Neuro-Symbolic Continual Learning: Knowledge, Reasoning Shortcuts and Concept Rehearsal

Emanuele Marconato [*1 2] **Gianpaolo Bontempo** [*1 3] **Elisa Ficarra** [3] **Simone Calderara** [3] **Andrea Passerini** [2] **Stefano Teso** [4 2]

Abstract

We introduce Neuro-Symbolic Continual Learning, where a model has to solve *a sequence of neuro-symbolic tasks*, that is, it has to map sub-symbolic inputs to high-level concepts and compute predictions by *reasoning* consistently with prior knowledge. Our key observation is that neuro-symbolic tasks, although different, often share concepts whose *semantics* remains stable over time. Traditional approaches fall short: existing continual strategies ignore knowledge altogether, while stock neuro-symbolic architectures suffer from catastrophic forgetting. We show that leveraging prior knowledge by combining neuro-symbolic architectures with continual strategies *does* help avoid catastrophic forgetting, but also that doing so can yield models affected by *reasoning shortcuts*. These undermine the semantics of the acquired concepts, even when detailed prior knowledge is provided upfront and inference is exact, and in turn continual performance. To overcome these issues, we introduce COOL, a **CO**ncept-level c**O**ntinual **L**earning strategy tailored for neuro-symbolic continual problems that acquires high-quality concepts and remembers them over time. Our experiments on three novel benchmarks highlights how COOL attains sustained high performance on neuro-symbolic continual learning tasks in which other strategies fail.[1]

1. Introduction

We initiate the study of Neuro-Symbolic Continual Learning (NeSy-CL), in which the goal is to *solve a sequence of neuro-symbolic tasks*. As is common in neuro-symbolic (NeSy) prediction (Manhaeve et al., 2018; Xu et al., 2018; Giunchiglia & Lukasiewicz, 2020; Hoernle et al., 2022; Ahmed et al., 2022a), the machine is provided *prior knowledge* relating one or more target labels to symbolic, high-level concepts *extracted* from sub-symbolic data, and has to compute a prediction by *reasoning* over said concepts. The central challenge of Nesy-CL is that the data distribution and the knowledge may vary across tasks. E.g., in medical diagnosis knowledge may encode known relationships between possible symptoms and conditions, while different tasks are characterized by different distributions of X-ray scans, symptoms and conditions. The goal, as in continual learning (CL) (Parisi et al., 2019), is to obtain a model that *attains high accuracy on new tasks without forgetting what it has already learned* under a limited storage budget.

Existing approaches are insufficient for NeSy-CL: neuro-symbolic models are designed for offline learning and as such suffer from *catastrophic forgetting* (Parisi et al., 2019), while continual learning strategies are designed for neural networks that neglect prior knowledge, preventing applications to tasks where compliance with regulations is key, e.g., safety critical tasks (Ahmed et al., 2022a; Hoernle et al., 2022). It is tempting to tackle NeSy-CL by pairing a SotA neuro-symbolic architecture, such as DeepProbLog (Manhaeve et al., 2018), with a proven rehearsal or distillation strategy, for instance dark experience replay (Buzzega et al., 2020). This yields immediate benefits, in the sense that prior knowledge makes the model more robust to catastrophic forgetting, as we will show. However, we show also that it is flawed, because it cannot prevent the model from acquiring *reasoning shortcuts* (defined in Section 3), through which it attains high task accuracy by acquiring unintended concepts with *task-specific semantics*, as illustrated in Figure 1. In turn, reasoning shortcuts entail poor cross-task transfer.

Our key observation is that, even though neuro-symbolic tasks may differ in terms of knowledge and distribution, *the semantics of the concepts they rely on must remain stable*

[*]Equal contribution [1]University of Pisa, Italy [2]DISI, University of Trento, Italy [3]University of Modena and Reggio Emilia, Italy [4]CIMeC, University of Trento, Italy. Correspondence to: Emanuele Marconato <emanuele.marconato@unitn.it>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]The data and code are available at https://github.com/ema-marconato/NeSy-CL

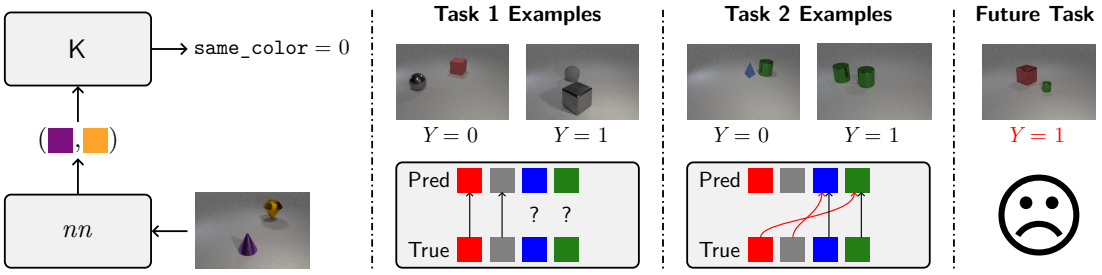

Figure 1. **Left**: DeepProbLog extracts concepts $\mathbf{c}$ from a sub-symbolic input $\mathbf{x}$ and reasons over prior knowledge K provided upfront to obtain a prediction $\mathbf{y}$. **Right**: Simplified illustration of our CLE4EVR benchmark, restricted to color concepts only. The full setting is reported in Section 5. The goal is to predict whether two objects have the $Y =$ `same_color`. The first task includes examples of **gray** and **red** objects only, and the model classifies them by learning the intended mapping between input colors and concepts. The second one includes **green** and **blue** objects only, and the model can learn a *reasoning shortcut* mapping the four colors to two concepts, achieving high accuracy on both tasks but compromising performance on *future* tasks. This is exactly the issue addressed by our approach, COOL.

over time. For instance, in automated protein annotation inferred signal peptides entail a catalytic function (Rost et al., 2003) regardless of the specific protein under examination, and in vehicle routing presence of pedestrians on the road is sufficient to rule out certain routes (Xu et al., 2020) regardless of location and weather conditions.

Prompted by this insight, we propose COOL, a simple but effective COncept-level cOntinual Learning strategy, that aims at *acquiring high-quality concepts and preserving them across tasks*. COOL makes use of a small amount of concept supervision to acquire high-quality concepts and explicitly preserves them with a *concept* rehearsal strategy, avoiding reasoning shortcuts in all tasks. COOL is applicable to a variety of NeSy architectures, and – as shown by our experiments with DeepProbLog (Manhaeve et al., 2018) and Concept-based Models (Koh et al., 2020) – easily outperforms state-of-the-art continual strategies on three novel, challenging NeSy-CL problems, achieving better concept quality and predictive accuracy on past and OOD tasks.

Contributions. Summarizing, we:

1. Introduce neuro-symbolic continual learning as a novel and challenging machine learning problem.
2. We show that knowledge readily improves forgetting in some scenarios, but also that it is insufficient to prevent reasoning shortcuts – which worsen forgetting and compromise transfer to new tasks – in others.
3. Propose new NeSy-CL benchmarks for evaluating continual performance with and without reasoning shortcuts.
4. Introduce COOL, a novel continual strategy that supports identifying concepts with the intended semantics and preserves them across tasks.
5. Show empirically that COOL outperforms state-of-the-art continual strategies on these challenging benchmarks.

2. Neuro-Symbolic Continual Learning

Notation. Throughout, we indicate scalar constants x in lower-case, random variables X in upper case, and ordered sets of constants $\mathbf{x}$ and random variables $\mathbf{X}$ in bold typeface. The symbol $[n]$ stands for the set $\{1, \ldots, n\}$ and $\mathbf{x} \models \mathsf{K}$ indicates that $\mathbf{x}$ satisfies a logical formula K. We say that a distribution $p(\mathbf{A} \mid \mathbf{B}; \mathsf{K})$ is *consistent* with K, written $p \models \mathsf{K}$, if it holds that $p(\mathbf{a} \mid \mathbf{b}; \mathsf{K}) > 0$ implies $(\mathbf{a}, \mathbf{b}) \models \mathsf{K}$, i.e., if p associates zero mass to all states that violate K.

2.1. Problem Statement

We are concerned with solving a *sequence of neuro-symbolic prediction tasks*, each requiring to learn a classifier mapping a (partially) sub-symbolic input $\mathbf{x} \in \mathbb{R}^d$ to $n \geq 1$ labels $\mathbf{y} \in \mathbb{N}^n$. What makes them neuro-symbolic is that: (*i*) The labels $\mathbf{y}$ depend entirely on the state of k *symbolic concepts* $\mathbf{c} = (c_1, \ldots, c_k)^\top$ capturing high-level aspects of the input $\mathbf{x}$. (*ii*) The concepts $\mathbf{c}$ depend on the *sub-symbolic input* $\mathbf{x}$ in an intricate manner and are best extracted using deep learning techniques. (*iii*) The way in which the labels $\mathbf{y}$ depend on the concepts $\mathbf{c}$ is specified by *prior knowledge* K, necessitating reasoning during inference.

We make the natural assumption that the *semantics of the concepts appearing in the various tasks remain constant over time*. This assumption lies at the heart of knowledge representation and ontology design, where concepts serve as a *lingua franca* for the exchange and reuse of knowledge across application boundaries (Gruber, 1995) and with human stakeholders (Kambhampati et al., 2022).

Formally, each task $t \in \mathbb{N}$ is defined by a data generating distribution $p^{(t)}(\mathbf{X}, \mathbf{C}, \mathbf{Y}; \mathsf{K}^{(t)})$ that factorizes as:

$$p^{(t)}(\mathbf{Y} \mid \mathbf{C}; \mathsf{K}^{(t)}) \cdot p(\mathbf{C} \mid \mathbf{X}) \cdot p^{(t)}(\mathbf{X}) \qquad (1)$$

Here, $\mathsf{K}^{(t)}$ denotes the knowledge relevant to the t-th task.[2] As customary in NeSy, we assume the knowledge to cor-

[2]The knowledge might depend also on discrete variables in $\mathbf{X}$; we suppress this dependency in the notation for readability.

rectly describe the ground-truth generative process and that, therefore, the label distributions are *consistent* with their respective prior knowledge, i.e., $p^{(t)}(\mathbf{Y} \mid \mathbf{X}; \mathsf{K}^{(t)}) \models \mathsf{K}^{(t)}$ for all t. The key feature of Equation (1) is that $p(\mathbf{C} \mid \mathbf{X})$ does not depend on t, capturing our assumption that concept semantics are stable. For instance, if $C_{\texttt{dog}}$ represents the notion of "dog", then $p(C_{\texttt{dog}} \mid \mathbf{X})$ only depends on whether an image $\mathbf{x}$ in fact depicts a dog, regardless of context, style, likelihood of observing a dog, and role of dogs in determining the label. See Appendix A.3 for an in-depth discussion. Critically, the distribution of observed inputs, concepts and labels, and the knowledge *are allowed to differ between tasks*. This means that, e.g., known concepts may stop occurring, play different roles in $\mathsf{K}^{(t+1)}$ than they did in $\mathsf{K}^{(t)}$, and entirely new concepts may appear.

Handling catastrophic forgetting. At step t, the machine obtains a task $\mathcal{T}^{(t)} = (\mathcal{D}^{(t)}, \mathsf{K}^{(t)})$ consisting of a data set $\mathcal{D}^{(t)}$, sampled i.i.d. according to Equation (1), and knowledge $\mathsf{K}^{(t)}$. The goal is to find parameters θ that achieve low *average risk* over all tasks observed so far, defined as:

$$\mathcal{L}(\theta, \mathcal{T}^{1:t}) = \frac{1}{t} \sum_{s \in [t]} \mathcal{L}(\theta, \mathcal{T}^{(s)}) \tag{2}$$

$$= \frac{1}{t} \mathcal{L}(\theta, \mathcal{T}^{(t)}) + \frac{t-1}{t} \mathcal{L}(\theta, \mathcal{T}^{1:(t-1)}) \tag{3}$$

where $\mathcal{T}^{1:t} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(t)}\}$ is the collection of all tasks observed so far, $\mathcal{L}(\theta, \mathcal{T}) := \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \ell(\theta, (\mathbf{x}, \mathbf{y}); \mathsf{K})$, and ℓ is a loss function over $\mathbf{y}$.

As in regular continual learning, we assume storage is insufficient to hold data from all tasks, meaning that $\mathcal{L}(\theta, \mathcal{T}^{1:(t-1)})$ cannot be evaluated exactly. Ignoring this term, as done by offline approaches, leads to *catastrophic forgetting*: by focusing on the loss of the current task, models tend to forget the information necessary to solve the previous tasks. CL algorithms mitigate forgetting using strategies like rehearsal (Buzzega et al., 2020; Parisi et al., 2019; De Lange et al., 2021; Boschini et al., 2022; Rebuffi et al., 2017) (i.e., replaying few examples of previous tasks), regularization (Huszár, 2018; Li & Hoiem, 2017) (i.e., slowing down parameter shift through additional terms in the loss function), or architectural modifications (Rusu et al., 2016) (i.e., freezing and adding new parameters at each task). See Section 6 for an overview.

Importantly, all these strategies focus on *optimizing the accuracy on the labels only*. This is sensible in CL but, as shown in Section 3, insufficient in NeSy-CL.

2.2. DeepProbLog

DeepProbLog (Manhaeve et al., 2018) is a state-of-the-art model ideally suited to solve tasks of the form in Equation (1). It decomposes prediction into two steps, cf. Figure 1 (left). At the lower level, it implements each concept as a Boolean or categorical random variable C_j, whose distribution $p_\theta(C_j \mid \mathbf{x})$ is flexibly parameterized by a neural network $nn_j(\mathbf{x}; \theta)$. This implies that concepts are mutually independent given the input, i.e., $p_\theta(\mathbf{C} \mid \mathbf{x}) = \prod_{j \in [k]} p_\theta(C_j \mid \mathbf{x})$. At the upper level, it models the distribution of labels conditioned on concepts as a *uniform distribution* over the support of K, and specifically as:[3]

$$u_{\mathsf{K}}(\mathbf{y} \mid \mathbf{c}) = \frac{1}{Z(\mathbf{c}; \mathsf{K})} \cdot \mathbb{1}\{(\mathbf{c}, \mathbf{y}) \models \mathsf{K}\} \tag{4}$$

where $Z(\mathbf{c}; \mathsf{K}) = \sum_{\mathbf{y}} \mathbb{1}\{(\mathbf{c}, \mathbf{y}) \models \mathsf{K}\}$ is a normalization constant. The overall label distribution is obtained by marginalizing over $\mathbf{C}$:

$$p_\theta(\mathbf{y} \mid \mathbf{x}; \mathsf{K}) = \sum_{\mathbf{c}} u_{\mathsf{K}}(\mathbf{y} \mid \mathbf{c}) \cdot \prod_{j \in [k]} p_\theta(c_j \mid \mathbf{x}) \tag{5}$$

Since the indicator function in Equation (4) evaluates to zero for all values of $\mathbf{c}$ that violate K, the label distribution in Equation (5) is *by construction* consistent with K.

Example 2.1. `MNIST-Addition` *(Manhaeve et al., 2018) is a prototypical neuro-symbolic task that requires learning a mapping from pairs of MNIST digits* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ *to their sum* Y*, e.g., from* $\mathbf{x} =$ () *to* $y = 5$*. It can be readily modelled in DeepProbLog using two concepts* C_1 *and* C_2 *ranging in* $\{0, \ldots, 9\}$*, each predicted by a convolutional network* $nn(\mathbf{x}_j)$*, and a constraint* $\mathsf{K} = (C_1 + C_2 = Y)$.

Given a task $\mathcal{T} = (\mathcal{D}, \mathsf{K})$, the parameters θ are usually learned by maximizing the log-likelihood $\mathcal{L}(\theta, \mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \log p_\theta(\mathbf{y} \mid \mathbf{x}; \mathsf{K})$ via stochastic gradient descent. Computing the (gradient of the) likelihood $p_\theta(\mathbf{y} \mid \mathbf{x})$ and the most likely prediction $\hat{\mathbf{y}} \in \operatorname{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x}; \mathsf{K})$ requires to evaluate Z, which is intractable in general. To make inference practical, DeepProbLog exploits knowledge compilation (Darwiche & Marquis, 2002) to convert the distribution u_{K} into a probabilistic circuit. Once in this format, the above operations take time linear in the size of the circuit (Choi et al., 2020; Vergari et al., 2021).

In the remainder of the paper, we focus on DeepProbLog as it offers a sound probabilistic architecture and exact inference. Our results, however, do transfer to many other neuro-symbolic architectures, as discussed in Section 6.

3. Knowledge and Reasoning Shortcuts

All neuro-symbolic architectures, including DeepProbLog, are designed for *offline* settings, and as such they easily fall prey of catastrophic forgetting when applied to NeSy-CL problems. This issue is illustrated by the following example and demonstrated empirically in Section 5.

[3] Non-uniform distributions consistent with the knowledge can also be modelled, as done for instance by Ahmed et al. (2022a).

Example 3.1. *We introduce* `MNAdd-Seq`*, a continual extension of* `MNIST-Addition` *in which tasks differ in what digits are observed. Specifically, each task* $t = 0, \dots, 8$ *consists of all pairs of digits* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ *whose sum is either* $2t$ *or* $2t+1$*. By construction, sums above* 9 *cannot be obtained by adding smaller digits, so these no longer occur in later tasks. Since DeepProbLog maximizes the likelihood of the current task, it quickly forgets small digits at* $t \geq 5$ *and can no longer classify sums involving them correctly.*

3.1. Knowledge Helps Remembering ...

A natural first step toward solving NeSy-CL is to bundle a NeSy predictor – say, DeepProbLog – with any state-of-the-art CL strategy. Focusing on experience replay, doing so amounts to storing a handful of well chosen labeled (or predicted) examples $(\mathbf{x}, \mathbf{y})$ from past tasks $1, \dots, t-1$, and replaying them when fitting DeepProbLog on the current task t together with the corresponding prior knowledge K^t.[4]

Doing so immediately brings a number of benefits. First and foremost, the knowledge encodes the valid, stable relationship between the concepts and the labels to be prediction. This implies that predicted concepts can be always correctly mapped to a corresponding label, and that this inference step is immune from forgetting. This is especially significant considering that the top layers of neural networks are those most affected by catastrophic forgetting (Wu et al., 2019). This effect is clearly visible in our experiments, cf. Section 5.

Conversely, prior knowledge effectively reduces the space of candidate concepts, providing further guidance to the model. If it reduces the space to only those having the intended semantics, then this simple setup can be very effective at tackling NeSy-CL problems.

3.2. ... But Does Not Prevent Reasoning Shortcuts

In general, however, this setup is insufficient. The core issue is that knowledge might not be enough to identify the right concept distribution $p(\mathbf{C} \mid \mathbf{X})$ using label annotations alone, in the sense that – depending on how the knowledge and training data are structured – it may be possible (in both offline and continual settings) to *correctly classify all training examples even using concepts with unintended semantics*. We refer to these situations as *reasoning shortcuts*.

This intuition is formalized in Theorem 3.2. Here, we write Θ to indicate the set of all possible parameters of (the neural networks implementing) $p_\theta(\mathbf{C} \mid \mathbf{X})$, and $\Theta^*(\mathsf{K}, \mathcal{D}) \subseteq \Theta$ for the parameters that maximize the log-likelihood $\mathcal{L}(\theta, \mathcal{D})$. Also, $\mathsf{K}[\mathbf{V}/\mathbf{v}]$ is the knowledge obtained by substituting all occurrences of variables $\mathbf{V}$ with constants $\mathbf{v}$. For instance, in `MNIST-Addition` $\mathsf{K}[Y/2]$ amounts to $C_1 + C_2 = 2$.

Theorem 3.2. *A model with parameters* θ *attains maximal likelihood, i.e.,* $\theta \in \Theta^*(\mathsf{K}, \mathcal{D})$*, if and only if, for all* $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$*, it holds that* $p_\theta(\mathbf{C} \mid \mathbf{x}) \models \mathsf{K}[\mathbf{Y}/\mathbf{y}]$.

All proofs can be found in Appendix A. Theorem 3.2 states that, as long as the concept distribution output by the learned neural network satisfies the knowledge for each training example, the log-likelihood is maximal.[5] The ground-truth concept distribution $p(\mathbf{C} \mid \mathbf{X})$ is a possible solution, but it is not necessarily *the only one*. In this case, fitting DeepProbLog – and indeed any NeSy approach that optimizes for label accuracy only – *does not guarantee that the learned concepts have the correct semantics*. To see this, consider the following example.

Example 3.3. *Consider* `MNIST-Addition` *and take a subset* $\mathcal{D}$ *including only pairs of examples of four possible sums:* 0 + 6 = 6, 4 + 6 = 10, 2 + 8 = 10, *and* 4 + 8 = 12. *Then, there exist many concept distributions that satisfy the knowledge on all examples, including:*

$$0 \mapsto 0, \quad 2 \mapsto 2, \quad 4 \mapsto 4, \quad 6 \mapsto 6, \quad 8 \mapsto 8$$
$$0 \mapsto 5, \quad 2 \mapsto 7, \quad 4 \mapsto 9, \quad 6 \mapsto 1, \quad 8 \mapsto 3$$

where the remaining concepts are allocated arbitrarily and $\mathbf{x} \mapsto c$ *is a shorthand for* $p(C = c \mid \mathbf{x}) = \mathbb{1}\{C = c\}$*. Only the first distribution has the intended semantics, whereas the second one is a reasoning shortcut. We remark that DeepProbLog does acquire this shortcut in practice, as illustrated by our experiments. Appendix D explains how shortcuts emerge in the data sets used in our experiments.*

Notice that the Theorem applies to both offline learning (i.e., $\mathcal{D}$ is fixed) and NeSy-CL (i.e., $\mathcal{D}$ indicates the training set of any given task). Yet, reasoning shortcuts are especially impactful in the latter. This is exemplified in Figure 1 (right). Here, DeepProbLog has learned high-quality concepts to solve the first task, but quickly forgets them when solving the second task, precisely because it falls pray of a reasoning shortcut that achieves high training and rehearsal accuracy on both tasks by satisfying the knowledge using concepts with unintended semantics. We provide additional concrete examples in Appendix D. In turn, reasoning shortcuts can dramatically affect forgetting and performance on future and OOD NeSy tasks, as shown by our experiments.

4. Addressing NeSy-CL with COOL

To this end, we introduce COOL, a COncept-level cOntinual Learning that acquires concepts with the intended semantics and preserves them over time, attaining sustained high

[4] The only non-trivial aspect is that, in addition to the replay buffer, we also has to store the past knowledge, so as to ensure be able to match the updated concepts with the past labels.

[5] This theorem essentially shows that, from the neural network's perspective, the reasoning layer of DeepProbLog has the same effect as the Semantic Loss (Xu et al., 2018).

performance. Formally, COOL is designed to satisfy two desiderata: (D1) $p_\theta(\mathbf{C} \mid \mathbf{X})$ should quickly approximate $p(\mathbf{C} \mid \mathbf{X})$, and (D2) $p_\theta(\mathbf{C} \mid \mathbf{X})$ should remain stable across tasks. D2 is straightforward, however we stress that it is only meaningful if D1 also holds: unless the learned concepts are high-quality, there is little benefit in remembering them.

In order to comply with D1, COOL makes use of a small number of densely annotated examples to quickly identify high-quality concepts, which – as we have shown – cannot always be guaranteed using knowledge alone. In practice, an average cross-entropy is added to the loss of these examples. To cope with D2, COOL implements a novel *concept rehearsal* strategy that stabilizes $p_\theta(\mathbf{C} \mid \mathbf{X})$ across tasks. This is motivated by the fact that concept stability helps to upper bound the average risk. Specifically,

Theorem 4.1. *Consider tasks $\mathcal{T}^{1:t}$. If the current model θ and the past one $\theta^{(t-1)}$ assign non-zero likelihood to all examples in $\mathcal{D}^{1:t}$, there exists a finite constant γ, depending only on the model architecture, knowledge and data, such that the average risk in Equation (3) is upper bounded by:*

$$\frac{1}{t}\mathcal{L}(\theta, \mathcal{D}^{(t)}) + \frac{t-1}{t}\Big[\mathcal{L}(\theta^{(t-1)}, \mathcal{D}^{1:(t-1)}) + \gamma \sum_{s \leq t} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}^{(s)}} \|p_\theta(\mathbf{C} \mid \mathbf{x}) - p_{\theta^{(t-1)}}(\mathbf{C} \mid \mathbf{x})\|_1\Big] \tag{6}$$

In words, this means that if the past model $\theta^{(t-1)}$ performs well on all past tasks (i.e., the middle term in Equation (6) is small), a new model that performs well on the current task (the first term is small) and whose concept distribution is close to that of the old model (the last term is small), also performs well on past tasks (the average risk in Equation (3) is small). Critically, this results holds regardless of how the prior knowledge $\mathsf{K}^{1:t}$ of the various task is chosen.

COOL implement this requirement by combining the original training loss with an extra penalty $\mathcal{L}_{\text{COOL}}$, defined as:

$$\mathcal{L}_{\text{COOL}} := \frac{1}{|\mathcal{M}|} \sum_{(\mathbf{x},\tilde{\mathbf{q}}_c,\mathbf{y}) \in \mathcal{M}} \big[\alpha \cdot \mathsf{KL}\big(p_\theta(\mathbf{C} \mid \mathbf{x}) \,\|\, \tilde{\mathbf{q}}_c\big) - \beta \cdot \log p_\theta(\mathbf{Y} = \mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(t)})\big] \tag{7}$$

Here, $\mathcal{M}$ denotes the mini-batch of examples extracted from the replay buffer, α denotes the scalar weight associated to the concept-reharsal strategy, and β the weight of the replay strategy on $\mathbf{y}$. The KL term is evaluated between the predicted concept distribution and the stored one $\tilde{\mathbf{q}}_c = p_{\theta^{(t-1)}}(\mathbf{C} \mid \mathbf{x})$. Notice that, by Pinsker's inequality, the KL upper bounds the (square of the) L_1 distance, meaning that COOL indirectly optimizes the bound in Equation (6).

4.1. Benefits and Limitations

COOL is explicitly designed to acquire high-quality concepts and retain them across tasks by combining knowledge, concept rehearsal, and a modicum of concept supervision. This substantially improves performance on past, future, and OOD tasks sharing these concepts, as demonstrated in Section 5. COOL works even if the knowledge $\mathsf{K}^{(t)}$ changes across tasks and new concepts appear over time: these can be encoded as additional neural predicates in DeepProbLog, and COOL will take care of remembering the known concepts while leaving room to learn the new ones.

One limitation of COOL is that, in the general case, it requires a handful of densely annotated examples. The same requirement can be found in other settings where concept quality is critical. For instance, concept supervision is key in concept-based models – which strive to generate concept-level explanations for their predictions – to ensure the acquired concepts are *interpretable* (Koh et al., 2020; Chen et al., 2020; Marconato et al., 2022b). It is also a prerequisite for guaranteeing that learned representations acquired by general (deep) latent variable models are *disentangled*, as shown theoretically (Locatello et al., 2019) and empirically (Locatello et al., 2020). We stress that concept supervision is *not* required if knowledge and data disallow shortcut solutions, as is the case in `MNAdd-Seq` (see Section 5), although it does help avoiding sub-optimal parameters even in this case. If reasoning shortcuts *are* possible, however, concept supervision becomes essential, because – by construction – knowledge and labels alone are insufficient to pin down the correct semantics, hindering concept quality. Moreover, in many situations, annotating just *some* concepts is sufficient to rule out reasoning shortcuts.

5. Empirical Evaluation

We address empirically the following research questions:

Q1: Does knowledge help to stabilize the continual learning process and reduce the need for supervision?

Q2: Does COOL help avoid reasoning shortcuts when knowledge alone fails, thus facilitating past and future continual performance?

Q3: How much concept supervision does COOL need?

To answer these questions, we compared COOL against several representative continual strategies on three *novel* and challenging NeSy-CL benchmarks. Additional results and details on data sets, metrics, and hyperparameters can be found in the Appendices.

Data sets. Existing NeSy and CL benchmarks are designed for offline settings or lack any sort of prior knowledge, respectively. Hence, in order to evaluate COOL, we introduce three novel NeSy-CL benchmarks specifically designed to evaluate impact of knowledge, concept quality and robustness to reasoning shortcuts, briefly described next.

MNAdd-Seq is the problem introduced in Example 3.1 and it is designed *not* to contain reasoning shortcuts. In short, inputs $\mathbf{x}$ are pairs of MNIST (LeCun, 1998) digits labeled with their sum y; each digit is mapped to a concept, and K specifies that their sum must match the label. In each task $t = 0, \ldots, 8$ includes only examples with labels $y \in \{2 \cdot t, 2 \cdot t + 1\}$, making this problem both *label incremental* (only two out of 18 possible labels are observed per task) and *concept incremental* (higher digits only appear in later tasks, while lower digits disappear, see Appendix C). The data set holds 42k training examples, of which we used 8.4k for validation, and 6k test examples.

MNAdd-Shortcut is a simple two-task version of MNIST-Addition used here to illustrate the impact of *reasoning shortcuts*. The first task includes only even digits and the second one only odd ones. In the first task we include 4 types of examples: (i) 0 + 6 = 6, (ii) 4 + 6 = 10, (iii) 2 + 8 = 10, and (iv) 4 + 8 = 12. In the second task, we allow all possible sums of odd digits $\{1, 3, 5, 7, 9\}$. As shown in Example 3.3 and further discussed in Appendix D, the four sums in the first task are not sufficient to identify the correct digits, i.e., different shortcuts are possible. This data consists of 13.8k examples, 2.8k of which are reserved for validation and 2k for testing. We also include an additional OOD test set with 4k unseen combinations of all concepts, like sums involving an odd and an even digit, allowing us to probe the efficacy on a plausible future task.

CLE4EVR is a challenging new *concept-incremental* NeSy-CL benchmark based on CLEVR (Johnson et al., 2017). Inputs $\mathbf{x}$ are renderings of two randomly placed 3D objects with several possible shapes, colors, materials, and sizes. The goal is to predict whether the objects have the same color, same shape, both, or neither. The knowledge simply defines the three labels using four (one-hot encoded) concepts encoding shape and color of the two objects. There are 5 tasks. Objects in each task have only two colors out of ten and two shapes out of ten, with no overlap between tasks. Knowledge and labels allow for a large number of *reasoning shortcuts*, as illustrated in Figure 1 and detailed in Appendix D. To evaluate quality of learned concepts, we also define an OOD test set containing *unseen combinations* of training objects. Overall, the dataset contains almost 5.5k training data, 500 data for validation and 2.5k data for test.

Metrics. We evaluate all models using common CL metrics (De Lange et al., 2021), namely class-incremental accuracy (Class-IL) of labels $\mathbf{Y}$ and concepts $\mathbf{C}$, and forward transfer of labels (FWT). For MNAdd-Shortcut and CLE4EVR we also measure label and concept accuracy on the OOD test set.

Competitors. We compare COOL against the following *label-based* continual strategies: NAÏVE fine-tunes the old model on each new task without any continual strategy.

	Strategy	Class-IL $\mathbf{Y}$ (↑)	Class-IL $\mathbf{C}$ (↑)	FWT (↑)
CBM @ 10%	NAÏVE	11.71 ± 0.8	36.2 ± 2.6	7.5 ± 0.3
	RESTART	10.78 ± 0.1	29.7 ± 0.1	7.3 ± 0.2
	LWF	18.08 ± 1.8	63.2 ± 4.4	-4.7 ± 1.1
	EWC	11.57 ± 0.6	37.4 ± 0.6	7.6 ± 0.4
	ER	13.29 ± 0.4	43.5 ± 2.0	13.4 ± 1.6
	DER	18.63 ± 2.5	53.1 ± 1.7	15.7 ± 0.9
	DER++	18.17 ± 1.6	54.1 ± 3.0	16.6 ± 1.8
	COOL	$\mathbf{38.0 \pm 1.9}$	$\mathbf{78.1 \pm 2.5}$	$\mathbf{29.0 \pm 4.8}$
DeepProbLog	NAÏVE	6.9 ± 0.2	6.7 ± 0.4	6.2 ± 0.2
	RESTART	9.6 ± 0.3	0.2 ± 0.1	6.9 ± 0.8
	LWF	6.8 ± 0.5	10.8 ± 4.6	18.3 ± 0.2
	EWC	6.8 ± 0.4	7.8 ± 0.6	6.1 ± 0.3
	ER	44.3 ± 9.7	62.0 ± 8.6	8.2 ± 4.1
	DER	68.3 ± 9.4	81.3 ± 6.9	44.5 ± 23.7
	DER++	62.2 ± 5.4	77.1 ± 4.2	27.1 ± 5.2
	COOL	$\mathbf{71.9 \pm 2.9}$	$\mathbf{84.5 \pm 1.9}$	$\mathbf{83.2 \pm 0.9}$

Table 1. **Knowledge helps, and COOL helps even more.** Top block: results on MNAdd-Seq for all competitors + CBM with 10% concept supervision, averaged over 10 seeds. Bottom block: same for DeepProbLog with 0% supervision. COOL + DeepProbLog outperforms the neural baseline (despite the gap in supervision) and the other continual strategies. The additional results in Appendix E support these conclusions.

RESTART fits a model from scratch for each task, without any continual strategy. RESTART and NAÏVE serve as baselines to quantify the impact of forgetting. LWF: Learning without Forgetting (Li & Hoiem, 2017), a regularization approach that performs knowledge distillation from the past model. EWC: Elastic Weight Consolidation (Kirkpatrick et al., 2017), a regularization approach that avoids drastic updates to important parameters based on the Fisher values of the previous model. ER: Experience Replay (Riemer et al., 2019), a popular rehearsal approach that stores a random selection of past examples and replays them when training on the new task. DER: Dark Experience Replay (Buzzega et al., 2020), a state-of-the-art rehearsal approach similar to ER that stores and distills the logits of the past model. DER++: an improvement of DER that also stores the true label. We do not compare against prototype-based strategies, like iCaRL (Rebuffi et al., 2017), because class prototypes cannot be easily defined in structured representation spaces. We also consider OFFLINE learning over the union of all tasks as an ideal upper bound. COOL is implemented as in Equation (7). Following DER, replay examples are selected with *reservoir sampling* (Vitter, 1985), an efficient incremental method for random sampling with uniformity guarantees. All hyperparameters were chosen to optimize last-task Class-IL on the validation labels.

Q1: Knowledge helps, COOL helps even more. We evaluate the impact of knowledge on MNAdd-Seq, where shortcuts are *absent*. Specifically, we evaluate different continual strategies paired with DeepProbLog (Manhaeve et al., 2018) and Concept-bottleneck Models (CBMs) (Koh et al., 2020). Both architectures extract concepts using a convolutional network, but differ in how they infer the label. DeepProbLog uses a probabilistic-logic layer encoding the available knowledge (see Section 2.2). CBMs aggregate

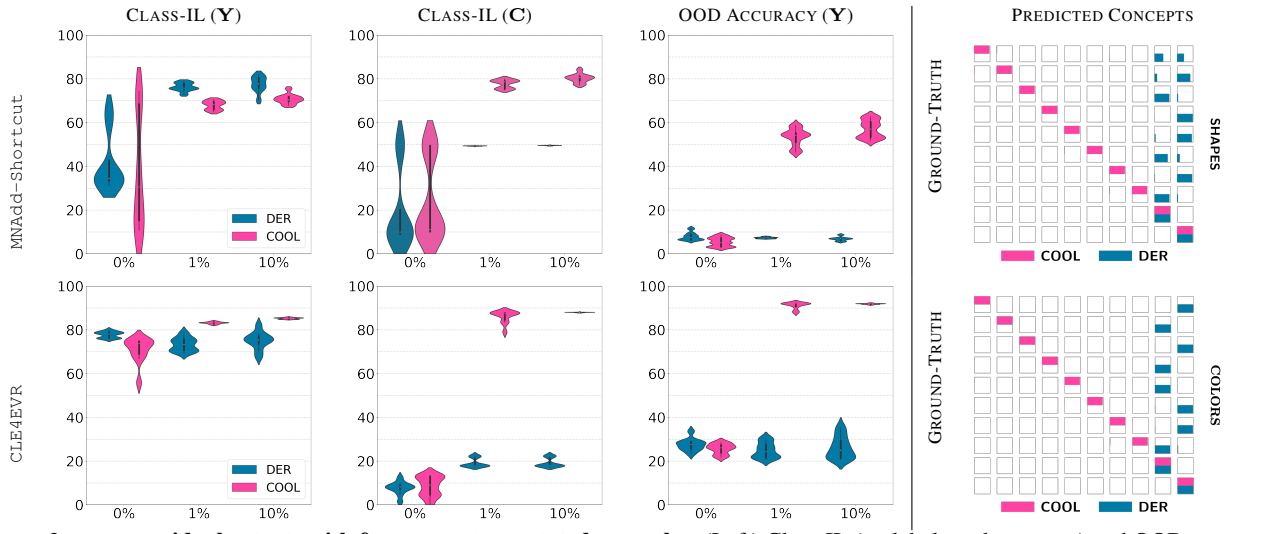

Figure 2. **COOL avoids shortcuts with few concept-annotated examples.** (Left) Class-IL (on labels and concepts) and OOD accuracy for DER and COOL on `MNAdd-Shortcut` (top) and `CLE4EVR` (bottom). The x-axis is the % of concept annotated examples per task. (Right) Confusion matrices of `shape` (top) and `color` (bottom), computed on the last task of `CLE4EVR`, obtained by DeepProbLog paired with COOL (in **blue**) vs. DER (in **red**) with 10% concept supervision. COOL is the only strategy that acquires and maintains the intended semantics. The complete numerical results are reported in Appendix E.

the concepts using a learnable network *independent* of the knowledge, and serve as a purely neural baseline. Additional architectural choices are reported in Appendix B. The two models were trained for 25 epochs per task, using a fixed buffer size of 1000, but received different amounts of concept supervision: 10% for CBMs, which is enough to learn the intended concepts in the offline setting, and *none* for DeepProbLog.

The *offline* performance of CBM and DeepProbLog are excellent, achieving around 96% label accuracy and 99% concept accuracy, showing that both are capable of solving the learning task. In the *continual* setting, however, the gap between the neural and NeSy models widens noticeably. The results are reported in Table 1. Despite DeepProbLog being harder to learn than CBMs (as shown by RESTART and NAÏVE), all replay strategies – i.e., ER, DER, DER++, and COOL– perform much better when paired with DeepProbLog than CBMs: remarkably, label Class-IL sees gains close to 50% for DER, and similarly FWT for COOL. This highlights the benefits of knowledge, which apply despite DeepProbLog having access to *no concept supervision*. The regularization strategies LWF and EWC are not informative, as they struggle to improve on the NAÏVE and RESTART baselines both *with* and *without* knowledge.[6]

Overall, Table 1 indicates that *when reasoning shortcuts are absent*, knowledge facilitates identifying better concepts, and thus better predictions. By retaining these concepts, COOL manages to outperform all other competitors on both CBMs and DeepProbLog. The runner-up, DER, keeps up only when knowledge is available and with a substantial margin in terms of FWT (45% vs. 83%). In contrast, even though CBMs can acquire good concepts, this does not always yield good predictions, chiefly because the top layer undergoes forgetting, cf. Section 3.1. Thanks to knowledge, DeepProbLog avoids this issue altogether.

Q2: COOL avoids reasoning shortcuts. Next, we evaluate the impact of concept quality and rehearsal in `MNAdd-Shortcut` and `CLE4EVR`, which *are* affected by reasoning shortcuts. Given the sub-par performance of CBMs, we focus on DeepProbLog from now on. Also, we restrict our attention to COOL and DER, the runner up in the previous experiment. Results for all other competitors are available in Appendix E. For `MNAdd-Shortcut` we set a buffer size of 1000 and 100 epochs per task, and to 250 examples and 50 epochs for `CLE4EVR`.

The results in Figure 2 shows that, when no concept supervision is in place, the presence of shortcuts complicates retaining the correct concepts, as displayed by low values of Class-IL (**C**). This does not impact directly Class-IL (**Y**) in the case of `CLE4EVR`, but yields extremely low OOD generalization, around 10% for `MNAdd-Shortcut` and 25% for `CLE4EVR`. The effect on increasing concept supervision on DER is only seemingly positive, as label accuracy does improve in both data sets. However, Class-IL on concepts (about 20–50%) and OOD accuracy (10%–30%) are very poor, despite the supervision. What happens is precisely the issue depicted in Figure 1: the model acquires good concepts for one task, but – due to reasoning shortcuts and lack of concept rehearsal – these get corrupted when fitting on the next task. Since COOL retains the high quality

[6]The only exception is LWF on CBM, which displayed pathological behavior, see Appendix E.

concepts identified via supervision, the latter leads to clear improvements in label accuracy, concept accuracy *and* OOD accuracy for COOL. As a result, COOL improves on DER by about $+30\%$ and $+60\%$ in terms of Class-IL (**C**) and $+40\%$ and $+60\%$ in OOD, in the two data sets respectively.

We stress that label-based strategies inevitably fall for reasoning shortcuts *even if concept supervision is provided*. This is clearly shown by the concept confusion matrices reported in Figure 2 (right). Notice that, out of *all* strategies, only COOL manages to prevent shortcuts. Further details are available in Appendix E.

Q3: COOL requires minimal concept supervision. Figure 2 shows that COOL identifies high-quality concepts when given dense annotations for only 1% of the training set. This translates to about 30 examples per task in `MNAdd-Shortcut`, and to only 12 in `CLE4EVR`. Increasing concept supervision to 10% improves Class-IL (**C**) by 3% and shrinks its variance, but 1% is enough to substantially outperform DER in our tests.

6. Related Work

Neuro-symbolic integration. NeSy encompasses a diverse family of methods integrating learning and reasoning (De Raedt et al., 2021). Here, we focus on approaches for encouraging neural networks to output structured predictions consistent with prior knowledge.The two main strategies introduce an additional loss penalizing inconsistent predictions (Xu et al., 2018; Fischer et al., 2019; Ahmed et al., 2022b) or a top reasoning layer (Manhaeve et al., 2018; Giunchiglia & Lukasiewicz, 2020; Hoernle et al., 2022; Ahmed et al., 2022a). Since the former cannot guarantee that the model outputs consistent predictions, we focus on the latter. In either case, end-to-end training requires to differentiate through the knowledge. One option is to soften the knowledge using fuzzy logic (Diligenti et al., 2012; Donadello et al., 2017), but doing so can introduce semantic and learning artifacts (Giannini et al., 2018; van Krieken et al., 2022a). An alternative is to cast reasoning in terms of probabilistic logics (De Raedt & Kimmig, 2015), which preserves semantics and allows for sound inference and learning. DeepProbLog is just an example of Nesy strategies (Manhaeve et al., 2021; Huang et al., 2021; Winters et al., 2022; Ahmed et al., 2022a; van Krieken et al., 2022b). All NeSy approaches are *offline* and suffer from catastrophic forgetting, and existing continual strategies do not protect them from reasoning shortcuts, as shown in Section 5. Since these depend only on the latent nature of concepts, they affect probabilistic-logic and fuzzy logic architectures alike. COOL applies to all these, cf. Appendix A.4.

Continual Learning. CL algorithms attempt to preserve model plasticity while mitigating catastrophic forgetting (Robins, 1995) using a variety of techniques (van de Ven et al., 2022; Qu et al., 2021). A first group of strategies, like Experience Replay (Riemer et al., 2019) and ER-ACE (Caccia et al., 2022), store and rehearse a limited amount of examples from previous tasks. Doing so ignores additional "dark knowledge" learned by the past model, so techniques like DER (Buzzega et al., 2020), DER++, and others (Rebuffi et al., 2017; Li & Hoiem, 2017; Castro et al., 2018; Hou et al., 2019), drop rehearsal in favor of distillation. COOL follows the same strategy. Popular alternatives include architectural approaches (Rusu et al., 2016), which freeze or add model parameters as needed, and regularization strategies (De Lange & Tuytelaars, 2021; Kirkpatrick et al., 2017; Aljundi et al., 2018; Zenke et al., 2017). These introduce extra penalties in the loss function to discourage changing parameters essential for discriminating classes, but can struggle with complex data (Aljundi et al., 2019). To the best of our knowledge, CL has only been tackled in flat prediction settings (e.g., classification), and existing strategies focus on preserving label accuracy only. The only work on forgetting in CBMs is (Marconato et al., 2022a), which however ignores knowledge altogether.

Reasoning shortcuts. In machine learning, "shortcuts" refer to models that exploit *spurious* correlations between inputs and annotations to achieve high training accuracy (Ross et al., 2017; Lapuschkin et al., 2019). Proposed solutions include dense annotations (Ross et al., 2017), out-of-domain data (Parascandolo et al., 2020), and interaction with annotators (Teso et al., 2022). Stammer et al. (2021) have investigated shortcuts in NeSy and proposed to fix them using knowledge, under the assumption that concepts are high-quality. We make no such assumption. Our work is the first to investigate *reasoning* shortcuts that knowledge cannot always prevent and their preminence in NeSy-CL.

7. Conclusion

We initiated the study of Neuro-Symbolic Continual Learning and showed that knowledge, although useful, can be insufficient to prevent acquiring reasoning shortcuts that compromise concept semantics and cross-task transfer. Our approach, COOL, acquires and preserves high-quality concepts, attaining better concepts and performance than existing CL strategies in three new NeSy-CL benchmarks.

Acknowledgements

We acknowledge the support of the MUR PNRR project FAIR - Future AI Research (PE00000013) funded by the NextGenerationEU. The research of AP and ST was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation program under GA No 952215. We acknowledge the CINECA award under the

ISCRA initiative, for the availability of high performance computing resources and support. The research of SC was partially supported by Italian Ministerial grant PRIN 2020 "LEGO.AI: LEarning the Geometry of knOwledge in AI systems", n. 2020TA3K9N. The research of EF was partially supported by the European Union's Horizon 2020 research and innovation program DECIDER under Grant Agreement 965193. We acknowledge Angelo Porrello for his useful discussion with us.

References

Ahmed, K., Teso, S., Chang, K.-W., Van den Broeck, G., and Vergari, A. Semantic Probabilistic Layers for Neuro-Symbolic Learning. In *NeurIPS*, 2022a.

Ahmed, K., Wang, E., Chang, K.-W., and Van den Broeck, G. Neuro-symbolic entropy regularization. In *UAI*, 2022b.

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

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

Boschini, M., Bonicelli, L., Buzzega, P., Porrello, A., and Calderara, S. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Caccia, L., Aljundi, R., Asadi, N., Tuytelaars, T., Pineau, J., and Belilovsky, E. New insights on reducing abrupt representation change in online continual learning. *ICLR*, 2022.

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

Chen, Z., Bei, Y., and Rudin, C. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2020.

Choi, Y., Vergari, A., and Van den Broeck, G. Probabilistic circuits: A unifying framework for tractable probabilistic models. *UCLA*, 2020.

Darwiche, A. and Marquis, P. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

De Lange, M. and Tuytelaars, T. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8250–8259, 2021.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *PAMI*, 2021.

De Raedt, L. and Kimmig, A. Probabilistic (logic) programming concepts. *Machine Learning*, 2015.

De Raedt, L., Dumančić, S., Manhaeve, R., and Marra, G. From statistical relational to neural-symbolic artificial intelligence. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 4943–4950, 2021.

Diligenti, M., Gori, M., Maggini, M., and Rigutini, L. Bridging logic and kernel machines. *Machine learning*, 86(1): 57–88, 2012.

Donadello, I., Serafini, L., and Garcez, A. D. Logic tensor networks for semantic image interpretation. In *IJCAI*, 2017.

Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C., and Vechev, M. Dl2: Training and querying neural networks with logic. In *International Conference on Machine Learning*, pp. 1931–1941. PMLR, 2019.

Giannini, F., Diligenti, M., Gori, M., and Maggini, M. On a convex logic fragment for learning and reasoning. *IEEE Transactions on Fuzzy Systems*, 2018.

Giunchiglia, E. and Lukasiewicz, T. Coherent hierarchical multi-label classification networks. *NeurIPS*, 2020.

Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995.

Hoernle, N., Karampatsis, R. M., Belle, V., and Gal, K. Multiplexnet: Towards fully satisfied logical constraints in neural networks. In *AAAI*, 2022.

Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

Huang, J., Li, Z., Chen, B., Samel, K., Naik, M., Song, L., and Si, X. Scallop: From probabilistic deductive databases to scalable differentiable reasoning. *NeurIPS*, 2021.

Huszár, F. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 115(11):E2496–E2497, 2018.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

Kambhampati, S., Sreedharan, S., Verma, M., Zha, Y., and Guan, L. Symbols as a Lingua Franca for Bridging Human-AI Chasm for Explainable and Advisable AI Systems. In *Proceedings of Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B., and Liang, P. Concept bottleneck models. In *ICML*, 2020.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K.-R. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8, 2019.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

Li, Z. and Søgaard, A. Qlevr: A diagnostic dataset for quantificational language and elementary visual reasoning. In *Findings of NAACL*, 2022.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, 2019.

Locatello, F., Tschannen, M., Bauer, S., Rätsch, G., Schölkopf, B., and Bachem, O. Disentangling factors of variations using few labels. In *International Conference on Learning Representations*, 2020.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. DeepProbLog: Neural Probabilistic Logic Programming. *NeurIPS*, 2018.

Manhaeve, R., Marra, G., and De Raedt, L. Approximate inference for neural probabilistic logic programming. In *KR*, 2021.

Marconato, E., Bontempo, G., Teso, S., Ficarra, E., Calderara, S., and Passerini, A. Catastrophic forgetting in continual concept bottleneck models. In *Image Analysis and Processing. ICIAP 2022 Workshops: ICIAP International Workshops, Lecce, Italy, May 23–27, 2022, Revised Selected Papers, Part II*, pp. 539–547. Springer, 2022a.

Marconato, E., Passerini, A., and Teso, S. Glancenets: Interpretabile, leak-proof concept-based models. *NeurIPS*, 2022b.

Misino, E., Marra, G., and Sansone, E. VAEL: Bridging Variational Autoencoders and Probabilistic Logic Programming. *NeurIPS*, 2022.

Parascandolo, G., Neitz, A., ORVIETO, A., Gresele, L., and Schölkopf, B. Learning explanations that are hard to vary. In *International Conference on Learning Representations*, 2020.

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Qu, H., Rahmani, H., Xu, L., Williams, B., and Liu, J. Recent advances of continual learning in computer vision: An overview. *arXiv preprint arXiv:2109.11369*, 2021.

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

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. 2019.

Robins, A. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995.

Ross, A. S., Hughes, M. C., and Doshi-Velez, F. Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2662–2670, 2017.

Rost, B., Liu, J., Nair, R., Wrzeszczynski, K. O., and Ofran, Y. Automatic prediction of protein function. *Cellular and Molecular Life Sciences CMLS*, 60(12):2637–2650, 2003.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Stammer, W., Schramowski, P., and Kersting, K. Right for the Right Concept: Revising Neuro-Symbolic Concepts by Interacting with their Explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3619–3629, 2021.

Teso, S., Alkan, Ö., Stammer, W., and Daly, E. Leveraging explanations in interactive machine learning: An overview. *arXiv preprint arXiv:2207.14526*, 2022.

van de Ven, G. M., Tuytelaars, T., and Tolias, A. S. Three types of incremental learning. *Nature Machine Intelligence*, pp. 1–13, 2022.

van Krieken, E., Acar, E., and van Harmelen, F. Analyzing differentiable fuzzy logic operators. *Artificial Intelligence*, 2022a.

van Krieken, E., Thanapalasingam, T., Tomczak, J. M., van Harmelen, F., and Teije, A. t. A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. *arXiv preprint arXiv:2212.12393*, 2022b.

Vergari, A., Choi, Y., Liu, A., Teso, S., and Van den Broeck, G. A compositional atlas of tractable circuit operations for probabilistic inference. *Advances in Neural Information Processing Systems*, 34, 2021.

Vitter, J. S. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1): 37–57, 1985.

Winters, T., Marra, G., Manhaeve, R., and De Raedt, L. DeepStochLog: Neural Stochastic Logic Programming. In *AAAI*, 2022.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. Large scale incremental learning. In *CVPR*, 2019.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Broeck, G. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, 2018.

Xu, Y., Yang, X., Gong, L., Lin, H.-C., Wu, T.-Y., Li, Y., and Vasconcelos, N. Explainable object-induced action decision for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9523–9532, 2020.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence, 2017.

A. Proofs

A.1. Proof of Theorem 3.2

Taking Equation (5) as reference, the log-likelihood of $\mathcal{D}$ can be rewritten as:

$$\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \log p_\theta(\mathbf{y} \mid \mathbf{x}; \mathsf{K}) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \log \langle u_{\mathsf{K}}(\mathbf{y} \mid \cdot), p_\theta(\cdot \mid \mathbf{x})\rangle \tag{8}$$

Here, the inner product runs over all possible values of $\mathbf{c}$. To see what the optima of this quantity look like, fix a single example $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ and let $\mathcal{C}_{\mathbf{y}}$ be the set of values $\mathbf{c}$ that satisfy the knowledge $\mathsf{K}[\mathbf{Y}/\mathbf{y}]$ and $\mathcal{C}_{\bar{\mathbf{y}}}$ be those that violate the knowledge. The likelihood of $(\mathbf{x}, \mathbf{y})$ amounts to:

$$\langle u_{\mathsf{K}}(\mathbf{y} \mid \cdot), p_\theta(\cdot \mid \mathbf{x})\rangle = \sum_{\mathbf{c}\in\mathcal{C}_{\mathbf{y}}} u_{\mathsf{K}}(\mathbf{y} \mid \mathbf{c}) p_\theta(\mathbf{c} \mid \mathbf{x}) + \sum_{\mathbf{c}\in\mathcal{C}_{\bar{\mathbf{y}}}} \underbrace{u_{\mathsf{K}}(\mathbf{y} \mid \mathbf{c})}_{=0} p_\theta(\mathbf{c} \mid \mathbf{x}) \tag{9}$$

The inner product is maximized whenever $p_\theta(\mathbf{C} \mid \mathbf{x})$ allocates *all* probability mass to values $\mathbf{c}$ that satisfy $\mathsf{K}[\mathbf{Y}/\mathbf{y}]$, because the remaining ones do not contribute anything to the likelihood. Hence, in order to maximize Equation (8) it is sufficient that, for every $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$, $p_\theta(\mathbf{C} \mid \mathbf{x})$ assigns zero probability to all concept configurations $\mathbf{c}$ that are ruled out by the knowledge $\mathsf{K}[\mathbf{Y}/\mathbf{y}]$.

□

Remarks: This result essentially states that DeepProbLog's reasoning layer has the same effect as the Semantic Loss (Xu et al., 2018) on the underlying neural network, and it is of independent interest. Notice that Theorem 3.2 also holds for non-uniform label distributions without any change to the proof.

A.2. Proof of Theorem 4.1

We start by proving a general lemma:

Lemma A.1. *Consider tasks $\mathcal{T}^1, \ldots, \mathcal{T}^{(t)}$ and two parameter configurations $\varphi, \psi \in \Theta$. If both models assign non-zero likelihood to all examples in $\mathcal{D}^{1:t}$, there exists a finite constant γ, depending only on the model architecture, knowledge and data, such that:*

$$|\mathcal{L}(\varphi, \mathcal{T}^{1:t}) - \mathcal{L}(\psi, \mathcal{T}^{1:t})| \leq \gamma \sum_{s\leq t} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}^{(s)}} \|p_\varphi(\mathbf{C} \mid \mathbf{x}) - p_\psi(\mathbf{C} \mid \mathbf{x})\|_1 \tag{10}$$

Proof. The left-hand side can be expanded to:

$$\frac{1}{t}\left|\sum_{s\leq t} \frac{1}{|\mathcal{D}^{(s)}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}^{(s)}} \Big(\log p_\varphi(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)}) - \log p_\psi(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)})\Big)\right| \tag{11}$$

$$\leq \frac{1}{t}\sum_{s\leq t} \frac{1}{|\mathcal{D}^{(s)}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}^{(s)}} \Big|\log p_\varphi(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)}) - \log p_\psi(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)})\Big| \tag{12}$$

Recall that, for any $a, b \in [\beta, \infty]$, it holds that $|\log a - \log b| \leq \frac{1}{\beta}|a - b|$. For any $(\mathbf{x}, \mathbf{y})$ and task $s \leq t$, it holds that:

$$|\log p_\varphi(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)}) - \log p_\psi(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)})| \leq \frac{1}{\beta}\big|p_\varphi(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)}) - p_\psi(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)})\big| \tag{13}$$

$$= \frac{1}{\beta}\big|\langle u_{\mathsf{K}^{(s)}}(\mathbf{y} \mid \cdot), p_\varphi(\cdot \mid \mathbf{x}) - p_\psi(\cdot \mid \mathbf{x})\rangle\big| \leq \frac{1}{\beta}\|u_{\mathsf{K}^{(s)}}(\mathbf{y} \mid \cdot)\|_\infty \cdot \|p_\varphi(\cdot \mid \mathbf{x}) - p_\psi(\cdot \mid \mathbf{x})\|_1 \tag{14}$$

The first step follows by choosing $\beta := \min_{(\mathbf{x},\mathbf{y})\in\mathcal{D}^{1:t}} \min_{\theta\in\{\varphi,\psi\}} p_\theta(\mathbf{y} \mid \mathbf{x}; \mathsf{K}^{(s)}) > 0$, the second one from Equation (5), and the last one from Hölder's inequality. By Equation (4), the max norm of $u_{\mathsf{K}^{(s)}}$ amounts to:

$$\|u_{\mathsf{K}^{(s)}}(\mathbf{y} \mid \cdot)\|_\infty = \max_{\mathbf{c}} \frac{\mathbb{1}\{(\mathbf{c}, \mathbf{y}) \models \mathsf{K}^{(s)}\}}{Z(\mathbf{c}; \mathsf{K}^{(s)})} = \frac{1}{\min_{\mathbf{c}} Z(\mathbf{c}; \mathsf{K}^{(s)})} \leq \frac{1}{\zeta} \leq 1 \tag{15}$$

where we chose $\zeta = \min_{\mathbf{c}} \min_{s \leq t} Z(\mathbf{c}; \mathsf{K}^{(s)})$. Therefore,

$$\frac{1}{\beta}\|u_{\mathsf{K}^{(s)}}(\mathbf{y} \mid \cdot)\|_\infty \cdot \|p_\varphi(\cdot \mid \mathbf{x}) - p_\psi(\cdot \mid \mathbf{x})\|_1 \leq \frac{1}{\beta\zeta}\|p_\varphi(\cdot \mid \mathbf{x}) - p_\psi(\cdot \mid \mathbf{x})\|_1 \tag{16}$$

Taking $\gamma = \max_s \frac{1}{\beta\zeta|\mathcal{D}^{(s)}|t}$ and replacing the above in Equation (12) yields the claim. □

In order to prove Theorem 4.1, let $\theta^{(t-1)}$ be the parameters learned after observing $t-1$ tasks and θ to be learned at the current iteration. Applying Lemma A.1 with $\varphi = \theta^{(t-1)}$ and $\psi = \theta$ to Equation (3) yields the desired result.

Remark: In the worst case ζ can be as small as 1, which occurs if in all tasks $s \leq t$ the knowledge $\mathsf{K}^{(s)}$ accepts a single concept configuration $\mathbf{c}$ for every example $(\mathbf{x}, \mathbf{y})$; more commonly, ζ is exponential in the number of concepts k. Also, β, which is the minimum likelihood attained by either p_φ or p_ψ on the data sets $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(t)}$, is actively maximized during learning.

A.3. What Are Correct Semantics?

Strictly speaking, the only concept distribution with the actual *correct semantics* is the ground-truth distribution $p(\mathbf{C} \mid \mathbf{x})$.

Our assumption is that the ground-truth concept distribution we are given is always consistent with the knowledge, in the sense that $p(\mathbf{C} \mid \mathbf{x}) \models \mathsf{K}^{(t)}[\mathbf{X}/\mathbf{x}, \mathbf{Y}/\mathbf{y}]$ for every possible task $\mathcal{T}^{(t)} = (\mathcal{D}^{(t)}, \mathsf{K}^{(t)})$ and example $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}^{(t)}$. In other words, we assume that the knowledge correctly reflects how the world works. Under this assumption, having the correct semantics is useful *in practice* because, when paired with the knowledge, the ground-truth distribution by construction always yields correct labels in every possible task the learner can in principle receive. This is a form of *systematic generalization*.

Naturally, if the learned distribution $p_\theta(\mathbf{C} \mid \mathbf{x})$ matches the ground-truth distribution exactly, then it will also achieve systematic generalization. This condition, however, is very restrictive. Pragmatically, we can relax this requirement, and say that a distribution encodes the correct semantics if it is *indistinguishable from the ground-truth distribution in terms of what concepts it predicts*. Formally, we say that a distribution $p_\theta(\mathbf{C} \mid \mathbf{x})$ is *semantically equivalent* to the ground-truth distribution on data $\mathcal{D}$ if it allows us to infer the same concept configuration for all data points, or formally:

$$\forall \mathbf{x} \in \mathcal{D} \,.\, \operatorname*{argmax}_{\mathbf{c}} \; p(\mathbf{c} \mid \mathbf{x}) \equiv \operatorname*{argmax}_{\mathbf{c}} \; p_\theta(\mathbf{c} \mid \mathbf{x}) \tag{17}$$

where we used $\equiv$ to indicate set equivalence. This is quite intuitive: any distribution satisfying Equation (17) will yield the same concepts $\mathbf{c}$ as $p(\mathbf{C} \mid \mathbf{x})$ for every $\mathbf{x}$, and therefore also the same MAP states $\mathbf{y}$ under any choice of knowledge K.

The opposite is *not* generally true: the knowledge K might have multiple possible solutions, in the sense that different choices of concepts $\mathbf{c}$ might yield the same label $\mathbf{y}$. In this case, the label does not carry enough information to recover the ground-truth concepts $\operatorname{argmax}_{\mathbf{c}} p(\mathbf{c} \mid \mathbf{x})$, and therefore also a concept distribution $p_\theta(\mathbf{C} \mid \mathbf{x})$ semantically equivalent to $p(\mathbf{C} \mid \mathbf{x})$. This is exactly what we mean by *reasoning shortcuts*: concept distributions that achieve high performance on the observed task(s) but have no guarantee of systematically generalizing to future tasks, or more formally:

$$\operatorname*{argmax}_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}; \mathsf{K}) \equiv \operatorname*{argmax}_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x}; \mathsf{K}) \qquad \wedge \qquad \operatorname*{argmax}_{\mathbf{c}} p(\mathbf{c} \mid \mathbf{x}) \not\equiv \operatorname*{argmax}_{\mathbf{c}} p_\theta(\mathbf{c} \mid \mathbf{x}) \tag{18}$$

A.4. Reasoning Shortcuts in other NeSy Architectures

While Theorem 3.2 shows that reasoning shortcuts do affect DeepProbLog, which maximizes for label likelihood, we remark that they are a *general* phenomenon. It is easy to see that reasoning shortcuts occur whenever the prior knowledge admits deducing the correct *label* $\mathbf{y}$ from *concepts* $\mathbf{c}$ that do not have the correct semantics, as this makes it is impossible for a model to distinguish between concepts with "correct" vs. "incorrect" semantics by maximizing accuracy alone. This impacts *offline* NeSy prediction tasks and NeSy-CL problems alike; indeed, Theorem 3.2 makes no assumption on how the training set $\mathcal{D}$ has been generated.

Reasoning shortcuts are not specific to DeepProbLog. On the contrary, this situation can be triggered by a variety of other NeSy architectures, including but not limited to:

(*i*) NeSy predictors that rely on a top reasoning layer to ensure predictions are consistent with prior knowledge, which are typically trained to maximize some surrogate of the label accuracy, including (Ahmed et al., 2022a; Giunchiglia & Lukasiewicz, 2020; Hoernle et al., 2022; Huang et al., 2021; Winters et al., 2022; van Krieken et al., 2022b).

(*ii*) NeSy predictors that rely on *relaxed* reasoning layers obtained by softening the logical prior knowledge (Diligenti et al., 2012; Donadello et al., 2017), because this transformation usually preserves existing optima of the label accuracy. As such, it also preserves unintended optima – that is, reasoning shortcuts.

(*iii*) Neural networks trained to maximize accuracy and consistency with prior knowledge using the Semantic Loss and similar techniques (Xu et al., 2018; Fischer et al., 2019; Ahmed et al., 2022b). In fact, Theorem 3.2 shows that DeepProbLog is affected by shortcuts precisely because, from the neural network's perspective, its reasoning layer acts exactly like the Semantic Loss; see our remark in Appendix A.

More generally, reasoning shortcuts impact NeSy tasks and architectures beyond these, at least as long as models are trained by optimizing loss functions that do not measure or correlate with concept quality. We leave a detailed analysis of specific cases to future work.

B. Implementation Details

In this Section, we report useful details for the models and the metrics adopted in the evaluation.

B.1. Hardware and Software Implementation

The code for the project was developed on top `mammoth` (Boschini et al., 2022), a well-known CL framework. We included the implementation of DeepProbLog for `MNIST-Addition` from VAEL (Misino et al., 2022). The generation of `CLE4EVR` was adapted from (Stammer et al., 2021). All experiments were implemented using Python 3 and Pytorch (Paszke et al., 2019) and run on a server with 128 CPUs, 1TiB RAM, and 8 A100 GPUs.

B.2. Metrics

We adopted standard CL measures, namely task-incremental (Task-IL) and class-incremental (Class-IL) accuracy, applied here to both labels and concepts predictions, as well as forward transfer (FWT) and backward transfer (BWT) on the labels, see also (Buzzega et al., 2020). Below we write T to indicate the last task.

- **Class-IL** measures the average accuracy on the test sets of all tasks T. In Table 1, we report Class-IL at the very last task T, defined as:

$$\textsc{Class-IL}_{\mathbf{Y}}(\theta_T) = \frac{1}{T}\sum_{s=1}^{T} \mathcal{A}_{\mathbf{Y}}(\theta_T, s) \tag{19}$$

  where $\mathcal{A}_{\mathbf{Y}}(\theta_t, s)$ denotes the accuracy on the labels evaluated on the test set of task s. Class-IL for concepts is analogous, but builds on the average accuracy over *all* concepts.

- **Task-IL** is the average accuracy over the test sets of all tasks up to t, evaluated only on examples annotated with the classes or concepts appearing in task t. The definition is identical to Equation (19) except that we mask the prediction of model so as to place mass only on the labels appearing in $\mathcal{D}^s$, with $s \leq t$.

- **FWT** evaluates the adaptability of the model at each time-step to the successive task. Formally, at each t, FWT measures the average gain in accuracy between θ_t and a random baseline θ_{rand} when predicting the labels of the task $t+1$. This can be written as:

$$\text{FWT} = \frac{1}{T-1}\sum_{t=1}^{T-1} \mathcal{A}_{\mathbf{Y}}(\theta_t, t+1) - \mathcal{A}_{\mathbf{Y}}(\theta_{\text{rand}}, t+1) \tag{20}$$

  where θ_{rand} denotes the initialized model with random weights.

- **BWT** measures how much forgetting the model undergoes by looking at how much accuracy for each task t is retained after the last task. Formally:

$$\text{BWT} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{A}_{\mathbf{Y}}(\theta_t, t) - \mathcal{A}_{\mathbf{Y}}(\theta_T, t) \tag{21}$$

For the sake of brevity, in the main paper we reported only Class-IL on labels and concepts, and FWT. Task-IL was omitted because it does not account for accuracy on past concepts that no longer occur in the last task, and BWT because it does not as informative as Class-IL. All results for these extra metrics on MNAdd-Seq are reported in Appendix E.

B.3. Architectures & Models Details

MNIST-Addition: For both CBMs and DeepProbLog we adopted the same architecture for extracting concepts – henceforth, *encoder* – and we implemented it as a standard convolutional neural network, with dropout set at 50% after each convolution module. We also inject a noise term $\epsilon \sim \mathcal{N}(0, 0.1)$ after the encoder to stabilize the overall training process. The complete structure is reported in Table 2.

Table 2. CNN Encoder for MNIST-Addition

INPUT SHAPE	LAYER TYPE	PARAMETERS	ACTIVATION
$(28, 28, 1)$	Convolution	depth=64, kernel=4, stride=2, padding=1	ReLU
$(14, 14, 64)$	Dropout	$p = 0.5$	
$(14, 14, 64)$	Convolution	depth=128, kernel=4, stride=2, padding=1	ReLU
$(7, 7, 128)$	Dropout	$p = 0.5$	
$(7, 7, 128)$	Convolution	depth=256, kernel=4, stride=2, padding=1	ReLU
$(3, 3, 256)$	Flatten		
(2304)	Linear	dim=10, bias = True	

In both CBMs and DeepProbLog, each input digit $\mathbf{x}^{(i)}$ is predicted independently and mapped to a 10-dimensional bottleneck $\mathbf{z}^{(i)}$. Then, the two encodings are stacked together, obtaining the overall representation $\mathbf{z} = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)})$.

The classifier (top layer) of the CBM is designed to predict the sum the $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$. A simple linear layer, which is the standard choice in CBM (Koh et al., 2020), is insufficient to successfully address the task. Therefore, we implemented the classifier via a bi-linear operation on the encodings, i.e.,:

$$p_\theta(y|\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) = \text{softmax}\left(\mathbf{z}^{(1)} \cdot W^y \mathbf{z}^{(2)}\right)$$

where W^y is a learnable class-specific 10×10 tensor of real entries, and the softmax is over all classes $y \in \{0, \dots, 17\}$.

Conversely, the *reasoning* (top) layer of DeepProbLog is implemented as in Equation (5). Specifically, each $\mathbf{z}^{(i)}$ encodes the logits of the probability $p_\theta(C_i|\mathbf{x}^{(i)})$ for the i-th digit, which we convert into a categorical probability distribution through a softmax activation.

CLE4EVR: In a first step, we used Faster-RCNN (Ren et al., 2015) to extract the bounding boxes associated to objects in all images. The bounding box predictor is a pretrained MobileNet (Howard et al., 2017) fine-tuned on training images from the first task only, using the ground-truth bounding boxes of CLE4EVR images. The bounding box predictor is kept frozen in all successive tasks. We discarded all examples $\mathbf{x}_i$ with less than 2 predicted bounding boxes. Concept supervision was transferred from the ground-truth bounding boxes to the predicted ones based on overlap.

We scale each predicted bounding box to an image of size $28 \times 28 \times 3$ which is then passed to the encoder, implemented once again using a CNN with dropout with $p = 50\%$ after each convolution layer and normal noise $\epsilon \sim \mathcal{N}(0, 0.1)$ added to the final output. The architecture is the same as in Table 2, except that the input has depth 3 instead of 1 and that the bottleneck is 20-dimensional, with 10 dimensions allocated for the shape and 10 for the color of the input object, each with its own softmax to produce shape and color probability distributions. The DeepProbLog *reasoning* layer is as in Equation (5).

During inference, the prediction returns invalid ($\perp$) whenever the number of predicted bounding boxes is less than 2. We counted invalid predictions as wrong predictions in all metrics evaluated in the test set.

B.4. Hyper-parameter Selection

All continual strategies have been trained with the same number of epochs and buffer dimension. The actual values depend on the specific benchmark: 25 epochs per task and a buffer size of 1000 examples for MNAdd-Seq, 100 epochs and 1000 examples for MNAdd-Shortcut, and 50 epochs each task and 250 examples for CLE4EVR. In all experiments, we employed the Adam optimizer (Kingma & Ba, 2015) combined with exponential decay ($\gamma = 0.95$). The initial learning rate is restored at the beginning of a new task.

For each data set, we optimized the weight of the concept supervision w_c based on the Class-IL $(\mathbf{Y})$ performance of ER using grid-search on the validation set (union of all tasks). Then, for each strategy, we selected the best learning rate and strategy-specific hyperparameters through a grid-search on the validation set, so as to optimize Class-IL $(\mathbf{Y})$ on a single random seed. The learning rate was chosen from the range of $[10^{-5}, 10^{-2}]$. The exact values can be found in the source code.

C. NeSy-CL Benchmarks

In this section, we provide a more detailed description of the benchmarks introduced in Section 5.

C.1. `MNAdd-Seq`

We derived `MNAdd-Seq` from the `MNIST-Addition` data set of Manhaeve et al. (2018). Here, the knowledge encodes the following constraint:

$$\mathsf{K} = \forall c_1, c_2 \in \{0, \dots, 9\}\ (C_1 = c_1 \wedge C_2 = c_2) \implies Y = (c_1 + c_2) \tag{22}$$

for a total of 19 possible sums. `MNAdd-Seq` is both *label-incremental* and *concept-incremental*; in each task only two possible sums appear, obtaining in total 9 tasks. Specifically:

Task 1: $Y \in \{0, 1\}$ and $C \in \{$,  $\}$;

Task 2: $Y \in \{2, 3\}$ and $C \in \{$  ,  , ,  $\}$;

Task 3: $Y \in \{4, 5\}$ and $C \in \{$  ,  , ,  ,  ,  $\}$;

Task 4: $Y \in \{6, 7\}$ and $C \in \{$, , , , , ,  ,  $\}$;

Task 5: $Y \in \{8, 9\}$ and $C \in \{$  ,  , ,  ,  ,  ,  ,  ,  ,  $\}$;

Task 6: $Y \in \{10, 11\}$ and $C \in \{$  , ,  ,  ,  ,  ,  ,  ,  $\}$;

Task 7: $Y \in \{12, 13\}$ and $C \in \{$, , , , , , $\}$;

Task 8: $Y \in \{14, 15\}$ and $C \in \{$, , , , $\}$;

Task 9: $Y \in \{16, 17\}$ and $C \in \{$, , $\}$;

In total, the data set counts 42k training and 6k test examples.

C.2. `MNAdd-Shortcut`

This benchmark is a case-study composed of two task, built considering the following constraints:

- In the first task, we present only even digits and four possible sums: (i) $+$$= 6$, (ii) $+$$= 10$, (iii) $+$$= 10$, and (iv) $+$$= 12$.
- In the second task, only odd numbers are considered, i.e., $C \in \{$, , , , $\}$, and all their possible sums.

The rationale behind this construction is that it makes it possible to satisfy the knowledge in both tasks by leveraging reasoning shortcuts, and specifically those described in Appendix D.

The overall data set contains 13.8k training examples and 2k test data. We also collected an OOD test set containing examples not appearing in the training, validation and test sets, which comprise sums of odd and even digits, e.g., $+$$= 1$, and unseen combinations of even numbers, e.g., $+$$= 16$.

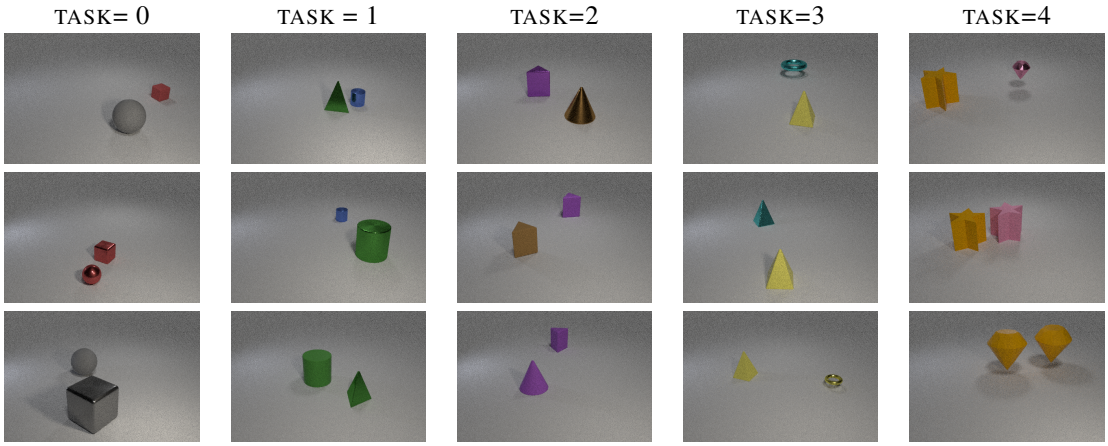

Figure 3. Examples images for each tasks in the CLE4EVR benchmark.

C.3. CLE4EVR

Following Stammer et al. (2021), the CLE4EVR data set was generated using Blender (Blender Online Community, 2018), using additional objects from (Li & Søgaard, 2022) as well as three custom shapes.

We generated different objects with 10 possible shapes and colors, and with 2 free variations on material $\{\texttt{rubber}, \texttt{metal}\}$ and size $\{\texttt{small}, \texttt{big}\}$, that play no role in determining the label. Each image is of size $256 \times 430 \times 3$, and contains two non-overlapping objects over a light-gray background.

Table 3 reports what combinations of objects appear in each task. All tasks are composed of 1.1k training examples, 100 validation examples, and 500 test examples. Each task includes only two possible shapes (out of ten) and two colors (out of ten), without any overlap between tasks. An illustration is given in Figure 3.

The knowledge $\mathsf{K} = \mathsf{K}' \wedge \mathsf{K}'' \wedge \mathsf{K}'''$ encodes the following constraints:

$$\mathsf{K}' = (C_{\texttt{shape},1} = C_{\texttt{shape},2}) \iff \texttt{same\_shape} \tag{23}$$
$$\mathsf{K}'' = (C_{\texttt{color},1} = C_{\texttt{color},2}) \iff \texttt{same\_color} \tag{24}$$
$$\mathsf{K}''' = (\texttt{same\_shape} \wedge \texttt{same\_color}) \iff \texttt{same} \tag{25}$$

with three output variables $Y_1 = \texttt{same\_shape}$, $Y_2 = \texttt{same\_color}$ and $Y_3 = \texttt{same}$, giving rise to four mutually exclusive classes: 0 = different shape and color, 1 = same shape and different color, 2 = different shape and same color, 3 = same shape and same color.

We also generated an additional OOD test set, comprising 300 images depicting *unseen combination* of training objects, e.g., redsquares and pinkdiamonds (which occur in none of the tasks). All OOD examples all have label $\mathbf{Y} = (0, 0, 0)$.

All generated images come with ground-truth bounding boxes annotated with the properties (i.e., concepts) of the objects they contain, as well as annotations for Y_1, Y_2, and Y_3. The concept annotations are transferred to the bounding boxes predicted by Faster R-CNN during pre-processing, cf. Appendix B.

Table 3. Task Organization in CLE4EVR

TASK	COLORS	SHAPES
1	red, gray	sphere, cube
2	green, blue	cylinder, tetrahedron
3	brown, purple	cone, triangular prism
4	yellow, cyan	pyramid, toroid
5	orange, pink	diamond, star prism

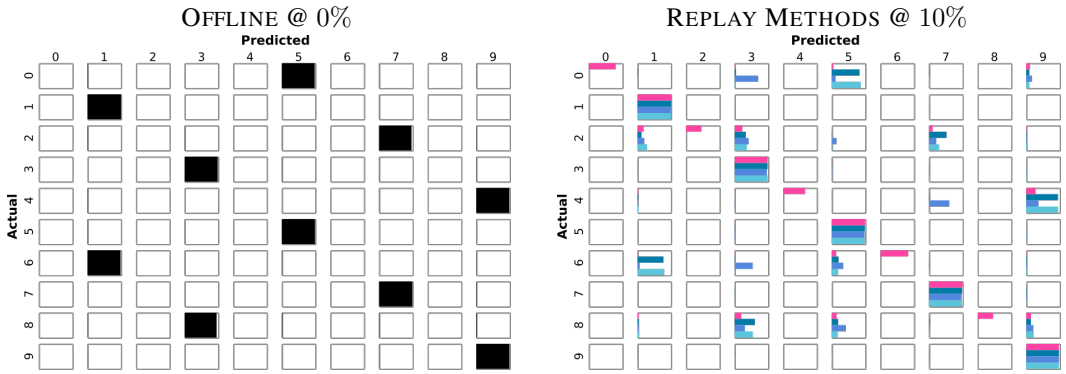

Figure 4. Confusion matrices of the learned concepts at the end of the very last task, for a single run. **Left**: The confusion matrix on concepts for OFFLINE without concept supervision. It shows that it is very likely to opt for a reasoning shortcut. **Right**: Confusion matrices on a single run for all replay-based methods. Here, COOL is pictured in **red**, while DER, DER++, and ER in shades of **blue**. Only COOL retains the correct semantics of the concepts, whereas the other replay methods are very likely to pick the shortcut of OFFLINE.

D. Shortcut Solutions in `MNAdd-Shortcut` and `CLE4EVR`

In this section, we provide a more detailed account on the shortcut solutions for the continual scenarios introduced.

D.1. Reasoning Shortcuts Due to Low-Level Correlations

Before proceeding, we observe that simply predicting multiple concepts jointly by a single neural network is sufficient to enable reasoning shortcuts. Intuitively, this happens because, since the network has access to all properties of all objects, it can automatically exploit correlations between them to satisfy the knowledge without the need for extracting any "proper" concepts. For instance, in `MNIST-Addition` the knowledge can simply group pairs of digits (both of which it has access to) into two single combinations of concepts that always yield the right labels. To see this, consider the following example:

Example D.1. *Consider a single* `MNIST-Addition` *task consisting of digits summing to either* 2 *or* 3*. By Theorem 3.2,* $\Theta^*(\mathsf{K}, \mathcal{D})$ *contains* φ *with* $p_\varphi(\mathbf{C} \mid \mathbf{x}) \not\equiv p(\mathbf{C} \mid \mathbf{x})$*, such that:*

$$p_\varphi(C_1 \mid \mathbf{x}) = \mathbb{1}\{C_1 = 2\}$$
$$p_\varphi(C_2 \mid \mathbf{x}) = \begin{cases} \mathbb{1}\{C_2 = 0\} & \text{if } \mathbf{x} = (\textbf{0}, \textbf{2}) \text{ or } (\textbf{1}, \textbf{1}) \\ \mathbb{1}\{C_2 = 1\} & \text{otherwise} \end{cases}$$

This distribution fits the data perfectly and is consistent with the knowledge, and thus cannot be distinguished from the ground-truth distribution based on likelihood alone.

Notice that the two learned concepts ignore the value of the individual digits, and rather depend on the correlation between them. in `MNIST-Addition`, it is straightforward to avoid this situation *by construction* by simply processing the two digits independently. The same can be done when processing objects in `CLE4EVR`. This partially motivates our choice of neural architecture, described in Appendix B. More precisely, in `MNIST-Addition` the adopted architecture factorizes the joint probability distribution on the concepts in:

$$p(\mathbf{c}_1, \mathbf{c}_2 | \mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = p(\mathbf{c}_1 | \mathbf{x}^{(1)}) \cdot p(\mathbf{c}_2 | \mathbf{x}^{(2)}) \tag{26}$$

While this is sufficient to guarantee independence among distinct objects, the prior knowledge can still admit reasoning shortcuts. Next, we describe the concrete reasoning shortcuts appearing in `MNAdd-Shortcut` and `CLE4EVR`.

D.2. Shortcuts in `MNAdd-Shortcut`

In order to understand reasoning shortcuts in `MNAdd-Shortcut`, it is useful to view the sums as constraints on the possible values attributed to each concept C_j. From this perspective, reasoning shortcuts occur whenever the combinations of digits present in the training data (which are five in each task) are insufficient to uniquely determine the four possible sums.

In particular, the problem of assigning the intended semantics of each learned concept can be expressed as a system of 4 linear equations with 5 variables, which in task 1 of MNAdd-Seq can be written as:

$$\begin{cases} c_0 & & + c_6 & & = 6 \\ & & c_4 + c_6 & & = 10 \\ & c_2 & & + c_8 & = 10 \\ & & c_4 & + c_8 & = 12 \end{cases} \tag{27}$$

The first equation states that whatever values are assigned to the concepts that fire when a 0 and a 6 are present in the input $\mathbf{x}$, have to satisfy the condition that their sum is 6 (in all examples in which they appear). It should be clear that the linear system is undetermined and infinitely many real solutions can be found for c_i, all of which except one do not capture the intended semantics of digits.

One of these unintended solutions is very often picked by label-replay strategies. Specifically, the input 4 can be easily confused for a 9, due to input similarity, which brings the model towards the assignment $c_4 = 9$. This immediately yields the following unintentional mappings for all other digits: $c_0 = 5$, $c_2 = 7$, $c_6 = 1$, and $c_8 = 3$. This reasoning shortcut is often found when training offline on this task and also when training on both tasks of MNAdd-Shortcut, as done in our experiments, by ER, DER, and DER++, as shown in Figure 4.

D.3. Shortcuts in CLE4EVR

Several shortcut exist in CLE4EVR. Recall that:

- CLE4EVR is composed of 5 tasks, with four possible outcomes (*different objects*, *same shapes*, *same colors*, and *same objects*).
- In each task only two possible shapes and colors are observed, and are never seen again in other tasks.

In order to correctly classify the same_color and same_shape labels, the model must acquire at least two distinct concepts for shape and two distinct concepts for colors in each task, but the knowledge provides little guidance as to *which* shapes or colors need to be associated to the input objects. This leaves ample room for reasoning shortcuts.

Let us focus on shapes only. Letting $\mathcal{S}$ be the set of the 10 possible shapes s_i, the model needs at least $10 \cdot (10-1)/2$ different negative examples of the knowledge to uniquely identify all possible shapes (up to permutation), one for each pair of mismatching shapes. Consider the map $\pi : \mathcal{S} \to \mathcal{S}$ mapping from ground-truth shape of the input object to the learned concept for shape. Ideally, we would like π to be injective, so that no two distinct ground-truth shapes are mapped to the same learned shape.

Consider a task with 4 possible shapes $s_1 = \texttt{cube}$, $s_2 = \texttt{cone}$, $s_3 = \texttt{cylinder}$, and $s_4 = \texttt{toroid}$. In order to guarantee injectivity, the data has to include enough combinations of shapes so that the map π satisfies the following condition:

$$\begin{aligned} &\pi(s_1) \neq \pi(s_2), \quad \pi(s_1) \neq \pi(s_3), \quad \pi(s_1) \neq \pi(s_4), \\ &\pi(s_2) \neq \pi(s_3), \quad \pi(s_2) \neq \pi(s_4), \\ &\pi(s_3) \neq \pi(s_4) \end{aligned} \tag{28}$$

Notice that this map *is* injective, in the sense that $s_i \neq s_j \implies \pi(s_i) \neq \pi(s_j),\ \forall i \neq j$. All tasks in CLE4EVR, however, are designed to distinguishing between only *two* possible shapes (and colors), hence the condition in Equation (28) is never satisfied. This is what allows for reasoning shortcuts.

As a matter of fact, without concept supervision, all values for shapes and colors are equally likely to be picked up to solve the task. We observed this phenomenon in the case of 0% supervision reported in Figure 5.

Figure 5. Confusion matrices for COOL (in **red**) and the other replay strategies (in shades of **blue**) in CLE4EVR with no concept supervision, at the end of all tasks. All methods fails to attribute the correct semantics to the concepts and learn, instead, a shortcut which optimizes the Class-IL ($\mathbf{Y}$).

	MNAdd-Seq (BUFFER SIZE = 1000)		CLE4EVR (BUFFER SIZE = 250)	
	TIME PER EPOCH	MEMORY OCCUPATION	TIME PER EPOCH	MEMORY OCCUPATION
NAÏVE	0.238 s	-	0.335 s	-
ER	0.385 s	6.280 Mb	0.344 s	1.570 Mb
DER	0.475 s	6.348 Mb	0.412 s	1.587 Mb
DER++	0.667 s	6.356 Mb	0.482 s	1.589 Mb
COOL	0.476 s	6.360 Mb	0.439 s	1.590 Mb

Table 4. **Time per epoch and memory occupation** for MNIST-Addition and CLE4EVR.

E. Additional Results and Metrics

E.1. Time Overhead and Memory Occupation of COOL

We evaluate the time overhead and the memory requirements of COOL compared to other replay-based strategies in Table 4. All these strategies impose an additional time overhead due to the selection and replay of past examples.

We measured the time required for completing a single epoch in the first task of MNIST-Addition and CLE4EVR. NAÏVE provides the lower bound for the computation time, as it just fine-tunes the model on the new examples. ER in MNIST-Addition strategy requires almost ~ 0.1 s more than NAÏVE for storing and replaying. In CLE4EVR, however, the gap is less evident between the two strategies. For MNAdd-Seq, the time per epoch of COOL amounts to ~ 0.476 s, which is comparable to that of DER ~ 0.475 s and lower than that of DER++ ~ 0.667 s. For CLE4EVR COOL requires 0.439 s per epoch, slightly increasing compared to DER ~ 0.412 s, but still being lower w.r.t. DER++ ~ 0.482 s.

In terms of memory occupation, COOL requires slightly more memory than other replay strategies, due to the need of storing the *logits* of the learned concepts. However, most of the overhead is due to storing the instances $\mathbf{x}$ themselves, which is the very same for all strategies and amounts to 6.272 Mb for MNIST-Addition and 1.568 Mb for CLE4EVR.

E.2. MNAdd-Seq

We report in Table 5 results for all competing strategies and performance measures used (including those omitted in the main text), namely Class-IL and Task-IL on labels and concepts (denoted as $\mathbf{Y}$ and $\mathbf{C}$, respectively), backward transfer (BWT), and forward transfer (FWT).

These results confirm the ones reported in Section 5. Specifically, COOL attains higher performance w.r.t. all metrics in both CBM and DeepProbLog. They also show that COOL outperforms all other approaches in terms of BWT when paired with DeepProbLog, and is the runner-up with CBMs.

The method with the best BWT on CBM is LWF, which however displays a pathological learning behavior, as made clear by the fact that it is the only method to have *negative* forward transfer. The reason is that LWF struggles to learn the first few tasks properly, but performs reasonably on the latter ones.

		Labels (**Y**)		Concepts (**C**)			
	Strategy	Class-IL (↑)	Task-IL (↑)	Class-IL (↑)	Task-IL (↑)	BTW (↑)	FWT (↑)
CBM @ 10%	naïve	11.71 ± 0.8	27.5 ± 3.9	36.2 ± 2.6	50.8 ± 1.7	-91.1 ± 1.2	7.5 ± 0.3
	restart	10.78 ± 0.1	32.9 ± 2.1	29.7 ± 0.1	43.1 ± 0.6	-98.2 ± 0.1	7.3 ± 0.2
	lwf	18.08 ± 1.8	55.7 ± 5.3	63.2 ± 4.4	78.6 ± 2.5	$\mathbf{-18.5 \pm 1.8}$	-4.7 ± 1.1
	ewc	11.57 ± 0.6	38.3 ± 11.2	37.4 ± 0.6	52.1 ± 1.7	-90.4 ± 0.8	7.6 ± 0.4
	er	13.29 ± 0.4	32.5 ± 3.2	43.5 ± 2.0	67.7 ± 3.3	-88.0 ± 0.7	13.4 ± 1.6
	der	18.63 ± 2.5	50.3 ± 3.4	53.1 ± 1.7	73.1 ± 2.0	-88.6 ± 2.9	15.7 ± 0.9
	der++	18.17 ± 1.6	48.9 ± 2.4	54.1 ± 3.0	73.1 ± 4.5	-89.2 ± 1.8	16.6 ± 1.8
	cool	$\mathbf{38.0 \pm 1.9}$	$\mathbf{78.4 \pm 3.8}$	$\mathbf{78.1 \pm 2.5}$	$\mathbf{90.4 \pm 1.6}$	-68.3 ± 2.1	$\mathbf{29.0 \pm 4.8}$
DeepProbLog @ 0%	naïve	6.9 ± 0.2	7.6 ± 0.2	6.7 ± 0.4	16.4 ± 1.0	-63.1 ± 0.6	6.2 ± 0.2
	restart	9.6 ± 0.3	22.2 ± 0.8	0.2 ± 0.1	11.6 ± 0.5	-69.5 ± 1.9	6.9 ± 0.8
	lwf	6.8 ± 0.5	11.0 ± 1.6	10.8 ± 4.6	21.7 ± 8.3	-71.2 ± 5.2	18.3 ± 0.2
	ewc	6.8 ± 0.4	7.8 ± 0.4	7.8 ± 0.6	18.3 ± 0.6	-62.9 ± 0.4	6.1 ± 0.3
	er	44.3 ± 9.7	81.2 ± 6.5	62.0 ± 8.6	82.8 ± 5.1	-48.3 ± 8.8	8.2 ± 4.1
	der	68.3 ± 9.4	94.8 ± 3.2	81.3 ± 6.9	93.8 ± 3.0	-30.5 ± 8.5	44.5 ± 23.7
	der++	62.2 ± 5.4	93.5 ± 2.1	77.1 ± 4.2	92.8 ± 2.9	-34.8 ± 5.6	27.1 ± 5.2
	cool	$\mathbf{71.9 \pm 2.9}$	$\mathbf{96.6 \pm 0.8}$	$\mathbf{84.5 \pm 1.9}$	$\mathbf{95.4 \pm 0.4}$	$\mathbf{-28.7 \pm 3.2}$	$\mathbf{83.2 \pm 0.9}$

Table 5. **Additional results for** `MNAdd-Seq`**.**

Surprisingly, this oddball behavior is sufficient for LWF to beat the baselines (NAÏVE and RESTART) in terms of Class-IL **Y** (at around 18%), but not enough to outperform COOL, and also yields the aforementioned issue with FWT. We stress that this implies that LWF generalizes to forward tasks *worse than random*.

E.3. `MNAdd-Shortcut`

We report here in Table 6, all results obtained for all strategies in `MNAdd-Shortcut`. We performed the comparison adopting only DeepProbLog with increasing amount of concept supervision.

	Supervision 0%			Supervision 1%			Supervision 10%		
	Class-IL(**Y**)	Class-IL(**C**)	OOD (**Y**)	Class-IL(**Y**)	Class-IL(**C**)	OOD (**Y**)	Class-IL(**Y**)	Class-IL(**C**)	OOD (**Y**)
naïve	59.9 ± 3.2	49.4 ± 0.4	10.7 ± 2.3	57.5 ± 0.5	49.2 ± 0.1	12.9 ± 0.4	59.9 ± 0.9	49.4 ± 0.1	12.3 ± 0.5
restart	59.1 ± 0.6	$\mathbf{49.7 \pm 0.1}$	$\mathbf{12.9 \pm 0.4}$	58.1 ± 1.6	49.3 ± 0.1	12.9 ± 0.5	59.1 ± 0.9	49.6 ± 0.1	13.2 ± 0.6
ewc	55.9 ± 8.9	45.6 ± 11.4	12.5 ± 1.4	59.2 ± 1.5	49.5 ± 0.1	12.6 ± 0.8	59.7 ± 0.8	49.6 ± 0.1	12.4 ± 0.3
lwf	$\mathbf{62.2 \pm 1.8}$	49.1 ± 0.2	11.1 ± 1.2	55.8 ± 1.4	48.7 ± 0.1	12.5 ± 0.7	58.0 ± 1.2	49.2 ± 0.1	12.6 ± 0.6
er	$\mathbf{45.0 \pm 24.7}$	23.8 ± 19.1	3.0 ± 2.5	70.0 ± 7.4	48.9 ± 0.3	7.6 ± 1.5	77.6 ± 1.8	49.3 ± 0.1	6.9 ± 0.5
der	41.4 ± 13.1	19.6 ± 17.4	7.9 ± 1.7	76.2 ± 1.9	49.3 ± 0.1	7.3 ± 0.4	77.5 ± 3.6	49.5 ± 0.1	6.8 ± 0.9
der++	36.3 ± 21.7	23.6 ± 18.0	3.7 ± 3.5	$\mathbf{76.8 \pm 7.3}$	49.0 ± 0.6	5.9 ± 0.5	$\mathbf{82.1 \pm 7.1}$	49.4 ± 0.2	5.0 ± 1.6
cool	38.8 ± 26.3	24.1 ± 18.1	4.9 ± 2.0	67.9 ± 2.0	$\mathbf{77.67 \pm 1.9}$	$\mathbf{53.2 \pm 3.9}$	70.7 ± 2.1	$\mathbf{80.2 \pm 1.9}$	$\mathbf{57.1 \pm 3.9}$

Table 6. `MNAdd-Shortcut` Additional results.

With 0% concept supervision, all methods perform poorly, i.e., worse or comparably to NAÏVE and RESTART in terms of Class-IL. Variance is also quite large for EWC, ER, DER, DER++, and COOL. The sole exception is LWF, which obtains higher performance and small variance in label classification. On the other hand, the results in OOD accuracy are all below 13%, indicating that no strategy can successfully identify high-quality concepts that transfer across tasks.

The picture at 1% and 10% supervision is very similar: COOL outperforms all approaches in terms of concept quality and OOD accuracy by a large margin, while the other approaches pick up the reasoning shortcut, thus achieving high label accuracy only.

E.3.1. `CLE4EVR`

The complete results for `CLE4EVR`, reported in Table 7, show the same trend as those on `MNAdd-Shortcut`. For completeness, we report the evolution across tasks of concept confusion matrices for all methods in Figure 6. The impact of the reasoning shortcut on DER, and its lack of impact on COOL, are clearly visible.

	Supervision 0%			Supervision 1%			Supervision 10%		
	Class-IL($\mathbf{Y}$)	Class-IL($\mathbf{C}$)	OOD ($\mathbf{Y}$)	Class-IL($\mathbf{Y}$)	Class-IL($\mathbf{C}$)	OOD ($\mathbf{Y}$)	Class-IL($\mathbf{Y}$)	Class-IL($\mathbf{C}$)	OOD ($\mathbf{Y}$)
NAÏVE	44.9 ± 0.7	9.2 ± 1.2	25.0 ± 0.9	43.5 ± 2.6	19.9 ± 2.6	27.9 ± 11.4	39.9 ± 1.6	17.8 ± 0.1	16.9 ± 3.6
RESTART	39.1 ± 0.9	10.5 ± 1.4	11.1 ± 3.7	39.9 ± 1.0	17.8 ± 0.1	14.8 ± 4.1	39.2 ± 1.1	17.9 ± 0.1	13.7 ± 3.8
EWC	41.5 ± 7.1	8.3 ± 3.2	15.7 ± 8.4	38.5 ± 6.4	17.2 ± 1.4	23.5 ± 8.6	41.8 ± 1.8	17.8 ± 0.1	16.8 ± 4.3
LWF	47.1 ± 0.8	6.6 ± 2.2	27.2 ± 2.8	41.9 ± 1.8	19.1 ± 2.9	29.0 ± 15.2	39.5 ± 7.5	17.4 ± 3.9	44.8 ± 24.3
ER	85.3 ± 0.3	10.2 ± 5.5	26.3 ± 3.5	72.8 ± 4.9	20.5 ± 3.9	28.8 ± 9.9	66.8 ± 8.4	18.3 ± 1.3	23.3 ± 4.0
DER	77.7 ± 1.4	7.7 ± 2.8	$\mathbf{27.5 \pm 2.7}$	73.3 ± 3.0	19.1 ± 1.9	24.9 ± 3.4	75.1 ± 3.9	19.2 ± 2.0	26.1 ± 5.0
DER++	$\mathbf{85.5 \pm 0.2}$	11.1 ± 4.4	23.2 ± 1.8	83.0 ± 1.1	20.9 ± 3.9	32.4 ± 12.1	81.3 ± 2.3	20.1 ± 3.9	28.4 ± 9.3
COOL	70.7 ± 5.7	8.5 ± 4.5	25.6 ± 2.3	$\mathbf{83.2 \pm 0.5}$	$\mathbf{85.9 \pm 2.9}$	$\mathbf{91.1 \pm 1.5}$	$\mathbf{85.2 \pm 0.3}$	$\mathbf{87.9 \pm 0.1}$	$\mathbf{91.9 \pm 0.2}$

Table 7. CLE4EVR Full results

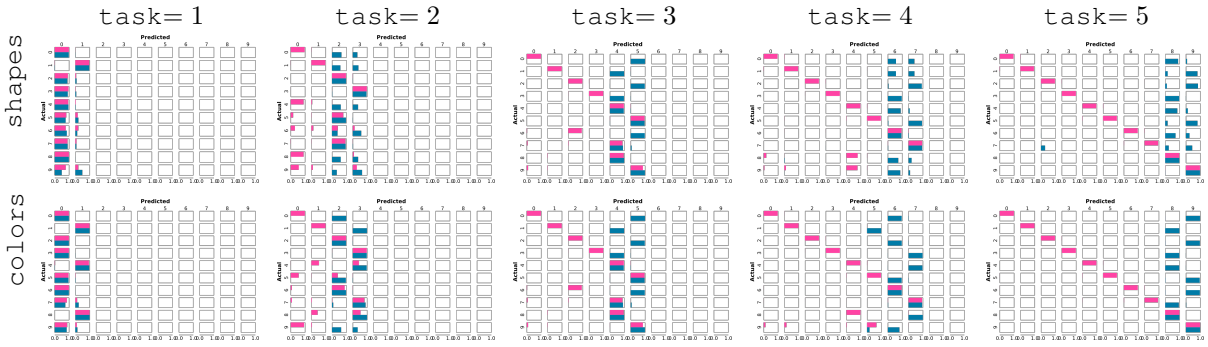

Figure 6. Dynamics of confusion matrices for shapes and colors encodings on CLE4EVR with 10% concept supervision. COOL in **red** preserves the correct concepts, while DER (in **blue**) suffers for shortcuts. Shape and color range in $\{0, \ldots, 9\}$.