que d'autres approches dans ce contexte, en particulier PODNet, Rebalance et GFR qui reposent toutes sur la mise à jour d'une architecture ResNet.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 19, 22, 23, 30, 33

BiC Bias Correction. 43

CIFAR Canadian Institute For Advanced Research. 28

CL Continual Learning. 14, 21–25, 27–30, 32, 33, 37, 40, 42–45, 48, 49, 51, 52, 62, 65, 66, 69, 72, 73, 75, 76, 78, 81, 83, 84, 89, 91, 93, 95, 99, 101–103, 105, 107–109

CORe50 Continuous Object Recognition. 28

DEN Dynamically Expandable Networks. 77, 78

DGR Deep Generative Replay. 44, 45, 71–73

DL Deep Learning. 19, 20, 29

EWC Elastic Weight Consolidation. 78

FDA Food and Drug Administration. 30, 32

FPGA Field-programmable gate array. 84

GAN Generative Adversarial Network. 44, 45

GEM Gradient Episodic Memory. 42, 43, 71, 72

GFR Generative Feature Replay. 45, 101, 102, 104

Glow Generative Flow. 97, 100–105

iCaRL Incremental Classifier and Representation Learning. 41–43, 45, 48, 71–73, 75, 77, 84, 88, 89

ICDAR International Conference on Document Analysis and Recognition. 19

IJCNN International Joint Conference on Neural Networks. 19

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 19, 29

INN Invertible Neural Network. 53–55, 58–60, 62, 63, 65–70, 78–81, 87, 90, 91, 95–97, 99, 102, 108–110

ML Machine Learning. 19, 107

MNIST Modified National Institute of Standards and Technology. 28

MTL Medial Temporal Lobe. 26, 27

NICE Non-linear Independent Components Estimation. 60, 67

OOD out-of-distribution. 62, 63, 66

OvA-INN One-versus-All Invertible Neural Network. 66, 67, 69, 70, 72–78, 80, 81, 84–86, 88–90, 93, 99, 101–105, 108–110

PGMA Parameter Generation and Model Adaptation. 44, 45, 71, 72

PODNet Pooled Outputs Distillation Network. 41, 43, 100, 101, 104

ResNet Residual neural network. 55, 61, 74, 75, 77, 78, 80, 81, 90, 94, 99–104, 109

RPSnet Random Path Selection for Incremental Learning. 46, 47, 71, 72, 75, 77

SWS Slow Wave Sleep. 27

t-SNE t-distributed stochastic neighbor embedding. 78, 80

Chapter 1

Introduction

By interacting with the world, humans are able to acquire new skills and new knowledge. This ability to adapt a behavior to fulfill a need is readily available from birth and is continuously operating throughout every moment of one's life. We are constantly learning things and changing our view of our world. Back in the mid-20$^{\text{th}}$ century, early computer science pioneers such as Alan Turing had already envisioned that machines could replicate this learning process. Ever since, considerable effort has been invested to make computers able to learn automatically. However, as most students or former students can relate, it is not enough to acquire knowledge at a given point in time. Most of the challenge consists in fact in assimilating and keeping available a previously acquired knowledge. This thesis is dedicated to this second aptitude: make computers able to develop new skills automatically, while maintaining their expertise on previously acquired skills.

1.1 Quick overview of the state of the art of Deep Neural Networks

In the last decade, Artificial Intelligence (AI) has gained major attention from the academic community, the industrial sector, and also the general public. The Oxford Dictionary defines AI as the theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages. This field of study is, however, not new. The term dates back to a conference at Dartmouth College, Hanover, in 1956. But interest for this subject has encountered ups and downs because of the high expectations from the investors and the hard technological challenges that needed to be solved. This situation led to periods of limited funding and slower advances called "AI winters" from 1974 to 1980 and from 1987 to 1993.

The recent high degree of interest can be explained by three factors, namely:

- the availability of massive amounts of data,
- the progress in highly efficient computing systems,
- the broad accessibility of these technologies and knowledge.

These trends have greatly favored the development of Machine Learning (ML), which consists in designing algorithms that are able to solve problems by acquiring knowledge from data and interaction with the world. Among the variety of ML algorithms, Deep Learning (DL) is currently one of the most successful approaches, as it has proven a tremendously efficient ability to take advantage of the previous three factors. Starting in 2011, deep neural network models have won multiple computer vision challenges including International Joint Conference on Neural Networks (IJCNN) traffic signs, International Conference on Document Analysis and Recognition (ICDAR) Chinese handwriting, and ImageNet Large Scale Visual Recognition Challenge (ILSVRC), beating other approaches by a large margin [1–3]. Since then, numerous advances have been made to improve those methods and exploit their whole potential in various applications such as self-driving cars, virtual assistants, face recognition, colourisation of black and white images, speech synthesis, protein structure prediction, automatic game playing, ...

Since the beginning of the DL era, it is often observed in experimental results that those models tend to perform even better as the models gain in complexity and as more data is provided to train them. Consequently, a tremendous amount of work has been dedicated to improve their ability to scale in complexity, namely by designing methods that allow the training of models with more and more parameters, reaching tens of billions of parameters for natural language processing models (See Figure 1.1). Computer vision models also provide unprecedented results on larger architectures such as Noisy Student [4] with 480 million parameters or BiT [5] with 928 million parameters. At the same time, a lot of effort has also been devoted to harvest the most resources from the Internet. This can be achieved with a combination of weakly annotated data (using metadata) and manually annotated data (using crowdsourcing), leading to gigantic databases such as JFT-300M [6] with its hundreds of billions of images (See Figure 1.2).

On one hand, it is highly encouraging to see that those approaches produce better results as we provide them more computing power and data. On the other hand, this tendency is not sustainable in the long run [7]. While there is active research looking at decreasing the training time by relying on a large-scale distributed training on a GPU cluster, allowing the training of deep neural networks on ImageNet in hundreds of seconds, these types of infrastructures are not within everyone's reach [8]. Hence, for most users and applications, it still requires several days or weeks to complete this kind of training on a single GPU.

Furthermore, whether it was performed on a single machine or on several of them in parallel, the training of large models can be prohibitively expensive in electricity (reaching over 4.6 million US Dollars for a full training of the GPT-3 language model, as estimated by [9]).

To avoid the burden of long and costly training phases, a widely adopted method consists in using a pretrained network and slightly updating it to fit a new database (or only retraining the last layers). This strategy, called transfer learning, works usually fine on well-defined tasks. However, as soon as the task changes (with new classes to recognize

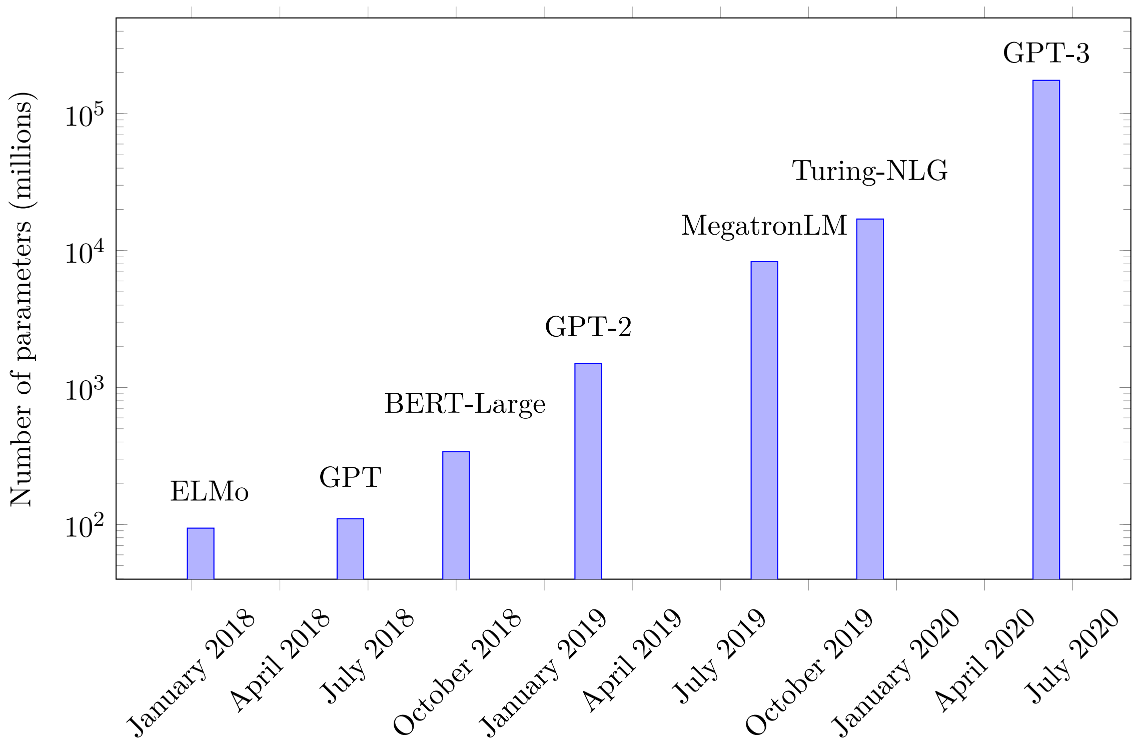

Figure 1.1: Evolution of the architecture size of recent deep learning models.

for example), it is mandatory to adapt the model. This can be a limiting factor if the data is continuously changing, or even a severe issue if previous data is no longer available. In this last case, it is simply not possible with a traditional training to tune a network and expect it to function correctly without the previous data. This limitation has motivated researchers to investigate new ways of updating a neural network and guarantee that it will continue to perform well even with limited or no access to previous data. This research area is called Continual Learning (CL).

1.2 Continual Learning framework

Continual Learning, sometimes also called Lifelong Learning, or Incremental Learning, designates a training procedure that allows a model to be trained on a sequence of tasks, and ensuring that this model is able to perform well on each of these tasks after the training phase. The difficulty is that when learning a new task, we suppose that the model has no

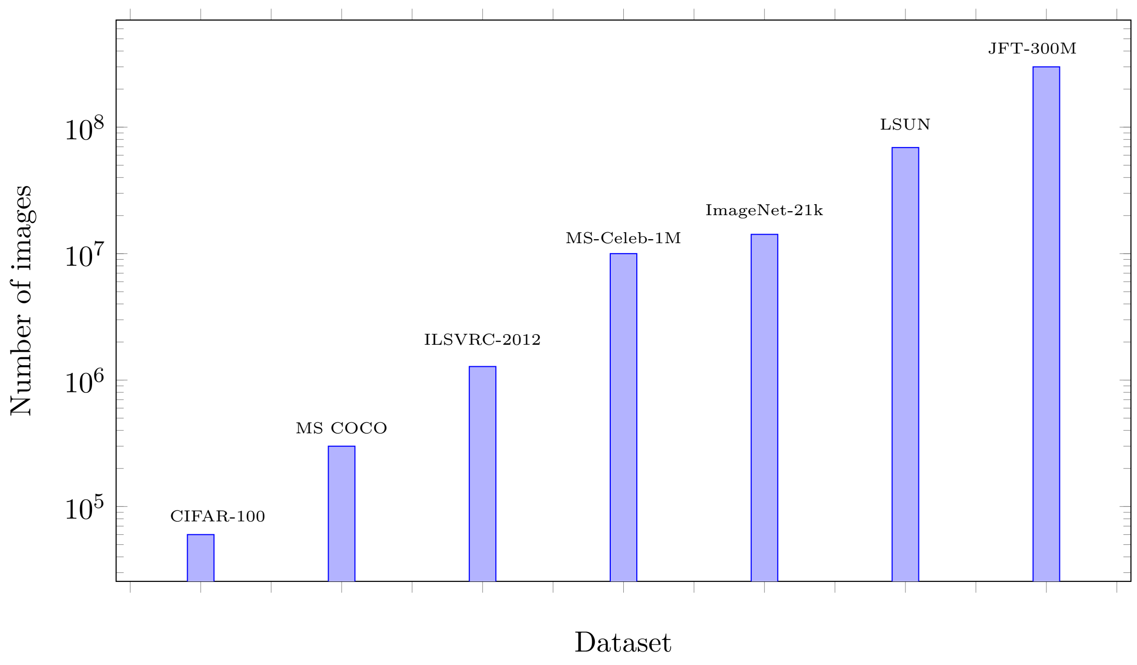

Figure 1.2: Comparison of the size of various image datasets.

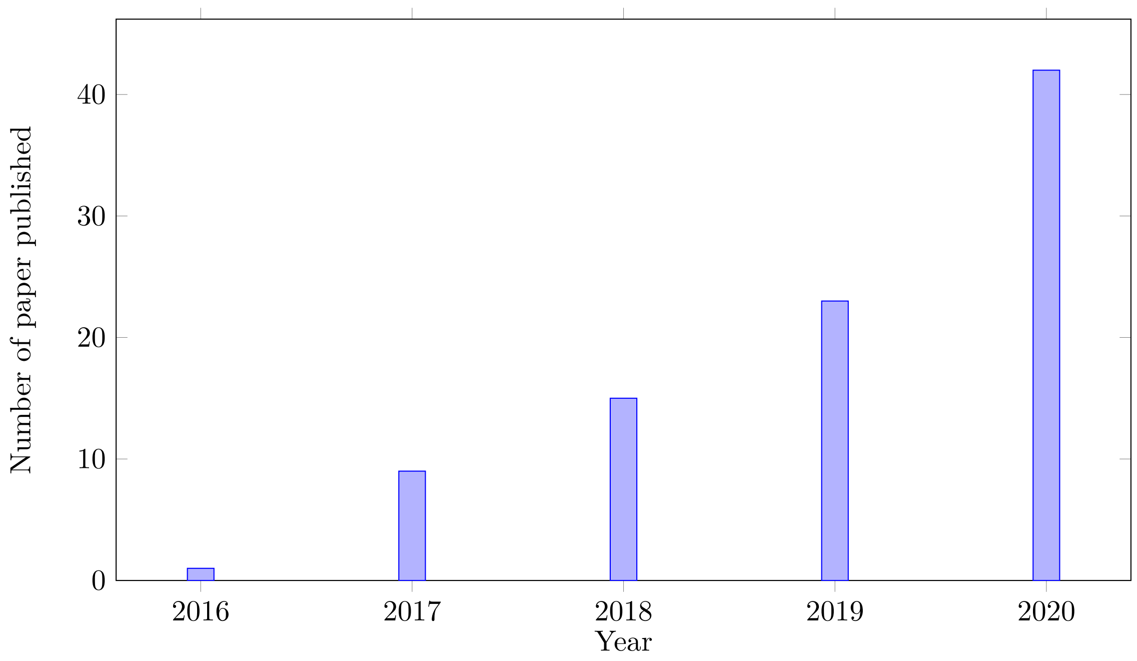

Figure 1.3: Evolution of the number of CL related articles published in major AI conferences (Source : [10]).

access to the data from previous tasks.

This constitutes a very general definition of Continual Learning, and its underlying problem can be addressed in various ways. Therefore, multiple PhD dissertations have been dedicated to adapt the training procedure of Deep Neural Networks in order to accommodate the CL constraint. We can notably cite the recent works of Aljundi [11], Lomonaco [12], Lesort [13], and Liu [14]. We can also notice a growing interest for CL research in major AI conferences (See Figure 1.3).

1.2.1 Problem description

Formal definition

In a typical supervised classification setting, the goal is to learn a set of parameters w using an independently and identically distributed (i.i.d.) labelled training dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}$ to predict $p(y^*|w, x^*)$ for an unseen (x^*, y^*) pair. In a CL setting however, members of $\mathcal{D}$ are not i.i.d. but can be considered drawn from different sets $\mathcal{D}_t = \{(x_t^{(i)}, y_t^{(i)})\}$. Each of these distributions is called a task. The goal is to be able to perform well on every tasks after training sequentially on the data of each task individually.

Let us consider that we have previously learned $T-1$ tasks with parameters $w_1, ..w_{T-1}$ after each training update. Noting L the loss and f our model, we want to solve the following optimization problem to find the parameters w_T for the T^{th} task

$$\begin{aligned} \underset{w}{\arg\min} \quad & L(f_w, \mathcal{D}_T) + \sum_{t=1}^{T-1} \eta_t \\ \text{s.t.} \quad & L(f_w, \mathcal{D}_t) \leq L(f_{w_t}, \mathcal{D}_t) + \eta_t \quad && \forall t \in [1, ..T-1] \\ & \eta_t \geq 0 && \forall t \in [1, ..T-1] \,, \end{aligned} \tag{1.1}$$

where the η_t represent the error tolerance that we allow our model to have on previous tasks, which should be as small as possible.

As we can observe, solving such an optimization problem appears fairly challenging because, for a given task T, we cannot compute the constraints, since we have no access to $\mathcal{D}_t$ for $t < T$. On the left side, we have a term corresponding to the plasticity of the network, which is its ability to learn accurately a new task $(L(f_w, \mathcal{D}_T))$; and on the right side, we have a term corresponding to the stability of the network, which defines its ability to retain previous knowledge ($\sum_{t=1}^{T-1} \eta_t$).

Without an intelligent strategy to perform the training, this second term can be overlooked in the absence of data for previous tasks. This will inevitably lead to a drastic decrease in performance on the previous tasks, a phenomenon known in the literature as catastrophic forgetting. The objective of adequately balancing between those two terms is referred to as the stability-plasticity dilemma [15].

Single-head and multi-head learning

Various contexts of CL can lead to different ways of evaluating a CL method. In the literature, the two main performance evaluation criteria are called single-head and multi-head situations. The distinction between both metrics lies in the knowledge of the task at test time.

Specifically, multi-head learning relies on the assumption that when making a prediction on a sample, the algorithm already know from which task the sample is sampled from. By noting $f : \mathcal{X} \to \mathcal{Y}$ the prediction function and $\mathcal{Y}_t$ the classes of the task t, when predicting the class of a sample from task i among T tasks, in single-head learning we have $\mathcal{Y} = \bigcup_{t=1}^{T} \mathcal{Y}_t$ and in multi-head we have $\mathcal{Y} = \mathcal{Y}_i$.

As an example, we can imagine a CL setting which would consist in two tasks with labels $\mathcal{Y}_1 = [y_0, y_1]$ and $\mathcal{Y}_2 = [y_2, y_3]$. After the algorithm has been trained on the whole dataset, when predicting a sample with a ground truth label y_0, a single-headed algorithm has to make a prediction among the classes y_0, y_1, y_2 and y_3, whilst a multi-headed algorithm has only to choose between y_0 and y_1 since we allow it to know that the sample is coming from $\mathcal{Y}_1$.

As one could expect, an algorithm designed for a multi-head goal cannot handle a single-head goal in a straightforward way. Hence, it is important to take into account for which metric each approach has been optimized, as otherwise the performances can drastically drop (as suggested in [16]).

Desiderata for practical implementation

In order to make such algorithms suitable for practical use, the community has come up with a list of characteristics that illustrates the behavior of an ideal CL method:

- **Limited memory consumption**: Since one of the main purpose of CL is to avoid the storing of large amount of data, it is legitimate to expect that a CL algorithm can limit its impact on memory when learning a new task. This can be enforced by setting a bound on data storage or on the network expansion.
- **No task boundary**: In the case where the data is not available by batches from only one task and is also not independently and identically distributed like in conventional training, the algorithm must be able to update itself accordingly.
- **Forward transfer**: The training process on new tasks should benefit from previously acquired knowledge to perform better than only training on each task independently.
- **Backward transfer**: One could also expect that acquiring knowledge with subsequent tasks should improve the performances on previous tasks.
- **Task size independence**: The algorithm should be able to perform in various conditions, in particular when the number of classes in a task can be as low as only one class per task.

Designing an algorithm that is able to meet all the previous requirements is an arduous challenge and is clearly out of the scope of our study. We will essentially focus on the memory consumption criteria, since our motivation is to produce an algorithm that could be used on embedded devices, and on the task size criteria to ensure that our algorithm is able to deal with tasks consisting of a single class only.

1.2.2 Relation to cognitive science

Learning, the ability to acquire new knowledge and adapt a behavior as a result of experience, is a vital skill for animals. The organ responsible for such a task is the brain. It

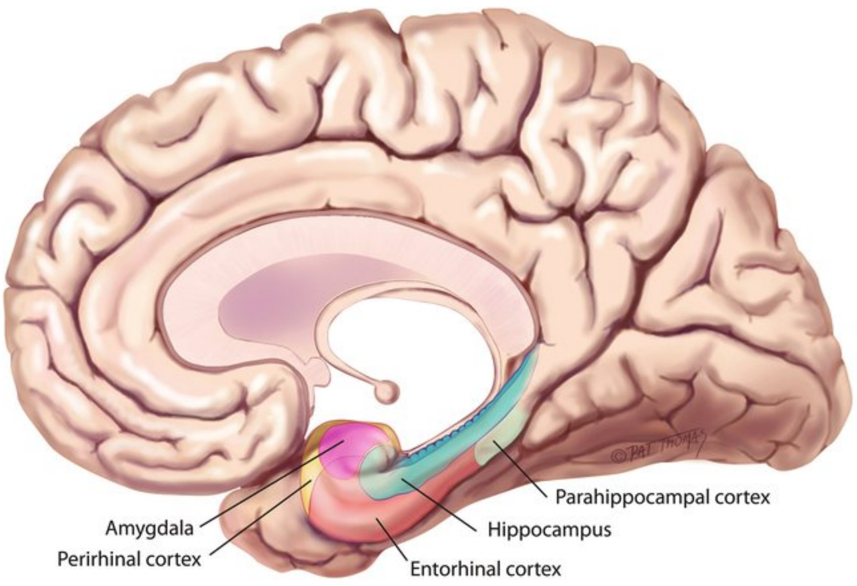

Figure 1.4: The Medial Temporal Lobe is the area of the brain responsible for memory formation (reproduced from [17])

is the outcome of hundreds of millions of years of evolution. It has therefore always been a source of inspiration for intelligent algorithms. We will subsequently present a quick overview of some aspect of this complex biological structure that would allow us to draw some analogies with our approach.

In his seminal work on memory, Tulving proposed to differentiate between episodic memory, which would store information about temporally dated episodes and their relations, and semantic memory, which would contain facts, ideas, meaning and concepts about our general knowledge of the world [18]. These two types of memory fall under the category of declarative memory, which is the memory that can be consciously retrieved. On the other hand, non-declarative memory is acquired and used unconsciously, and can affect behaviour or produce spontaneous thoughts.

In the cognitive science community, experimental evidences have shown that the hippocampus and the surrounding Medial Temporal Lobe (MTL) structures (See Figure 1.4) are the parts of the brain responsible for storing new information and managing short term memory [19]. This observation comes from studies of patients with hippocampal damage who had profound deficits in acquiring new information about their daily lives and experiences but could still remember old memories [20].

Contemporary models of memory formation consider that new experiences are indeed initially encoded in both hippocampal and cortical networks but that the dedicated cortical connections are unstable at the beginning. Successive coordinated reactivation of the hippocampal and cortical connections will lead to a progressive consolidation of the cortical connections that will eventually become independent of the hippocampal connections. This explains the formation of long-term memories unaffected by hippocampus impairments.

While the hippocampus and MTL are actively gathering information during wakefulness, it has been observed that they shared reactivation patterns with cortical areas during sleep time. More precisely, in the case of declarative memory, this happens during a specific sleep phase called Slow Wave Sleep (SWS) that can last between 20 and 40 minutes and is repeated for each sleep cycle. On average, normal subjects spend between 5 and 20% of their total sleep time in SWS, depending on their age and their gender [21].

By analogy, in the CL community, we can consider the learning of a classification task as an update of the semantic memory. That way, we can see the hippocampus and MTL structures as a temporary memory storage for data from one or a few classes. The learning phase of a new batch of classes would be an implementation of the consolidation operated during the SWS phase.

1.2.3 Common benchmark datasets

The most common scenario for evaluating CL algorithms consists in training the model on the sequence of tasks one after the other. After the learning is completed on one task, we compute the accuracy of the model on a test set with all the previously encountered tasks.

We will now present the two datasets that we will use throughout our study and discuss other options. Note that even though it would be possible to adapt any classification dataset to the task of CL, it is important to carefully select the right datasets in order to ensure fair comparisons with preexisting works and avoid datasets that would generate irrelevant difficulties for the current state of CL, since it is already a quite challenging task (as observed in [22]). In particular, datasets that would produce tasks with too small amount of data are out of the scope of this study.

Modified National Institute of Standards and Technology (MNIST)

This dataset consists of 70,000 black-and-white images of size 28×28 pixels. There are ten classes representing each of the ten digits. When the full dataset is available, it is straightforward to achieve very good results, even with simple machine learning approaches. However, training a model in a continuous fashion is still very challenging on this dataset, hence the high number of benchmarks available in the literature.

Canadian Institute For Advanced Research (CIFAR)

CIFAR-100 is a dataset consisting of 60,000 images of size 32×32 pixels. There are 100 classes, each of them containing 600 images. The categories of these images include animals, flowers, fruits, vegetables, trees, household devices, and vehicles.

Other datasets

Finally, we discuss a couple of datasets that may be encountered in CL literature, but that will be discarded from our study.

Permuted MNIST is a rather common benchmark for CL algorithms. It was introduced in [23] where the authors propose to train a model on MNIST and consider that each subsequent task consist in a permutation of the pixels of each digit. Hence, for each new task, a new random permutation is drawn and applied to all images from that task.

Despite its popularity and its simplicity, this benchmark does not appear to be relevant for evaluating CL methods for real world applications, as noticed in [22]. The limitation comes from the observation that permuting pixels does not induce an ambiguity between different tasks. In real-life situations, the distinction between two classes is more complex than just a different ordering of the data.

Continuous Object Recognition (CORe50) has been introduced in [24]. The data was gathered by filming 50 objects at 20 frames per second, for 15 seconds each, with a

Microsoft Kinect sensor producing RGB-D frames of size 128 × 128 pixels.

Each object comes from one of ten categories: plug adapters, mobile phones, scissors, light bulbs, cans, glasses,balls, markers, cups or remote controls. Each object was recorded in 11 different sessions, which defines 11 different backgrounds.

We will not provide experimental results for this dataset because the diversity of represented objects is very low, which is very advantageous for specific CL algorithms, namely exemplar-based ones.

ILSVRC-2012 is also sometimes used to illustrate the efficiency of CL methods over a very long sequence of tasks (ILSVRC-2012 features 1,000 classes). However, in these cases, the authors usually choose to evaluate their approaches on batches of 100 classes to avoid drastic loss in performance and to ensure that enough data is available to learning anything properly. This behaviour makes comparison with the one-class approach pursued in our work unfair. Furthermore, as our method will rely on a feature extractor pretrained on this dataset (for the most part), it does not make sense to evaluate its CL ability on the same dataset.

1.3 Relevance to industrial needs

Whilst the rise of DL has sparkled considerable interest from software technology giants such as Google for text translation, or Facebook for image recognition, the demand is also growing for a variety of actors with very specific needs. The domain of CL is no exception for this trend: we can already envision cutting-edge applications of this field for some of the most flourishing industries.

1.3.1 Self-driving cars

Since 2017, Waymo (a subsidiary of Alphabet Inc.) has been testing its autonomous cars in the city of Phoenix, Arizona, USA. Over the almost 10 million kilometers of automated driving, Waymo reported 18 cases of events involving contact with another object (none of which resulted in severe injuries [25]). The Chinese company Baidu, with its fleet of

300 autonomous vehicles, announced that it achieved over 2 million kilometers across 13 cities. Cruise (a subsidiary of General Motors) has also recently been experimenting its self-driving cars in the streets of San Francisco, California, USA, whilst Tesla has released a beta version of its software for full self-driving to a selected cohort of users.

The sector of autonomous driving is a fast-paced industry with colossal financial stakes. Beyond the technical challenges to overcome to make this technology largely accessible, public authorities are also concerned about the safety of such endeavour. Relying on an external source to make a software operate implies security risks that urgently need to be taken care of. This is typically relevant when an update of the software occurs: such software modifications from the outside should be sparse to avoid any malicious attack that would compromise the safety of the vehicle. The software should instead rely its own sensors (as we can see in Figure 1.5) to directly gather information from the outside world and update itself accordingly. Those local modifications are critical for optimal performances since each vehicle can evolve in very different contexts (weather, lighting, road infrastructure) but all situations cannot be taken into account during simulations.

For these reasons, CL appears as a necessary mechanism to ensure a safe experience for users of autonomous vehicles.

1.3.2 Healthcare system

The use of AI-powered software in healthcare environments has increased steadily in recent years for various applications, including diseases diagnostics and prediction, living assistance, biomedical information processing, and biomedical research [28].

Currently, such software has to pass a series of tests in order to be approved for clinical applications. This is a mandatory procedure to ensure that a software used with real patients in real situations is reliable in regard to medical standards. However, these tests can take several months and any modification implies a new evaluation. This setting is particularly unsuited for AI algorithms, since they can usually provide better results when provided with more data (new manual grading by medical professionals or new outcomes over time) as explained in [29].

In light of these limitations, the FDA has released a new preliminary guidance on what

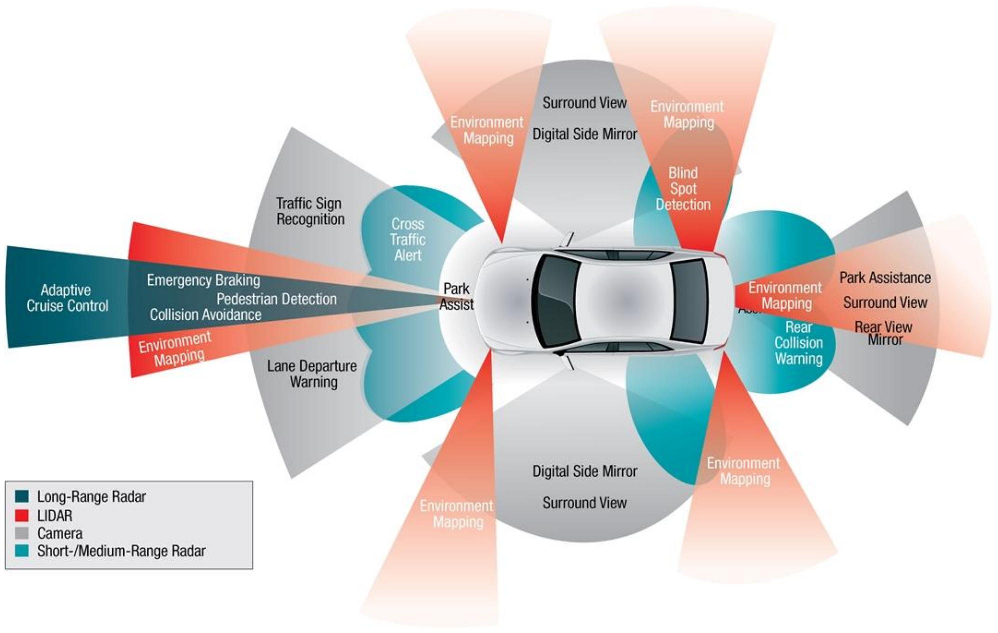

Figure 1.5: Various sensors continuously collect data from all sources and all directions to operate an autonomous car (reproduced from [26]). Typical autonomous cars do not rely solely on motion sensors because they can be unreliable in some situations (bad weather, distance between cars). Autonomous cars also heavily rely on image sensors that are able to identify textures and can be more reliable to noise affecting motion sensors.

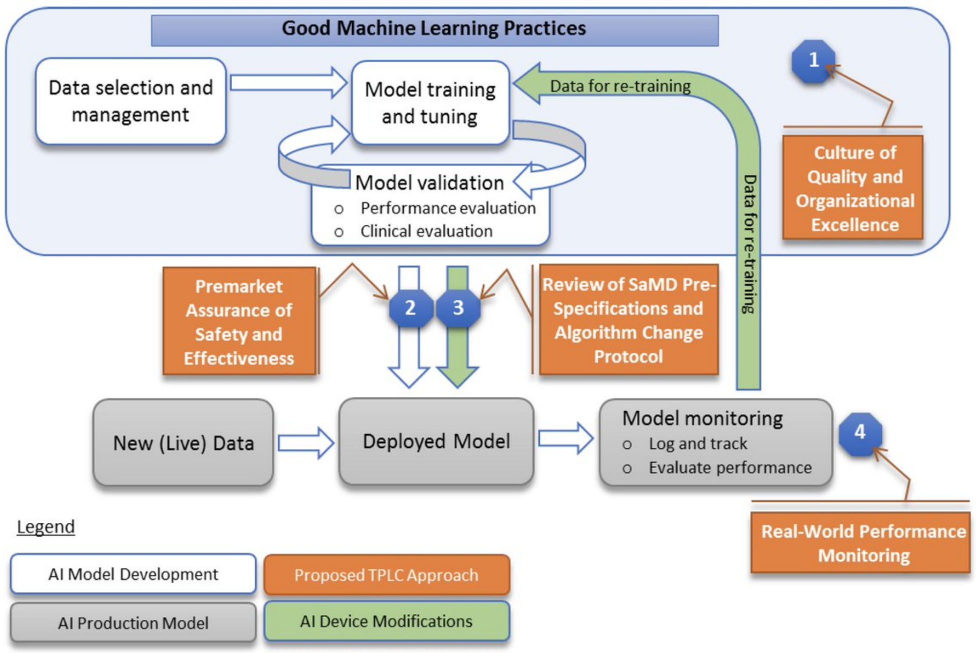
Good Machine Learning Practices
Data selection and management
Model training and tuning
Data for re-training
Model validation
Performance evaluation
Clinical evaluation
Data for re-training
1
Culture of Quality and Organizational Excellence
Premarket Assurance of Safety and Effectiveness
2
3
Review of SaMD Pre-Specifications and Algorithm Change Protocol
New (Live) Data
Deployed Model
Model monitoring
Log and track
Evaluate performance
4
Real-World Performance Monitoring
Legend
AI Model Development
Proposed TPLC Approach
AI Production Model
AI Device Modifications

Figure 1.6: The total product lifecycle regulatory approach in healthcare proposed by the Food and Drug Administration (FDA). We can notice that if the data for retraining (green arrow) can only be stored for a limited time then this system can be identified as a CL environment (from [27]).

they call a “total product life cycle regulatory approach” (see Figure 1.6) for continuously updating AI clinical software [27]. It would allow future AI algorithms and their self-updated versions to be certified without having to pass new tests. Although it would be theoretically possible to retrain from scratch such algorithms with more data over time, this operation would be very costly and may not be compliant with patient data protection laws such as the American Health Insurance Portability and Accountability Act.

Hence, CL algorithms appear as a perfect fit for this kind of application by reducing the computational cost and ensuring the privacy of patient data. In the long term, CL may lead to efficient personalized medicine where each patient is associated with his own algorithms continuously trained with his own data.

1.4 Thesis outline

This thesis will present our work toward more efficient methods of training neural networks that are compliant with CL requirements. That is to say, we are interested in algorithms for training neural networks on a sequence of tasks, each task to be trained solely with its own data; the goal is to ensure that the learning of a new task does not interfere with the knowledge acquired on previous tasks.

We will notably investigate the challenging case of single class incremental CL, where data is fed one class at a time: each task consists of only one class. The objective being to be able to discriminate between several classes learned one by one.

In Chapter 2, we will present an extensive review of the current state of the art in CL. This study will allow us to discuss the limitations of each approach of the literature.

In Chapter 3, we will introduce the theoretical core of our approach. We will present the framework of Invertible Neural Networks from a general standpoint, before showing how to apply them in the context of CL in the next chapters.

In Chapter 4, we will detail our main algorithm: OvA-INN. In this first set of experiments, we will rely on a feature extractor pretrained on external data. The performances will be compared with state-of-the-art methods on standard vision datasets.

In Chapter 5, we will show a way to improve the memory efficiency of OvA-INN to make it more compatible with embedded systems.

In Chapter 6, we will propose to apply OvA-INN in a setting without external data and show that, even in this regime, it can improve the performance over standard approaches.

We will finally conclude the thesis with a discussion on the meaning of our results and provide some insights for future research directions.

1.5 List of publications

Peer-Reviewed International Conference Proceedings

- Guillaume Hocquet, Olivier Bichler, and Damien Querlioz

  OvA-INN: Continual Learning with Invertible Neural Networks.

  In IEEE 2020 International Joint Conference on Neural Networks (IJCNN)., doi: 10.1109/IJCNN48605.2020.9206766, preprint available at `https://arxiv.org/abs/2006.13772`.

  The conference, initially planned in Glasgow, UK, was held virtually in July 2020.

- Guillaume Hocquet, Olivier Bichler, and Damien Querlioz

  Memory Efficient Invertible Neural Networks for Class-Incremental Learning.

  In 2021 International Conference on Artificial Intelligence Circuits and Systems (AICAS)., doi: 10.1109/AICAS51828.2021.9458549.

  The conference, initially planned in Durham, North Carolina, USA, was held virtually in June, 2021.

Chapter 2

Background on Continual Learning approaches

The research community has been very active in the Continual Learning (CL) field for the past five years and has proposed a variety of approaches to tackle this challenge from different viewpoints. In this chapter, we will cover the spectrum of the different approaches.

As was already observed in the previous chapter, the definition of CL is very broad. In the interest of clarity, we will focus our review of the literature on algorithms that are closely related to our current objective. In particular, we suppose that each task is constituted of a batch of several labeled samples that we can iterate on. By doing so, we will not consider "no task boundary" or "open world" approaches [30–32].

2.1 Previous works

For each path of research, we will present its pros and cons and review the theoretical aspects of the most popular methods. In particular, we will pay special attention to their limitations and to the meaning of their assumptions.

2.1.1 Regularization-based approaches

Regularization-based approaches aim to mitigate the catastrophic forgetting effect by applying a constraint on the loss when learning new tasks. This constraint is designed to prevent the weights of the updated network from becoming too different compared to the weights of the previous network. The idea is that if the shift necessary to learn a new task is as small as possible, the impact on the network should be negligible, and the network should be able to perform almost as well as before being updated with a new task. A large variety of such constraints have been proposed, we will present the most popular ones.

A simple constraint consists in computing the prediction of a pretrained network on the new data, keeping the results only for the previous tasks and penalizing when the prediction from the updated network on previous tasks deviate from the previously computed results. This constraint is known as distillation loss and was proposed in “Learning without forgetting” [33].

For k previously learned tasks, noting $\hat{y}_o$ the predictions of the current network for the old tasks, $\hat{y}_n$ the predictions of the current network for the new task, y_o the predictions of the previous network for the old task and y_n the ground truth of the new data, the goal is to minimize:

$$\mathcal{L}_{dist} = -y_n \log \hat{y}_n - \sum_{i=1}^{k} y_o \log \hat{y}_o. \tag{2.1}$$

As we can see, the new network is only required to memorize the predictions of the previous network to compensate the penalty term.

In order to improve the constraint, a more sophisticated version consists in not only recording the output but also the attention maps provided by Grad-Cam [34]. Those attention maps indicate which input pixels are responsible for the prediction. This has

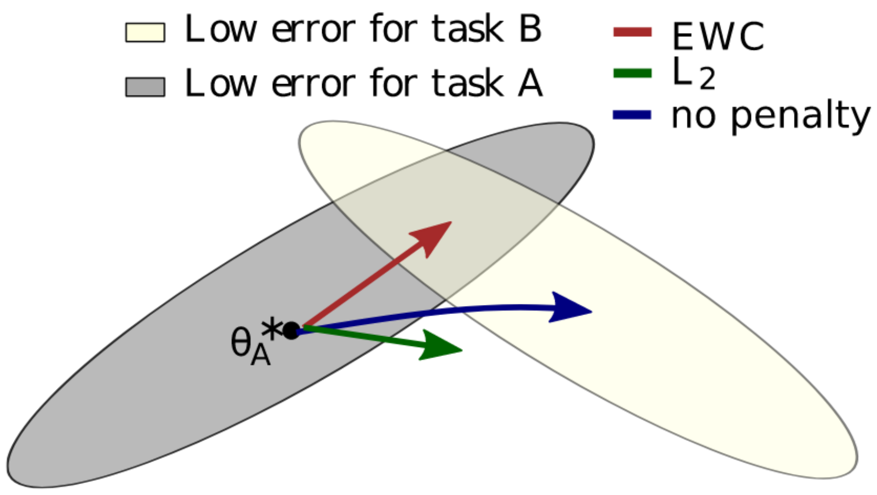

Figure 2.1: In regularization-based approaches, the parameters θ are updated from θ_A^* and are constrained to stay in the domain of both tasks (red arrow). With a simple L2 penalty, the parameters are constrained to stay close to θ_A (green arrow). Whilst with no penalty, the parameters would completely forgotten the previous task (blue arrow) (reproduced from [36])

been proved even more efficient in "Learning without Memorizing" [35].

Other approaches are directly applying a constraint on weights deviation. The idea is to find which weights have a low impact on the loss of previous tasks and which weights have a big impact.

For k previously learned tasks, noting $\mathcal{L}_{k+1}$ the loss on the current task, $\theta^{(k)}$ the parameters trained on the previous k tasks, θ the current parameters and $\omega^{(k)}$ the importance of parameters, the loss to optimize is:

$$\mathcal{L}_{imp} = \mathcal{L}_{k+1} + \sum_i \omega_i^{(k)} \left(\theta_i - \theta_i^{(k)}\right)^2 \tag{2.2}$$

Noting q the output of the model and g_{θ_i} the gradient of the loss with respect to the parameter θ_i, various approaches have been proposed to estimate ω_i:

- Elastic Weight Consolidation [36] proposes to compute an approximation of the Fisher information matrix

$$\omega_i = \mathbb{E}_x \mathbb{E}_y [g_{\theta_i}(x, y)]. \tag{2.3}$$

- Memory Aware Synapses [37] proposes to rely on the sensitivity of the last layer

$$\omega_i = \mathbb{E}_x \left[\left| \frac{\partial \|q(x)\|^2}{\partial \theta_i} \right| \right]. \tag{2.4}$$

- Synaptic Intelligence [38] evaluates the importance of a parameter by computing each parameter's contribution to the change in the total loss between time 0 and T

$$\omega_i = \sum_{t=0}^{T-1} g_{\theta_i}^t \cdot (\theta_i(t+1) - \theta_i(t)). \tag{2.5}$$

Although those three approaches are based on different motivations, it has been shown theoretical and empirically in [39] that they are actually very similar and can be unified in a common framework.

As can be noticed in Figure 2.1, those methods can only be effective if one can find an overlap between low error spaces of different tasks. Classifier-Projection Regularization [40] proposes to add a regularization term to the importance based losses to promote wide local minima. This way, the low error spaces are more likely to overlap for subsequent tasks.

To conclude on regularization-based approaches, we would like to emphasize that those methods have been originally designed for multi-head CL, when the task identifier is known at inference (see Section 1.2.1). Hence, they usually provide very limited results on single-head CL [41].

In the next paragraph, we will present methods that build on these previous metrics to overcome catastrophic forgetting in the case of single-head CL.

2.1.2 Exemplar Rehearsal methods

Rehearsal techniques rely on the storage of a subset of previous data in order to mitigate the catastrophic forgetting effect. In those approaches, the quality and quantity of stored data, which are called exemplars, can have a strong impact on the final performances. For these reasons, those approaches are usually compared for a fixed size of data storage.

Although some work has been proposed for exemplar selection [43], it remains very

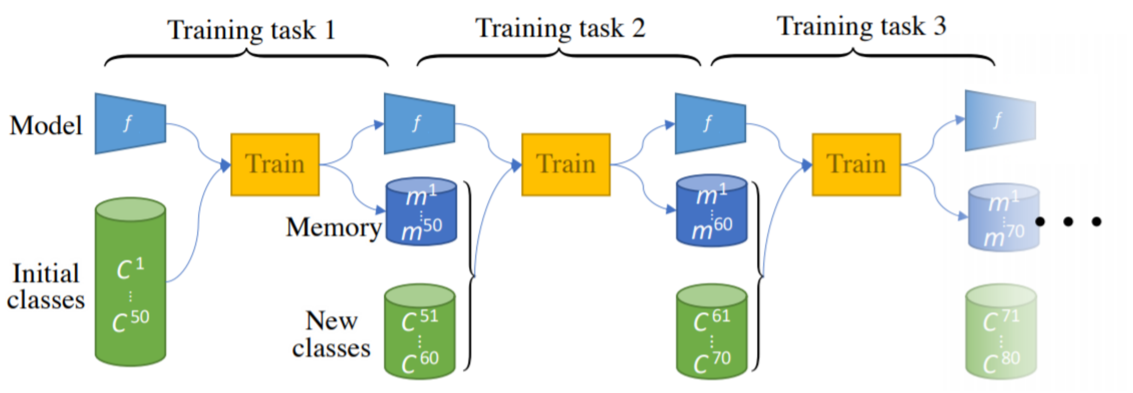

Figure 2.2: In exemplar rehearsal approaches, the network f is updated with all the data from the current task, noted C, and with a subset of the data from previous tasks, noted m. (from [42])

difficult to design a general procedure to choose which sample to keep and which to delete. The most appropriate choice can indeed depend on the data or the learning algorithm. That is why comparison between exemplar-based methods are often conducted with random sampling [44].

Incremental Classifier and Representation Learning (iCaRL) [45] proposes to use a distillation loss on exemplars predictions and a cross-entropy loss on the new data predictions.

iCaRL avoids the need of a classification layer by relying on a nearest-mean-of-exemplars approach. This process consists in computing the mean of the output from exemplars for each class y after a training phase on a network f

$$\mu_y = \frac{1}{|\mathcal{D}_y|} \sum_{x \in \mathcal{D}_y} f(x), \tag{2.6}$$

and then to assign the class label from the closest μ:

$$y^* = \underset{y=1,\dots t}{\arg\min} \|f_y(x) - \mu_y\|_2^2. \tag{2.7}$$

Pooled Outputs Distillation Network (PODNet) [42] proposed to apply the distillation loss not only on the output, but also after each layer of the network. This improvement indeed appeared very efficient in the single class incremental learning scenario.

End-to-end IL [46] is similar to iCaRL but showed that a classification layer could be used with a cross-entropy loss by carefully training the network on each task. Namely, the authors of this technique first update the network with all available data (stored old data and all new data) and then perform a second training phase with the old data and a subset of the new data to enforce a balance between old and new data. This fine-tuning phase is executed with a small learning rate and has proven useful in their ablation study. Other additional methods have been proposed in Rebalance [47] to counteract the imbalance between previous and new data.

Rather than relying on distillation, Gradient Episodic Memory (GEM) [48] proposes to impose a constraint on the previous losses to prevent them from increasing. This can be formulated by an optimization problem that can be solved using the stored exemplars.

The combination of two losses at the same time (optimizing the performances on new data and ensuring that these modifications have a negligible impact on previous performances) can lead to undesirable optimization behavior if one loss is more important to optimize than the other. In TPCIL [49], the authors observe that, in the previously proposed exemplar-based approaches, the first epochs are more focused on learning the new data than preventing modifications to the previous network. They propose to implement a smoother transition when learning a new task by relying on a graph that models the feature space.

Variational Continual Learning [50] is an approach based on Bayesian inference. The authors derive a regularization term to alleviate the catastrophic forgetting effect when learning subsequent tasks. This regularization term can be computed without stored exemplars but can also benefit from them if available. Although this approach appears to perform well on small datasets with fully connected networks, there is still work to be done to adapt Bayesian inference to more sophisticated layers that would be essential to tackle more interesting computer vision problems.

Although CL is still challenging for rather small datasets, a direction of research has been dedicated to scale up these methods to higher number of classes per task [51]. In particular, the authors of Bias Correction (BiC) [52] noticed that the imbalance problem between old and new data can be even more problematic in the case of tasks involving tens of classes.

The authors of BiC also proposed a two-step learning procedure. The first step is the usual training with stored old data and new data. The second step is performed on a held out set of old and new data, its goal is to fine tune a linear transform on the output logits of the new classes to improve generalization performances on this small validation dataset. As one can notice, this approach requires a significant number of samples in order to construct a sufficiently consistent validation set. That is why it is only relevant for tasks with several classes.

On a final note, after having presented a variety of approaches that take advantage of exemplars to overcome catastrophic forgetting, we would like to emphasize that these results should be taken with caution, since it has been shown, in a recent paper, that it is possible to reach comparable performances without any continual adaptation! This is achieved by GDumb[53], a very simple heuristic that retrains from scratch the network with only the data from the new class and the exemplars. Such a result illustrates that the previous approaches do not seem to provide any advantage over a simple retraining.

We can see that exemplar rehearsal methods have been extensively studied and their comparative performances have been assessed in a variety of scenarios [41]. However, for many applications, these approaches cannot be satisfactory: it will often not be possible to allow the storage of data because of privacy issues or memory limitation.

In the next paragraph, we will dive into a set of methods that are able to perform rehearsal without storing real data.

In our experiments, we will compare our results to those of iCaRL, GEM, End-to-end IL, Rebalance and PODNet.

2.1.3 Pseudo-rehearsal approaches

A promising research direction for CL consists in relying on generative models to produce artificial samples of old data to update a network on a new task, a procedure called

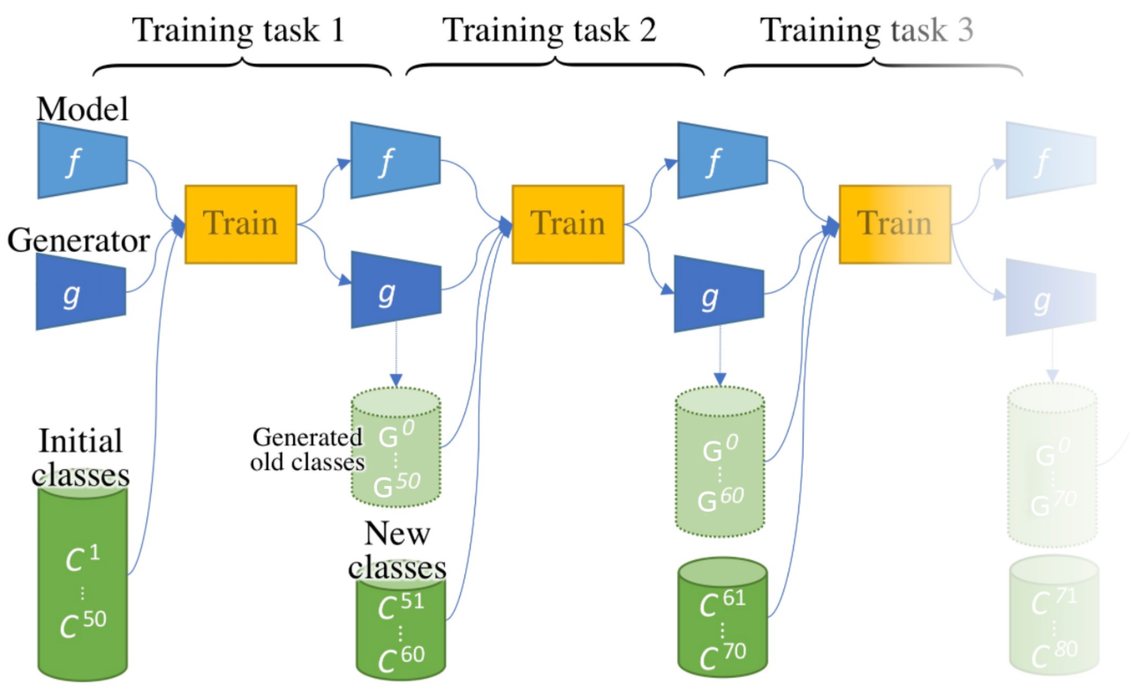

Figure 2.3: In pseudo-rehearsal approaches, a GAN composed of a discriminator f and a generator g is updated with all the data from the current task, noted C, and with data of previous tasks sampled with g, noted G. (from [54])

pseudo-rehearsal.

Generative Adversarial Networks (GANs) are a commonly used generative models for images and have naturally been used for the CL of image datasets. The idea is to train a GAN alongside a classification network. When given a new task to learn, the GAN is used to generate labeled samples of old data that will be added to the new data. With this new dataset consisting of data from all the task seen to the point, it is possible to update the CL network and the GAN to be respectively able to classify data from all the tasks and generate data from all the tasks.

The authors of Deep Generative Replay (DGR) [55] and Parameter Generation and Model Adaptation (PGMA) have proposed approaches to update continuously a GAN, but their results are limited to simple datasets such as MNIST, SVHN and CIFAR-10. It is indeed a challenging task to ensure the stability of a GAN training phase, especially with complex data and in a CL setting.

MeRGAN [56] further improves on the previous approach using a conditional GAN to generate images from the LSUN dataset [57]. But the authors of this approach provide results for only four sequential classes from this complex dataset. Even Lifelong GAN [58] which uses a sophisticated Bicycle GAN architecture is only evaluated on five classes for complex datasets.

Whereas those approaches have only been evaluated on their ability to generate data, the authors of Parameter Generation and Model Adaptation [59] proposed to evaluate those generative models in a CL experiment. They trained a GAN to sequentially learn several tasks with the MNIST and CIFAR-10 datasets. But their results did not match those from real exemplar based methods such as iCaRL [45].

It seems in fact very unlikely that a GAN, as sophisticated as it can be, could capture all the variety and complexity of a natural dataset. However, this ability is crucial to ensure an efficient learning for the subsequent tasks. The consequence of this lack of quality in data generation over time has been highlighted in [60], where the authors used a GAN to generate data along with a buffer of natural old exemplars. They showed that even a small buffer of exemplars could significantly boost the performance.

A compromise between pseudo-rehearsal and regularization-based approaches has been proposed in a model called Generative Feature Replay (GFR) [61]. The idea consists in preventing the forgetting in the feature extractor using a distillation loss, while the classifier is updated with feature samples generated by a GAN. Likewise, FearNet [62] uses a fixed feature extractor and produce feature samples with an autoencoder.

In our experiments, we will compare our results to those of DGR, PGMA, GFR and FearNet.

2.1.4 Expandable models

All previously presented methods have considered a fixed model size. Some could have increased memory usage for exemplars or data statistic storage, but all of them used a fixed number of model parameters along their continual training. However, it seems only

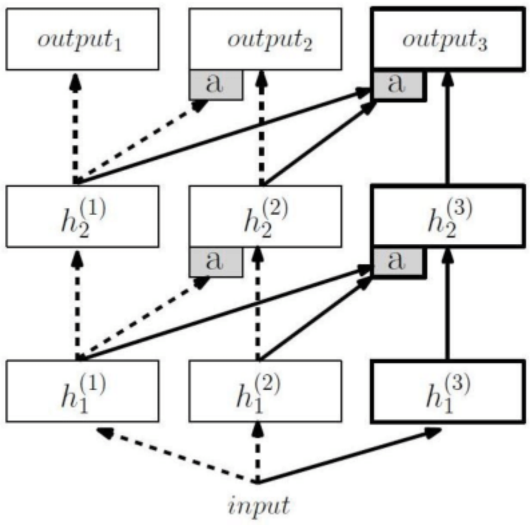

Figure 2.4: Progressive neural networks are an architecture designed to learn several tasks by adding a new stack of layers connected with previous layers each time a new task is presented in a multi-head setting. The parameters unrelated to the new task stay fixed (dotted arrow), and intermediate layers noted a are added to ensure dimensional reduction. (from [63])

natural that the more tasks a model learns, the more complex it must be to maintain its performance.

Consequently, Progressive Neural Networks [63] have been proposed as a simple way to grow a network with the number of tasks. The idea is to add new layers connected to the previously trained layers. The new layers can be trained on the new task but the previous layers remain fixed (See Figure 2.4).

Dynamically Expandable Network [64] performs a retraining of some of the parameters of the network by exploiting task relatedness, while increasing its number of parameters when necessary to account for new knowledge required when learning new tasks.

These approaches have been used in a multi-head setting, which is not as constraining as a single-head setting. In this second scenario, Random Path Selection for Incremental Learning (RPSnet) [65] proposes to add new modules for each task and freeze parts of the network that will not be updated anymore. RPSnet also relies on exemplars to operate. But its training cost is significantly higher than typical exemplar-based approaches in order to accommodate the training of new modules and the choice of these modules.

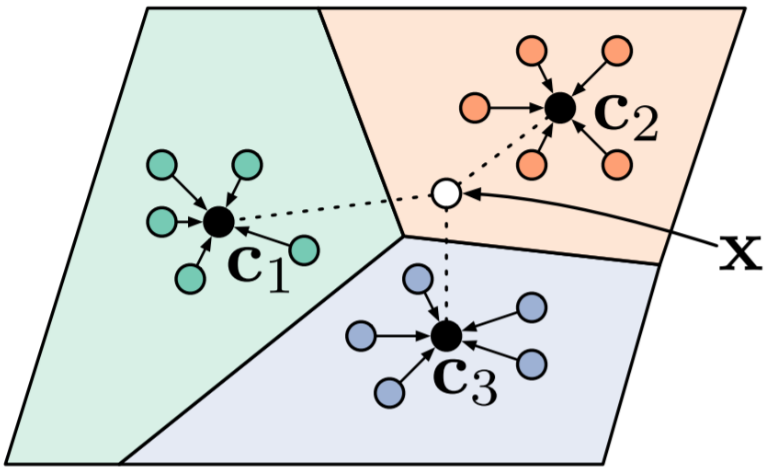

Figure 2.5: Each dot corresponds to the output of the prototypical network for a given data point. The prototypes c_k are defined as the means of the output of the data from each class k (each class is represented by a color). Predicting the class of a new sample x consists in finding the closest prototype c_k. (from [66])

We can observe that expanding the size of a network to allow it to acquire additional knowledge is not a straightforward operation in the single-head setting. Conversely, in the next paragraph, we will present methods that try to update as little as possible the network when presented with a new task to learn.

In our experiments, we will compare our results to those of RPSnet.

2.1.5 Distance-based models

The idea of distance-based models consists in learning an embedding network that will be used to project samples into a low dimensional space where the similarity in this embedding can be used to predict the label of a new sample.

This type of networks is commonly used in a related field called few-shot learning, where samples from new classes have to be matched with other classes from a small set. The challenge of few-shot learning is to design networks that are able to work with very small amount of data from new classes. In such a setting, updating the network for each new class is not effective, hence the idea to rely on fixed embedding networks.

Matching Networks [67] embed a sample $\hat{x}$ into a low dimensional space, in which the predicted label $\hat{y}$ can be obtained by performing a generalized form of nearest-neighbours classification following:

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i, \tag{2.8}$$

where $(x_i,\ y_i)$ is a labeled samples from class i and $a(\cdot,\cdot)$ is an attention kernel. In the paper, the authors propose a simple kernel:

$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}), f(x_i))}}{\sum_{j=1}^{k} e^{c(f(\hat{x}), f(x_j))}}, \tag{2.9}$$

where c denotes the cosine similarity.

Subsequently, Prototypical Networks [66] have been proposed as a simplification of the previous approach. It is based on the observation that samples from a given class tend to form a cluster in the embedding space. After learning the embedding network, they compute the mean of the data output for each class, which they call a prototype (similarly to iCaRL in equation 2.6). Classification is then performed by finding the nearest prototype (see Figure 2.5).

In our experiments, we will compare our results to a nearest-neighbours baseline.

2.2 Comparisons between the different approaches

In the previous paragraphs, we have observed that regularization-based approach and expandable models provide a satisfying solution for the multi-head CL setting but are very limited in a single-head setting. Updating model parameters without any samples from previous class always results in some form of overfitting on new data. A simple fix and very efficient solution consists in storing a few samples from previous data as proposed by exemplar-based approaches. This way, the parameters of the whole model can be safely updated as long as the device capacity allows to store enough data to perform the rehearsal procedure. Pseudo-rehearsal approaches have been explored to avoid the storage of data and still allow rehearsal by generating samples of previous data but the results are still limited because this method can lead to very unstable learning phases that will not provide reliable sampling.

We have also noticed that distance-based models proved that it is not always required to make significant changes to a model to make it usable on new tasks, nor it is required to store previous data. While the current state-of-the-art methods in distance-based models are usually focused on learning simple distances for few-shot learning, we can foresee a potentially efficient strategy for single-head CL by learning a more sophisticated distance metric. In few-shot learning, it is indeed assumed that very few data can be used to learn new tasks, but in general single-head CL, we can assume that we have access to a substantially more significant set of data for new tasks, which would be very beneficial for learning a better distance metric.

2.3 The special case of class-by-class learning

Throughout this chapter, we have presented a variety of approaches to alleviate the problem of catastrophic forgetting. We should however emphasize that although these approaches are usually assumed to work for any number of classes per task, they are most of the time not tested in the harder case of single-class incremental learning. As a matter of fact, when provided with data from only one class, learning algorithms are even more prone to the effect of catastrophic forgetting. This setting is indeed not providing any negative data. In such a training regime, artificial neural networks will typically degenerate and blindly return the same answer whatever the input is. This behavior is caused by the fact that with only data from one class, the optimal strategy is to systematically predict this class without even looking at the input data. Also, for a fixed dataset, learning tasks with fewer classes imply learning with less data, which can lead to other learning difficulties such as overfitting.

The ability to update a network with only one more class is however not a simple corner case but a rather highly relevant feature for some applications. Thus, it appears critical to assert the behavior of learning algorithms in this particular situation. This evaluation is even more significant since each approach can perform very differently in a single-class incremental learning scenario. For example, distance-based models are invariant with the number of classes since they do not require negative data during their training phase. On the contrary, regularization-based approaches seem to completely degenerate in the absence of negative data.

Because of these specific issues, the single-class incremental learning scenario has been largely overlooked in recent research efforts. Our work aims to illustrate that we can design very efficient learning processes that are also compliant with single-class tasks.

Chapter 3

Invertible Neural Networks: a model for Continual Learning?

3.1 Introduction

In Section 2.2 of the previous chapter, we reviewed the major state-of-the-art Continual Learning (CL) approaches. We paid special attention to their application in the case of single-class CL and noticed that distance-based models presented promising features to tackle this challenge. In this spirit, in this thesis, we propose to rely on a fixed feature extractor when learning new tasks. This way, we only have to develop a strategy to exploit, as best as possible, the output of this feature extractor.

The main limitation of classical distance-based models is that they rely on simple distance metrics such as Euclidean distance or cosine similarity. In this work, we rather propose to build a sequence of models, one for each class. Each model task will be to compute the probability that a sample belongs to the distribution of its training data. In particular, the training of each model will rely solely on data from a single class.

In order to achieve this, we need a model that is able to construct an estimate of the probability density function of the data for each class. With this estimate, we will be able to predict to which class a given sample belongs to by identifying the class with the highest probability.

A relevant approach for computing density estimation is a particular class of neural networks called Invertible Neural Networks. In this chapter, we will present their properties and various ways of implementing them. More details on their use in a CL application will be exposed in Chapter 4.

3.1.1 Why use Invertible Neural Networks?

In a typical classification problem, models are trained to estimate the parameters of a decision boundary in order to predict the class of a sample. Such models are called discriminative models; they are designed to compute the conditional probability of the target y, given an observation x : $P(y|x)$. In our case, we cannot rely on such models, since the number of classes is continuously growing and we have no access to previous data: it is unfeasible to estimate boundaries.

Instead, our objective is rather to train a model for each class that will output the probability for a sample x to belong to a class y: $P(x|y)$. The function that measures the "goodness of fit" of a statistical model to a sample of data is the likelihood. Hence, the ideal model for our purpose is a model providing exact and efficient estimate of its likelihood on a data sample. This is one of the objectives of generative models.

Deep generative models are an active research field [68] and a variety of approaches have already proved successful for different applications:

- Energy-Based Models [69] have been studied for a long time, but these approaches are difficult to scale on complex datasets.

- Generative Adversarial Networks [70] are a widely adopted architecture for generating convincing looking images, but they do not rely on likelihood estimation.

- Variational Autoencoders [71] are a simple but efficient way of generating data, but they provide only an approximation of the value of the latent variables.
- Autoregressive models [72] provide a way of computing explicit probability densities, but have a high computational cost, as they rely on several iterations to produce a sample.
- Invertible Neural Networks (INNs) are a recently developed framework that is rapidly gaining attention for its ability to provide a tractable access to the exact value of the likelihood.

Therefore, INNs appear as a favorable choice for our purpose. Before diving deeper into the underlying theory of INNs, we will briefly review a few currently investigated applications of those models.

3.1.2 Examples of application of Invertible Neural Networks

Although INNs have only recently been actively adopted, and there is still important ongoing research to improve their performances, we can already find very promising applications of this approach.

INNs have been successfully applied in the context of image super-resolution. While other generative approaches are only trained to provide a single high-resolution image for a given low-resolution input, the INN-based approach, SRFlow [73], is able to generate a multitude of potential high-resolution images. Figure 3.1 presents a striking example.

In the domain of audio synthesis, the FloWaveNet approach [74] has been proposed as a very efficient alternative to popular autoregressive approaches. By relying on the efficient framework of INN, this model can provide real-time parallel audio synthesis without depending on the costly training procedure of other comparable approaches.

INNs have also been exploited in the field of memory-efficient training. They indeed provide a way of propagating the gradient without the need to store the activations of the intermediate layers [75]. These activations can be retrieved during the backward pass using the inverse operation. In this way, the memory cost of the training phase is constant regardless of the number of layers!

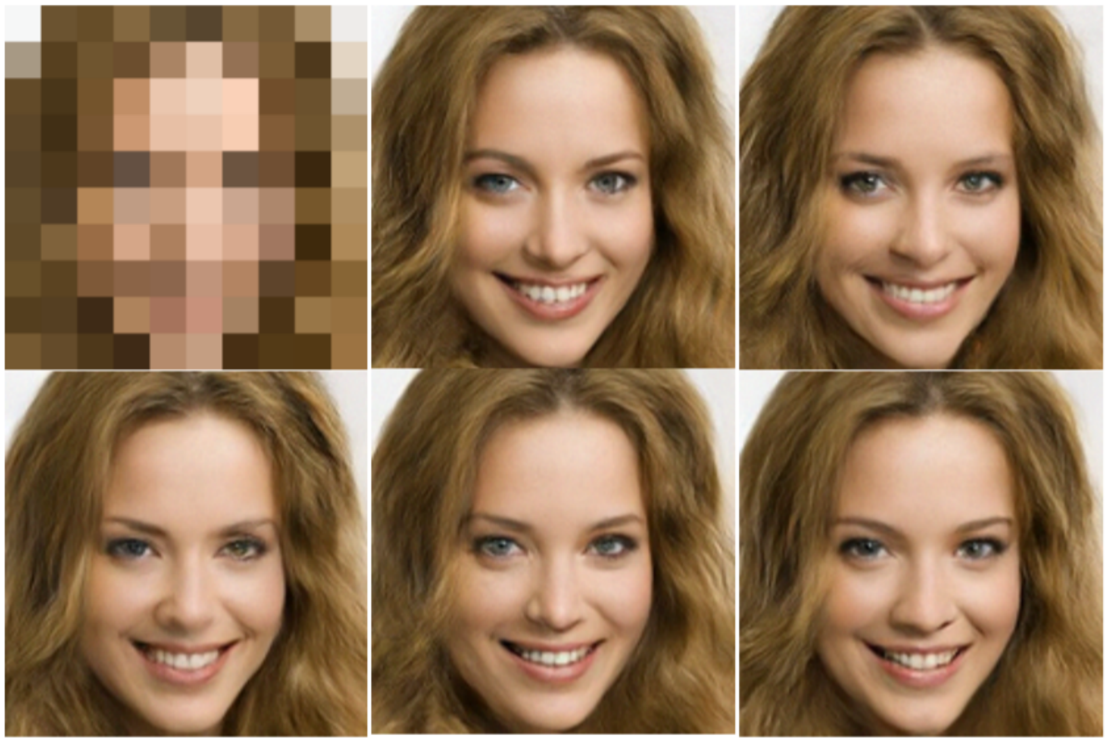

Figure 3.1: By learning the conditional distribution of the output given the low-resolution input, INN open the way to new applications. Here, SRFlow can produce several high-resolution images corresponding to a low-resolution one (top left). (reproduced from [73])

For image classification tasks, the authors of the *i*-RevNet [76] approach demonstrate that it is possible to design a fully invertible Residual neural network (ResNet) with performances approaching the original non-invertible one. Finally, for image generation tasks, the Glow [77] approach was able to produce very convincing images.

3.2 Theory

The goal of INNs is to model complex data in a way that ensures the access to density estimation. In order to achieve this, this type of networks relies on a technique called normalizing flow, which consists in modeling a complex probability density as an invertible transformation of a simple base density. Namely, to compute an estimate of the intractable probability density $p_X(x)$ of a sample data x, the network is trained to transform x into a different representation z in such a way that $p_Z(z)$ is easily tractable and can be used to recover $p_X(x)$.

In the next section, we will provide a few definitions and present some properties that will be useful to understand how INNs work and can be implemented.

3.2.1 Mathematical tools

Jacobian Matrix

Given a function mapping a n-dimensional input vector x to a m-dimensional output vector, $f : \mathbb{R}^n \to \mathbb{R}^m$, the matrix of all first-order partial derivatives of this function is called the Jacobian matrix, J_f where one entry on the i^{th} row and j^{th} column is $J_{i,j} = \frac{\partial f_i}{\partial x_j}$:

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}. \tag{3.1}$$

Determinant

The determinant is a scalar value that is a function of the entries of a square matrix. For a matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$, it is defined by:

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}, \tag{3.2}$$

where S_n is the set of all permutations of the set set $\{1, 2, ..., n\}$ and $\operatorname{sgn}(\sigma)$ is the signature of σ, a value that is 1 whenever the reordering given by σ can be achieved by successively interchanging two entries an even number of times, and -1 otherwise.

Change of variable theorem

Let us consider a random variable X with a probability density function $p_X(x)$ and a bijective mapping f, we are interested in the probability density function of $Z = f(X)$. In the univariate case, it is given by

$$\begin{aligned} p_Z(z) &= p_X(f^{-1}(z))|(f^{-1})'(z)| \\ &= p_X(x)|(f^{-1})'(z))|. \end{aligned} \tag{3.3}$$

In the multivariate case, this result is generalized by

$$p_Z(z) = p_X(x)|\det(J_{f^{-1}}(z))|. \tag{3.4}$$

We can see that the right hand-side involves a term called the Jacobian determinant: $\det(J_{f^{-1}}(z))$. This term can be expressed in a more convenient way

$$\begin{aligned} \det(J_{f^{-1}}(z)) &= \det[(J_f(x))^{-1}] \\ &= \det(J_f(x))^{-1}. \end{aligned} \tag{3.5}$$

The first transformation comes from the inverse function theorem and the second from a property of determinants. By induction, we can easily generalize this result for compositions of bijective functions. Let us consider $f = f_n \circ f_{n-1} \circ .... \circ f_1$, and noting

$x_i = f_i \circ f_{i-1} \circ .... \circ f_1(x)$ with $z = x_n$ and $x = x_0$

$$\begin{aligned} p_Z(z) &= p_{X_{n-1}}(x_{n-1})|\det(J_{f_n}(x_{n-1}))^{-1}| \\ &= p_{X_{n-2}}(x_{n-2})|\det(J_{f_{n-1}}(x_{n-2}))^{-1}||\det(J_{f_n}(x_{n-1}))^{-1}| \\ &= p_X(x)\prod_{i=1}^{n}|\det(J_{f_i}(x_{i-1}))^{-1}|. \end{aligned} \tag{3.6}$$

Kullback–Leibler divergence

The Kullback-Leibler divergence (KL divergence), or relative entropy, is a commonly used measure that quantifies how much one probability distribution differs from another probability distribution. It is always non-negative, with a lower value for closer probability distributions, and is equal to zero if and only if the two distributions are the same almost everywhere. The KL divergence is defined for two distributions p and q as

$$D_{KL}(q||p) = \mathbb{E}_{x\sim q}[\log q(x) - \log p(x)]. \tag{3.7}$$

The KL divergence is an asymmetric measure: $D_{KL}(q||p)$ and $D_{KL}(p||q)$ are not, in general, equal.

Maximum Likelihood Estimation

Maximum Likelihood Estimation is a method used to find the best parameters of a model to represent a population. Noting q the unknown real density and p its estimate, for a model with parameters θ and a population X of size N, it consists in maximizing the following loss:

$$\mathcal{L}(X;\theta) = \frac{1}{N}\sum_{x\in X}\log p(x|\theta). \tag{3.8}$$

This term is actually a sampled approximation of the expected log-likelihood by the law of large numbers:

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim q}[\log p(x|\theta))]. \tag{3.9}$$

Notably, this last expression is linked to the definition of the Kullback–Leibler divergence:

$$\begin{aligned} D_{KL}(q||p) &= \mathbb{E}_{x\sim q}[\log q(x)] - \mathbb{E}_{x\sim q}[\log p(x|\theta)] \\ &= \mathbb{E}_{x\sim q}[\log q(x)] - \mathcal{L}(\theta). \end{aligned} \tag{3.10}$$

As can be observed, the first term in the right hand-side is independent of θ, so we can conclude that maximizing the log-likelihood is equivalent to minimizing the KL-divergence between the estimated density p and the real density q.

3.2.2 Normalizing flow principle

To allow the computation of exact and efficient density estimation, INNs follow a particular architecture called Normalizing Flow. A normalizing flow is a process that transforms a simple distribution p_X into a complex distribution p_X by applying a sequence of bijective functions, as represented in Figure 3.2.

Noting $g = g_1 \circ g_2 \circ .... \circ g_n$ the transformation from simple to complex, and its reciprocal $f = f_n \circ f_{n-1} \circ .... \circ f_1$, we can use the change of variable formula (see Section 3.2.1) to express the density of one distribution given the other:

$$p_Z(z) = p_X(x) \prod_{i=1}^{n} |\det(J_{g_i}(x_{i-1}))| \tag{3.11}$$

where $x_i = f_i \circ f_{i-1} \circ .... \circ f_1(x)$, $x = x_0$ and $z = x_n$.

In the reverse way:

$$p_X(x) = p_Z(z) \prod_{i=1}^{n} |\det(J_{f_i}(x_i))|. \tag{3.12}$$

Usually, p_Z is chosen to be a standard normal distribution with independent components. For d components, its probability density function is

$$p_Z(z) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_i^2}. \tag{3.13}$$

During the training phase, the network is optimized following the Maximum Likelihood Estimation method (see Section 3.2.1) to find the parameters that will best fit p_Z with

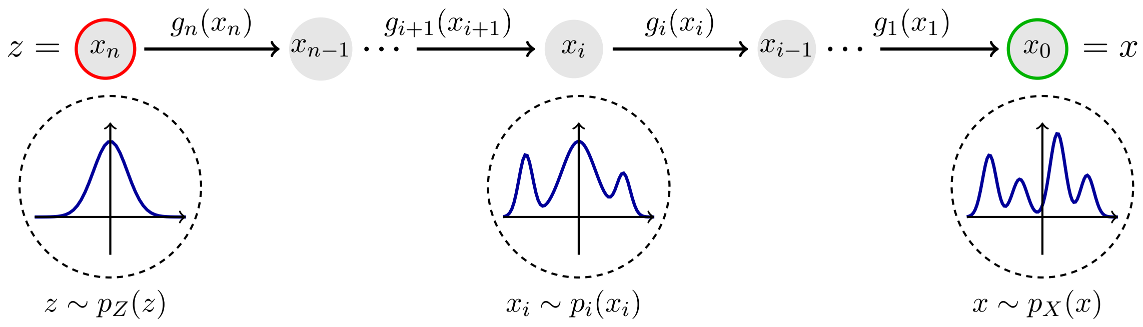

Figure 3.2: A normalizing flow is a sequence of functions g_i that transforms a simple distribution p_Z intro a complex one p_X. (from [78])

data from a dataset X of size N:

$$\mathcal{L}(X; f) = \frac{1}{N} \sum_{x \in X} \log p_Z(f(x)). \tag{3.14}$$

Interestingly, there are no constraint on the f functions, apart from being invertible. Hence, it is possible to implement them as neural networks, which are then called Invertible Neural Networks. In the next section, we will review different strategies to build the invertible layers that will compose these INNs.

In practice, there are a few desirable properties that this type of neural networks should satisfy to make them compatible with complex data:

- The output dimension has to be the same as the input dimension (otherwise it would not be invertible).
- Each layer should be expressive enough to represent complex distributions.
- The inverse function and its Jacobian determinant should be efficiently computable.

3.3 Different implementations of Invertible Neural Networks

A variety of approach have been proposed to implement INNs. We will describe the most popular ones and analyze the computational efficiency of their inverse and Jacobian determinant.

3.3.1 Elementwise Flows

The simplest form of invertible layer can be constructed with a bijective scalar function $h : \mathbb{R} \to \mathbb{R}$ that we apply on each component:

$$g(x) = (h(x_1), h(x_2), ...h(x_n)). \tag{3.15}$$

The determinant and inverse are trivial to compute but the representation power of this layer is very limited.

3.3.2 Linear Flows

A linear invertible layer can be built with an invertible matrix A:

$$g(x) = Ax + b. \tag{3.16}$$

Notice that in this general form, the computations of the determinant and the inverse are expensive, with a complexity of $O(n^3)$. To address this limitation, A can be constrained to be a triangular matrix, whose determinant is obtained in $O(n)$ and inverse in $O(n^2)$.

3.3.3 Coupling Flows

First introduced by Non-linear Independent Components Estimation (NICE) [79], coupling flows consist in splitting an input x into two subvectors x_1 and x_2 of equal size; then successively applying two (non-necessarily invertible) functions f_1 and f_2 following the equation:

$$\begin{cases} y_1 = f_1(x_2) + x_1 \\ y_2 = f_2(y_1) + x_2, \end{cases} \tag{3.17}$$

and finally, concatenate y_1 and y_2.

The inverse operation can be computed with

$$\begin{cases} x_2 = y_2 - f_2(y_1) \\ x_1 = y_1 - f_1(x_2). \end{cases} \tag{3.18}$$

As we can see, the design of this layer ensures that the inverse can be very easily computed and that the Jacobian determinant is constant (so it is not necessary to compute it).

3.3.4 Autoregressive Flows

Let $h(\cdot;\theta)$ be a bijective scalar function $h : \mathbb{R} \to \mathbb{R}$ parameterized by θ. We call an autoregressive [80] layer g a function that outputs $y = g(x)$ conditioned on the previous entries of the input such that

$$y_t = h(x_t; \Theta_t(x_{1:t-1})), \tag{3.19}$$

where Θ_t are mappings from $\mathbb{R}^{t-1}$ to the set of parameters, and Θ_1 is a constant. While it is possible to compute very efficiently the Jacobian and the forward pass, the invertible pass is inherently sequential.

3.3.5 Continuous Flows

Neural ordinary differential equations [81] are a recent design of deep neural networks that formulates the forward pass of a deep network as the solution of an ordinary differential equation. Initial works were motivated by the similarity between the expression of a layer of a ResNet and a step of an Euler solver for ordinary differential equation. They benefit of various advantageous properties compared to classical neural networks, among them their limited memory usage since several layers can be expressed with one set of parameters; and a trade-off between prediction accuracy and speed of execution due to their execution time flexibility. They are also invertible, making them a very promising tool for computing normalizing flows.

Notably, the authors of Free-form Jacobian of Reversible Dynamics (FFJORD) [82] have proposed a very efficient way of computing the likelihood of a sample.

3.3.6 Others

A large variety of formulations of invertible layers have been proposed and their expressive power is still under ongoing research. We should also mention Planar and Radial flows [83] that respectively allow simple non-linear scaling transformations with respect to a subspace in the Cartesian coordinate system and to a point in the polar coordinate system; with linear-time computation of the determinant. The authors of Emerging Convolutions [84] also propose a way to formulate invertible convolutions that have receptive fields identical to standard convolutions.

3.4 Discussion on Out-of-Distribution Data Detection

The computation of the likelihood is a key component for various tasks. Among them is the detection of out-of-distribution (OOD) data, which consists in identifying if a given sample belongs or not to the training distribution of a model. The idea is that *a model trained on data from a distribution should have higher likelihood on data sampled from this distribution than from a different distribution.* Our proposed approach for CL, that will be presented in the next chapter, heavily rely on the principle of OOD.

It has been observed that the general idea of OOD data detection does not turn out to be systematically true in practice [85]. Namely, a generative model trained on pictures of clothing may assign higher likelihood to pictures of handwritten digits. The precise reason for this phenomenon is still unknown and is not specific to INN. Several potential explanations have been proposed including a limitation of likelihood computation for high-dimensional data, a tendency of the model to learn low-level statistics rather than high-level semantics.

Noticeably, when it comes to data generation, it has been observed that a model can have poor likelihood and produce great samples, or have great likelihood and produce poor samples as shown in [86]. This is a counter-intuitive result, but future architectures of INNs may provide the inductive bias that would correct this mismatch between high likelihood and good sample generation.

Although these observations seem to limit the relevance of INNs for OOD detection, we should emphasize that it will not directly affect our approach since:

- Our INNs are not trained directly on image data, but rather on extracted features. This strategy allows for better OOD detection as shown by the authors of [87].
- We do not use our INNs to generate data, but only to compute the likelihood of data samples.
- Each data distribution is represented by its own INN. This way, even if a network assigns inadequately high likelihood on an OOD data sample, this data sample should have even higher likelihood for the network of its corresponding class.

Chapter 4

Applying Invertible Neural Networks on Continual Learning tasks

4.1 Introduction

In this chapter, we will present our first experiments to tackle the problem of Continual Learning (CL) in the challenging case of single class incremental learning. Our approach is based on the model of Invertible Neural Networks (INNs), introduced in Chapter 3, which presents particular properties that we will rely on to mitigate the catastrophic forgetting effect[1].

[1] This chapter is based on a publication published at the International Joint Conference on Neural Networks, 2020 [88]

4.1.1 Motivations and Challenge

We investigate the problem of training several datasets in a sequential fashion with batches of only one class at a time. We have seen in Chapter 2 that most approaches of the state-of-the-art rely on updating a feature extractor when data from a new class are available. However, this strategy is unreliable in the special case that we are interested in, namely batches of data from only one class. With few or no sample of negative data, it is highly inefficient to update the weights of a network because the setting of deep learning typically involves vast amounts of diverse data to be able to learn to extract valuable features. Without enough negative samples, the training is prone to overfit the new class.

4.1.2 Out-of-distribution detection for Continual Learning

Our approach consists in interpreting a CL problem as several out-of-distribution (OOD) detection problems. OOD detection has already been studied for neural networks and can be formulated as a binary classification problem which consists in predicting if an input x was either sampled from the same distribution as the training data or from a different distribution [89, 90].

This way, for each class, we can train a network to predict if an input x is likely to have been sampled from the distribution of this class. The class that got the highest confidence can be considered as the prediction of the class of x. This training procedure is particularly suitable for CL, as the training of each network does not require any negative sample. Our approach can be seen as a One-versus-All method for multi-class classification based on INNs, hence its name One-versus-All Invertible Neural Network (OvA-INN).

For a class i, we can train an INN f_i to fit a prior distribution p and compute the exact log-likelihood l_i on a sample x

$$l_i(x) = \log(p(f_i(x))). \tag{4.1}$$

As it is usually assumed in the literature, for computational efficiency reasons, we consider the case where p is a distribution with independent components p_d:

$$p(f_i(x)) = \prod_d p_d(f_{i,d}(x)). \tag{4.2}$$

In our experiments, we considered p_d to be standard normal distributions. Although it is possible to learn the parameters of the distributions, we found experimentally that doing so decreases the accuracy of the results.

Under these design choices, the computation of the log-likelihood becomes

$$\begin{aligned} l_i(x) &= \sum_d \log(p_d(f_{i,d}(x)) \\ &= -\sum_d \frac{1}{2} f_{i,d}(x)^2 + \sum_d \log\left(\frac{1}{\sqrt{2\pi}}\right) \\ &= -\frac{1}{2}\|f_i(x)\|_2^2 + \beta \end{aligned} \tag{4.3}$$

where $\beta = -n \log\left(\sqrt{2\pi}\right)$ is a constant term, with n the size of x.

Hence, identifying the network with the highest log-likelihood is equivalent to finding the network with the smallest output norm.

4.1.3 Our approach

As we have seen in Chapter 3, a variety of options for implementing INNs is available. In this work, we will rely on coupling layers, as proposed in the Non-linear Independent Components Estimation (NICE) [91] approach. This choice is a simple yet very versatile architecture that will allow us to compute the likelihood without the cumbersome computations of the Jacobian determinant (see Section 3.3.3).

In this architecture, OvA-INNs are obtained by cascading invertible blocks. An invertible block (see Figure 4.1) consists in splitting the input x into two subvectors x_1 and x_2 of equal size; then successively applying two (non necessarily invertible) networks g_1 and g_2 following the equation:

$$\begin{cases} y_1 = g_1(x_2) + x_1 \\ y_2 = g_2(y_1) + x_2, \end{cases} \tag{4.4}$$

and finally, concatenate y_1 and y_2.

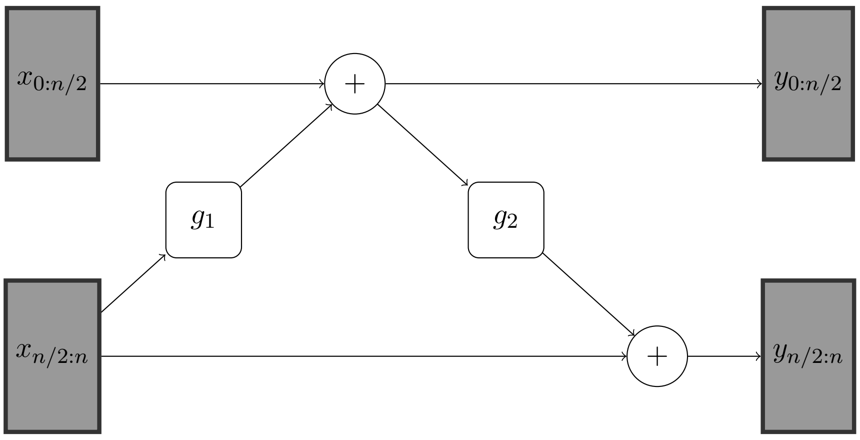

Figure 4.1: Diagram of a coupling layer. x is split into $x_{0:n/2}$ and $x_{n/2:n}$. g_1 and g_2 can be any type of Neural Networks as long as their output dimension is the same as their input dimension. In our experiments, we stack two of these blocks one after the other and use fully-connected feedforward layers for g_1 and g_2.

We propose to specialize each INN to a specific class by training each of them to maximize the log-likelihood on samples from their class. This goal is equivalent to outputting a vector with small norm when an INN is presented with data samples from its class, as showed in Equation 4.3. Given a dataset $\mathcal{X}_i$ of class i and an INN f_i, our objective is, therefore, to minimize the loss

$$\mathcal{L}(\mathcal{X}_i) = \frac{1}{|\mathcal{X}_i|} \sum_{x \in \mathcal{X}_i} \|f_i(x)\|_2^2. \tag{4.5}$$

Once the training has converged, the weights of this network will not be updated when new classes are added. At inference time, after learning t classes, the predicted class y^* for a sample x is obtained by running each network and identifying the one with the smallest output:

$$y^* = \underset{y=1,\ldots t}{\arg\min} \|f_y(x)\|_2^2. \tag{4.6}$$

In theory, one could train a network end-to-end for each class. However, it is usually unnecessary to retrain low level features to discriminate new classes, as those are general features. In some situations, it can be more efficient to apply a preprocessing step by relying on a common pretrained feature extractor beforehand. Using a fixed pretrained feature extractor allows saving computation time and memory when learning a new class. This fixed representation of data can be transferred from a model trained on ImageNet or from unlabeled data [92]. Noting ϕ this fixed pretrained model, the inference equation can be expressed as

$$y^* = \underset{y=1,\dots t}{\arg\min} \| f_y(\phi(x)) \|_2^2. \tag{4.7}$$

4.2 Experimental Results

We compare our method against several state-of-the-art baselines for single-head learning on the MNIST and CIFAR-100 datasets.

4.2.1 Evaluation on MNIST

We start by considering the MNIST dataset [93]. MNIST is a canonical benchmark that is now considered a particularly easy task. However, MNIST remains challenging in the case of single-head CL.

Topology of OvA-INN and implementation details

The OvA-INNs used for training MNIST are obtained by cascading two invertible blocks (see Figure 4.2). For MNIST, no preprocessing is used: the OvA-INNs take the MNIST data as a flattened input directly.

Due to the bijective nature of INNs, the output size of invertible blocks of Figure 4.1 is the same as their input size. As the MNIST input data are vectors of size 784, g_1 and g_2 functions have to take inputs of half this size, that is $n = 392$.

It is, however, possible to choose the depth of the networks g_1 and g_2 or to optimize their number of parameters. In our experiments, we use fully connected layers for the

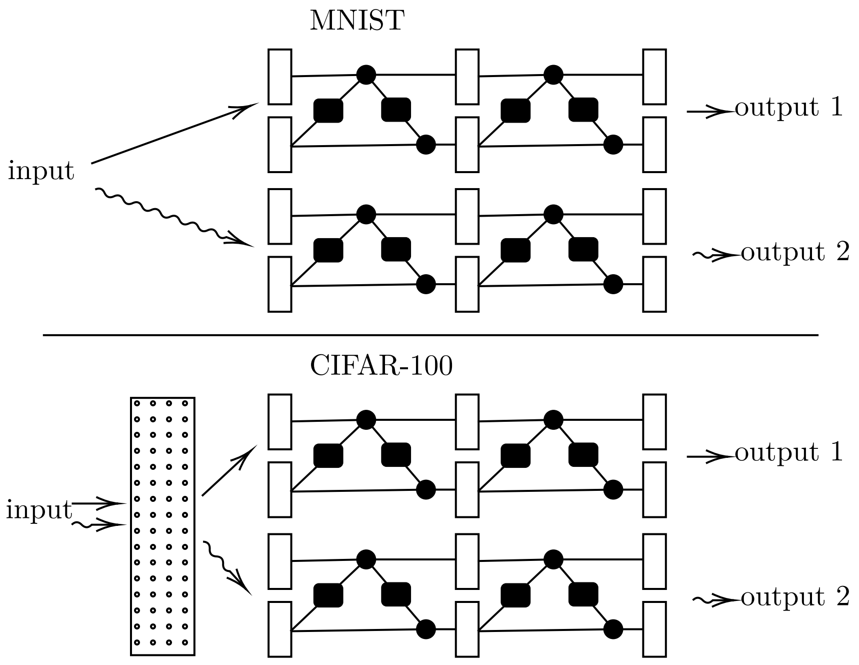

Figure 4.2: This diagram illustrates our implementations of OvA-INN for our experiments. We only represent the case of two classes. On top, the inputs are directly fed into the INNs, each one being constituted of two blocks in cascade. The details of these blocks are displayed in Figure 4.1. On the bottom, we apply a pretrained feature extractor represented by a rectangle filled with dots.

networks g_1 and g_2. To to reduce the memory footprint, we replace the square matrix of parameters W of size $n \times n$ by a product of matrices AB of sizes $n \times m$ and $m \times n$.

For all our MNIST experiments, we use $m = 16$.

The hyperparameters for the MNIST simulations are reported in Table 4.2a. No data augmentation was used.

Methods

Our implementation is done using the Pytorch deep learning framework [94], and the simulations are performed on single precision Nvidia Graphics Processing Units.

We use the adaptive moment estimation (Adam) optimizer [95] and a scheduler that reduces the learning rate by a factor of 0.5 when the loss stops improving.

Baselines

We compare our approach with methods based on exemplars storage (detailed in Section 2.1.2) such as Incremental Classifier and Representation Learning (iCaRL) [45], SupportNet [43], Gradient Episodic Memory (GEM) [48] and Random Path Selection for Incremental Learning (RPSnet) [65]; and with methods based on generative models (detailed in Section 2.1.3) such as Parameter Generation and Model Adaptation (PGMA) [59] and Deep Generative Replay (DGR) [55].

We report the results from the original papers; except for iCaRL and SupportNet where we use the code provided with the SupportNet paper [43] to compute the results for MNIST. For these baselines, we use networks employing two layers of convolutions with poolings and a fully connected last layer. We have also set the coreset size to $s = 800$ samples.

All the baselines use batches of two classes, as their performances would be drastically reduced with batches containing a single class.

Results and analysis

We report the average accuracy over all the classes after the networks have been trained on all batches (see Table 4.1).

Table 4.1: Comparison of accuracy and memory cost in number of parameters (and memory usage for storing samples if relevant) of different approaches on MNIST at the end of the CL.

Model	Accuracy	Memory cost	Classes per batch
Parameter Generation and Model Adaptation (PGMA) [59]	81.70%	6,000k	2 by 2
SupportNet [43]	89.90%	940k	2 by 2
Gradient Episodic Memory (GEM) [48]	92.20%	4,919k	2 by 2
Deep Generative Replay (DGR) [55]	95.80%	12,700k	2 by 2
Incremental Classifier and Representation Learning (iCaRL) [45]	96.00%	940k	2 by 2
Random Path Selection for Incremental Learning (RPSnet) [65]	96.16%	4,919k	2 by 2
Our approach: One-versus-All Invertible Neural Network (OvA-INN)	**96.40%**	**520k**	**1 by 1**

Table 4.2: Hyperparameters

(a) MNIST

Hyperparameter	**Value**
Learning Rate	2e-3
Number of epochs	200
Weight decay	0.0
Number of invertible blocks	2
Compression factor m	16
Patience	20

(b) CIFAR-100

Hyperparameter	**Value**
Learning Rate	2e-3
Number of epochs	1000
Weight decay	2e-4
Number of invertible blocks	2
Compression factor m	32
Patience	30

Our approach presents better results than all the other reference methods: the accuracy reaches 96.4%, while the accuracy of the baselines ranges between 81.7% and 96.16%. This is impressive, as our approach is the only one being trained by batches of a single class, and it also features the lowest memory cost! OvA-INN uses 520,000 parameters, whereas the baselines use between 940,000 (iCaRL) and 12,700,000 parameters (DGR). These first results suggest the high potential of the OvA-INN approach.

It should be noted that the obtained 96.4% accuracy remains significantly below the MNIST accuracies obtained in a non-CL context (that can reach 99.8% [96]). This difference highlights the difficulty of single-head CL, even for a simple task like MNIST.

4.2.2 Evaluation on CIFAR-100

It is now well known that impressive results obtained on the MNIST dataset do not necessarily transfer to more sophisticated datasets. For this reason, we now consider a more complex image dataset with a greater number of classes: CIFAR-100 [97], already introduced in section 1.2.3. This new benchmark allows us to make comparisons in the case of a long sequence of data batches and to illustrate the value of using a pretrained feature extractor for CL.

Topology of OvA-INN and implementation details

Preprocessing The CIFAR-100 images are first preprocessed through the convolutional layers of a Residual neural network (ResNet) with 32 layers pretrained on ImageNet.

Rescaling As ResNet-32 has been trained on ImageNet images of size 224×224 pixels, we rescale CIFAR-100 images to match the size of images from ImageNet. For this purpose, we use the resize transformation from the torchvision library, which employs bilinear interpolation.

OvA-INN topology As in the MNIST case, the OvA-INNs used for training CIFAR-100 are obtained by cascading two invertible blocks (see Figure 4.2).

As the output of the convolutional layers of ResNet-32 are of size 512, the g_1 and g_2 functions in the invertible blocks take inputs of size $n = 256$. As in the MNIST case, we use fully connected layers for the networks g_1 and g_2, and to reduce the memory footprint, we replace the square matrix of parameters W of size $n \times n$ by a product of matrices of sizes $n \times m$ and $m \times n$. For the CIFAR-100 experiments, we use $m = 32$.

The hyperparameters for the CIFAR-100 simulations are listed in Table 4.2b.

Regularization When performing learning one class at a time, the amount of training data can be highly reduced: only 500 training samples per class for CIFAR-100! To avoid overfitting the training set, we found that adding a weight decay regularization could increase the validation accuracy.

Methods

As in the MNIST case, our implementation is done using the Pytorch and single precision Nvidia Graphics Processing Units. We use the adaptive moment estimation (Adam) optimizer, and a scheduler that reduces the learning rate by a factor of 0.5 when the loss stops improving.

Baselines

We compare OvA-INN results with the following baselines:

- FearNet [62] is based on a generative model. It uses a pretrained ResNet48 features extractor. In their experiments, the authors rely on a warm-up phase. Namely, the network is first trained with all the first 50 classes of CIFAR-100, and subsequently learns the next 50 classes one by one in a continual fashion.
- iCaRL [45], End-to-end IL [46] both use 2k exemplars from previous classes and retrain a ResNet32 features extractor respectively with a distillation loss and with a cross-entropy together with distillation loss.
- RPSnet [65] also use 2k exemplars from previous classes but it trains several ResNet18 in parallel and assign different paths for predicting each class.
- Nearest prototype is our implementation of the method consisting in computing the mean vector (prototype) of the output of a pretrained ResNet32 for each class at train time. Inference is performed by finding the closest prototype to the ResNet output of a given sample.

Results and analysis

Fig. 4.3 presents the CIFAR-100 CL results of OvA-INN and our baselines. Image data are provided by batch of classes. When the training on a batch ($\mathcal{D}_i$) is completed, the accuracy of the classifier is evaluated on the test data of classes from all previous batches ($\mathcal{D}_1, ..., \mathcal{D}_i$). We report the results from the literature with various sizes of batch when they are available.

OvA-INN uses the weights of a ResNet-32 pretrained on ImageNet and never updates them. FearNet also uses pretrained weights from a ResNet. iCaRL, End-to-End IL and RPSnet use a similar architecture but retrain it from scratch at the beginning and fine-tune it with each new batch.

The performance of the Nearest prototype baseline proves that there is high benefit in using pretrained feature extractor on this kind of dataset. FearNet shows better performance by taking advantage of a warm-up phase with 50 classes.

We can see that OvA-INN is able to clearly outperform all the other approaches, reaching 72% accuracy after training on 100 classes. This result is particularly impressive.

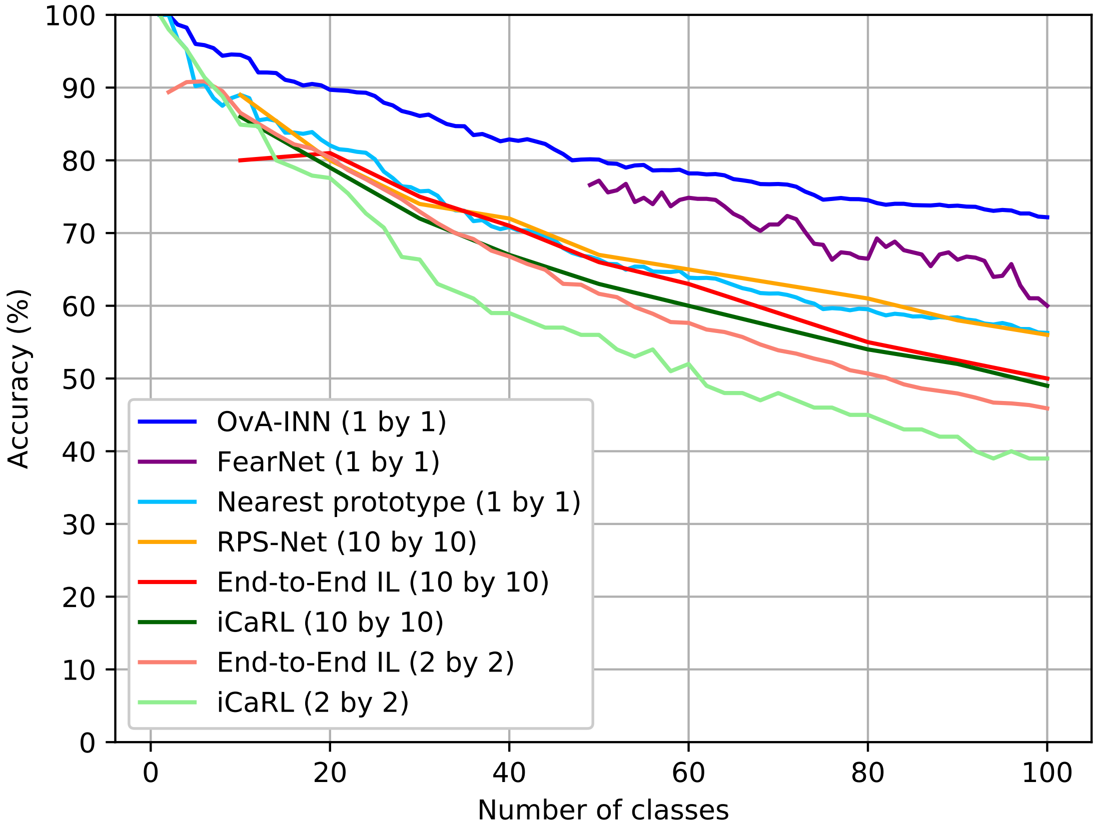

Figure 4.3: Comparison of the accuracy of several CL methods on CIFAR-100 with various batches of classes. In parenthesis is indicated the number of classes per batch. FearNet's curve has no point before 50 classes because the first 50 classes are learned in a non-continous fashion. OvA-INN, FearNet and Nearest prototype all benefit from a fixed pre-trained feature extractor. OvA-INN and Nearest prototype are our own implementations, the other results are reported from their original paper.

For comparison, we were only able to reach 76% accuracy on a ResNet trained on all the CIFAR-100 data at once.

We can see that the performances of methods retraining ResNet from scratch (iCaRL, End-to-End IL and RPSnet) quickly deteriorate compared to those using pretrained parameters. Even with larger batches of classes, the gap is still present. Notice that one cannot simply adapt those methods by using them with a pretrained feature extractor like we do. They are not design to take advantage of pretrained features. Their updating procedure rely on carefully modifying a complex model to make it compatible with new data. By only updating the last layers, their value is very limited. The authors of FearNet have provided experiments that still show inferior performances for iCaRL even with a fixed feature extractor.

It can be surprising that at the end of its warm-up phase, FearNet still has an accuracy bellow OvA-INN, even though it has been trained on all the data available at this point. It should be noted that FearNet is training an autoencoder and uses its encoding part as a features extractor (stacked on the ResNet) before classifying a sample. This can diminish the discriminative power of the network since it is also constrained to reproduce its input (only a single autoencoder is used for all classes).

4.2.3 Experiments in multi-head Continual Learning

We provide additional experimental results in the multi-head setting, which consists in learning independent tasks as previously defined in Section 1.2.1. We propose to perform the learning of CIFAR-100 with 10 tasks of 10 classes each. The results are displayed in Table 4.3. The training procedure of OvA-INN does not change from the usual single-head learning but, at test time, the evaluation is processed by batches of 10 classes (instead of the whole dataset). The accuracy score is the average accuracy over all 10 tasks.

We report the results from various methods of multi-head learning. Although our approach is able to match state-of-the-art results in accuracy, it should be noticed that it is drastically more memory and time consuming than some baselines. OvA-INN requires to train entire new layers whilst Dynamically Expandable Networks (DEN) is optimized to use as little additional parameters as possible to learn a new task. That being said,

Table 4.3: Comparison of the accuracy of several multi-head CL methods on CIFAR-100 on 10 tasks of 10 classes.

Model	Accuracy
Elastic Weight Consolidation (EWC) [36]	81.34%
Progressive Networks [63]	88.19%
Dynamically Expandable Networks (DEN) [98]	92.25%
Our approach: OvA-INN	**92.58%**

none of the compared methods can be applied in the single-head setting.

4.3 Discussion

4.3.1 Visualization

To further understand the effect of an INN on the feature space of a sample, we propose to project the different feature spaces in two dimensions. To perform this projection, we rely on the t-distributed stochastic neighbor embedding (t-SNE) algorithm, which is a non-linear dimensionality reduction technique commonly used for high dimensional data visualization [99].

In Figure 4.4, we project the features of the first five classes of CIFAR-100 test set. The t-SNE hyperparameters used for obtaining this Figure are listed in Table 4.4. Each class is represented with a different color.

The projection of features extracted by the ResNet is displayed on a single plot (Figure 4.4a), since those features are computed from a single network. Cluster centers are represented by black crosses. A sample closer to a cluster center indicates a higher confidence of the network to predict the class corresponding to the cluster.

In the ResNet projections of Figure 4.4a, we can see that some samples appear in

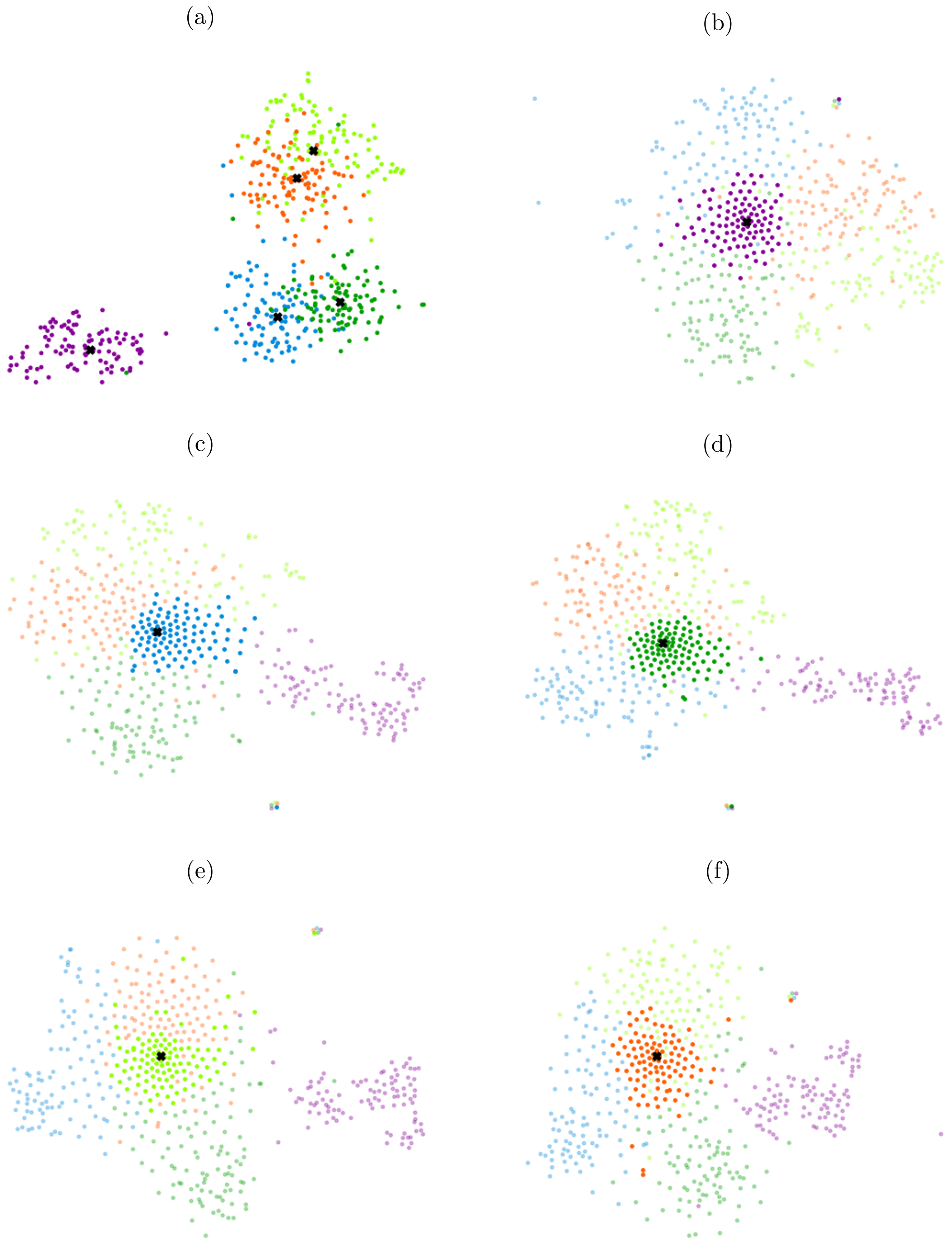

Figure 4.4: t-SNE projections of feature spaces for five classes from CIFAR-100 test set (colors are given by the ground truth). *(a)*: feature space before applying INNs (black crosses are the clusters centers). *(b),(c),(d),(e),(f)*: each feature space after the INN of each class. The samples of a class represented by a network are clustered around the zero vector (black cross) whilst the samples from other classes appear further away from the cluster.

a cluster of a different class. Those samples are likely to be misclassified by a distance-based method since the dimensionality reduction did not manage to distinguish them from samples of other classes.

For the INNs, we display the projection of features computed by each of the five networks on separate plots (Figures 4.4b, 4.4c, 4.4d, 4.4e, and 4.4f for networks 1, 2, 3, 4, and 5, respectively). In this case, we observe that classes that are already well represented in a cluster with ResNet features (like the violet class) are clearly separated from the clusters of INNs. On the other hand, classes represented with ambiguity by ResNet features (like the light green and the red classes) are better clustered in the INN space. Figure 4.4, therefore, highlights the power of OvA-INN in a very visual manner.

Table 4.4: t-SNE Hyperparameters

Hyperparameter	**Value**
Perplexity	15.0
Principal Components	50
Steps	400

4.3.2 Limitations

A limiting factor in our approach is the necessity to add a new network each time one wants to learn a new class. This makes the memory and computational cost of OvA-INN linear with the number of classes. Recent works in networks merging could alleviate the memory issue by sharing weights [100] or relying on weights superposition [101]. We will propose our own method to further improve memory efficiency in Chapter 5.

Another constraint of using INNs is to keep the size of the output equal to the size of the input. When one wants to apply a feature extractor with a high number of output channels, it can have a very negative impact on the memory consumption of the invertible layers. Feature Selection or Feature Aggregation techniques may help to alleviate this issue [102].

Finally, we can notice that our approach is highly dependent on the quality of the pretrained feature extractor. In our CIFAR-100, we had to rescale the input to make it compatible with ResNet. Nonetheless, recent research works show promising results in training feature extractors in very efficient ways [103]. Because it does not require to retrain its feature extractor, we can foresee better performance in class-by-class learning with OvA-INN as new and more efficient feature extractors are discovered. In this regard, we will present results on feature extractor design in Chapter 6.

4.4 Conclusion

In this chapter, we proposed a new approach for the challenging problem of single-head CL without storing any of the previous data. On top of a fixed pretrained neural network, we trained for each class an INN to refine the extracted features and maximize the log-likelihood on samples from its class. This way, we show that we can predict the class of a sample by running each INN and identifying the one with the highest log-likelihood.

This setting allows us to take full benefit of pretrained models, which results in very good performances on the class-by-class training of MNIST and CIFAR-100 compared to prior works. The next chapters present two optimizations of the OvA-INN scheme to enhance its applicability in real-life settings.

Chapter 5

Memory Efficient Invertible Neural Networks

5.1 Introduction

A consistent body of work has been dedicated to developing hardware accelerators to improve the training efficiency of neural networks [104, 105]. The goal of these works is to allow the training procedure to be performed on devices at the edge, with limited computational and memory resources. Most of the research has focused on optimizing the representation of network parameters, but the cost of storing a sufficiently large database to perform the training is also a critical issue. Continual Learning (CL) could provide a very profitable solution for the training of neural networks on embedded devices.

This chapter is based on a publication published at the International Conference on Artificial Intelligence Circuits and Systems, 2021 [106]

5.2 Related works in the circuits and systems community

The circuits and systems community is increasingly engaged in building efficient hardware to make the training of neural networks at the edge a reality [107–110]. Recently, the authors of [111] proposed to investigate the performance of quantization techniques for CL and have simulated various bits representations for the weights, activations and gradients values during training. In their experiments, they use the method proposed by Incremental Classifier and Representation Learning (iCaRL). We can observe that this type of quantized training regime can lead to inefficient learning for complex tasks in the case of exemplars-based methods. Indeed, updating the whole network requires to be cautious to avoid the collapse of previously acquired knowledge, but this objective is particularly difficult to ensure with quantized gradient introducing noise in the training procedure.

Another approach has been studied, using self-organization neural networks, where a Field-programmable gate array (FPGA) based custom hardware architecture is able to dynamically grow additional neurons and connections to improve its knowledge representation when presented with new data [112]. In particular, this system does not rely on stored data from previous classes. The major limitation of this approach is that it is designed to learn with batches of several classes, which can lead to poor performances with batches of a single class.

In this chapter, we propose using One-versus-All Invertible Neural Network (OvA-INN) for CL at the edge. The largest challenge of this technique for embedded systems, however, is that it requires storing each network, resulting in a high cost in memory. In the following sections, therefore, we propose to reduce the impact in memory of networks storage for OvA-INN. By using an efficient factorization of its matrices, we show that OvA-INN can have a significantly better trade-off between accuracy and memory usage compared to other solutions such as FearNet and iCaRL.

5.3 Memory impact analysis

The memory cost of OvA-INN depends on the number of invertible layers L per network, the number of classes T and the matrix size N (for $N \times N$ weight matrices). Notice that N depends on the data size and cannot be changed. Otherwise, the network would not be invertible. A naive implementation would have a memory impact of

$$\mathcal{M}_{\text{naive}} = 2TL(N^2 + N), \tag{5.1}$$

expressed as a number of parameters.

In the previous formulation of OvA-INN (Chapter 4), the memory used is reduced by using low rank matrices. Hence, each matrix of size $N \times N$ is written as the product of two smaller matrices of size $N \times M$ and $M \times N$ (Figure 5.1, top). The previous memory cost is, therefore,

$$\mathcal{M}_{\text{original}} = 2TL(2MN + N). \tag{5.2}$$

To further decrease the memory cost to store the weight matrices, we propose to represent them with two common matrices for each layer operation. Consider a given set of matrices in a given layer (Figure 5.1 bottom). We can write the associated matrices for each class t as $\mathbf{W}_t = \mathbf{A}\mathbf{V}_t\mathbf{B}$ where the $\mathbf{V}_t$ are of size $M \times M$, and $\mathbf{A}_t$ and $\mathbf{B}_t$ are the common matrices. With this modification, the memory cost is now

$$\mathcal{M}_{\text{efficient}} = 4NML + 2TL(M^2 + N). \tag{5.3}$$

We can see that this memory cost value features a first term independent on T and that in the second term, the product NM has been replaced by M^2, which is very advantageous. M can indeed be chosen arbitrarily small, as long as the network is still able to perform well. A simple illustrating case with $L = 1$ and $T = 2$ is represented in Figure 5.1.

Since this matrix formulation implies to update all matrices, we had to add a new term to the loss to limit the update on matrices of previous classes. When learning a new class t, we store the weight matrices from the previous $t-1$ classes and label them as $\mathbf{A}^{\text{old}}$, $\mathbf{B}^{\text{old}}$

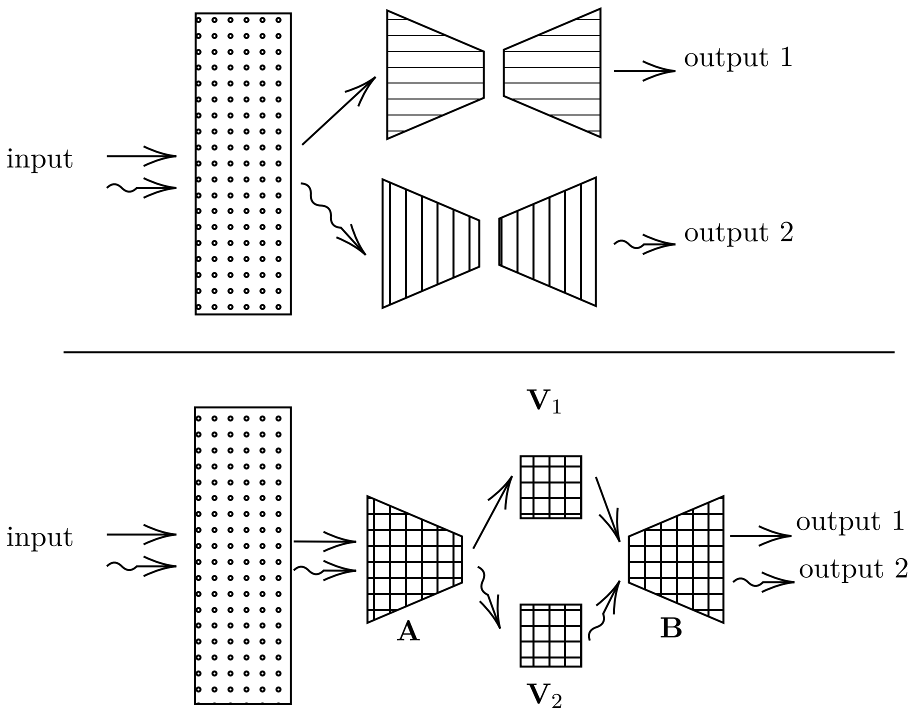

Figure 5.1: Diagram illustrating the differences between the original implementation of OvA-INN (top) and the optimized method (bottom). This figure only represents the case of the sequential learning of two classes with one layer (with one operation for each class). The fixed feature extractor is represented by the dotted rectangles. OvA-INN first update the parameters for the class 1, then the parameters for the class 2 during a second step. With this new approach, each network still has its own path but some parts are shared, this implies that the whole network has to be updated during the second phase.

and $\mathbf{V}_i^{\text{old}}$ for each $i < t$. These matrices will be used as a constraint when learning the new class and will be discarded afterward. The weight for the new class $\mathbf{W}_t$ will also be discarded as we will use the newly computed $\mathbf{V}_t$ ($\mathbf{W}_t$ is only used to stabilize the learning process). The constraint term for a set of matrices S is then

$$\mathcal{L}_t^S = \sum_{i=1}^{t-1} \|\mathbf{A}^{\text{old}}\mathbf{V}_i^{\text{old}}\mathbf{B}^{\text{old}} - \mathbf{A}\mathbf{V}_i\mathbf{B}\|^2 + \|\mathbf{W}_t - \mathbf{A}\mathbf{V}_t\mathbf{B}\|^2. \tag{5.4}$$

As the network will update itself with data of new classes, the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{V}_t$ will slowly drift away from their initial values, especially for the earliest learned classes. But if this constraint is enforced, the drift will be limited and it will have a negligible impact on the network accuracy.

The constraint must be applied to all sets of matrices used in each layer, leading to

$$\mathcal{L}_t^1 = \sum_S \mathcal{L}_t^S. \tag{5.5}$$

The optimization loss $\mathcal{L}_t^0$ for maximizing the log-likelihood of an Invertible Neural Network (INN) f_t with a feature extractor ϕ on a class t being:

$$\mathcal{L}_t^0 = \frac{1}{2} \sum_{x \in \mathcal{X}_t} \|f_t(\phi(x))\|_2^2, \tag{5.6}$$

we can write the final loss when learning a new class t with our proposed method as

$$\mathcal{L}_t = \mathcal{L}_t^1 + \lambda \mathcal{L}_t^0, \tag{5.7}$$

where λ is a constant hyperparameter.

5.4 Experiments

We conduct our experiments on the CIFAR-100 dataset, as in Chapter 4. The data for each class is fed one class after the other and the final accuracy is reported on the CIFAR-

100 test dataset. The implementation details and hyperparameters choice is similar to the implementation proposed in previous chapters, and reported in Table 5.1.

Table 5.1: Hyperparameters. Intervals indicate that various values have been chosen for the experiments.

Hyperparameter	**Value**
Learning rate	3e-3
Number of epochs	3000
Weight decay	2e-4
Batch size	125
Regularization term λ	0.2
Number of layers L	$[1, 2]$
Matrix size N	256
Compression parameter M	[2...64]
Number of classes T	100

5.4.1 Baselines

We compare the performances of our method with the original OvA-INN, presented in Chapter 4. Additionally, we report the performances from FearNet [113], which generates samples from previous classes to update itself. This generative process requires to store statistics for each class and to have a consequently large model to ensure a sufficient quality of generated samples. Those requirements have a significant impact on its memory consumption. We also report the performances of iCaRL [45] adapted with a fixed feature extractor (taken from [113]).

Finally, we add a simple baseline called Nearest Neighbour, which consists in computing the mean of the extracted features for each class. By computing the features of a new sample, the class can be predicted by finding the closest mean. No additional learning is required for this baseline and it only require to store the mean of the features of each class.

5.4.2 Results

Figure 5.2 presents the accuracy results of our technique and of the baselines. Each point corresponds to the final accuracy after learning 100 classes one by one for a given model size, expressed in megabytes, and assuming four bytes by parameter. The weights of the feature extractor are not counted in the model size, as they are common to all baselines.

For the OvA-INN based approaches, we vary the parameter M from 2 to 64, to get different model sizes. We observe a natural trade-off between accuracy and model size. We evaluate our approach in the case of one and two stacked invertible layers for each class.

We can see that our approach is less affected by the decrease in accuracy with smaller model size than the original OvA-INN. For the smallest models, our approach can even match the accuracy performance of the original, with a memory cost reduced by a factor five. Our method has also a better accuracy than the Nearest Neighbour (62% against 56%) with almost the same model size.

The one-layer OvA-INN system has a trade-off between accuracy and model size that is slightly better than the two-layers OvA-INN. However, one-layer OvA-INN is limited to an accuracy of 66%.

We can notice that iCaRL performances are beneath the other methods. This approach has indeed not been designed to take into account that the feature extractor is fixed. But updating the feature extractor in a CL setting is a very costly operation and does not scale efficiently when combined with weight quantization. Our experiments show that approaches based on a fixed pretrained feature extractor are a promising candidate for implementing more efficient and reliable CL algorithms on embedded systems.

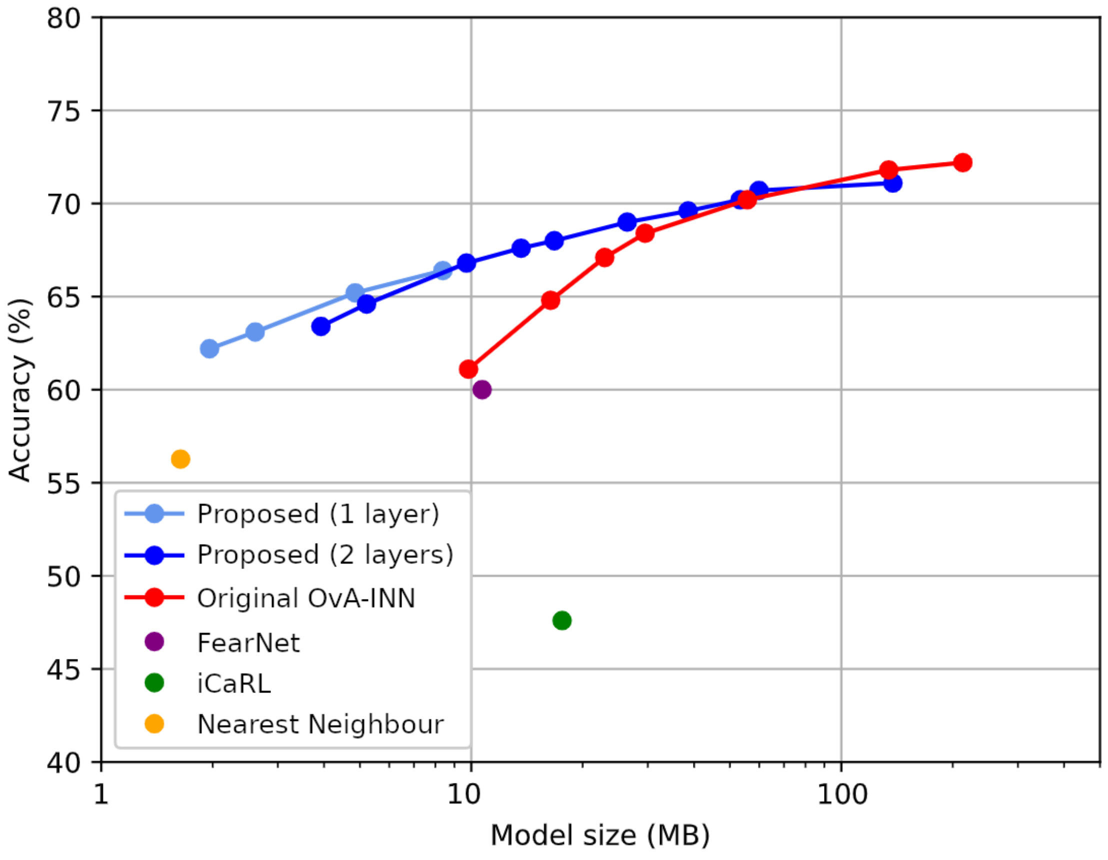

Figure 5.2: Comparison of the trade-off between accuracy and memory consumption of various approaches for the single class incremental learning of CIFAR-100. All methods use a pre-trained Residual neural network (ResNet)32 but its memory impact is not taken into account for the model size in this figure. The original implementation of OvA-INN uses two layers of INN. But we can see that with our approach, for really small models at a given model size, it is more efficient to use only one layer rather than two more compressed layers. On the contrary, with bigger models the factorization proposed in this article cannot compete with the original implementation.

5.5 Conclusion

In this work, we have proposed a method for reducing the memory impact of training INNs in the case of single class incremental learning. We have conducted experiments on the CIFAR-100 dataset to illustrate the trade-off between the final accuracy and the memory cost of this implementation. Our results show that this approach is more efficient than related state-of-the-art methods.

Because it does not require to update the entire network and does not rely on any previous data, this implementation provides a promising solution for the CL of neural networks on embedded systems. Future research could examine the impact of quantization methods on this implementation and deploy it on a chip.

Chapter 6

Learning a common feature extractor for Continual Learning

In the previous chapters, we have been relying on a pretrained feature extractor as the basis for the One-versus-All Invertible Neural Network (OvA-INN) approach. This technique can be a very efficient way of training a model on a new database, especially when few labeled data are available. However, it is not always straightforward to find a pretrained model for any data type. End-to-end Continual Learning (CL) is therefore also highly desirable.

For this purpose, in this chapter, we explore what makes a feature extractor compliant with CL. The core idea of the chapter is to train a feature extractor on a subset of classes from a dataset and subsequently learn the remaining classes in a continuous way without updating this extractor.

We will first explain our intuition for using a model based on invertible layers as a feature extractor, then present an architecture suited for our purpose and finally provide experimental results on the CIFAR-100 dataset confirming the viability of our approach.

6.1 Limitations of a pretrained feature extractor

Pretraining the model on external data, as we have done in the previous chapters, can be a convenient approach for learning each class of a dataset one by one with batches of limited size. However, pretrained feature extractor also bring some important challenges:

- **Input preprocessing.** As we have remarked in our previous experiments, we had to resize the images from CIFAR in order to make it compatible with the pretrained Residual neural network (ResNet). This had to be done since the CIFAR and ImageNet data have very different scales. Such preprocessing steps can rapidly become cumbersome, and they could be avoided with a suitable feature extractor.
- **Memory size.** When importing a model from the literature, there only exists a handful of possible models to choose from (unless we are willing to conduct new experiments on the whole source dataset). This means that we may end up with an unnecessary large model that requires large amount of memory to be stored. Designing our own model can allow us to ensure an optimal use of available memory.
- **Computational cost.** For the same reasons as the previous point, an unnecessarily large model would require more computations than needed.
- **Features relevance.** A model trained on external data may not provide the optimal features for the data of the current task. Changes in scale, lighting or the way the data is produced can have very negative impact on performances.

For all these reasons, recent research has taken interest in a very compelling alternative setting, where we would be to have access to a pool of data from several classes for training a first model and then have only access to future single-class batch one by one for updating the model. This situation may allow harnessing the benefits of both pretrained and dedicated feature extractors: we can first train a model end-to-end on a set of multiple classes, and then update it with batches of a single unseen class [42, 47, 53].

The work reported in this Chapter follows this trend and applies it to the case of Invertible Neural Networks (INNs).

6.2 Proposed architecture

As we have seen in the previous section, training a model end-to-end for CL is of high interest. However, most of the work in the literature is dedicated to designing new training strategies to tackle this challenge. In other words, they are looking for answers to "How should a network be trained in these conditions?". As for us, we think it is also essential to ask for "Which type of networks are compatible with this style of learning?". In the next paragraphs, we will present our point of view on the potential limiting factors of commonly used architecture for CL and propose a different approach.

6.2.1 Motivations and challenges

Typical neural architectures can be viewed as a stack of transformations that synthesize increasingly abstract concepts from the data in order to answer a problem (e.g., classification, segmentation, regression,...). The last layers contain all the information needed to make a prediction. That is why it is a common practice to use a pretrained model to answer a different problem, provided that the data of the new problem is similar enough to the data of the previous problem.

This is certainly a relevant approach for traditional learning tasks when all the data is available from the start. Neural networks are trained to discover useful patterns in the data and discard any information that cannot provide insights to answer the problem at hand.

However, this strategy appears limited in the case of CL where a good model should retain as much information about the data as possible, since this information may become useful for a future task. For example, if only shape detection is necessary for a model

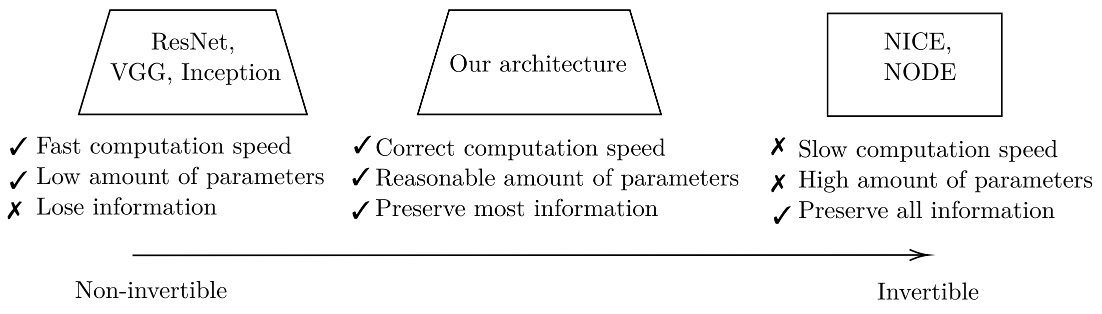
ResNet, VGG, Inception
Our architecture
NICE, NODE
✓ Fast computation speed
✓ Low amount of parameters
✗ Lose information
✓ Correct computation speed
✓ Reasonable amount of parameters
✓ Preserve most information
✗ Slow computation speed
✗ High amount of parameters
✓ Preserve all information
Non-invertible
Invertible

Figure 6.1: Our proposed architecture of feature extractor is designed in a way to take advantage of both standard non-invertible networks and INNs.

to classify a set of images, it will not learn any feature related to color. So, if we were to introduce new classes to this classification task, and that these classes required color features to distinguish them from old classes, the previously learned features would be irrelevant for this new task.

Elaborating features without losing any information is precisely the goal of INNs. By allowing the full recovery of the input given their output, they are indeed guaranteeing that no information is discarded during the learning phase.

However, INNs also have limitations as feature extractors. Requiring that an invertible layer output must have the same size as its input implies two main drawbacks:

- The number of parameters of each layer depends on the size of the input, which can be very high for large images.
- The output size of the feature extractor is the same as its input size. This can lead to important memory consumption for the INNs that will use these features.

It is therefore crucial to design an architecture that offers a good balance between invertible and non-invertible networks (see Figure 6.1).

6.2.2 Components of the architecture

We propose a new architecture building on the recent Generative Flow (Glow) model [77]. As the first INN approach succeeding in synthesizing realistic images from large datasets, Glow appears as a favorable choice for creating an efficient feature extractor.

Our strategy consists in stacking multiple blocks with Glow components and dropping a part of the features after each block, as represented in Figure 6.2. This way, the output size of each layer can be chosen as an hyperparameter of the model. Although dropping features is detrimental to the invertibility of the network, each layer is still invertible on its own. Our hypothesis is that even though some information is lost in this process, the information remaining is still more valuable than in a conventional neural network architecture.

We will now give a quick review of the components of Glow, more detailed descriptions can be found in the original paper [77].

- **Actnorm** is a normalization layer whose parameters are trained (not computed over batches).
- **Invertible** 1×1 **convolution** is a layer that can be easily inverted and can be see as a way to shuffle the channels.
- **Flow** is a block constituted by stacking a layer of Actnorm followed by Invertible 1×1 followed by a coupling layer.
- **Split** is a layer that separates its input into two halves along the channel dimension.
- **Squeeze** is a layer that reshapes a tensor of size $H \times W \times C$ into a tensor of size $H/2 \times W/2 \times 4C$.

Furthermore, we replaced the activation functions of the original Glow by Swish activations [114]:

$$f(x) = x \times \text{sigmoid}(x). \tag{6.1}$$

This choice is favorable, as Swish activations do not rely on a threshold (unlike the ReLU activations) and do not saturate (unlike sigmoid activations).

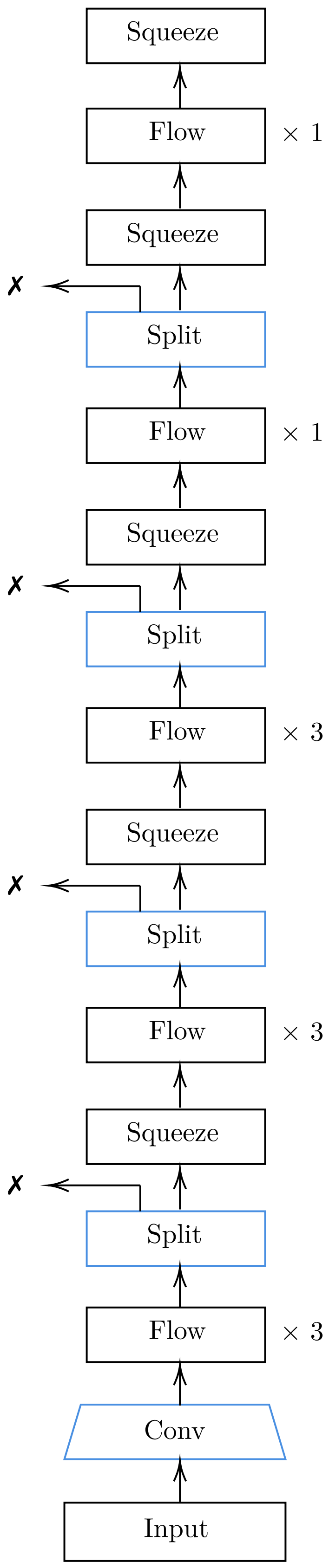

Figure 6.2: The architecture of feature extractor used in our experiment. The non-invertible operations are represented in blue. The sign ×3 indicates that the corresponding layer is stacked 3 times. The cross ✗ indicates that half of the channels are being dropped after a Split.

6.3 Experimental Results

We propose to evaluate the performances of OvA-INN associated with this architecture and compare them with the performances of OvA-INN associated with ResNet and other approaches using ResNet (and updating it).

6.3.1 Experimental protocol

We will rely on the CIFAR-100 dataset. In this setting, the training is divided in two phases. First, each method has access to the first 50 classes of the dataset and can train on all of them together. Then, the next 50 classes are sent to be trained one by one. Contrarily to the previous chapters, none of the approach are based on a pretrained feature extractor,

During the first training phase with 50 classes, we stack a fully connected layer on top of our proposed architecture, namely after the last Squeeze layer in Figure 6.2. The loss used at this point is the cross-entropy. After the training has converged, we can drop this layer and replace it with 50 different INNs. This way, the first INNs are trained over fixed feature, just like the next that will be leaned in a continuous fashion. Ensuring this symmetry between the INNs trainings is crucial to guarantee good performances.

To avoid losing performances when transitioning from fully connected to INNs, we apply a sigmoid function on the features and constrain the linear layer weights to stay in the interval $[-1, 1]$. This makes the features easier to learn for the INNs. With these modifications, we observe a reduction lesser than 1% in accuracy for the INNs compared to the fully connected layer.

6.3.2 Implementation details

We based our implementation on a github repository [115]. The authors indicate that the optimizer Adamax [95] provides optimal results, so we also used this optimizer.

The hyperparameters for the CL phase are similar to the ones previously used in Chapter 4, and the hyperparameters of the feature extractor training are summarized in Table 6.1.

Note that unlike previous experiments, no scaling of the input images is necessary. Namely, the network takes directly images of size 32×32 as input.

During the first training phase, with 50 classes, we rely on data augmentation to improve the performances of the feature extractor. This includes standard augmentation operations such as random cropping, random horizontal flipping and random rotations. Since our Glow-based model has a slightly higher number of parameters compared to its ResNet counterpart, we additionally rely on a mixup data augmentation step for its training [116].

Table 6.1: Hyperparameters

Hyperparameter	**Value**
Learning rate	5e-3
Number of epochs	300
Weight decay	2e-4
Batch size	250
Actnorm scale	1.0
First convolution size C_0	16
Coupling layer type	affine
Mixup parameter α	0.2

6.3.3 Evaluation on CIFAR-100

We reproduce the experimental setting proposed in Pooled Outputs Distillation Network (PODNet) [42]. It consists in providing the network with the first 50 classes of CIFAR-100 as a pretraining phase, then update the network with batches of a single class. In their work, all the baselines rely on a custom ResNet32 feature extractor. This is a lighter version of the original ResNet, specifically made for CIFAR (see Section 4.2 of the ResNet paper [117]).

Table 6.2: Here are displayed for each approach its feature extractor architecture, its training strategy and its mean accuracy. The performances of OvA-INN are greatly improved by choosing our proposed feature extractor, allowing it to outmatch all baselines without updating its feature extractor.

Method	Feature extractor	Feature extractor training	Mean accuracy
GFR	ResNet	Trained on 50 classes, then partially updated with each class	54.97
OvA-INN	ResNet	Trained on 50 classes, then fixed	56.06
Rebalance	ResNet	Trained on 50 classes, then updated with each class	57.08
PODNet	ResNet	Trained on 50 classes, then updated with each class	61.44
OvA-INN Glow	**Glow-based**	**Trained on 50 classes, then fixed**	**64.81**

Some of the baselines rely on exemplars storage: PODNet [42] and Rebalance [47]. While Generative Feature Replay (GFR) [61] use a generator to produce features for replay. The various methodologies are summarized in Table 6.2. This table also displays the mean accuracy, which corresponds to the mean over the performances after each of the 50 phases of CL.

Performance analysis

In Figure 6.3, PODNet and Rebalance results are reported from [42]; GFR results are reported from [61]. We display the performances of OvA-INN with a ResNet and with our new Glow-based architecture.

We can see that OvA-INN with ResNet is one of the least efficient methods. The ResNet indeed focused its representation capacities on the 50 first classes and those are not reliable for learning the next classes. Other approaches are probably doing better thanks to their ability to update their ResNet with stored data. Except for GFR which does not store exemplars and ends up as the least efficient method in this setting.

However, we can clearly see that OvA-INN with our proposed architecture of feature extractor is performing well above the others. This confirms our hypothesis that invertible layers convey more useful informative features for future learning phases. Notably, we can notice that this architecture is not simply more powerful for data discrimination than the other. Its performances for the first 50 classes are indeed lower than all the other baselines. By adopting invertible layers as feature extractor, we traded a small amount of performances at the beginning for more sustainability during the following CL phase.

Memory impact

We did not use the memory optimization proposed in Chapter 5 but rather relied on the original implementation. Our goal is not to study the trade-off between memory and accuracy. We are interested in providing the best accuracy possible, but we emphasize that it is always possible to decrease the memory impact of OvA-INN at the cost of a few points in accuracy.

Still, our implementation remains competitive compared to the exemplars-based ones, as shown in Table 6.3. The top layer parameters column represents the cost to store a fully connected layer for ResNet and all the INNs for the Glow-based architecture. The Data memory column indicates the cost of storing exemplars. In fine, our proposed architecture is able to provide drastically superior performances with a lower memory impact compared to the other baselines.

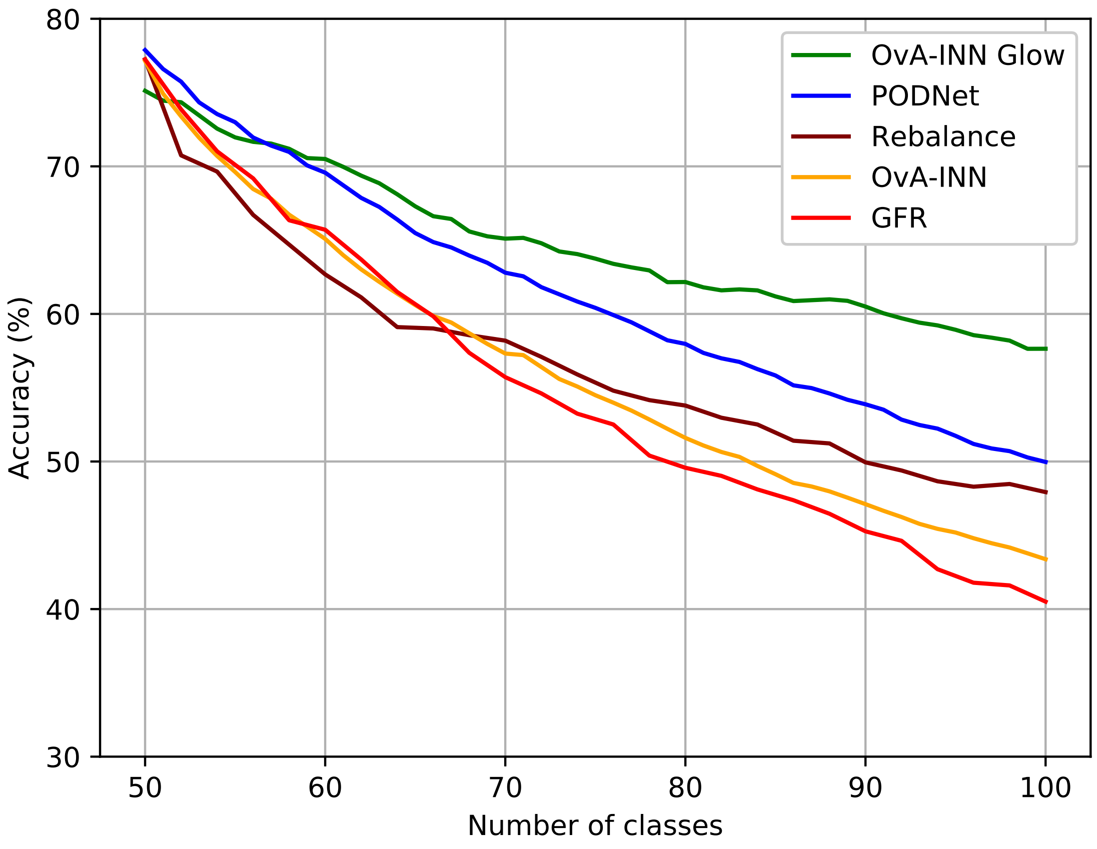

Figure 6.3: Comparison between Glow-based and ResNet feature extractors. We can see that the choice of feature extractor can be determinant for future performances in a CL setting: OvA-INN clearly benefits from the proposed Glow-based architecture compared to a ResNet architecture.

Table 6.3: Comparison of ResNet and Glow-based feature extractors in terms of memory consumption and final accuracy.

Architecture	Method	Network params	Top layer params	Data memory	Accuracy
ResNet *with exemplars*	PODNet, Rebalance	463k	6.5k	6,144k	Average
ResNet	GFR	507k	6.5k	0	Low
ResNet	OvA-INN	463k	3,379k	0	Low
Glow-based	OvA-INN Glow	543k	3,379k	0	High

6.4 Discussion

The results of the experiments proposed in this chapter are strong arguments in favor of exemplar-free methods for CL. Whilst updating the feature extractor was considered an efficient strategy thanks to the use of stored exemplars, we showed that an adequate design of the feature extractor could ensure event better performances.

This is also an encouraging development for methods relying on fixed feature extractor, the kind of OvA-INN. It would indeed appear unlikely that methods updating the feature extractor could benefit as much from invertible layers. Any update could have harmful consequences on the performances of previous task. The whole point of using invertible layer is indeed to benefit from previously learned feature for old classes; while still having access to already useful features for the new classes without the need to make any update.

We should emphasize that these results are preliminary. There is still a lot to uncover about what makes a good feature extractor for CL. This work is meant to foster more research in this direction. We consider that there should be a trade-off between the expressive power of a network for full batch learning, namely the first phase of the experiment, and the expressive power for CL, the second phase. This is clearly visible in Figure 6.3 since OvA-INN Glow is a little bit underperforming compared to the other baselines for 50 classes but quickly compensates when provided with more classes in a CL fashion.

6.5 Conclusion

In this chapter, we proposed a feature extractor architecture specifically designed for CL tasks. We built our approach with invertible components from the Glow model of generative flows. We obtained noticeably good performances when pretraining this architecture on the first 50 classes of CIFAR-100 before learning the next 50 classes in a CL fashion. This work was a clear departure from the usual vision that a good model for full batch learning should also be a good model for CL tasks.

Chapter 7

Conclusion

7.1 Summary of our results

We started our study by illustrating how Continual Learning (CL) is a key component of efficient and powerful deep neural networks training. With increasing opportunities for application in a variety of fields and a growing in complexity of tasks to be learned, it is clear that the challenge of learning in a continuous fashion is a subject of high in interest for the future of Machine Learning (ML).

We presented a general definition of the goal of CL and insisted on the variety of ways to address it. The literature is indeed filled with articles focused on different aspects of the problem, under different assumptions (single-head, multi-head, single class, multi class, without boundary,...). Each research team is tackling the CL problem from its own point of view, sometimes with its own metrics, leading to a sprawling development. However, recent effort have been made to unify the methodology of the community [41, 118].

We have proposed an extensive review of the literature in CL for deep neural networks.

This allowed us to build our approach based on some limitations of the previous works. The fundamental observation that we made was that retraining the entire model with each newly added class was a very complex operation and was arguably unnecessary. So, we set out to design a method based on a fixed feature extractor (either pretrained on external data or a subset of the data at hand).

Before diving into the details of our approach, we dedicated a chapter to introduce a modern neural network architecture with interesting properties. Invertible Neural Networks (INNs) recently gained a lot of interest in the deep neural networks community for their mathematical grounding and computation efficiency. Notably, they are a reliable approach for computing density estimations.

In the next chapter, we detailed our main contribution, One-versus-All Invertible Neural Network (OvA-INN), which we used to tackle CL by relying on a set of independently trained INNs, one for each class. At test time, each INN is computed, and the one with the highest confidence indicates the class of the sample. Our experiments, conducted on the MNIST and CIFAR-100 datasets, showed superior results compared to other approaches relying or not on a pretrained feature extractor.

Although we did ensure that our approach provided better performances compared to other methods with a similar number of parameters, it was still a costly operation. To improve this, we dedicated a chapter to the analysis of the memory impact of OvA-INN and proposed a different training process to optimize memory usages. In our experiments, we were able to illustrate a trade-off between the performances and the memory cost of our approach. In particular, comparable performances with other approaches could be attained with a significantly lower memory impact.

Finally, we investigated new ideas in support of a pretrained feature extractor for CL. Namely, we paid special attention to the architecture of the feature extractor to make it more reliable for CL. While commonly used neural networks appear prone to overly specialize on original data, we hypothesized that it is possible to design neural architectures that could extract more relevant features for the long term. To support our claim, we proposed a new model of feature extractor based on INNs components. Experiments conducted on CIFAR-100 showed a clear advantage of our custom feature extractor compared to a Residual neural network (ResNet) when used with OvA-INN. Notably, the combination of this model with OvA-INN was the best performing method compared to all the other approaches.

7.2 Potential improvements to this work

While OvA-INN presents very convincing advantages for its ability to learn with a single class batch and without stored data, we should also highlight some current limitations that may serve as a starting point for future improvement.

7.2.1 Memory consumption

Although we put effort into making our approach as memory efficient as possible, the memory cost is still linear with the number of classes. That being said, our optimization strategy of matrix factorization remains applicable for any implementation of INNs, leaving room for improvement with more memory efficient networks such as continuous flows [81].

7.2.2 Architecture of the feature extractor

Our experiments in Chapter 6 showed that a feature extractor based on INNs components appears highly beneficial for CL. But these are only preliminary results that need more theoretical investigation and empirical testing of other architectures. We think there is a high potential for improvement in this direction. What makes a good multi-scale INN is still an open question but progress is being made steadily [119].

7.2.3 Choice of the invertible neural network model

In this study, we chose to rely on coupling layers to build OvA-INN, as it was a convenient way to perform our experiments and demonstrate the validity of our hypothesis. We would like to emphasize that, since the beginning for our work, INNs have been greatly improved over time and a lot of ongoing research is still happening to make them even more efficient [82, 120, 121]. There would be a clear benefit in using these new architectures in our model. It may even be possible to design INNs specifically optimized for the setting of OvA-INN.

7.3 Future research directions

A main takeaway from our work is the high value of a carefully designed feature extractor associated with adequate density estimation methods. Although we have exclusively investigated fixed feature extractors, there is still room for improvement by updating the feature extractor. Notably, the generative ability of INNs could be used to produce samples at the feature level or at the input level, allowing for modifications of the model much like pseudo-rehearsal approaches [77].

Scaling our method to more complex datasets with higher number of classes is also an important research avenue. In addition to the usual scaling challenges, an intelligent strategy for choosing which INNs to compute at test time should be proposed to avoid unnecessary computations. In practice, it is indeed usually possible to divide a set of classes into subsets of closed classes. By finding in which subset a sample belongs to, it would be possible to avoid computing density estimations for the other subsets, a costly and useless operation. More complex data may also require to identify features in a larger dimensional space. Therefore, at some point, it will become crucial to rely on feature selection [122] to avoid each INN to have to deal with all the features.

Chapter 8

Bibliography

[1] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1918–1921, San Jose, CA, 2011. IEEE.

[2] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks`.

[4] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2019.

[6] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, pages

843–852. IEEE Computer Society, 2017. ISBN 978-1-5386-1032-9. URL `http://dblp.uni-trier.de/db/conf/iccv/iccv2017.html#SunSSG17`.

[7] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The computational limits of deep learning, 2020.

[8] Hiroaki Mikami, Hisahiro Suganuma, Yoshiki Tanaka, Yuichi Kageyama, et al. Imagenet/resnet-50 training in 224 seconds. *arXiv preprint arXiv:1811.05233*, pages 1–8, 2018.

[9] Chuan Li. OpenAI's GPT-3 language model: A technical overview. `https://lambdalabs.com/blog/demystifying-gpt-3/`, 2020. Accessed: 2020-09-03.

[10] Xialei Liu. Awesome incremental learning. `https://github.com/xialeiliu/Awesome-Incremental-Learning`, 2020. Accessed: 2020-11-18.

[11] Rahaf Aljundi. *Continual Learning in Neural Networks*. PhD thesis, KU Leuven, 2019. URL `http://arxiv.org/abs/1910.02718`.

[12] Vincenzo Lomonaco. *Continual Learning with Deep Architectures*. PhD thesis, alma, Aprile 2019. URL `http://amsdottorato.unibo.it/9073/`.

[13] Timothée Lesort. *Continual Learning: Tackling Catastrophic Forgetting in Deep Neural Networks with Replay Processes*. PhD thesis, ENSTA, 2020.

[14] Xialei Liu. *Visual recognition in the wild: learning from rankings in small domains and continual learning in new domains*. PhD thesis, Universitat Autònoma de Barcelona, 2019.

[15] 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, 08 2013. doi: 10.3389/fpsyg.2013.00504.

[16] Yen-Chang Hsu, Yen-Cheng Liu, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *ArXiv*, abs/1810.12488, 2018.

[17] Flavius Raslau, I Mark, Andrew Klein, J Ulmer, V Mathews, and L Mark. Memory part 2: The role of the medial temporal lobe. *AJNR. American journal of neuroradiology*, 36, 11 2014. doi: 10.3174/ajnr.A4169.

[18] Endel Tulving et al. Episodic and semantic memory. *Organization of memory*, 1: 381–403, 1972.

[19] Joseph O'Neill, Barty Pleydell-Bouverie, David Dupret, and Jozsef Csicsvari. Play it again: Reactivation of waking experience and memory. *Trends in neurosciences*, 33:220–9, 03 2010. doi: 10.1016/j.tins.2010.01.006.

[20] Melissa C. Duff, Natalie V. Covington, Caitlin Hilverman, and Neal J. Cohen. Semantic memory and the hippocampus: Revisiting, reaffirming, and extending the reach of their critical relationship. *Frontiers in Human Neuroscience*, 13:471, 2020. ISSN 1662-5161. doi: 10.3389/fnhum.2019.00471. URL https://www.frontiersin.org/article/10.3389/fnhum.2019.00471.

[21] Bryce Mander, Joseph Winer, and Matthew Walker. Sleep and human aging. *Neuron*, 94:19–36, 04 2017. doi: 10.1016/j.neuron.2017.02.004.

[22] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

[23] Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In Yoshua Bengio and Yann LeCun, editors, *ICLR (Poster)*, 2014. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2014.html#GoodfellowMDCB13.

[24] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *CoRL*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017. URL http://proceedings.mlr.press/v78/lomonaco17a.html.

[25] Matthew Schwall, Tom Daniel, Trent Victor, Francesca Favaro, and Henning Hohnhold. Waymo public road safety performance data, 2020.

[26] Steven Ashley. Centimeter-accurate gps for self-driving vehicles. `https://www.sae.org/news/2016/10/centimeter-accurate-gps-for-self-driving-vehicles`, 2016. Accessed: 2020-10-12.

[27] Proposed regulatory framework for modifications to artificial intelligence/machine learning (ai/ml)-based software as a medical device (samd) - discussion paper and request for feedback. https://www.fda.gov/media/122535/download, 2019.

[28] Guoguang Rong, Arnaldo Mendez, Elie Bou Assi, Bo Zhao, and Mohamad Sawan. Artificial intelligence in healthcare: Review and prediction case studies. *Engineering*, 6, 01 2020. doi: 10.1016/j.eng.2019.08.015.

[29] Cecilia S Lee and Aaron Y Lee. Clinical applications of continual learning machine learning. *The Lancet Digital Health*, 2(6):e279 – e281, 2020. ISSN 2589-7500. doi: https://doi.org/10.1016/S2589-7500(20)30102-3. URL `http://www.sciencedirect.com/science/article/pii/S2589750020301023`.

[30] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline, 2020.

[31] Chen Zeno, Itay Golan, E. Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv: Machine Learning*, 2018.

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

[33] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

[34] 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 *ICCV*, pages 618–626. IEEE Computer Society, 2017. ISBN 978-1-5386-1032-9. URL http://dblp.uni-trier.de/db/conf/iccv/iccv2017.html#SelvarajuCDVPB17.

[35] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5133–5141, 2019.

[36] 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*, page 201611835, 2017.

[37] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[38] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3987–3995. JMLR.org, 2017.

[39] Frederik Benzing. Understanding regularisation methods for continual learning, 2020.

[40] Sungmin Cha, Hsiang Hsu, Flávio P. Calmon, and Taesup Moon. CPR: classifier-projection regularization for continual learning. *CoRR*, abs/2006.07326, 2020. URL https://arxiv.org/abs/2006.07326.

[41] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation, 2020.

[42] Arthur Douillard, M. Cord, Charles Ollion, T. Robert, and E. Valle. Podnet: Pooled

outputs distillation for small-tasks incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[43] Yu Li, Zhongxiao Li, Lizhong Ding, Yuhui Hu, Wei Chen, and Xin Gao. SupportNet: a novel incremental learning framework through deep learning and support data. *bioRxiv*, 2018. doi: 10.1101/317578.

[44] Khurram Javed and F. Shafait. Revisiting distillation and incremental classifier learning. In *ACCV*, 2018.

[45] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, July 2017. doi: 10.1109/CVPR.2017.587.

[46] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018.

[47] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[48] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

[49] Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIX*, volume 12364 of *Lecture Notes in Computer Science*, pages 254–270. Springer, 2020. doi: 10.1007/978-3-030-58529-7_16. URL https://doi.org/10.1007/978-3-030-58529-7_16.

[50] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[51] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 583–592, 2019.

[52] Y. Wu, Yan-Jia Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 374–382, 2019.

[53] Ameya Prabhu, Philip Torr, and Puneet Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *The European Conference on Computer Vision (ECCV)*, August 2020.

[54] Learning deep neural networks incrementally forever. `https://arthurdouillard.com/post/incremental-learning/`, 2019. Accessed: 2021-03-10.

[55] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2990–2999. Curran Associates, Inc., 2017.

[56] Chenshe 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 *Conference on Neural Information Processing Systems (NIPS)*, 2018.

[57] 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.

[58] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong gan: Continual learning for conditional image generation. In *ICCV*, pages

2759–2768. IEEE, 2019. ISBN 978-1-7281-4803-8. URL http://dblp.uni-trier.de/db/conf/iccv/iccv2019.html#ZhaiCTHNM19.

[59] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[60] Amanda Rios and Laurent Itti. Closed-loop memory gan for continual learning. In Sarit Kraus, editor, *IJCAI*, pages 3332–3338. ijcai.org, 2019. URL http://dblp.uni-trier.de/db/conf/ijcai/ijcai2019.html#RiosI19.

[61] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 226–227, 2020.

[62] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[63] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. URL http://arxiv.org/abs/1606.04671.

[64] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=Sk7KsfW0-.

[65] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and

Ling Shao. Random path selection for incremental learning. *Advances in Neural Information Processing Systems*, 2019.

[66] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4077–4087, 2017.

[67] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3637–3645, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[68] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *ArXiv*, abs/2103.04922, 2021.

[69] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *PREDICTING STRUCTURED DATA*. MIT Press, 2006.

[70] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014. URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets`.

[71] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

[72] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Pro-*

ceedings of the 33rd International Conference on International Conference on Machine Learning, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/oord16.html.

[73] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srflow: Learning the super-resolution space with normalizing flow. In *ECCV*, 2020.

[74] Sungwon Kim, Sang-Gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. FloWaveNet : A generative flow for raw audio. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3370–3378. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/kim19b.html.

[75] Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. The reversible residual network: Backpropagation without storing activations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 2211–2221, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[76] Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. i-RevNet: Deep invertible networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJsjkMb0Z.

[77] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

[78] Lilian Weng. Flow-based deep generative models. *lilianweng.github.io/lil-log*, 2018. URL http://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html.

[79] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA,*

May 7-9, 2015, Workshop Track Proceedings, 2015. URL http://arxiv.org/abs/1410.8516.

[80] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf.

[81] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6572–6583, Red Hook, NY, USA, 2018. Curran Associates Inc.

[82] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

[83] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/rezende15.html.

[84] Emiel Hoogeboom, Rianne Van Den Berg, and Max Welling. Emerging convolutions for generative normalizing flows. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2771–2780. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/hoogeboom19a.html.

[85] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Why normaliz-

ing flows fail to detect out-of-distribution data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/ecb9fe2fbb99c31f567e9823e884dbec-Abstract.html.

[86] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016. URL http://arxiv.org/abs/1511.01844.

[87] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *ICLR (Poster)*. OpenReview.net, 2019. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2019.html#NalisnickMTGL19.

[88] Guillaume Hocquet, Olivier Bichler, and Damien Querlioz. OvA-INN: Continual learning with invertible neural networks. In *2020 international joint conference on neural networks (IJCNN)*. IEEE, 2020.

[89] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

[90] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[91] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[92] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-

supervised GANs via auxiliary rotation loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12154–12163, 2019.

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

[94] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

[95] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

[96] Adam Byerly, Tatiana Kalganova, and Ian Dear. A branching and merging convolutional network with homogeneous filter capsules. *arXiv preprint arXiv:2001.09136*, 2020.

[97] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[98] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[99] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[100] Yi-Min Chou, Yi-Ming Chan, Jia-Hong Lee, Chih-Yi Chiu, and Chu-Song Chen. Unifying and merging well-trained deep neural networks for inference stage. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2049–2056. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/283. URL `https://doi.org/10.24963/ijcai.2018/283`.

[101] Brian Cheung, Alex Terekhov, Yubei Chen, Pulkit Agrawal, and Bruno A. Olshausen. Superposition of many models into one. *CoRR*, abs/1902.05522, 2019. URL `http://arxiv.org/abs/1902.05522`.

[102] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data classification: Algorithms and applications*, page 37, 2014.

[103] Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi. Surprising effectiveness of few-image unsupervised feature learning. *arXiv preprint arXiv:1904.13132*, 2019.

[104] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12): 2295–2329, 2017.

[105] J. Chen and X. Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019. doi: 10.1109/JPROC.2019.2921977.

[106] Guillaume Hocquet, Olivier Bichler, and Damien Querlioz. Memory efficient invertible neural networks for class-incremental learning. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 1–4, 2021. doi: 10.1109/AICAS51828.2021.9458549.

[107] Ghouthi B Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel, and Michel Jezequel. Efficient hardware implementation of incremental learning and inference on chip. In *2019 17th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 1–4. IEEE, 2019.

[108] I. A. Lungu, S. Liu, and T. Delbruck. Fast event-driven incremental learning of hand symbols. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 25–28, 2019. doi: 10.1109/AICAS.2019.8771472.

[109] Axel Laborieux, Maxence Ernoult, Tifenn Hirtzlin, and Damien Querlioz. Synaptic metaplasticity in binarized neural networks. *CoRR*, abs/2003.03533, 2020.

[110] Maxence Ernoult, Julie Grollier, Damien Querlioz, Yoshua Bengio, and Benjamin Scellier. Updates of equilibrium prop match gradients of backprop through time in

an RNN with static input. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7079–7089, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/67974233917cea0e42a49a2fb7eb4cf4-Abstract.html.

[111] Y. Hu, T. Delbruck, and S. Liu. Incremental learning meets reduced precision networks. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2019.

[112] D. Piyasena, M. Thathsara, S. Kanagarajah, S. K. Lam, and M. Wu. Dynamically growing neural network architecture for lifelong deep learning on the edge. In *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pages 262–268, 2020. doi: 10.1109/FPL50879.2020.00051.

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

[114] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. In *ICLR (Workshop)*. OpenReview.net, 2018. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2018w.html#RamachandranZL18.

[115] Joost van Amersfoort. Glow-pytorch. https://github.com/y0ast/Glow-PyTorch, 2019.

[116] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR (Poster)*. OpenReview.net, 2018. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2018.html#ZhangCDL18.

[117] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[118] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele

Graffieti, Tyler L Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido van de Ven, et al. Avalanche: an end-to-end library for continual learning. *arXiv preprint arXiv:2104.00405*, 2021.

[119] Hari Prasanna Das, Pieter Abbeel, and Costas J. Spanos. Likelihood contribution based multi-scale architecture for generative flows, 2019.

[120] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf`.

[121] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2722–2730, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/ho19a.html`.

[122] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014. ISSN 0045-7906. doi: https://doi.org/10.1016/j.compeleceng.2013.11.024. URL `https://www.sciencedirect.com/science/article/pii/S0045790613003066`. 40th-year commemorative issue.

Titre: Apprentissage continu classe par classe pour les réseaux de neurones profonds

Mots clés: Apprentissage continu, Réseaux de neurones profonds, Oubli catastrophique, Vision artificielle, Réseaux inversibles

Résumé: Nous nous intéressons au problème de l'apprentissage continu de réseaux de neurones artificiels dans le cas où les données ne sont accessibles que pour une seule catégorie à la fois. Pour remédier au problème de l'oubli catastrophique qui limite les performances d'apprentissage dans ces conditions, nous proposons une approche basée sur la représentation des données d'une catégorie par une loi normale. Les transformations associées à ces représentations sont effectuées à l'aide de réseaux inversibles, qui peuvent alors être entraînés avec les données d'une seule catégorie. Chaque catégorie se voit attribuer un réseau pour représenter ses caractéristiques. Prédire la catégorie revient alors à identifier le réseau le plus représentatif. L'avantage d'une telle approche est qu'une fois qu'un réseau est entraîné, il n'est plus nécessaire de le mettre à jour par la suite, chaque réseau étant indépendant des autres. C'est cette propriété particulièrement avantageuse qui démarque notre méthode des précédents travaux dans ce domaine. Nous appuyons notre démonstration sur des expériences réalisées sur divers jeux de données et montrons que notre approche fonctionne favorablement comparé à l'état de l'art.
Dans un second temps, nous proposons d'optimiser notre approche en réduisant son impact en mémoire en factorisant les paramètres des réseaux. Il est alors possible de réduire significativement le coût de stockage de ces réseaux avec une perte de performances limitée.
Enfin, nous étudions également des stratégies pour produire des réseaux capables d'être réutilisés sur le long terme et nous montrons leur pertinence par rapport aux réseaux traditionnellement utilisés pour l'apprentissage continu.

Title: Class Incremental Continual Learning in Deep Neural Networks

Keywords: Continual Learning, Deep Neural Networks, Catastrophic forgetting, Computer vision, Invertible Neural Networks

Abstract: We are interested in the problem of continual learning of artificial neural networks in the case where the data are available for only one class at a time. To address the problem of catastrophic forgetting that restrain the learning performances in these conditions, we propose an approach based on the representation of the data of a class by a normal distribution. The transformations associated with these representations are performed using invertible neural networks, which can be trained with the data of a single class. Each class is assigned a network that will model its features. In this setting, predicting the class of a sample corresponds to identifying the network that best fit the sample. The advantage of such an approach is that once a network is trained, it is no longer necessary to update it later, as each network is independent of the others. It is this particularly advantageous property that sets our method apart from previous work in this area. We support our demonstration with experiments performed on various datasets and show that our approach performs favorably compared to the state of the art.
Subsequently, we propose to optimize our approach by reducing its impact on memory by factoring the network parameters. It is then possible to significantly reduce the storage cost of these networks with a limited performance loss.
Finally, we also study strategies to produce efficient feature extractor models for continual learning and we show their relevance compared to the networks traditionally used for continual learning.