Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, Natalia Díaz-Rodríguez

Version of Record: https://www.sciencedirect.com/science/article/pii/S1566253519307377
Manuscript_779b3bc79cf4dbaf021bc1f2a556bdf2

Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges

Timothée Lesort[*,1,2], Vincenzo Lomonaco[*,3], Andrei Stoian[2], Davide Maltoni[3], David Filliat[1], and Natalia Díaz-Rodríguez[*,1]

[1]Flowers Team (ENSTA Paris, Institute Polytechnique de Paris & INRIA).
[2]Thales, Theresis Laboratory.
[3]Department of Computer Science and Engineering - University of Bologna
[*]Equal contribution.

Abstract

Continual learning (CL) is a particular machine learning paradigm where the data distribution and learning objective change through time, or where all the training data and objective criteria are never available at once. The evolution of the learning process is modeled by a sequence of learning experiences where the goal is to be able to learn new skills all along the sequence without forgetting what has been previously learned. CL can be seen as an online learning where knowledge fusion needs to take place in order to learn from streams of data presented sequentially in time. Continual learning also aims at the same time at optimizing the memory, the computation power and the speed during the learning process.

An important challenge for machine learning is not necessarily finding solutions that work in the real world but rather finding stable algorithms that can *learn* in real world. Hence, the ideal approach would be tackling the real world in a embodied platform: an autonomous agent. Continual learning would then be effective in an autonomous agent or robot, which would learn autonomously through time about the external world, and incrementally develop a set of complex skills and knowledge.

Robotic agents have to learn to adapt and interact with their environment using a continuous stream of observations. Some recent approaches aim at tackling continual learning for robotics, but most recent papers on continual learning only experiment approaches in simulation or with static datasets. Unfortunately, the evaluation of those algorithms does not provide insights on whether their solutions may help continual learning in the context of robotics. This paper aims at reviewing the existing state of the art of continual learning, summarizing existing benchmarks and metrics, and proposing a framework for presenting and evaluating both robotics and non robotics approaches in a way that makes transfer between both fields easier. We put light on continual learning in the context of robotics to create connections between fields and normalize approaches.

Keywords: Robotics, Reinforcement Learning, Deep Learning, Lifelong Learning, Continual Learning, Catastrophic Forgetting

1 Introduction

Machine learning (ML) approaches generally learn from a stream of data randomly sampled from a stationary data distribution. This is often a *sine qua non* condition to learn efficiently. However, in the real world, this setting is rather uncommon. *Continual Learning* (CL) [128] gathers together work and approaches that tackle the problem of learning when the data distribution changes over time, and where knowledge fusion over never-ending streams of data needs to be accounted for. Consequently, CL is the paradigm to deal with catastrophic forgetting [102, 47].

For convenience, we can empirically split the data stream into several subsections temporally bounded we call tasks. We can then observe what we learn or forget when learning a new task. Even if there is no mandatory constraint on a task, a task often refers to a particular period of time where the data distribution may (but not necessarily) be stationary, and the objective function constant. Tasks can be disjoint or related to each other, in terms of learning objectives, and depending on the setting.

One solution to Continual Learning would be saving all data, shuffle it, and come back to a traditional machine learning setting. Unfortunately, in this case, this is not always possible nor optimal. Here are several examples of settings where continual learning is necessary:

- You have a trained model, you want to update it with new data but the original training data was discarded or you do not have the right to access it any longer.
- You want to train a model on a sequence of tasks but you can not store all your data or you do not have the computational power to retrain the model from all data (e.g., in an embedded platform).
- You want an agent to learn multiple policies but you do not know when the learning objective changes nor how.
- You want to learn from a continuous stream of data that may change through time but you do not know how and when.

In order to handle such settings, representations should be learned in an online manner [87]. As data gets discarded and has a limited lifetime, the ability to forget what is not important and retain what matters for the future are the main issues that continual learning targets and focuses on.

From a robotics point of view, CL is the machine learning answer to developmental robotics [93]. Developmental robotics is the interdisciplinary approach to the autonomous design of behavioural and cognitive capabilities in artificial agents that directly draws inspiration from developmental principles and mechanisms observed in children's natural cognitive systems [18, 93][1].

In this context, CL must consist of a process that learns cumulative skills and that can progressively improve the complexity and the diversity of tasks handled.

Autonomous agents in such settings learn in an open-ended [36] manner, but also in a continual way. Crucial components of such developmental approach consist of learning the ability to autonomously generate goals and explore the environment, exploiting intrinsic motivation [113] and computational models of curiosity [112].

We propose a framework to link continual learning to robotics. This framework also sets the opportunities for continual learning to have a framed mathematical formulation to present approaches in a clear and systematic way.

First we present the context and the history of continual learning. Secondly, we aim at disentangling vocabulary around continual learning to have a clear basis. Thirdly, we introduce our framework as a standard way of presenting CL approaches to help transfer between different fields of continual learning, especially to robotics. Fourthly, we present a set of metrics that will help to better understand the quality and shortcomings of every family of approaches. Finally, we present the specifics and opportunities of continual learning in robotics that make CL so crucial.

We kept the sections definitions, framework, strategies and evaluation general enough to both robotics and non-robotics domains. Nevertheless, the last section, *Continual Learning for Robotics* (Section 6) benefits from the content of previous sections to present the specificities of Continual Learning in the field of robotics.

2 Definition of Continual Learning

Given a potentially unlimited stream of data, a Continual Learning algorithm should learn from a sequence of partial experiences where all data is not available at once. A non-continual learning setting would then be when the algorithm can have access to all data at once and can process it as desired. Continual learning algorithms may have to deal with imbalanced or scarce data problems [154], catastrophic forgetting [47], or data distribution shifts [50].

As a more constrained version of on-line learning, CL needs to implicitly or explicitly account for knowledge fusion at different levels over time. Firstly, CL is required to support data-level fusion and, at the same time, be able to preserve learned knowledge from data that may disappear (e.g. due to inability to re-process certain data, due to the *right to be forgotten* of EU GDPR[2], or simply legacy reasons like for medical records). CL requires as well fusion at model level, since

[1] Synonyms of Developmental Robotics include *cognitive developmental robotics*, *autonomous mental development* as well as *epigenetic robotics*

[2] Art. 17 GDPR – Right to erasure (*right to be forgotten*) https://gdpr-info.eu\T1\guilsinglrightart-17-gdpr

different tasks to be learned may require different model architecture components that in the end must act as one.

Lastly, fusion needs also to occur at the knowledge or conceptual level, since memorization of raw data has to be avoided, but without incurring catastrophic forgetting. We consider continual learning a synonym of *Incremental Learning* [50, 125], *Lifelong Learning* [24, 159] and *Never Ending Learning* [19, 107]. For the sake of simplicity, in the remaining of the article we refer to all Continuous, Incremental and Lifelong learning synonyms as Continual Learning (CL).

In this section we first present the history and motivation of continual learning, then we present several definitions of terms related to CL and, finally, we present challenges addressed by CL in machine learning.

2.1 History and Motivation

The concept of learning continually from experience has always been present in artificial intelligence and robotics since their birth [162, 169]. However, it is only at the end of the 20^{th} century that it has began to be explored more systematically. Within the machine learning community, the lifelong learning paradigm has been popularized around 1995 by [159] and [128], while the robotics field only later catches up with a renewed interest in developmental robotics [93].

Between the end of the 90s and the first decade of the 21^{st} century, sporadic attention has been devoted to the topic within the supervised, unsupervised and reinforcement learning domains. However, despite the first pioneering attempts and early speculations, research in this area has never been carried out extensively until the recent years [114, 24]. We argue that this is because there were more complex and fundamental problems to solve and a number of additional constraints:

- *Lack of systemic approaches*: Machine learning research for the past 20 years has focused on statistical and algorithmic approaches on simple tasks (e.g., tasks where the distribution of data is assumed static). CL typically needs a systems approach that combines multiple components and learning algorithms in complex and dynamic tasks. The complexity of tasks and their multiple uses in continual learning greatly complicates training and evaluation procedures. Disentangling *"static"* learning performance from continual learning side effects is important for the very incremental nature of the research and to facilitate comparison between approaches in this area.

- *Limited amount of data and computational power*: Digital data is a luxury of the 21^{st} century. Before the big data revolution, collecting and processing data was a daunting task. Moreover, the limited amount of computational power available at the time did not allow complex and expensive algorithmic solutions to run effectively, especially in a continual learning setting which undoubtedly makes learning more complex by having to deal with multiple tasks at the same time, as well as having to incorporate the concept of time into the learning process.

- *Manually engineered features and ad-hoc solutions*: Before early 2000s and first works on representation learning, creating a machine learning system meant to handcraft features and finding ad-hoc solutions, which may differ significantly depending on the task or domain. Having a general algorithm with a more systematic approach seemed for a long time a very distant goal. Manually engineered features is also a clear limitation to achieve autonomy, as new tasks need to have the same features or re-engineered ones.

- *Focus on supervised learning*: creating labeled data is probably the slowest and the most expensive step in most ML systems. This is why learning continuously has been for a long time not a viable and practical option.

The relaxation of these constraints, thanks to recent advancements and results in machine learning research, as well as the rapid technological progress witnessed in the last 20 years, have open the door for starting tackling more complex problems such as learning continually.

We argue that the robotics community, which has always been intrigued by endowing embodied machines with lifelong and open-ended learning [36] of new skills and new knowledge, would highly benefit by the recent advances of ML in this area. Robotics applications in unconstrained environments, indeed, have always posed questions out of reach for previous machine learning techniques. On the other hand, CL developed in the context of robotics is involved in understanding

the role and the impact of the concept of "embodiment" in intelligent machines that learn and think like humans.

Learning, embodiment, and reasoning are presented as the three great families of challenges for robotics in [158]. We postulate that CL tackles the learning problem, taking into account the importance and constraints of embodiment. At best, CL would also benefit from reasoning in order to maximize the learning process. Thus, continual learning lies in the intersection of crucial robotics challenges.

Despite lifelong learning approaches existing in different ML disciplines (such as evolutionary algorithms for example [5, 6, 13, 7]), in the rest of the article we focus on recent continual learning developments in the context of gradient-based neural network and deep learning approaches. For a more detailed description of many other classic approaches to continual learning with shallow architectures we refer the reader to [24].

2.2 Terminology Clarification

In this section we aim at clarifying the distinction and similarities of continual learning with related topics and terms used in the literature.

Online learning

Online learning is a special case of CL [66] where updates are done on per single data point basis and therefore, the batch size is one. Online learning algorithms are suited to scenarios where information should be processed instantly, either to adapt the model to learn as fast as possible or because data can not be saved.

Few-shot Learning

Few shot learning [76, 42] is the ability to learn to recognize new concepts based on only few samples of them. It may be used for continual learning problems when the number of data points is very low. The extreme case of zero-shot learning consists of the ability to detect new classes while being trained with a disjoint set of classes [166].

Curriculum Learning

Curriculum learning [9] is a training process that proposes a sequence of more and more difficult tasks to a learning algorithm in order to make it able to learn, at last, a generally harder task. The sequence of tasks is designed in order to be able to learn the last one. Both CL and curriculum learning learn on a sequence of tasks (or partial experience). However, in curriculum learning, tasks are chosen in a way that makes possible to learn tasks of different complexity, by taking into account the difficulty of them, while in CL, tasks are not voluntarily chosen nor ordered. Furthermore, while the interest of curriculum learning ultimately lies into solving the last task, the continual learning objective is to be able to solve all tasks.

Meta-learning

Meta-learning [12] is a learning process that uses meta-data about past experiences, such as hyper-parameters, in order to improve its capacity to learn on new experiences. It also learns several different tasks; however, its goal is not learning without forgetting but to progressively improve the learning efficiency while learning on more and more tasks. It is also called "learning to learn", and it can or not be used in a continual learning setting.

Transfer learning

Transfer learning [122, 44, 177] is the ability to use what has been learned from a previous task on a new task. The difference with continual learning is that transfer learning is not concerned about keeping the ability to solve previous tasks. In computer vision, transferring what has been learned from a past environment to new environments would be often referred to as *domain adaptation* [118, 31].

Active Learning

Active learning is a special case of semi-supervised machine learning in which a learning algorithm is able to interactively query the user (or some other information source) to obtain the desired output labels for new data points [147, 148]. Active learning may be used in CL to query new examples and have control of the data the algorithm has access to.

2.3 Challenges Addressed by CL

In this section we describe the specific problems addressed by continual learning; the kind of problems that arise when data cannot be assumed i.i.d., and when the hypothesis that the data distribution is static is not valid.

2.3.1 Catastrophic Forgetting

Catastrophic forgetting [102, 47] refers to the phenomenon of a neural network experiencing performance degradation at previously learned concepts when trained sequentially on learning new concepts [102]. Since by definition the continual learning setting deals with sequences of classes or tasks, the catastrophic forgetting is an important challenge to be tackled. Catastrophic forgetting might also be referred to as *catastrophic interference*. The notion of interference is pertinent since the acquisition of new skills interferes with past skills by modifying important parameters.

2.3.2 Handling Memories

One of the main components that distinguishes two CL approaches is the way they handle memories. In order to deal with catastrophic forgetting, each strategy should find a way to remember what gradient descent will make forget. Continual learning needs a mechanism to *store* memories of past tasks, which can take very different forms. It is important to note that memories can be saved in different manners: as raw data, as representations, as model weights, regularization matrices, etc. An efficient memory management strategy should only save important information, as well as be able to transfer knowledge and skills to future tasks. In practice, it is almost impossible to know what will be important and what could be transferable in the future; a trade off should then be found between the precision of the information saved and the acceptable forgetting. This trade-off problem is known as the stability/plasticity dilemma [103].

An important challenge inherent to handling memories is to automatically assess them. Learning new tasks may lead to degradation of the memories. Furthermore, the memory process needs mechanisms to evaluate how the memories are degraded, i.e., how it forgets. As no more data and labels from past tasks may be available, this check-up might be very challenging.

2.3.3 Detecting Distributional Shifts

When the distribution is not stationary, a shift into the data stream is observed. When there is no external information concerning this shift, the CL model has to detect it, and account for fixing it by itself. An undetected shift in the data distribution will irrevocably lead to forgetting. Changes in the data distribution over time are commonly referred to as *concept drift*. This idea is related to online change detection algorithms [140, 109] or Bayesian surprise [156] in ML. Two kinds of concept drift are defined [50]: Virtual and real concept drift. Virtual concept drift concerns the input distribution only, and can easily occur, e.g., due to imbalanced classes over time. Real concept drift, on the contrary, is caused by novelty on data or new classes, and can be detected by its effect, on e.g., classification accuracy. However shift may also happen when the task change. In RL for example an agent may have to solve a new task. Then the shift is not exactly in the data distribution but in the supervision signal. Regardless of where exactly the shift happened it has to be detected to avoid catastrophic interference with non related skills or knowledge.

2.4 Learning Paradigms Orthogonal to Continual Learning

In this section we describe the relationship of continual learning with respect to the main three, generally acknowledged machine learning paradigms: supervised, unsupervised and reinforcement learning.

2.4.1 Supervised Continual Learning

Supervised learning is the machine learning problem of learning from input-output example pairs [135]. For each input-output pair (X_t,Y_t), the model should learn to predict Y_t from X_t. X_t is the input data, Y_t is the supervision signal. Supervised continual learning is a particular case where the data is not available all at once. The function should then be learned from a sequence of data points in order to be able to map data to labels at the end of the sequence for the whole dataset. Supervised Continual Learning approaches have been mostly focused on classification [92, 97, 73].

While the study of continual learning in this context may help disentangling the complexity introduced by algorithms that learn continually, in the context of robotics, the lack of supervision does not allow, most of the time, to apply directly supervised methods.

2.4.2 Unsupervised Continual Learning

Unsupervised learning refers to machine learning algorithms that do not have labels or rewards to learn from. In the context of robotics, unsupervised continual learning may play an important role in building increasingly robust multi-modal representations over time to be later fine-tuned with an external and very sparse feedback signal from the environment. In order to learn robust and adaptive representations with unsupervised learning, the main objective is to find suitable surrogate and meaningful learning signals, as robotics priors [64, 84], self-supervised models or curiosity driven techniques.

A particular unsupervised task learned in a continual learning setting is the generation of images. Image generation is achieved by training generative models to reproduce images from a dataset. In a CL setting, the distribution changes over time and the generative model should be able to produce at the end images from the whole distribution. This problem has been studied for various generative models (cf. section 4) as adversarial models [171, 81], variational auto-encoders [111, 124, 1, 41, 81] and standard auto-encoders [161, 178].

There is also a different relation between unsupervised learning and CL, since unsupervised models can be used to learn representations from vast amounts of data sources and can then generate such data (cf section 4.4). This capacity can then be used to perform CL for classification [172, 150, 161, 83] or reinforcement learning tasks [20]. Another use case is using data generation as a data augmentation strategy.

2.4.3 Continual Reinforcement Learning

Reinforcement Learning is a machine learning paradigm where the goal is to train an agent to perform actions in a particular environment in order to maximize the expected cumulative reward. In traditional RL, the world is modeled as a stationary MDP: i.e., fixed dynamics and states that can recur infinitely often [129][3]. Since in general, complex RL environments have no access to all data gathered at once, RL could often be framed as a CL situation. Moreover, RL borrows several tools used in CL models, such as approximating data to an i.i.d. distribution, via either *i)* setting multiple agents or actors to learn in parallel [99], or *ii)* using a replay buffer (or experience replay [108]), that is equivalent to a particular category of CL (rehearsal, see Section 4.3). An analogy of a popular stable method in RL is PPO algorithm [143], which constrains learning by using the Fisher information matrix to improve learning continually, in the same way as some CL strategies (e.g., EWC, see Section 4.2.1). Most of Continual Learning approaches in RL have been applied in simulation settings such as Atari games [73]. However, many approaches [160, 68, 6, 13] also tackle use cases on real robots.

3 A Framework for Continual Learning

Despite the rapidly growing interest in continual learning and mainly empirical developments of the recent years [114], very little research and effort has been devoted to a common formalization of algorithms that learn continually in dynamic environments. However, the availability of a common ground for thoroughly evaluating and understanding continual learning algorithms is essential to reduce ambiguities, enhancing fair comparisons and ultimately better advancing research in this direction.

Being able to better compare and evaluate continual learning strategies, while still being general enough to overlook implementation-dependent details over different learning paradigms, becomes essential. This is specially true when targeting deployment of CL paradigms in real-word applications, such as robotics. Nowadays, despite the existence of a basic set of shared practices, many are the fundamental questions often overlooked in recent continual learning research. For example, questions about the data availability during training and evaluation, the amount of supervision with respect to the tasks separation and composition, as well as common but biased assumptions on the nature of the data among others. A list of questions of interest we would like to address and report are the following:

(a) Data Availability

[3]This MDP assumption was recognized and first removed in [129]

- $\boldsymbol{Q_1}$: *Does some data need to be stored? if yes, how and what for? (e.g. regularization, re-training, validation)?*
- $\boldsymbol{Q_2}$: *Is the algorithm tuned based on the final performance? I.e. is it possible to go back in time to improve performance?*
- $\boldsymbol{Q_3}$: *Are data distributions assumed i.i.d. at any point?*
- $\boldsymbol{Q_4}$: *Is each task assumed to be encountered only once?*

(b) Prior Knowledge

- $\boldsymbol{Q_5}$: *Is the continual learning algorithm agnostic with respect to the structure of the training data stream? (e.g. number of classes, numbers of tasks, number of learning objectives...)*
- $\boldsymbol{Q_6}$: *Does the approach need a pretrained model for the CL setting? If so, what is the new knowledge that needs to be acquired while learning continually?*

(c) Memory and Computational Constraints

- $\boldsymbol{Q_7}$: *How much available memory does the algorithm require while learning? Does the memory capacity requirement changes as more tasks are learned?*
- $\boldsymbol{Q_8}$: *Is the continual learning algorithm constrained in terms of computational overhead for each learning experience? Does the computational overhead increase over the task sequence?*
- $\boldsymbol{Q_9}$: *Is the continual learning algorithm agnostic with respect to the data type? (e.g. images, video, text,...)*
- $\boldsymbol{Q_{10}}$: *Is the continual learning algorithm able to handle situations where there is not enough time to learn?*

(d) Amount/Type of Supervision

- $\boldsymbol{Q_{11}}$: *In the presence of multiple tasks, is the task label available to the algorithm during the training phase? And during evaluation?*
- $\boldsymbol{Q_{12}}$: *Are all the data labeled? or only the first training set? Can the user provide sparse label/feedback (e.g. active learning) to correct the system errors?*

(e) Performance Expectation

- $\boldsymbol{Q_{13}}$: *What is expected from the algorithm to remember at the end of the full stream? Is it acceptable to forget somehow, when task, context or supervision change?*

To summarize these questions, in any new CL algorithm proposition, it is fundamental to clearly describe the data stream, its use, the algorithm functioning, its assumed prior knowledge , and its requirements in terms of supervision, memory and computation.

We will now propose a comprehensive and detailed framework to help distill and disentangle different approaches in different continual learning settings and help answer these questions.

Early theoretical attempts to formalize the CL paradigm are found in [129] as a combination between reinforcement learning and inductive transfer. More general framework approaches include the one on non i.i.d. tasks of [119]. As in [119], we assume CL is tackling a probably approximately correct (PAC) learnable problem in the approximation of a target hypothesis h^* as well as learning from a sequence of non i.i.d. training sets. Our framework could also be seen as a generalization of the one proposed in [92], where learning happens continuously through a *continuum* of data and a "task supervised signal" t may be provided along with each training example.

In continual learning data can be conveniently seen as drawn from a sequence of distributions D_i, and thus the need to redefine a CL framework taking into account this important property is defined as follows.

Definition 1 ***Continual Distributions and Training Sets***

In Continual Learning, $\mathcal{D}$ is a potentially infinite sequence of unknown distributions $\mathcal{D} = \{D_1, \ldots, D_N\}$ over $X \times Y$, with X and Y input and output random variables, respectively. At time i a training set Tr_i containing one or more observations is provided by D_i to the algorithm.

As the framework hereby proposed is supposed to be general enough to cover the orthogonal and classical unsupervised, supervised and reinforcement learning approaches, Tr_i, as better detailed in Definition 3, is a collection of training observations/data samples that act as signal of the joint distribution to be learned.

Definition 2 ***Task***
A task is a learning experience characterized by a unique task label t and its target function $g^*_{\hat{t}}(x) \equiv h^*(x, t = \hat{t})$, i.e., the objective of its learning.

It is important to note that the tasks are just an abstract representation of a learning experience represented by a task label. This label helps to split the full learning experience into smaller learning pieces. However, there is not necessarily a bijective correspondence between data distributions and tasks.

Definition 3 ***Continual Learning Algorithm*** *Given h^* as the general target function (i.e. our ideal prediction model), and a task label t, a continual learning algorithm A^{CL} is an algorithm with the following signature:*

$$\forall D_i \in \mathcal{D}, \qquad A^{CL}_i : \ < h_{i-1}, Tr_i, M_{i-1}, t_i > \rightarrow < h_i, M_i > \tag{1}$$

Where:

- *h_i is the current hypothesis at timestep i, or, practically speaking, the parametric model learned continually.*
- *M_i is an external memory where we can store previous training examples or partial computation not directly related to the parametrization of the model.*
- *t_i is a task label, that can be used to disentangle tasks and customize the hypothesis parameters. For simplicity, we can assume N as the number of tasks, one for each Tr_i.*
- *Tr_i is the training set of examples. Each Tr_i is composed of a number of examples e^i_j with $j \in [1, \dots, m]$. Each example $e^i_j =< x^i_j, y^i_j >$, where y^i is the feedback signal and can be the optimal hypothesis $h^*(x, t)$ (i.e., exact label y^i_j in supervised learning), or any real tensor (from which we can estimate $h^*(x, t)$, such as a reward r^i_j in RL).*

It is worth pointing out that each D_i, can be considered as a stationary distribution. However, this framework setting allows to accommodate continual learning approaches where examples can also be assumed to be drawn non i.i.d. from each D_i over $X \times Y$, as in [50, 56].

Definition 4 ***Continual Learning scenarios*** *A CL scenario is a specific CL setting in which the sequence of N task labels respects a certain "task structure" over time. Based on the proposed framework, we can define three different common scenarios:*

- *Single-Incremental-Task (SIT):* $t_1 = t_2 = \dots = t_N$.
- *Multi-Task (MT):* $\forall i, j \in [1, .., n]^2, i \neq j \implies t_i \neq t_j$.
- *Multi-Incremental-Task (MIT):* $\exists\ i, j, k : \ t_i = t_j$ *and* $t_j \neq t_k$.

Table 1 illustrates an example to clarify the definition of SIT, MT and MIT.

An example of Single-Incremental-Task (SIT) scenario is an ordinary classification task between cats and dogs, where the distribution changes through time. First, there may only be input images of white dogs and white cats, and later only black dogs and black cats. Therefore, while learning to distinguish black cats from black dogs the algorithm should not forget to differentiate white cats from white dogs. The task is always the same, but the concept drift might lead to forgetting.

However, in a classification setting, a Multi-Task (MT) scenario would first consist of learning cats versus dogs, and later cars versus bikes, without forgetting. The task label changes when the classes change, and the algorithm can use this information to maximize its continual learning performance. The Multi-Incremental-Task (MIT) is the scenario where the same task can happen several times in the sequence of tasks, but such task is not the only existing one.

In any learning problem (be it classification, RL or unsupervised learning), the ability to adapt to new concepts to be learned (from the PAC ML framework [163]), as well as new instances of

Table 1: Example: Sequential task labels (corresponding to different distribution $D_i \in \mathcal{D}$) to reflect differences among CL categorization w.r.t. number and unicity of tasks for SIT, MT and MIT. Notice that a MIT setting requires breaking the constraint definition of SIT but also breaking the constraint definition of MT, i.e., it corresponds to the case where not all the tasks are considered having the same *ID*, and not all the task are considered distinct.

Task ID/Session	**CL settings**		
Task ID	**SIT**	**MT**	**MIT**
t_1	0	1	0
t_2	0	2	1
t_3	0	3	0
...	...	...	...
t_i	0	i	...

each concept, should be accounted. This is the objective of the next definition where we formally set three different settings an algorithm is required to manage, as they can have very high impact on the algorithm performance.

Definition 5 ***Task label and concept drift scenarios*** *The task label can specify different assumptions made in a continual learning scenario. We can define three main categories of task label assumptions regarding concept drift:*

- *No task label: Changes in the distribution are not signaled by any task label. The task is always the same (equivalent to SIT scenario).*
- *Sparse task label: Changes in the distribution are sparsely signaled by the task label. There are several tasks but changes in distribution may as well happen inside a task.*
- *Task label oracle: Every change in the data distribution is signaled by the task label, which is given.*

We illustrate the different scenarios in Figure 1.

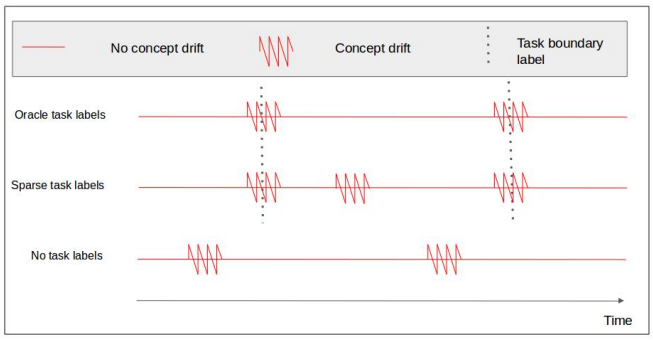

Figure 1: Task label and concept drift: illustration of the different scenarios.

Definition 6 ***Availability of task label.*** *When a task label is provided, it is worth distinguishing among two different cases:*

- *Learning labels: Task labels are provided for learning only. At test time, inference should be done without knowing from which task a data point is coming from.*
- *Permanent labels: The task labels are provided for learning, and it is assumed they will also be provided at test time for inference.*

Definition 7 ***Content Update Type*** *The nature of the data samples or observations contained in each Tr_i can be conveniently framed in three different categories:*

- *New Instances (NI): Data samples or observations contained in the training set at time-step i relate to the same dependent variable Y used in the past.*
- *New Concepts (NC): Data samples or observations contained in the training set at time-step i relate to a new dependent variable Y to be learned from the model.*
- *New Instances and New Concepts (NIC): Data samples or observations contained in the training set at time-step i relate to both, already encountered dependent variables, and new ones (Y).*

In order to exemplify the concept of *Content Update Type* defined in Definition 7, let us recover the aforementioned example of classification. If an algorithm learns the *cat vs dogs* classification problem on a dataset and then new images of cat vs dogs are provided to the algorithm, we are then in a *New Instances* case (NI), we have new data but no new concepts. If the new instances were of different classes (e.g. cars vs bikes) we then would face the *New Concepts* case (NC). The new instances and new concepts case would then have been a mix of both new images of known and new classes.

If a CL algorithm uses a network pretrained on a dataset, the features of such dataset will need to be accounted for as one more task or the same, depending on the distribution of new instances and new classes according to definitions 4 and 7. In other words, using a pretrained model is similar to assume there is a task already learned by the model, and the new learning experiences of the algorithm are just a continuum of learning curricula. If there is any intersection between the pretraining and the new tasks, it should be reported in the setting description. The pretraining effect can then be estimated with the metrics proposed in Section 5.2.

Constraints

Constraint 1 *For every step in time, the number of current examples contained in the memory is lower than the total number of previously seen examples*[4]*:* $\forall i \in [1, ..., n], |M_i| \ll \left| \bigcup_{i=1}^{i-1} Tr_i \right|$

Constraint 2 *Memory and computation for each iteration step i are bounded. Given two functions* $ops()$ *and* $mem()$ *that compute the number of operations and memory occupation required by* A_i^{CL}*, two reasonably small values* max_ops *and* max_mem *should exist, such that, for each i,* $ops(A_i^{CL}) < max\_ops$ *and* $mem(h_{i-1}, M_{i-1}) < max\_mem$*.*

max_ops and max_mem are the max throughput, in number of operations, and the max memory capacity of the system running A_i^{CL}. Having a memory and computational bounds for each iteration i is an important constraint for a continual learning algorithm. The reason is that the number of training sets Tr_i can potentially be unlimited, and thus, computation and memory should not be proportional to the number of hypothesis updates h_i over time. A finite upper bound should exist and be considered, especially with $n \to \infty$.

Relaxation and desiderata Given the difficult setting and the additional constraints imposed by Continual Learning with respect to the classic "static" setting, many researchers in the recent literature have proposed new CL strategies in slightly relaxed [137, 73, 95, 92] yet reasonable settings:

Relaxation 1 *Memory relaxation: Removes the fixed memory bound constraint over* $ops()$ *and* $mem()$*.*

Relaxation 2 *Computation relaxation: Removes the fixed computational bound constraint* $ops(h_i) < max\_ops$*.*

In both cases we assume that for practical applications, a finite (and reasonable) number of tasks N are encountered, hence, for many settings with a generous memory and computational bounds, many continual learning strategies that, in terms of complexity and memory usage, grow somehow proportional to the number of training sets Tr_i may still be a viable option, especially if they can guarantee better performance.

Having defined a formal framework for CL, we can therefore highlight a number of desiderata:

[4] I.e., if we could fit all previous examples in memory M, it would become a problem of scarce interest for the CL community, given that re-training the entire model h_i from scratch would be always possible [66]

Desideratum 1 *Storage-Free Continual Learning: Avoid the use of external memory M to store raw data.*

Desideratum 2 *Online Continual Learning: Limit the size of each training set, moving towards online learning so that $|Tr_i| = 1$.*

Being able to learn without storing any raw data would mean a large step towards continual learning. In fact, getting rid of storing raw data means that the learning algorithm is able to extract information from the current task that may be not only useful and accurate for the actual task, but also transferable for the future.

In our biological counterparts, namely the brain, a system-level consolidation process is often thought to take place, where memories are encoded, stored and than retrieved for rehearsal purposes [32]. However, the idea of storing high-dimensional perceptual data appears impractical given the incredible amount of information flowing into our brain every day from our multi-modal senses. Being able to process data online as well, is an important desideratum especially for reducing adaptation time and operational memory usage in an embedded or robotics setting.

Desideratum 3 *Task indicator free Continual Learning: Learning continually without help of an external signal t indicating the current task, in particular at test time, is strongly desirable.*

4 Continual Learning Strategies

In this section we present a summary of the most popular continual learning strategies in the literature (see Fig. 2). For a more in depth overview, we refer the reader to the recent overview in [114] that additionally exposes the bio-inspired aspects of existing continual approaches.

4.1 Dynamic Architectures Approaches

The architecture of learning models has a strong influence on how they learn. One approach to CL is to modify dynamically the architecture of a model to make it learn new concepts or skills without interfering with old ones. We present two types of dynamic architectures. Firstly, when the changes in the architecture are explicit; and secondly, when changes are implicit architectural changes by freezing weights. We also present an important architectural approach to CL: dual memory models.

4.1.1 Explicit Architecture Modification

Explicit dynamics architecture gather all methods that add, clone or save parts of parameters of the models to avoid catastrophic forgetting.

Progressive neural networks [137] is one of the first approaches within this paradigm for deep neural networks. For each new task to be learned, a new model is created connected to all past ones. The goal of this new model is to learn the new task by using what was already learned by previous models, and so develop the new skills needed. At test time, the proposed method needs to input data to all the neural networks previously created, and needs to know the task index to pick the right output. Because the weights are used to connect neural networks together, the growth of parameters is quadratic w.r.t. the number of tasks. This growth is generally to be prevented. Instead, layers may be dynamically expanded in a single network without the need of re-training or freezing previously learned parameters, improving model capacity over time [167].

Another type of dynamic architecture strategy consists of dynamically adding neurons for new tasks. As an example, output layers can be added in order not to change output parameters from previous tasks as in LWF approach [87]. This method ensures that the output layer will not be modified; however, as the feature extraction layers are shared between tasks, some parameters risk to be modified and forgotten. In addition, at test time, the method needs the task label.

It is worth mentioning that we consider as *dynamic architecture*, those approaches that adapt their architecture specifically with the aim of not forgetting, while similar mechanisms can be used for other purposes[5].

[5]If the architecture is changed without this objective, it is not considered as part of the CL approach. As an example, when new classes are available, we might choose to make the output size grow to handle these, without making it as a way to not forget.

4.1.2 Implicit Architecture Modification

Implicit architecture modification is the use of model adaptation for continual learning without modifying its architecture. This adaptation is typically achieved by inactivating some learning units or by changing the forward pass path.

We categorize the fact of dynamically freezing weights as an implicit dynamic architecture approach. It is implicit because the architecture of the model does not change; however, the capacity of the model to learn new tasks does in an inevitable way.

Freezing weights consist of choosing some weights at the end of a task that will no more change in the future. The backward pass will not be able to tune them anymore; however, they can still be used in the forward pass. This method assures that these weights will not forget, and tries to keep enough free parameters to learn in the future [96, 95, 146]. The difficulty lies in freezing enough weights to remember, but not too much to still be able to learn new skills. The way weight freezing is implemented in PackNet [96], Piggypack [95] or HAT [146] is by defining a special mask for each task that is used to both protect weights when new tasks are learned, and to define which weights to use at inference time for a given task. The use of masks to freeze important weights can be referred to as hard attention process [146]. Weight freezing can also be used to keep the decision boundary of the output unchanged [65].

An alternative to a weight freezing when tasks change is to define a dynamics path inside the model in order to use a specific path for a specific task and not modify already learned weights. This is the idea exploited in *PathNet* [43].

The use of implicit architecture modifications is not incompatible with explicit architecture modification as it is shown in [95, 146].

4.1.3 Dual Architectures

Dual approaches characterize architectures that are split in two models. One model is used in order to learn the actual task and should be easily adaptable, while the second model is used as a memory of past experiences. This approach can be linked to interactions between the hippocampus and neocortex to avoid catastrophic interference in mammals [101]. The stable network plays the role of the neocortex, and the flexible one plays the role of hippocampus [48, 50, 51, 97].

The use of dual architecture is explicit in many bio-inspired approaches such as [48, 51, 115, 154, 70]. Dual architectures are extended in [154] with the addition of an embedding model, and then, continual learning happens in the embedding space. The dual architecture can also be extended to more than two components, as in FearNet [70], which takes inspiration from the basolateral amygdala from the brain to add a third component that is able to choose between the flexible and the stable memory for recall.

4.2 Regularization Approaches

4.2.1 Penalty Computing

Regularization is a process of introducing additional information in order to prevent overfitting [14]. In the context of Continual Learning, the model should not overfit a new problem because it would make it forget it's previous skills. The regularization approaches in continual learning consist in modifying the update of weights when learning in order to keep memory of previous knowledge.

Basic regularization techniques that could be used for CL are weight sparsification, dropout [53], and early stopping [97]. These simple regularization techniques reduce the chance of weights being modified, and thus decrease the probability of forgetting. More complex methods consist in searching for important weights inside the models and protect them afterwards to prevent forgetting. The Fisher matrix can be used to estimate the importance of weights and produce an adapted regularization as for Elastic Weight Consolidation (EWC) approach [73]. For efficiency purpose, EWC only use the diagonal of the Fisher matrix to estimate importance. [131] proposes an alternative to get a better estimation of the Fisher matrix using the Kronecker factorization. EWC approach needs to have clear task delimitation to compute Fisher matrix at the end of the task, but Synaptic Intelligence (SI) [176] extended the method in an online learning fashion to relax this constraint. [80] propose to use a regularization method called *incremental moment matching* to overcome catastrophic forgetting. This method saves the moment posterior distribution of neural networks weights from past tasks and uses it to regularize learning of a new task. Two different

declinations of this method are proposed: one with the use of first order moment *IMM-mean* and one with second order moment *IMM-mode*.

Another method to apply regularization for continual learning is the use of *Conceptor* [62, 57]. Conceptor are memory mechanism that store learned patterns and representation. They are used to guide the gradient of the loss function to prevent forgetting and then favor modification for some weights and penalize others.

The regularization methods have been shown to be efficient in reinforcement learning [73], classification [73, 131, 176, 57] and also generative models [111, 145]. A limitation is that after several tasks the model may saturate because of a too high regularization, and finding a good trade-off between regularization that allows learning without forgetting may be hard.

4.2.2 Knowledge Distillation

Distillation techniques were introduced by [60] in order to transfer knowledge from neural network A to neural network B. The idea is that after A has learned to solve a task, we want B to share this skill with A. We then forward the same input to both A and B and impose B to have the same output as A. Distillation should be more efficient than retraining B because A produces a soft-target that helps B to learn faster. In order to apply this method for continual learning, after network A learned to solve the first task, and while B is learning the second one, we distill knowledge from A to B. In the end, B should be able to solve both tasks. This and related methods have been used in various approaches [171, 144, 48, 136, 68, 160, 33, 105]. A drawback of distillation is that it generally needs to preserve a reservoir of persistent data learned for each task in order to apply distillation from a teacher model to a student model. Distillation can also be used to transfer policy learning from one model to another [136].

4.3 Rehearsal Approaches

Rehearsal approaches gather all methods that save raw samples as memory of past tasks.

These samples are used to maintain knowledge about the past in the model. Ideally, those samples are carefully chosen in order to be representative of past tasks; by default, they can be randomly chosen.

The initial strategy is to save the representative samples and incorporate them in the new training set [125, 81]. In the second article samples are chosen randomly for continual learning of generative models but in [125] the set is carefully sorted in order to keep the most representative samples into a coreset. This process allows to dynamically adapt the weights of the feature extractor and strengthen the network connections for memories already learned without forcing to keep previous weights.

However, the coreset can also be used for regularization purpose and not just to be replayed from time to time along with new data in the learning process.

For example, the coreset can be used for distillation in [132] and in A-LTM (Active Long Term Memory Networks) [48] or to regularize the gradient when learning new tasks as in GEM (Gradient Episodic Memory) [92] and A-GEM (Averaged Gradient Episodic Memory) [23]. Coresets have also been used to regularize the continual learning of a generative model in the CloGAN approach [130]. In a bayesian learning setting the coreset can be incorporated into the prior to regularize learning update as in [111]. The authors experimented the use of a coreset to create a variational continual learning model (VCL).

The disadvantage of rehearsal approaches is the utilization of a separate memory of raw and unprocessed data which is a vanilla way of saving knowledge that does not respect data privacy. Nevertheless it ensure that the memories are not degraded through time.

4.4 Generative Replay

Instead of modeling the past from few samples as it is done in *Rehearsal* approaches, *Generative Replay* approaches train generative models on the data distribution. Therefore, they are able to afterwards sample data from past experience when learning on new data. By learning on actual data and artificially generated past data, they ensure that the knowledge and skills from the past is not forgotten. These methods have also been associated with the term *pseudo-rehearsal* [132] or *Intrinsic Replay* [37]. They could be understood as methods that perform *regeneration* of samples or internal states, and thus, they can be associated with model-based learning, where the model

learns the data distribution of past experiences. The generative models is generally a GAN [52] as in [172, 81, 150] or an auto-encoder as in [37, 70, 20, 69].

A classical method implementing a generative replay normally makes use of dual models [69, 150, 172, 41, 70]. One frozen model generates samples from past experiences and another learns to generate and classify actual samples in addition to the regenerated ones. When a task is over, we replace the frozen model by the actual one, freeze it, and initialize a new model to learn next task.

Generative Replay models can be categorized into two different approaches: "Marginal Replay" and "Conditional Replay" [83]. Techniques using *Marginal Replay* make use of standard generative models, while *Conditional Replay* are a particular case of the former where the generative model is conditional. Conditional models can generate data from a specific condition, e.g. a class or a task. In continual learning, it allows then to choose from which past learning experience we want to generate. It is important for example to balance data in generated replay [83].

While most of the Generative Replay based approaches are meant to solve classification tasks [70, 69, 150, 172, 130], some models use it for unsupervised learning [81, 171] or reinforcement learning [20].

4.5 Hybrid Approaches

Most CL approaches have an implicit dual architecture strategy, as they always have a slow learning and a fast learning mechanisms to learn continually. For example, in rehearsal approaches (Section 4.3) the stable model role is played by a memory that stores samples, in generative replay approaches (Section 4.4) a generative model plays the role of stable model, in some regularization approaches (Section 4.2.1) the stable model is played by the Fisher matrix which saves important weights.

Moreover, most of continual learning approaches do not rely on a single strategy to tackle catastrophic forgetting. As stated in previous sections, each approach offers advantages and disadvantages, but most of the times, combining strategies allows to find the best solutions. We summarize in Table 2 and Figure 2 the different approaches cited and the strategies they propose.

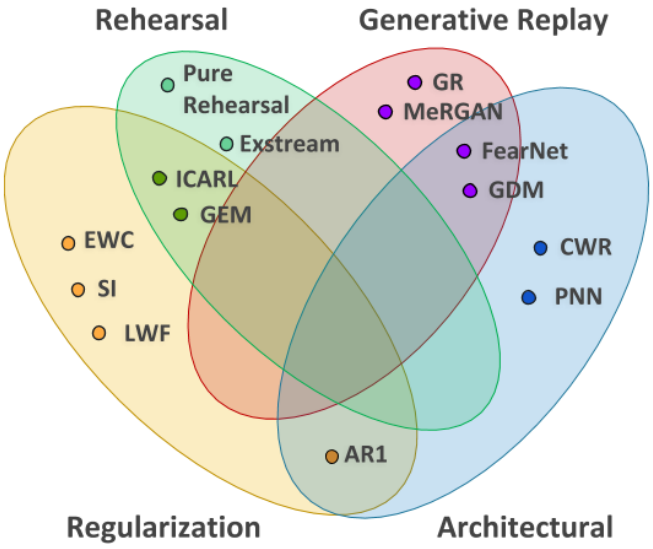

Figure 2: Venn diagram of some of the most popular CL strategies w.r.t the four approaches illustrated in Section 4: CWR [91], PNN [137], EWC [73], SI [176], LWF [87], ICARL [125], GEM [92], FearNet [70], GDM [115], ExStream [55], Pure Rehearsal, GR [150], MeRGAN [171] and AR1 [97]. Rehearsal and Generative Replay upper categories can be seen as a subset of replay strategies. Better viewed in color.

5 Evaluation of Continual Learning Algorithms

Before applying CL solutions to autonomous agents, they should be experimented and evaluated in simulation or toy examples. It is crucial to have a set of good evaluation metrics and benchmarks to

Table 2: Continual Learning Main Strategies

References	Regularization	Rehearsal	Architectural	Generative-Replay
Zhou et al. [178]			✓	
Goodfellow et al. [53]	✓			
Lyubova et al. [94]		✓		
Rusu et al. [136]	✓			
Camoriano et al. [17]	✓	✓		
Furlanello et al. [48]	✓		✓	
Li et al. [87] (LwF)	✓		✓	
Rusu et al. [137] (PNN)			✓	
Jung et al. [65]	✓		✓	
Aljundi et al. [3]			✓	
Rebuffi et al. [125] (Icarl)	✓	✓		
Kirkpatrick et al. [73] (EWC)	✓			
Fernando et al. [43]			✓	
Lee et al. [80]	✓			
Lee et al. [174]	✓			
Triki et al. [161]	✓			
Seff et al. [145]	✓			
Shin [150] (DGR)				✓
Velez et al. [165]	✓			
Lopez-Paz et al. [92] (GEM)	✓	✓		
Zenke et al. [176] (SI)	✓			
Nguyen et al. [111] (VCL)	✓	✓	✓	
Ramapuram et al. [124]	✓			✓
Mallya et al. [96]			✓	
Kamra et al. [69]				✓
Draelos et al. [37]				✓
Serra et al. [146]	✓			
Mallya et al. [95]			✓	
Parisi et al. [115] (GDM)	✓		✓	✓
He et al. [57]	✓		✓	
Hayes et al. [55]		✓		
Wu et al. [172]		✓		✓
Ritter et al. [131]	✓			
Schwarz et al. [144]		✓		
Maltoni et al. [97]	✓		✓	
Achille et al. [1]			✓	✓
Wu et al. [171] (MeRGAN)	✓			✓
Dhar et al.	✓			
Lesort et al. [81]				✓
Caselles-Dupré et al. [20]				✓
Riemer et al. [127] (MER)	✓	✓		
Rios et al. [130] (CloGAN)	✓	✓		✓
Lesort et al. [83]				✓
Sprechmann et al. [154]		✓	✓	
Kemker et al. [70] (FearNet)			✓	✓
Chaudhry et al. [23]	✓	✓		
Kalifou1 et al. [68]	✓	✓		

assess if the approaches are scalable to real problems or may not solve harder ones. In this section we summarize existing evaluation methods and benchmarks and highlight some of them we believe worth using when targeting the deployment of practical CL applications.

Table 3: Benchmarks and environments for continual learning. For each resource, paper use cases in the NI, NC and NIC scenarios are reported.

Benchmark	NI	NC	NIC	Use Cases
Split MNIST/Fashion MNIST		✓		[83, 81, 57, 130]
Rotation MNIST	✓			[92, 83, 127]
Permutation MNIST	✓			[53, 73, 43, 150, 176, 83, 57, 127]
iCIFAR10/100		✓		[125, 97, 70]
SVHN		✓		[71, 145, 130]
CUB200	✓			[80]
CORe50	✓	✓	✓	[91, 115, 97]
iCubWorld28	✓			[116, 90]
iCubWorld-Transformation		✓		[117, 16]
LSUN		✓		[171]
ImageNet		✓		[125, 95]
Omniglot		✓		[77, 144]
Pascal VOC		✓		[104, 151]
Atari	✓			[136, 73, 144]
RNN CL benchmark			✓	[153]
CRLMaze (based on VizDoom)	✓			[89]
DeepMind Lab	✓			[99]

5.1 Evaluation Protocols and Benchmarks

In continual learning, the difficulty of learning on a sequence of tasks is first of all dependant on the difficulty of each of the tasks separately. If a task is difficult to learn, a model will have to deeply modify its weights. If those weights contain knowledge from previous tasks, there is a high probability they will be degraded. On the other hand, the risk of forgetting is also dependant on the likelihood of tasks occurring. Indeed, after learning a task T_t, it is easier for a neural network to learn a radically different task T_{t+1} without forgetting, than learning a task T_{t+1} with similarities to T_t [41].

There are several kinds of similarities in a sequence of tasks:

- Similarities in learning objectives: They occur when the objective is similar from task to task. For example, in a classification setting, when the same classes are used from one task to another (e.g. Permuted MNIST), or in RL, the same tasks need to be achieved in different environments.
- Similarities in features: the features from task to task are the same or very similar (e.g. Rotation MNIST).

Beyond the similarity among tasks and the learnability of each task, the availability of data is primordial to evaluate the difficulty of a benchmark. For convenience, most of the classical benchmarks assume that each task is available long enough to learn a satisfying solution. Nevertheless, even when there is no constraint on the time to learn a task, data from the past can not be available again in the future. In several approaches, past data is used for model selection, however using the performance obtained on task T_t to fine-tune a model that will learn on T_0 violates temporal causality [120]. Data might be saved for later use as in rehearsal approaches, but this must be done before moving on to the next task.

Most CL benchmarks are benchmarks adapted from others fields, for instance:

- **Classification**: MNIST [79], Fashion-MNIST [173], CIFAR10/100 [74], Street View House Numbers (SVHN) [110], CUB200 [168], LSUN [175], ImageNet [75], Omniglot [77] or Pascal VOC [39] (object detection and segmentation).
- **Reinforcement Learning**: Arcade Learning Environment (ALE) [8] for Atari games, SURREAL [40] for robot manipulation and RoboTurk for robotic skill learning through imitation [98], *CRLMaze* extension of VizDoom [89] and DeepMind Lab [99].
- **Generative models**: Datasets that prevail in this domain are the same as those used in classification tasks.

These datasets are then split, artificially modified (e.g., with image rotations or permutation of pixels) or concatenated together to create sequences of tasks and build a continual learning setting. As an example, permuted MNIST [73] and rotated MNIST [92] are continual learning datasets artificially created from MNIST. Another possible continual learning scenario is the use of naturally non i.i.d. datasets (e.g. NICO [58]) or learning sequentially different datasets either on the same input space [80, 146] or in a multi-modal fashion [71]. However, only few datasets, such as CORe50 [91] or [153], are specifically built with continual learning in mind.

In robotics, numerous datasets are often recorded in a online fashion through video. Therefore, they are suitable to evaluate continual learning algorithms. As an example, those proposed by [116, 117, 4] are composed of sequences of images captured during robotics object manipulation; they are used for classification and detection algorithms. A summary of the main datasets and examples of their applications can be found in Table 3.

5.2 Continual Learning Metrics

Following the evaluation of an algorithm on a challenging benchmark, we should make sure that the evaluation criteria are rigorous and cover the whole aspect of the full learning problematic. It is not enough to observe good final accuracy on an algorithm to know if it is transferable to a robotics settings. We should also evaluate how fast it learns and forgets, if the algorithm is able to transfer knowledge from one task to another, and if the algorithm is stable and efficient while learning. In this section we gather a set of metrics to rigorously evaluate a CL approach.

For a rigorous evaluation, we can assume to have access to series of test sets Te_i. The aim is to assess and disentangle the performance of our hypothesis h_i as well as to evaluate if it is representative of the knowledge that should be learned by the corresponding training batch Tr_i.

For instance, one example of such evaluation is one of the first metrics proposed for CL [56]; it consists of an overall performance Ω in a supervised classification setting. It is based on the relative performance of an incrementally trained algorithm with respect to an offline trained algorithm (which has access to all the data at once). In our notation, Ω is:

$$\Omega = \frac{1}{N} \sum_{i=1}^{N} \frac{R_{i,i}}{R^C_{i,i}}. \tag{2}$$

Where N is the number of tasks encountered, $R^C_{i,j}$ is the potentially best accuracy we can have on Te^C_i if the model was trained with all data at once, i.e. on Tr^C_i (the accumulation of training sets Tr^C_t from t=0 to t=i). Te^C_i is the accumulation of all test sets Te^C_t from $t = 0$ to $t = i$. $\Omega = 1$ indicates identical performance to an off-line cumulative setting; an Ω larger than one is possible when the offline model is worse than trained in a CL paradigm.

In [146], instead, the authors try to directly model forgetting with the proposed *forgetting ratio* metric ρ after learning i tasks, defined as:

$$\rho^{j \leq i} = \frac{1}{N} \sum_{i}^{N} \sum_{j}^{N} \left(\frac{R_{ij} - R^R_j}{R^C_{ij} - R^R_j} - 1 \right) \tag{3}$$

Where, R^R_j is the accuracy of a random stratified classifier using the class information of task j.

Always in the same sequential setting, in [92] other three important metrics are proposed: *Average Accuracy* (ACC), *Backward Transfer* (BWT), and *Forward Transfer* (FWT). In this case, after the model finishes learning about the training batch Tr_i, its performance is evaluated on all (even future) test batches Te_j.

The larger these metrics, the better the model. If two models have similar ACC, the preferred one is the one with larger BWT and FWT. Note that it is meaningless to discuss backward transfer for the first batch, or forward transfer for the last batch. The metrics are extended for more fine grained, generic evaluation [34] so that the original accuracy [92] (as well as BWT and FWT) can account for performance at *every timestep in time*. Accuracy is defined as:

$$A = \frac{\sum_{i=1}^{N} \sum_{j=1}^{i} R_{i,j}}{\frac{N(N+1)}{2}} \tag{4}$$

where $R \in \mathbb{R}^{N \times N}$ is the training-test accuracy matrix that contains in each entry $R_{i,j}$ the test classification accuracy of the model on task t_j after observing the last sample from task t_i, Accuracy

(A) considers the average accuracy for training set Tr_i and test set Te_j by considering the diagonal elements of R, as well as all elements below it (i.e., averages $R_{i,j}$s where $i >= j$ see Table 4). Backward Transfer (BWT) measures the influence that learning a task has on the performance on

Table 4: Accuracy matrix R: elements accounted to compute A (white & cyan), BWT (cyan), and FWT (gray). $R^* = R_{ii}$, Tr_i = training, Te_i= test tasks.

R	Te_1	Te_2	Te_3
Tr_1	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$
Tr_2	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$
Tr_3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$

previous tasks. It is defined as the accuracy computed on Te_i right after learning Tr_i as well as at the end of the last task on the same test set (see Table 4 in light cyan).

$$BWT = \frac{\sum_{i=2}^{N}\sum_{j=1}^{i-1}(R_{i,j} - R_{j,j})}{\frac{N(N-1)}{2}} \tag{5}$$

The original BWT [22, 92] is extended into two terms to distinguish among two semantically different concepts (so that, as the rest of metrics, is to be maximized and in [0,1]).

$$REM = 1 - |min(BWT, 0)| \tag{6}$$

i.e., *Remembering*, and (the originally positive) BWT, i.e., improvement over time, *Positive Backward Transfer*:

$$BWT^+ = max(BWT, 0) \tag{7}$$

Likewise, the FWT redefined to account for the dynamics of CL at each timestep is

$$FWT = \frac{\sum_{i=1}^{j-1}\sum_{j=1}^{N} R_{i,j}}{\frac{N(N-1)}{2}} \tag{8}$$

FWT accounts for the train-test accuracy entries $R_{i,j}$ above the principal diagonal of R, excluding it (see elements accounted in Table 4 in light gray). Forward transfer can occur when the model is able to perform *zero-shot* learning.

A Learning Curve Area (LCA) ($\in [0, 1]$) metric to quantify the learning speed by a CL strategy is proposed in [23]. It uses the b-shot performance (where b is the mini-batch number) after being trained for all the N tasks:

$$Z_b = \frac{1}{N}\sum_{i=1}^{N} a_{i,b,i} \tag{9}$$

where $a_{i,k,j} \in [0, 1]$ is the accuracy evaluated on the test set of task j after the model has been trained with the k-th mini-batch of task i. This amount is equivalent to previous accuracy matrix entry R_{ij} but at a lower granularity of a batch level. $a_{i,k,j}$ is used to define a forgetting measure $\in [-1, 1]$ that quantifies the drop in accuracy on previous tasks [22]. f_j^k is the forgetting on task j after the model is trained with all mini-batches up to task k:

$$f_j^k = \max_{l \in 1,..,k-1} a_{l,B_l,j} - a_{k,B_k,j} \tag{10}$$

where B_i is all mini-batches corresponding to training dataset of task k ($\mathcal{D}_k$).

LCA_β is the area of the convergence curve Z_b during training as a function of $b \in [0, \beta]$:

$$LCA_\beta = \frac{1}{\beta+1}\int_0^\beta Z_b db = \frac{1}{\beta+1}\sum_{b=0}^{\beta} Z_b \tag{11}$$

The interpretation of LCA is intuitive: an LCA_0 is the average 0-shot performance (FWT), and LCA_β is the area under the Z_b curve, which is high if the 0-shot performance is good and if the learner learns quickly. LCA aims at disambiguating the performance of models that may have the

same Z_b or A_T, but very different LCA_β because despite both eventually obtaining the same final accuracy, one may learn much faster than the other.

While forgetting and knowledge transfer could be quantified and evaluated in various ways, as argued in [41, 56, 71], these may not suffice for a robust evaluation of CL strategies. For example, in order to better understand the different properties of each strategy in different conditions, especially for embedded systems and robotics, it would be interesting to keep track and unambiguously determine the amount of computation and memory resources exploited. In this context, the metrics proposed in [92] are extended in [34] to unify in a common evaluation framework different infrastructural and operational metrics. Other practical metrics included are Model Size (MS), Samples Storage Size (SSS) efficiency and Computational Efficiency (CE). We briefly describe them next.

The memory size of model h_i is quantified in terms of parameters θ at each task i, $Mem(\theta_i)$; with the idea that it should not grow too rapidly with respect to the size of the model that learned the first task, $Mem(\theta_1)$:

$$MS = min(1, \frac{\sum_{i=1}^{N} \frac{Mem(\theta_1)}{Mem(\theta_i)}}{N}) \tag{12}$$

Some CL approaches save training samples (or generative replay generated samples) as a replay strategy to not forget. The Samples Storage Size (SSS) efficiency establishes a metric for the memory occupation in bits by the samples storage memory M, $Mem(M)$, to be bound by the occupation of the total number of examples encountered at the end of last task:

$$SSS = 1 - min(1, \frac{\sum_{i=1}^{N} \frac{Mem(M_i)}{Mem(D)}}{N}) \tag{13}$$

where D is the lifetime dataset associated to all distributions $\mathcal{D}$.

A metric that bounds the Computational efficiency (CE) by the number of operations for training set Tr_i is defined as:

$$CE = min(1, \frac{\sum_{i=1}^{N} \frac{Ops\uparrow\downarrow(Tr_i)\cdot\varepsilon}{1+Ops(Tr_i)}}{N}) \tag{14}$$

where $Ops(Tr_i)$ is the number of (mul-adds) operations needed to learn Tr_i, $Ops\uparrow\downarrow(Tr_i)$ are the operations required to do one forward and one backward (backprop) pass on Tr_i, and ε is a scaling factor (associated to the nr of epochs needed to learn Tr_i). Overall CL_{score} and $CL_{stability}$ metrics are also finally proposed [34] in order to aggregate different criteria to be maximized that allow to rank CL strategies. In order to assess a CL algorithm A^{CL}, each criterion to be optimized by the CL model, $c_i \in \mathcal{C}$ (where $c_i \in [0, 1]$) is assigned a weight $w_i \in [0, 1]$ where $\sum_i^{\mathcal{C}} w_i = 1$. Each c_i is the average of r runs, and the final CL_{score} to maximize is computed as:

$$CL_{score} = \sum_{i=1}^{\#\mathcal{C}} w_i c_i \tag{15}$$

where each final criterion c_i is to be maximized by a CL algorithm. $CL_{stability}$ is the pondered standard deviation of each CL metric [34]:

$$CL_{stability} = 1 - \sum_{i=1}^{\#\mathcal{C}} w_i \sigma(c_i) \tag{16}$$

with $\sigma(c_i)$ the standard deviation of criterion c_i.

In future evaluation scenarios, particularly in robotics, stability is another important property that should be evaluated since in many robotic tasks and safety-critical conditions, potential abrupt performance drifts would be a major concern when learning continuously. The metrics presented here can also be combined to assess higher-level capabilities. As an example, if we are to assess the *scalability* of a CL algorithm, one could use a weighted average of *SSS*, *MS*, and *CE*.

The metrics presented in a supervised classification context [34] can also be generalized with different performance measure P, instead of accuracy, and used in different settings such as reinforcement and unsupervised learning. For instance, they can be extended to RL; the underlying performance metric is, instead of accuracy, the accumulated reward on test episodes. In general in

RL, cumulative reward plots over time are common norm to evaluate policy learning algorithms. Extra performance metrics in RL tasks will very much depend on the task being assessed, the reward function, and other evaluation metrics that act as evaluation *proxies*, as it is common in semi/unsupervised learning settings.

The evaluation of generative models in any setting is challenging. Fréchet Inception Score (FID) [59] is a common metric that compares features from generated data and true data. Inception Score (IS) [139] has also been widely used as a proxy to evaluate the quality of generative models. It measures if the class of generated samples are varied by making use of a model trained on ImageNet. One shortcoming of these scores is that they may be maximized by over-fitting generative models. Another evaluation method is using generated data to train a classifier and evaluate its accuracy on a test set of true data [85]. The test accuracy, called Fitting Capacity (FC) gives a proxy on the quality of the generated data. Fitting Capacity and Fréchet Inception Score were used in a CL setting in [83, 81].
More methods for evaluating generative models are described and assessed more in depth in [11, 61]; however, they have never been used in a CL setting. In any case, the need for real data is mandatory in most evaluation schemes. In a CL setting, evaluating the generation of data from past tasks may need to violate the data availability assumption. The different metrics for generative models may then be useful tools for example for evaluating generative replay methods; however, they have to be manipulated carefully to be incorporated into the continual learning spirit.

6 Continual Learning for Robotics

In the previous section we listed and described the different existing types of strategies to tackle continual learning. In this section, we will present real use cases of CL with an emphasis on robotics applications. First, we present why continual learning is crucial for robotics, and then, the challenges that robotics face in CL tasks. Finally, we present concrete robotic applications with potential insights to draw from CL.

6.1 Opportunities for Continual Learning in Robotics

A robot is an agent that interacts with the real world. It means that it can not go back in time to improve what it has learn in the past. These particularities of robotic platforms make them a natural playground for CL algorithms. Furthermore, robots suffer from several constraints in terms of power or memory, and that is exactly what CL intends to optimize, in the way it addresses learning problems. On the other hand, robots have rich information about their experiences. They are in control of their interaction with the environment, which may help them understanding the concept of causality, and extracting knowledge from different kinds of sensors (images, sound, depth...). This rich information helps machines to produce strong representations which are crucial for a well performing CL algorithm [82].

We could almost conclude that CL is born for robotics, and it may be true; however, today most of CL approaches are not robotics related and rather focus on experiments on image processing or simulated environments. Next section will present the challenges that make CL difficult to apply in robotic environments.

6.2 Challenges of Continual Learning in Robotics

6.2.1 Robotics Hardware

The first challenge to deal with when doing any experiment with robots is the hardware. Robots are known to be unstable and fragile. Robot failures are one of the main restrictions for researchers to propose new approaches on robotics tasks. They add unavoidable delay in any experiment and are expensive to fix. Moreover, if the failure is not hardware but software, since it is not possible to reset the state of the robot automatically, manual help is often needed, e.g., to put back the robot in his starting position or recover it from an irrecoverable state. Furthermore, most of the time building or buying a robot is itself quite costly. Once the robot is correctly working, one new problem arises, which is its autonomy in terms of energy. This aspect is also a main difficulty to deal with when experiments need to be set. It is difficult to program long experiments without manually recharging the robot and making sure that it will not stop by a lack of power supply

or failure. Lastly, robots are embedded platforms and, consequently, have limited memory and computation resources, which should be carefully managed to avoid overflow.

The difficulties of using robots in experiments explain why there are so few approaches of continual learning with robots in the literature. In the next section, we will see how robotic environments challenge continual learning algorithms.

6.2.2 Data Sampling

When a robot needs to learn a task in a known or unknown environment, it must collect its own training data in the real world [170]. Data serves as the basis for environment exploration and comprehension. This problematic is exactly the same as the one met by RL algorithms [157]. In infants, a crucial component of lifelong learning is the ability to autonomously generate goals and explore the environment driven by intrinsic motivation [113, 18]. Self-supervised approaches [121, 86, 170, 149] also help to automatically explore environments. Curiosity [15] and self-supervision [35] allow to search for new experiences (or data) and build a base of knowledge useful to achieve actual or future tasks via transfer learning [115]. As an example, manipulation tasks [72] such as grasping [121], reaching [123, 26], pushing buttons [84], throwing [155, 72] or stacking [26] objects (cubes, balls...) are common complex tasks built on comprehensive sets of experiments.

Data gathered in this way can then be used on the fly in an online learning process or stored for later processing.

However, in order to improve learning algorithms the need for annotations or external help is crucial. In the next subsection we will describe the particular needs for annotations in robotics.

6.2.3 Data Labeling

As seen in previous section, gathering a varied set of raw data is already a difficult task. However, using it and understanding it is even more tedious. In this section, we detail different needs for labelling that autonomous agents such as robots need. First of all, to understand its environment, a robot will need to apprehend the objects that compose it. To do so, the robot will need at some point that an external expert assesses that the object representation learned is good. This is the first kind of label the robot will need, i.e., object labelling [28, 30]. Secondly, if we want the robot to perform a certain task, it will need to get information about the goals we gave it and also what it should not do. This is generally done by a reward function that defines credit assignment [106], or it can also be defined internally by more abstract rules such as self-supervision [54, 152], intrinsic motivation or curiosity [113, 142] as in [46, 27, 30, 78]. Thirdly, the robot should know when the task changes, and what task it should try to solve. This process consists of labelling the task; and the label is called the task identifier [92].

All these types of labels are not mandatory, but they drastically help and impact the learning process. The downside of labelling is that it is expensive and time consuming, which slows down the learning algorithms. To tackle those two problems, CL needs to find efficient solutions that can make the best out of the available labels for learning.

The specific fields that aim at answering these questions are few-shot learning [76, 42] and active learning [148]. The former tries to grasp a concept from very few data points. Active learning aims at identifying and selecting the most needed labels in order to maximize learning. By combining optimization procedures in learning from few instances and minimizing the needs for labels, the field of robotics could be more suitable for leveraging continual learning settings in the real world. Furthermore, efficiency in learning reduces the risks of forgetting and degrading memories.

6.2.4 Learning Algorithms Stability

In continual learning, algorithms face several learning experiences in a row. From each learning experience, some memory should be saved to later prevent for not forgetting. The stability of learning algorithms is then crucial: if only one learning experience fails, the whole process may be corrupted. Moreover, if we respect the continual learning causality, we can not go back one or several tasks earlier in time in order to fix an actual problem. The corruption of one learning experience can lead to the corruption of memories and then to the model degradation when learning later tasks. The needs for robust mechanisms to validate or reject results of a learning algorithm is key to keep sane memories and weights; however, the instability of deep learning models must also be addressed to improve this drawback. As an example, generative models are powerful tools

for continual learning but their instability may make them unsuitable for this kind of setting [81]. Reinforcement learning algorithms are also known to be unstable and unpredictable, which is disastrous for continual learning.

6.3 Applications

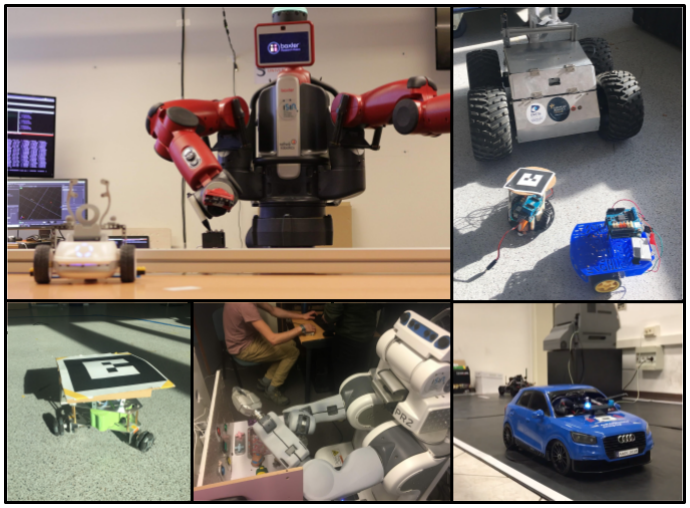

Figure 3: Sample tasks tested for unsupervised open-ended learning [123, 36] and continual learning settings [68] in a couple of robotics labs, among others, from the DREAM project (www.robotsthatdream.eu).

Real-word applications of continual learning are virtually unlimited. In fact, any learning algorithm that needs to deal with the real world will face a non i.i.d. data stream. This as well happens for autonomous robots that learn new manipulation tasks, for exploration policies, as well as for autonomous vehicles that need to learn and adapt to new circumstances [10, 25, 63, 126]. Non-static settings are also faced by algorithms that learn how to predict trends based on data streams from internet user activities, e.g., among others, for advertisement or finance. This problem is likewise confronted when an already trained algorithms needs to acquire new knowledge without forgetting, e.g., recognize new classes for classification, anomaly detection, etc. However, in this section we focus on specific continual learning use cases on robotics.

6.3.1 Perception

While the world of perception is a multi-faceted topic at the very center of every application on autonomous sytems, the vast majority of CL algorithms in the literature are evaluated on object recognition tasks. Most models, indeed, are evaluated on datasets including static or moving objects. This is motivated by the fact that before any further action or policy, an autonomous agent (or robot) needs to identify the different component of its environment. In the case of classification, the robot may be pre-trained from an initial dataset. However, in any environment the robot would probably need to learn new objects from the new domain, and new variants (different poses, lighting, aspect) of already learned objects should be leveraged to improve its recognition [94] capabilities. CL is crucial to tackle such dynamic scenarios. Initial progresses in this area have been proposed in [159, 116, 90, 16, 91]. Concrete Continual learning approaches to object segmentation can be found in [104, 105], and in object detection in [151].

Visual saliency for semantic segmentation and unsupervised object detection are other equally important applications in the context of perception which have been recently explored under continual learning and robotics settings [29]. RL-IAC (RL Intelligent Adaptive Curiosity), in particular, explores to learn visual saliency incrementally [30] with an articulated autonomous exploration technique based on curiosity to efficiently and continually learn a saliency model in a complex robotics environment tested in the real-world.

A classic problem in robotics within inherently continual learning settings are simultaneous localization and mapping (SLAM) [21] and navigation [159]. In [159], using a HERO-2000 mobile

robot with a radar sensor a continual learning algorithm based on explanation-based neural network learning (EBNN) is proposed to perform room mapping and navigation. Action models in EBNN *explain* (in terms of previous experiences) and analyze observations to transfer task-independent (navigation) knowledge via predicting collisions and their prediction certainty.

6.3.2 Reinforcement Learning

In reinforcement learning the data distribution is dependent on the actions taken by the controlled agent. Therefore, since the actions taken are not random, data is not i.i.d. and the data distribution is not stationary. In the context of reinforcement learning similar techniques to those proposed in CL are often adopted in order to learn over a data distribution which is approximately stationary. An example of such techniques is the use of a external memory for rehearsal purposes, also know as *experience* or *memory replay* buffer [88, 141, 55].

The first challenge for RL is the extraction of representations to understand and compact what is relevant from the input data [82]. Continual learning of state representations for RL is intrinsically close to unsupervised learning or representation learning for classification; the methods used in both cases may then be very similar or worth leveraging across.

The second RL challenge is learning a policy to solve a specific task. The CL challenge of policy learning is different because it often should take into account both state and context. Context is usually handled with recurrent neural networks, and this kind of model is not yet been studied extensively in CL; one example is in [153]. Different robot manipulation tasks such as grasping and reaching objects that are used as benchmarks can be seen in Fig. 3 and, for instance, in state representation learning for robotics goal-based tasks [123, 68]. These two challenges face continual learning problems, to learn representations and to learn policies from non stationary data distributions. However, it is worth distinguishing among both problems because learning and transfer between tasks are different challenges. Two tasks may need similar representations with different policies, while similar policies may require dissimilar representations.

In the context of robotics, fewer RL approaches have been proposed than in video-games or simulation settings. In particular, this is due to the low data efficiency of RL algorithms [123]. We can still note several approaches that successfully tackle this problem, either in an end-to-end manner [67, 121], or by splitting the two challenges to address them separately, i.e., by first learning a state representation [82] and later performing policy learning [45, 164, 100, 2, 38, 64]. Nevertheless, a solution to this problem is to learn the policy in simulation and transfer it to deploy it in a real world robot [10, 138, 49, 68].

6.3.3 Model-based Learning

Smoothly moving and interacting with always different, unpredictable environments, while constructing a coherent model of the external world, is one of the holy grail of robotics. For many years, researchers in this area have tried to propose robust and general enough sensory-motor solutions to complex problems such as navigation or object grasping. However, as it appears to be also true for humans, there will always be an environment or situation in which our biased model of the world fails and adaptation is needed.

Online (inverse dynamics) learning has also been applied in robotics, but generally not using deep learning. In [133, 17], the inverse and semiparametric dynamics of an iCub humanoid robot is learned in an incremental manner. This means both parametric modelling (based on rigid body dynamics equations) and nonparametric modelling (using incremental kernel methods) are used. In [134] it is shown that derivative-free models outperform numerical differentiation schemes in online settings when applied to non parametric (e.g. Gaussian processes with kernel function) model structures.

In the pioneering work by [159], a model of both the external world and the robot itself is incrementally learned through reinforcement learning in complex navigation tasks on a real robot. However, incrementally and autonomously building a causal model of the external world, still remains a poorly explored topic in the context of robotics. Nevertheless, as it has been shown in recent RL literature, a model-based approach may be of fundamental importance in the real-world where millions of trials and errors are not always conceivable.

7 Discussion and Conclusion

Several notions appear to be crucial to clearly describe learning algorithms in CL settings, fairly compare them and transfer them from simulation to real autonomous systems and robotics. First of all, identifying the exact problem we want to solve, and what are the existing constraints is mandatory. The framework we introduce in Section 3 should assist to achieve the characterization of these settings. This formal step helps finding the proper approach to use and identifying similarities with other settings. Secondly, in the same spirit of defining what we want to learn, it is important to define the level of supervision we are able to give to our learning algorithm. For example, if we can give it the task label, or some kind of information about the structure of the input data stream (number of classes, type of data distribution, number of instances of each task, etc.). This point is also discussed in our proposed framework (Section 3). Finally, it is important to exactly clarify what is the expected performance of the algorithm. The set of metrics and benchmarks gathered in Section 5 should help defining and articulating the dimension of evaluation for important properties worth considering in the development of embodied agents that learn continually.

For more concrete indications on what we consider worthwhile checking while creating a CL approach, we suggest a set of recommendations. After defining in Section 3 a set of assumptions, constraints, relaxations and desiderata of CL algorithms, the following concrete measure and action-based guidelines aim at being taken into account as general advice to palliate limiting factors of CL models in the literature.

Recommendation 1 *On-line capabilities: CL algorithms should not assume the number of total tasks to be solved is given beforehand.*

Recommendation 2 *Learning complexity: We recommend keeping the learning model complexity below an upper bound of a linear growth in terms of the number of parameter growth when performing architectural dynamic changes.*

Recommendation 3 *Scalability evaluation: In order to provide a proper evaluation of the scalability and continual learning performance, we recommend, as the authors from [41], to evaluate algorithms on more than two tasks.*

Recommendation 4 *Memory limitation: In order for realistic CL systems to be practical, they should not assume unlimited memory resources.*

Recommendation 5 *Reporting metrics: We recommend reporting as many metrics as possible and at least final performance, forward and backward (learning) transfer, the model's remembering capacity, model memory size, samples storage size, computational efficiency, CL score and stability metrics as described in Section 5.2.*

Recommendation 6 *Offline baselines: we recommend the usage of publicly available baselines for metrics computation and fair assessment for reproducibility purposes.*

Recommendation 7 *Ablation studies: we recommend reporting ablation studies to motivate as best as possible the different components and choices made in the CL algorithm such as initialization settings (using pre-trained network or not), optimization methods, hyper-parameters and surrogate losses used, etc.*

Recommendation 8 *Distributional shifts: We recommend to formally describe the mechanism to handle distributional shifts, not only when tasks change, but also among batches where data points conform to different distributions.*

Recommendation 9 *Benchmarks: We recommend the use of complex datasets with realistic and higher resolution scales than MNIST and CIFAR100; the use of the former is seen as a limiting factor and not a realistic robustness assessment method for CL (see Section 5.1).*

Recommendation 10 *Report precisely and clearly how an approach learns and the assumptions it makes, as described in the framework (Section 3).*

To summarize, in this paper, we proposed a generalized framework to hold a variety of CL strategies and make easier the connection between machine learning and robotics in continual learning settings. We reviewed the state of the art in continual learning and illustrated how to use the proposed framework to present various approaches. We also discussed benchmarks and evaluation techniques currently being used in continual learning algorithms. We hope it helps the AI community to better categorize and compare approaches, as well as to smoothly adapt to today's industry problems. Machine learning and robotics are fields undergoing an aggressive development period. We believe that pushing them forward to find formalization solutions to facilitate transfer between both fields is critical in order to understand each other, and make them profit from each other's successes.

Acknowledgments

This work is supported by the DREAM project[6] through the European Union Horizon 2020 FET research and innovation program under grant agreement No 640891.

References

[1] A. Achille, T. Eccles, L. Matthey, C. P. Burgess, N. Watters, A. Lerchner, and I. Higgins. Life-long disentangled representation learning with cross-domain latent homologies. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 9895–9905, USA, 2018. Curran Associates Inc.

[2] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 5074–5082. Curran Associates, Inc., 2016.

[3] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375, 2017.

[4] P. Azagra, F. Golemo, Y. Mollard, M. Lopes, J. C. Civera, and A. C. Murillo. A Multimodal Dataset for Object Model Learning from Natural Human-Robot Interaction. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017)*, Vancouver, Canada, Sept. 2017.

[5] F. Bellas, J. A. Becerra, and R. J. Duro. Using promoters and functional introns in genetic algorithms for neuroevolutionary learning in non-stationary problems. *Neurocomputing*, 72(10-12):2134–2145, 2009.

[6] F. Bellas, R. J. Duro, A. Faina, and D. Souto. Multilevel darwinist brain (mdb): Artificial evolution in a cognitive architecture for real robots. volume 2, pages 340–354, Dec 2010.

[7] F. Bellas, A. Faiña, G. Varela, and R. J. Duro. A cognitive developmental robotics architecture for lifelong learning by evolution in real robots. pages 1–8, 2010.

[8] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[10] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[11] A. Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2018.

[6] http://www.robotsthatdream.eu

[12] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to Data Mining*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[13] N. Bredeche, E. Haasdijk, and A. Prieto. Embodied evolution in collective robotics: A review. *Frontiers in Robotics and AI*, 5:12, 2018.

[14] P. Bühlmann and S. van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[15] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

[16] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta. Incremental robot learning of new objects with fixed update time. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3207–3214, May 2017.

[17] R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori. Incremental semiparametric inverse dynamics learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 544–550. IEEE, 2016.

[18] A. Cangelosi and M. Schlesinger. From babies to robots: The contribution of developmental robotics to developmental psychology. *Child Development Perspectives*, 2018.

[19] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3. Atlanta, 2010.

[20] H. Caselles-Dupré, M. Garcia-Ortiz, and D. Filliat. S-TRIGGER: Continual State Representation Learning via Self-Triggered Generative Replay. *arXiv e-prints*, page arXiv:1902.09434, Feb 2019.

[21] T. Cavallari, S. Golodetz, N. A. Lord, J. Valentin, L. Di Stefano, and P. H. Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4457–4466, 2017.

[22] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018.

[23] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with A-GEM. In *ICLR*, 2019.

[24] Z. Chen and B. Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[25] F. Codevilla, M. Muller, A. Dosovitskiy, A. L'opez, and V. Koltun. End-to-end driving via conditional imitation learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2017.

[26] C. Colas, P. Fournier, M. Chetouani, O. Sigaud, and P.-Y. Oudeyer. CURIOUS: Intrinsically motivated modular multi-goal reinforcement learning. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1331–1340, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[27] C. Colas, O. Sigaud, and P.-Y. Oudeyer. GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1039–1048, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[28] A. Collet, B. Xiong, C. Gurau, M. Hebert, and S. S. Srinivasa. Herbdisc: Towards lifelong robotic object discovery. *The International Journal of Robotics Research*, 34(1):3–25, 2015.

[29] C. Craye, D. Filliat, and J. Goudou. Exploration strategies for incremental learning of object-based visual saliency. In *2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 13–18, Aug 2015.

[30] C. Craye, T. Lesort, D. Filliat, and J.-F. Goudou. Exploring to learn visual saliency: The rl-iac approach. *Robotics and Autonomous Systems*, 112:244 – 259, 2019.

[31] G. Csurka. *A Comprehensive Survey on Domain Adaptation for Visual Applications*, pages 1–35. Springer International Publishing, Cham, 2017.

[32] J.-F. Delvenne. Science of memory: Concepts. henry l. roediger iii, yadin dudai, and susan m. fitzpatrick (eds.). oxford university press, new york, 2007. no. of pages 464. isbn 978-0-19-531044-3.(paperback). *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 23(6):895–896, 2009.

[33] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa. Learning without memorizing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[34] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni. Don't forget, there is more than forgetting: new metrics for Continual Learning. In *Workshop on Continual Learning, NeurIPS 2018 (Neural Information Processing Systems*, Montreal, Canada, Dec. 2018.

[35] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

[36] S. Doncieux, D. Filliat, N. Díaz-Rodríguez, T. Hospedales, R. Duro, A. Coninx, D. M. Roijers, B. Girard, N. Perrin, and O. Sigaud. Open-ended learning: a conceptual framework based on representational redescription. *Frontiers in Neurorobotics*, 2018.

[37] T. J. Draelos, N. E. Miner, C. C. Lamb, J. A. Cox, C. M. Vineyard, K. D. Carlson, W. M. Severa, C. D. James, and J. B. Aimone. Neurogenesis deep learning: Extending deep networks to accommodate new classes. *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 526–533, 2017.

[38] W. Duan. Learning state representations for robotic control. *M. Thesis*, 2017.

[39] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015.

[40] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, 2018.

[41] S. Farquhar and Y. Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733, Lifelong Learning: A Reinforcement Learning Approach (LLARLA) Workshop at FAIM 2018*, 2018.

[42] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[43] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734, 2017.

[44] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1126–1135. JMLR.org, 2017.

[45] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519, 2015.

[46] S. Forestier, Y. Mollard, and P.-Y. Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[47] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[48] T. Furlanello, J. Zhao, A. M. Saxe, L. Itti, and B. S. Tjan. Active Long Term Memory Networks. *ArXiv e-prints*, June 2016.

[49] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955, Sep. 2017.

[50] A. Gepperth and B. Hammer. Incremental learning algorithms and applications. In *European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2016.

[51] A. Gepperth and C. Karaoguz. A Bio-Inspired Incremental Learning Architecture for Applied Perceptual Problems. *Cognitive Computation*, 8:924 – 934, 2016.

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[53] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *ArXiv e-prints*, Dec. 2013.

[54] A. Gopnik, A. Meltzoff, and P. Kuhl. The scientist in the crib: Minds, brains and how children learn. *Journal of Nervous and Mental Disease - J NERV MENT DIS*, 189, 03 2001.

[55] T. L. Hayes, N. D. Cahill, and C. Kanan. Memory efficient experience replay for streaming learning. *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776, 2018.

[56] T. L. Hayes, R. Kemker, N. D. Cahill, and C. Kanan. New metrics and experimental paradigms for continual learning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2112–21123, June 2018.

[57] X. He and H. Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on Learning Representations*, 2018.

[58] Y. He, Z. Shen, and P. Cui. NICO: A Dataset Towards Non-I.I.D. Image Classification. *arXiv e-prints*, page arXiv:1906.02899, Jun 2019.

[59] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 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 6626–6637. Curran Associates, Inc., 2017.

[60] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop (2015)*, pages 1–9, 2015.

[61] D. J. Im, A. H. Ma, G. W. Taylor, and K. Branson. Quantitatively evaluating GANs with divergences proposed for training. In *International Conference on Learning Representations*, 2018.

[62] H. Jaeger. Using conceptors to manage neural long-term memories for temporal patterns. *Journal of Machine Learning Research*, 18(13):1–43, 2017.

[63] M. Jaritz, R. de Charette, M. Toromanoff, E. Perot, and F. Nashashibi. End-to-end race driving with deep reinforcement learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2070–2075, 2018.

[64] R. Jonschkowski and O. Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[65] H. Jung, J. Ju, M. Jung, and J. Kim. Less-forgetting learning in deep neural networks. *CoRR*, abs/1607.00122, 2016.

[66] C. Käding, E. Rodner, A. Freytag, and J. Denzler. Fine-tuning deep neural networks in continuous learning scenarios. In *Asian Conference on Computer Vision*, pages 588–605. Springer, 2016.

[67] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR, 29–31 Oct 2018.

[68] R. T. Kalifou, H. Caselles-Dupré, T. Lesort, T. Sun, N. Diaz-Rodriguez, and D. Filliat. Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer. In *ICML Workshop on Multi-Task and Lifelong Learning*, 2019.

[69] N. Kamra, U. Gupta, and Y. Liu. Deep Generative Dual Memory Network for Continual Learning. *ArXiv e-prints*, Oct. 2017.

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

[71] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI*, 2017.

[72] S. Kim, A. Coninx, and S. Doncieux. From exploration to control: learning object manipulation skills through novelty search and local adaptation. *CoRR*, abs/1901.00811, 2019.

[73] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proc. of the national academy of sciences*, 2017.

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

[75] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[76] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

[77] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[78] A. Laversanne-Finot, A. Pere, and P.-Y. Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 487–504. PMLR, 29–31 Oct 2018.

[79] Y. LeCun and C. Cortes. MNIST handwritten digit database. *public*, 2010.

[80] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*, pages 4652–4662, 2017.

[81] T. Lesort, H. Caselles-Dupré, M. Garcia-Ortiz, J.-F. Goudou, and D. Filliat. Generative Models from the perspective of Continual Learning. In *IJCNN - International Joint Conference on Neural Networks*, Budapest, Hungary, July 2019.

[82] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat. State representation learning for control: An overview. *Neural Networks*, 2018.

[83] T. Lesort, A. Gepperth, A. Stoian, and D. Filliat. Marginal replay vs conditional replay for continual learning. In *Artificial Neural Networks and Machine Learning - ICANN 2019: Deep Learning - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part II*, pages 466–480, 2019.

[84] T. Lesort, M. Seurin, X. Li, N. Díaz-Rodríguez, and D. Filliat. Deep unsupervised state representation learning with robotic priors: a robustness analysis. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2019.

[85] T. Lesort, A. Stoian, J. Goudou, and D. Filliat. Training discriminative models to evaluate generative ones. In *Artificial Neural Networks and Machine Learning - ICANN 2019: Image Processing - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part III*, pages 604–619, 2019.

[86] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[87] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[88] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[89] V. Lomonaco, K. Desai, E. Culurciello, and D. Maltoni. Continual reinforcement learning in 3d non-stationary environments. *arXiv preprint arXiv:1905.10112*, 2019.

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

[91] V. Lomonaco and D. Maltoni. CORe50: a New Dataset and Benchmark for Continuous Object Recognition. In S. Levine, V. Vanhoucke, and K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017.

[92] D. Lopez-Paz and M.-A. Ranzato. Gradient episodic memory for continual learning. 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 6467–6476. Curran Associates, Inc., 2017.

[93] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics: a survey. *Connection Science*, 15(4):151–190, 2003.

[94] N. Lyubova, S. Ivaldi, and D. Filliat. From passive to interactive object learning and recognition through self-identification on a humanoid robot. *Autonomous Robots*, page 23, 2015.

[95] A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[96] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

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

[98] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

[99] D. J. Mankowitz, A. Žídek, A. Barreto, D. Horgan, M. Hessel, J. Quan, J. Oh, H. van Hasselt, D. Silver, and T. Schaul. Unicorn: Continual learning with a universal, off-policy agent. *arXiv preprint arXiv:1802.08294*, 2018.

[100] J. Mattner, S. Lange, and M. A. Riedmiller. Learn to swing up and balance a real pole based on raw visual input data. In *Neural Information Processing - 19th International Conference, ICONIP 2012, Doha, Qatar, November 12-15, 2012, Proceedings, Part V*, pages 126–133, 2012.

[101] J. L. McClelland, B. L. McNaughton, and R. C. O'reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

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

[103] M. Mermillod, A. Bugaiska, and P. Bonin. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4(August):504, 2013.

[104] U. Michieli and P. Zanuttigh. Incremental learning techniques for semantic segmentation. In *International Conference on Computer Vision (ICCV), Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2019.

[105] U. Michieli and P. Zanuttigh. Knowledge distillation for incremental learning in semantic segmentation, 2019.

[106] M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, Jan 1961.

[107] T. Mitchell, W. Cohen, E. Hruscha, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohammad, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *AAAI*, 2015. : Never-Ending Learning in AAAI-2015.

[108] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.

[109] V. Moens and A. Zénon. Learning and forgetting using reinforced bayesian change detection. *PLOS Computational Biology*, 15(4):1–41, 04 2019.

[110] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[111] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[112] P. Oudeyer. Computational theories of curiosity-driven learning. *CoRR*, abs/1802.10546, 2018.

[113] P.-Y. Oudeyer, F. Kaplan, and V. Hafner. Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on*, 11(2):265–286, April 2007.

[114] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54 – 71, 2019.

[115] G. I. Parisi, J. Tani, C. Weber, and S. Wermter. Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. *Frontiers in Neurorobotics*, 12:78, 2018.

[116] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, and L. Natale. Teaching icub to recognize objects using deep convolutional neural networks. In *Proceedings of the 4th International Conference on Machine Learning for Interactive Systems - Volume 43*, MLIS'15, pages 21–25. JMLR.org, 2015.

[117] G. Pasquale, C. Ciliberto, L. Rosasco, and L. Natale. Object identification from few examples by improving the invariance of a deep convolutional neural network. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4904–4911, Oct 2016.

[118] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE Signal Processing Magazine*, 32(3):53–69, May 2015.

[119] A. Pentina and C. H. Lampert. Lifelong learning with non-iid tasks. In *Advances in Neural Information Processing Systems*, pages 1540–1548, 2015.

[120] B. Pfulb and A. Gepperth. A comprehensive, application-oriented study of catastrophic forgetting in DNNs. In *International Conference on Learning Representations*, 2019.

[121] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3406–3413, 2015.

[122] L. Y. Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pages 204–211, 1993.

[123] A. Raffin, A. Hill, K. R. Traoré, T. Lesort, N. Díaz-Rodríguez, and D. Filliat. Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. In *ICLR*, 2019.

[124] J. Ramapuram, M. Gregorova, and A. Kalousis. Lifelong generative modeling. *arXiv preprint arXiv:1705.09847*, 2017.

[125] 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.

[126] N. Rhinehart, R. McAllister, and S. Levine. Deep imitative models for flexible inference, planning, and control. *CoRR*, abs/1810.06544, 2018.

[127] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, , and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[128] M. B. Ring. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712, 1994.

[129] M. B. Ring. Toward a formal framework for continual learning. In *NIPS workshop on Inductive Transfer, Whistler, Canada.*, 2005.

[130] A. Rios and L. Itti. Closed-loop memory gan for continual learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, pages 3332–3338. AAAI Press, 2019.

[131] H. Ritter, A. Botev, and D. Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3738–3748. Curran Associates, Inc., 2018.

[132] A. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[133] D. Romeres, M. Zorzi, R. Camoriano, and A. Chiuso. Online semi-parametric learning for inverse dynamics modeling. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 2945–2950, Dec 2016.

[134] D. Romeres, M. Zorzi, R. Camoriano, S. Traversaro, and A. Chiuso. Derivative-free online learning of inverse dynamics models. *IEEE Transactions on Control Systems Technology*, pages 1–15, 2019.

[135] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[136] A. A. Rusu, S. Gomez Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy Distillation. *arXiv e-prints*, page arXiv:1511.06295, Nov 2015.

[137] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive Neural Networks. *ArXiv e-prints*, June 2016.

[138] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. *CoRR*, abs/1610.04286, 2016.

[139] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[140] P. Sarkar and W. Q. Meeker. A bayesian on-line change detection algorithm with process monitoring applications. *Quality Engineering*, 10(3):539–549, 1998.

[141] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[142] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[143] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[144] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4528–4537, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[145] A. Seff, A. Beatson, D. Suo, and H. Liu. Continual learning in generative adversarial nets. *CoRR*, abs/1705.08395, 2017.

[146] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *ICML*, 80:4548–4557, 10–15 Jul 2018.

[147] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[148] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[149] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[150] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[151] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 3420–3429, 2017.

[152] L. Smith and M. Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11:13–29, 12 2005.

[153] S. Sodhani, S. Chandar, and Y. Bengio. On training recurrent neural networks for lifelong learning. *CoRR*, abs/1811.07017, 2018.

[154] P. Sprechmann, S. Jayakumar, J. Rae, A. Pritzel, A. P. Badia, B. Uria, O. Vinyals, D. Hassabis, R. Pascanu, and C. Blundell. Memory-based parameter adaptation. In *International Conference on Learning Representations*, 2018.

[155] F. Stulp, L. Herlant, A. Hoarau, and G. Raiola. Simultaneous on-line discovery and improvement of robotic skill options. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 1408–1413. IEEE, 2014.

[156] Y. Sun, F. Gomez, and J. Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence*, pages 41–51. Springer, 2011.

[157] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

[158] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[159] S. Thrun and T. M. Mitchell. Lifelong robot learning. In *The biology and technology of intelligent autonomous agents*, pages 165–196. Springer, 1995.

[160] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, G. Cai, N. D. Rodríguez, and D. Filliat. Discorl: Continual reinforcement learning via policy distillation. *CoRR*, abs/1907.05855, 2019.

[161] A. Triki Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars. Encoder based lifelong learning. *IEEE International Conference of Computer Vision*, 2017.

[162] A. M. Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.

[163] L. G. Valiant. A theory of the learnable. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445. ACM, 1984.

[164] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3928–3934, Oct 2016.

[165] R. Velez and J. Clune. Diffusion-based neuromodulation can eliminate catastrophic forgetting in simple neural networks. *PLOS ONE*, 12(11):1–24, 11 2017.

[166] W. Wang, V. W. Zheng, H. Yu, and C. Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):13:1–13:37, Jan. 2019.

[167] Y.-X. Wang, D. Ramanan, and M. Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2471–2480, 2017.

[168] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[169] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, 2001.

[170] J. M. Wong. Towards lifelong self-supervision: A deep learning direction for robotics. *arXiv preprint arXiv:1611.00201*, 2016.

[171] C. Wu, L. Herranz, X. Liu, y. wang, J. van de Weijer, and B. Raducanu. Memory replay gans: Learning to generate new categories without forgetting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5962–5972. Curran Associates, Inc., 2018.

[172] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu. Incremental classifier learning with generative adversarial networks. *CoRR*, abs/1802.00853, 2018.

[173] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[174] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[175] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

[176] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[177] C. Zhao, T. M. Hospedales, F. Stulp, and O. Sigaud. Tensor based knowledge transfer across skill categories for robot control. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3462–3468. AAAI Press, 2017.

[178] G. Zhou, K. Sohn, and H. Lee. Online incremental feature learning with denoising autoencoders. In N. D. Lawrence and M. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 1453–1461, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.