Continuous Coordination As a Realistic Scenario for Lifelong Learning

Hadi Nekoei [*1] **Akilesh Badrinaaraayanan** [*1 2] **Aaron Courville** [1 2 3] **Sarath Chandar** [1 4 3]

Abstract

Current deep reinforcement learning (RL) algorithms are still highly task-specific and lack the ability to generalize to new environments. Lifelong learning (LLL), however, aims at solving multiple tasks sequentially by efficiently transferring and using knowledge between tasks. Despite a surge of interest in lifelong RL in recent years, the lack of a realistic testbed makes robust evaluation of LLL algorithms difficult. Multi-agent RL (MARL), on the other hand, can be seen as a natural scenario for lifelong RL due to its inherent non-stationarity, since the agents' policies change over time. In this work, we introduce a multi-agent lifelong learning testbed that supports both zero-shot and few-shot settings. Our setup is based on Hanabi — a partially-observable, fully cooperative multi-agent game that has been shown to be challenging for zero-shot coordination. Its large strategy space makes it a desirable environment for lifelong RL tasks. We evaluate several recent MARL methods, and benchmark state-of-the-art LLL algorithms in limited memory and computation regimes to shed light on their strengths and weaknesses. This continual learning paradigm also provides us with a pragmatic way of going beyond centralized training which is the most commonly used training protocol in MARL. We empirically show that the agents trained in our setup are able to coordinate well with unseen agents, without any additional assumptions made by previous works. The code and all pre-trained models are available at https://github.com/chandar-lab/Lifelong-Hanabi.

[*]Equal contribution [1]Mila [2]Université de Montréal [3]Canada CIFAR AI Chair [4]École Polytechnique de Montréal. Correspondence to: Hadi Nekoei <hadinekoei94@gmail.com>, Akilesh Badrinaaraayanan <akilesh041195@gmail.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

Deep reinforcement learning (RL) has shown an immense potential to achieve superhuman performance (Mnih et al., 2013; Silver et al., 2018) on some narrow and well-defined tasks. In contrast, humans can quickly and continually learn new tasks while maintaining the skills to solve previously learned tasks. The ability of an AI system to effectively update new information over time is known as lifelong learning (LLL) or continual learning, and one can postulate this as one of the fundamental ingredients of general AI. Balancing between learning from recent experiences while not forgetting the knowledge acquired from the past is a well-studied problem known as the stability-plasticity dilemma (Carpenter & Grossberg, 1987). Catastrophic forgetting is a phenomenon in which training the model with new information obstructs previously learned knowledge. This is a common failure case in training neural networks to adapt to new tasks or learning from non-stationary data streams (i.e. non-iid) (McCloskey & Cohen, 1989). Alleviating catastrophic-forgetting is crucial to enable real-world applications where input distributions can shift and where retraining on past data or from scratch is infeasible. While lifelong learning has been identified as an important and challenging problem decades ago (Thrun, 1998; Ring, 1998), it has recently seen a surge of interest (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018b; 2019; Kirkpatrick et al., 2017; Aljundi et al., 2018) with the success of deep learning.

Several standard benchmarks have been proposed to evaluate novel LLL approaches, mostly for supervised learning settings such as Permuted MNIST (Goodfellow et al., 2013), Split MNIST/CUB/CIFAR (Zenke et al., 2017; Chaudhry et al., 2018b). One fundamental issue with using datasets like MNIST as a source of data is the lack of resulting task complexity especially with the large capacity of modern neural networks. Another issue with most current LLL benchmarks is that the relation between tasks cannot be quantified easily. Consequently, most of the evaluation efforts have focused mainly on mitigating catastrophic forgetting, while an ideal LLL system should in addition measure forward and backward transfer. Some recent works have shown limitations of LLL benchmarks (Antoniou et al., 2020; Roady et al., 2020). For instance, it has been shown that after continual training, the performance of a model trained from scratch using only samples from the episodic memory at

test-time, is comparable to specifically designed LLL solutions for most of these benchmarks (Prabhu et al., 2020). There have been efforts to address this by proposing more challenging benchmarks like CORe50 (Lomonaco & Maltoni, 2017), CRIB (Stojanov et al., 2019), OpenLoris (Shi et al., 2020), Stream51 (Roady et al., 2020), and IIRC (Abdelsalam et al., 2020).

RL can be a natural fit for studying LLL as it provides an agent-environment interaction paradigm wherein the agent is exposed to non-stationary streams of data (Kaplanis et al., 2018; 2019). However, there is a dearth of well-established benchmarks to study progress in lifelong RL. Most of these benchmarks are hand-engineered customization to the standard RL environments (Bellemare et al., 2013; Brockman et al., 2016) adding synthetic non-stationarity to the environments (Henderson et al., 2017; Al-Shedivat et al., 2017) or ordering some completely unrelated environments in a sequence (Xu et al., 2020) to facilitate the evaluation of LLL performance (eg. a random sequence of Atari used in (Kirkpatrick et al., 2017)). Designing overly-tailored experiments for a specific lifelong RL problem can entail unwanted bias in the study (Khetarpal et al., 2020).

In this work, we propose a new lifelong RL setup based on Hanabi (Bard et al., 2020) called *Lifelong Hanabi*. Hanabi is a partially-observable, fully cooperative multi-agent game that consists of 2-5 players. In our setup, one agent (*learner*) is trained sequentially with a set of partners (tasks). The *learner* and its partners are sampled from a large pool of pre-trained agents ($\geqslant 100$). The pre-trained pool consists of agents trained with different MARL methods such as Independent Q-learning (IQL) (Tan, 1993), Value Decomposition Networks (VDN) (Sunehag et al., 2017), Simplified Action Decoder (SAD) (Hu & Foerster, 2019), Other-Play (OP) (Hu et al., 2020) with different architectures and seeds for each method that have shown good performance in Hanabi. Bard et al. (2020) show that agents trained even with the same MARL method but different seeds do not learn to cooperate in the zero-shot scenario, thereby suggesting that these agents converge to different strategies. This large strategy space of Hanabi makes it an ideal scenario for LLL. How far-apart the agents are in the strategy space can be measured through the *cross-play* (CP) matrix (Bard et al., 2020) that contains the gameplay scores obtained by pairing the agents with one another. CP scores can be used as a proxy for task-similarity to design tasks in *Lifelong Hanabi*.

Our contributions are as follows:

- We propose a new lifelong reinforcement learning benchmark that has the following desirable properties: (1) It is challenging for state-of-the-art (SOTA) lifelong learning algorithms, (2) It is straightforward to quantify the relation between tasks through the CP matrix, and (3) It is easily extendable to long sequences of diverse tasks without any synthetic modifications.
- We evaluate recent LLL algorithms on this benchmark in limited memory and computation regimes and highlight their strengths and limitations.
- We obtain comparable performance on zero-shot coordination in Hanabi even when coordinating with agents trained with MARL methods different from that of the *learner*, without any additional assumptions such as exploiting handcrafted symmetries (Hu et al., 2020) or having access to other agent's greedy action or policy (Hu & Foerster, 2019).

2. Related work

In this section, we will provide an overview of existing lifelong RL benchmarks. We will also review recent MARL algorithms since our benchmark is based on a challenging MARL problem.

2.1. Lifelong Reinforcement Learning Benchmarks

With regard to lifelong RL benchmarks, Henderson et al. (2017) proposed 50 new variations to OpenAI Gym environments through modifying some aspects of either the environments or agents like gravity, morphology of the agent's body, or goal positions. Al-Shedivat et al. (2017) introduced RoboSumo — a 3D environment based on MuJoCo physics simulator that allows pairs of agents to compete against each other. The robots differ in anatomy: the number of legs, their positions, and constraints on the thigh and knee joints. Lomonaco et al. (2020) designed CRLMaze based on VizDoom (Kempka et al., 2016), an object-picking LLL task that is composed of 4 scenarios (Light, Texture, Object, All) of incremental difficulty and a total of 12 maps. While these are interesting benchmarks, they still need synthetic modifications to either the environment or the agent in order to introduce non-stationarity. Recently, Coinrun (Cobbe et al., 2019) was proposed that is a procedurally generated environment having different training and testing sets to measure generalization in RL. Jelly Bean World (JBW) (Platanios et al., 2020) is a testbed introduced to develop agents with never-ending learning capabilities. It provides support to create non-stationary environments with wide range of tasks including multi-task and multi-modal settings.

2.2. Multi-agent RL

In recent years, there has been rapid progress on novel MARL algorithms that are based on centralized training with decentralized execution (Self-Play) as a training paradigm (Sunehag et al., 2017; Foerster et al., 2017; Hong et al., 2017; Foerster et al., 2019; Hu & Foerster, 2019). Sunehag et al. (2017) use a Value Decomposition Network (VDN) to learn to decompose the joint state-action value

into agent-wise Q-values that depend on local observations of each agent. Bayesian Action Decoder (BAD) (Foerster et al., 2019) and its simplified version (SAD) (Hu & Foerster, 2019) propose *public belief* MDP and use an approximate Bayesian update to tackle partially observable tasks. Hong et al. (2017) aim to learn models of opponents by learning policy features from raw observations of other agents. On the other hand, Omidshafiei et al. (2017) is one of the few efforts toward decentralized multi-agent learning which shows that optimistic learners can learn sample-efficient MARL policies. There have also been advances towards developing more complex MARL challenges. Stone et al. (2010) first introduced *ad-hoc* team-work challenge as a multi-agent task where autonomous agents need to collaborate with previously unknown teammates on tasks in which they are all individually capable of contributing as team members as well as characteristics of a good *ad-hoc* player. More recently, in the Hanabi challenge (Bard et al., 2020), the authors introduce *ad-hoc* scenario as a setting with the objective of making the RL agents adapt to play effectively with unknown partners or even humans. However, SP agents learn brittle policies that fail to cooperate in this scenario (Bard et al., 2020). Cooperative settings are more reflective of real-world scenarios such as autonomous driving and are crucial for human-AI collaboration (Crandall et al., 2018). The closest work to ours from the perspective of zero-shot coordination is Other-Play (Hu et al., 2020), which exploits the symmetry in the environment by training a self-play agent with shuffled observation space. This simple but elegant idea when combined with adding an auxiliary task, has been shown to be effective in training agents that are able to coordinate with other agents trained with the same MARL methods. However, other-play is still a self-play strategy and requires the knowledge of symmetry of the game upfront. In this work, we aim to go beyond these restricting assumptions and train a lifelong learner that is able to coordinate with unseen agents while not forgetting to play well with previous partners.

3. Multi-Agent RL and Lifelong Learning

MARL for LLL: Many machine learning algorithms make the assumption that the observations in the dataset are independent and identically distributed (i.i.d). However, in many real-world scenarios, this assumption is violated because the underlying data distribution is non-stationary. Lifelong learning tries to address this problem, where the non-stationarity of data is usually described as a sequence of distinct tasks. On the other hand, MARL is inherently non-stationary due to changing behavior of other agents present in the environment. Therefore, MARL is a realistic scenario for LLL. Another source of non-stationarity in MARL arises when the agent has to interact with different agents over its lifetime, even if the other agents are fixed. For example, in the game of Hanabi, we are interested in designing a single agent that can learn to coordinate well with a sequence of agents it will see over its lifetime. This is a lifelong learning problem.

LLL for MARL: Standard MARL methods typically focus on the centralized training with decentralized execution setting where agents have access to other agents' policies and observations during training (Zhang et al., 2019). Self-Play (SP) (Tesauro, 1995), the most common centralized training setting involves training a single agent against itself without any extra supervision. While this strategy works very well in competitive settings like playing the game of Go (Silver et al., 2016), in cooperative settings it can produce agents that establish highly specialized conventions that do not carry over to novel partners they have not been trained with (Bard et al., 2020). In particular, Bard et al. (2020) show that even though RL agents achieve a decent score after training in the SP setting, their performance drops off sharply in the zero-shot coordination scenario, with some agents scoring essentially zero. Therefore, SP agents fail to learn robust strategies that facilitate cooperation with *other* agents. Lifelong learning provides a natural framework to transfer knowledge from previous experience to future scenarios. Hence, in this paper, we consider lifelong learning as an alternative to self-play in MARL with the hope that lifelong learning algorithms can learn to coordinate well with unseen agents.

4. Lifelong Hanabi: A Benchmark for Lifelong Reinforcement Learning

In this section, we introduce our lifelong reinforcement learning benchmark based on Hanabi. Hanabi (Bard et al., 2020) is a partially-observable, fully cooperative multi-agent game that consists of 2-5 players. Each player can observe other players' hands except his/her own, making it a partially observable game. The objective of the game is to form ordered stacks of cards of respective colors (fireworks). The players can communicate with each other implicitly through actions or explicitly through *hints*, which are limited in number. Thus, Hanabi is a challenging game that requires the agent to possess the *theory of mind* (Premack & Woodruff, 1978; Rabinowitz et al., 2018) in order to cooperate effectively. The theory of mind is the ability of an agent to see the world through the lens of other agents. Our objective is to design a training paradigm that can learn zero-shot and few-shot coordination with unseen agents. To this end, we take inspiration from the recent Invariant Risk Minimization (IRM) (Arjovsky et al., 2019) that improves the Out-of-Distribution (OOD) generalization by training an algorithm on multiple environments. MARL provides such an environment naturally without any need to hand-engineer different features to come up with a set of diverse

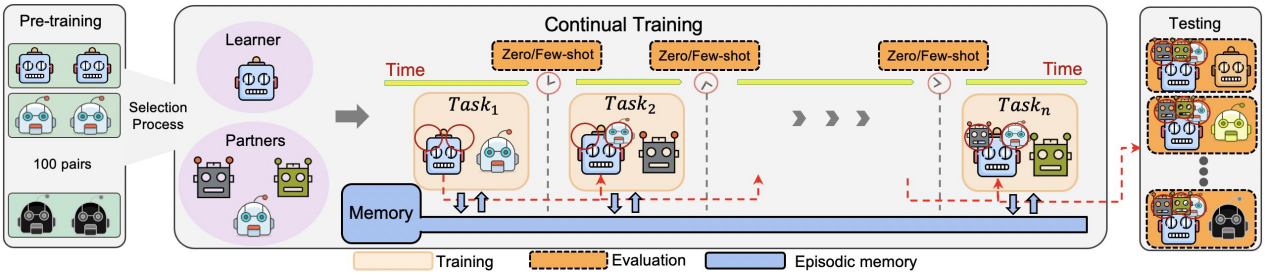


Figure 1. Our Lifelong Hanabi setup consists of three phases: **1- Pre-training (Optional):** In this phase, a pool of agents are trained through SP, **2- Continual training:** The *learner* is taken from the pool ($\sim p_{train}$) and trained sequentially with some *partners* ($\sim p_{train}$) and periodically evaluated against all the partners, **3- Testing:** The *learner* is evaluated with a set of random agents excluding its partners ($\sim p_{test}$) to measure generalization.

environments.

As shown in Figure 1, in our Lifelong RL setup, the *learner* ($\sim p_{train}$) is trained sequentially on a set of tasks $\mathbf{M} = \{M_t\}_{t=1}^{T}$ sampled from a distribution p_{train} over diverse strategies that perform well in Hanabi. The objective of the *learner* is to learn to coordinate well with its partners during continual training, with the ultimate goal of learning to coordinate well with unseen agents at the end of the training. During the testing phase, in order to measure generalization performance, the *learner* is evaluated with some random agents sampled from p_{test}. Although we consider a fully cooperative game in this work, our proposed lifelong RL setup can be easily extended to other multi-agent scenarios (e.g. fully competitive, mixed cooperative-competitive, etc.). Our proposed setup consists of three phases: (1) Pre-training, (2) Continual-training, and (3) Testing. The detailed description of each phase is as follows:

Pre-training: In this phase, agents are trained through SP to play the game of Hanabi. We consider several recent MARL methods for training the agents (IQL/VDN/OP/SAD, and their combinations) across different seeds and architectures leading to a pool of agents having diverse strategies.

Continual-training: An agent sampled from the pool is chosen as the *learner* and is trained sequentially with a set of agents (partners) for a fixed number of games per partner. The *learner* is also periodically evaluated with all its partners under both zero-shot and few-shot settings. In order to implement memory-based LLL algorithms (ER, A-GEM, etc.), we also include an episodic memory that is used to store some transitions from every task, which can be then be used for replaying in the future tasks.

Testing: The *learner* is evaluated with K random agents sampled from the pool excluding its partners in order to measure the generalization performance.

4.1. Evaluation methods

We consider two modes of evaluation in our setup : (a) zero-shot and (b) few-shot. In zero-shot setting, the *learner* is evaluated with another agent without providing it a chance to make updates to its own policy through its interaction with the other agent. On the other hand, in the few-shot setting, the *learner* plays a few games with the other agent, thereby adapting its policy through interaction, before being evaluated. We believe agents performing well under both these evaluation settings are crucial to developing AI systems that can adapt well to unknown partners not just in Hanabi but can also facilitate effective collaboration with humans. Few-shot evaluation setting also opens a door to explore recent advances in Meta-RL algorithms to enable fast adaptation.

4.2. Metrics

We measure the *learner*'s performance during continual training with some standard metrics from the LLL literature such as average score (A), forgetting (F), and forward transfer (FT) (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018a). We also define a metric for measuring OOD generalization in our setup called generalization improvement score (GIS), inspired by (Zhang et al., 2018). To calculate these metrics, we map Hanabi scores at the end of the game from [0, 25] to [0, 1] to have values more consistent with the literature.

Average score ($A \in [0, 1]$): Let $a_{i,j}$ be the score of the *learner* versus the j^{th} partner, after training it with the i^{th} partner in sequential training. The average score of the *learner* at task t (A_t) is defined as:

$$A_t = \frac{1}{t}\sum_{j=1}^{t} a_{t,j} \tag{1}$$

Forgetting($F \in [-1, 1]$): Let f_j^t represent the forgetting on task j after the *learner* is trained on task t and is computed

as:

$$f_j^t = \max_{l \in 1,\ldots,t-1} a_{l,j} - a_{t,j} \tag{2}$$

The average forgetting at task t is then defined as:

$$F_t = \frac{1}{t-1} \sum_{j=1}^{t-1} f_j^t \tag{3}$$

Average future score ($FT \in [0,1]$): Let $a_{i,j}$ be the score of the *learner* versus the j^{th} partner, after training it with the i^{th} partner in sequential training. The average future score (or forward transfer) of the *learner* at task t (FT_t) is defined as:

$$FT_t = \frac{1}{T-t} \sum_{j=t+1}^{T} a_{t,j} \tag{4}$$

Generalization Improvement Score ($GIS \in [0,1]$): Zhang et al. (2018) define in-task generalization as the difference in the performance of an RL algorithm between a set of training and testing trajectories all generated by the same simulator. In this case, the only source of variation is through random seeds. However, in our work, we are more interested in the out-of-task generalization that can be defined as follows. Let $a_{0,k}$ and $a_{N,k}$ be the score of the *learner* versus the k_{th} random agent sampled from the pool (different from its partners) before the start of LLL and at the end of continual training, respectively (T is number of tasks in LLL). The GIS is computed as follows:

$$GIS = \frac{1}{K} \sum_{k=1}^{K} (a_{T,k} - a_{0,k}) \tag{5}$$

where K is the total number of unseen agents.

5. Experiments

Figure 2 showcases how continual training can lead to improved scores and better zero-shot coordination. An IQL agent (pre-trained with self-play) is trained sequentially with 5 partners (ordering denoted by arrows in Figure 2), and then evaluated with both its partners seen during continual training as well as some unseen agents (SAD+AUX+OP, VDN+AUX). Sections (C) and (E) in Figure 2 show the performance of the IQL agent on unseen agents before and after continual training respectively, clearly indicating OOD generalization. Likewise, sections (A) and (D) show the performance before and after training respectively with its partners used in continual training.

As described in Section 4, we first pre-train a set of agents to play the game of Hanabi through SP. Our RL agents are based on the R2D2 architecture (Kapturowski et al., 2018) that are RNN-based DQN agents. The diversity in

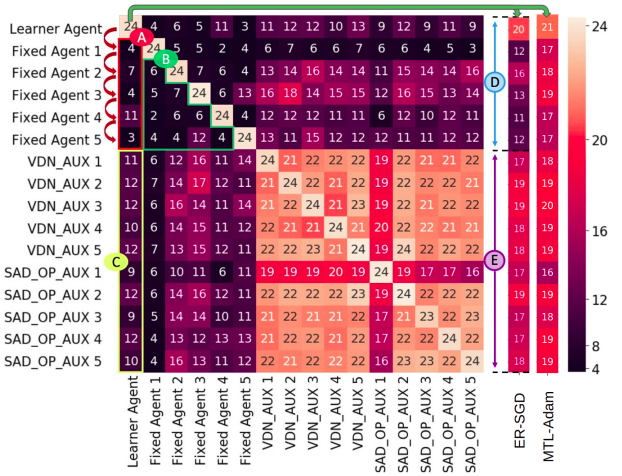


Figure 2. CP scores before (left) and after continual training (right) – $(i,j)_{th}$ element is the average score of agent i paired with j. [A-C] is before continual training – (A) Initial scores of the *learner* with its partners, (B) Cross-play scores amongst the partners, low scores indicate they are far apart in the strategy space, (C) Initial generalization scores with some unseen agents. The *learner* is then trained continually with its partners following the order indicated by the arrows. [D-E] is after continual training – (D) Scores with the original learner and its partners, (E) Generalization scores with the same unseen agents.

strategies learned by these SP agents are controlled by varying the seed, the MARL methods used for training (either IQL/VDN/OP/SAD and their combinations), type of recurrence (LSTM/GRU), number and dimension of recurrent layers, number and dimension of feed-forward layers before recurrence. The exact architecture details are described in the Appendix B. A pool of more than 100 SP agents is created this way. A subset of 100 agents with 10 agents from each of these 10 MARL methods are used to generate the *cross-play* (CP) matrix as shown in Appendix-A. The entries in this matrix (diagonal entries indicate self-play scores) are obtained through the gameplay of agents with each other for 5k games, and then averaging the scores.

We propose two levels of tasks based on these scores that have different difficulty levels: *easy* and *hard*. In both the settings, one of the agents is used as *learner* and the rest as its *partners* that are fixed during continual training i.e. they represent different tasks since these agents have different strategies. Both the *learner* and its *partners* are initialized with weights of the pre-trained agents as we found pre-training to be crucial to learn some basic knowledge of Hanabi. The names of these tasks are self-explanatory in the sense the *learner* in the *hard* version has to start from a much lower *cross-play* score and learn to achieve a good final score (out of 25). During continual training, the *learner* is trained sequentially with every *partner* for a fixed number of epochs and evaluated periodically with all its *partners*

under both zero-shot and few-shot settings. In the zero-shot setting, the *learner* is directly evaluated with all its *partners*, while in the few-shot setting, the *learner* is fine-tuned with its *partner* for a few gradient steps before evaluating against the same partner. In both these settings, scores are reported based on average over 5k gameplay.

Although the *hard* and *easy* settings consist of five tasks, our setup as such can be effortlessly extended to any number of tasks by selecting a different number of partners from the pre-trained pool and the pre-trained pool can itself be expanded by training more agents through SP. For instance, we report results on 10 tasks in section 5.3 as well as in Appendix C. Note that to choose the *partners*, we excluded all the agents using either SAD or AUX from the pre-trained pool as we wanted to compare the continually trained learner with them in terms of zero-shot coordination. We also wanted to select partners that have low CP scores so that the tasks are diverse.

R2D2 agents keep recent game transitions in a fixed-size prioritized replay buffer (Schaul et al., 2015). At the end of every task, the replay-buffer is sliced and stored in an episodic memory, which is then used for replay in different memory-based LLL algorithms that we consider in our benchmark. The *learner* can also start with random parameters (i.e. without pre-training), albeit, this setting is very hard for the game of Hanabi.

We aim to answer the following questions through our experiments : (1) How well standard LLL algorithms perform in our setup (section 5.1), (2) How well do these LLL algorithms fare under constrained memory and compute settings (section 5.2), (3) How lifelong RL methods perform in our setup (section 5.3), (4) How well the agents trained in our setup do in zero-shot and few-shot coordination scenarios in Hanabi (section 5.4), when compared to other recent methods such as OP (Hu et al., 2020).

5.1. Lifelong learning benchmarking:

We implement some standard LLL algorithms that are both replay-based and regularization-based.

Naive: This is the simplest algorithm in which the *learner* is trained sequentially on subsequent tasks, starting with the learned parameters at the end of the previous task, without any episodic memory or regularization.

Experience Replay (ER): We follow the procedure described in (Chaudhry et al., 2019) for implementing ER. We sample a mini-batch B_k from the current task (in which the *learner* plays with partner k), and a mini-batch B_m that consists of an equal number of samples from all previous tasks collectively. These mini-batches are stacked and a single gradient step is used to update the *learner*. Our implementation closely resembles the ring buffer strategy described in (Chaudhry et al., 2019), as there's equal representation from all previous tasks when sampling B_m, although the samples within each task itself are prioritized.

Averaged Gradient Episodic Memory (A-GEM): Mini-batches B_k and B_m are sampled as described in (Chaudhry et al., 2018b) and similar to ER. The gradients corresponding to these mini-batches are first computed denoted by g and g_{ref} respectively. If $g^T g_{ref} \geqslant 0$, then the gradient of the current task g is directly used to update the *learner*'s parameters, whereas if $g^T g_{ref} < 0$, g is first projected such that $g^T g_{ref} = 0$ before updating the *learner*. This projection ensures that the average loss over the previous tasks does not increase.

Elastic Weight Consolidation (EWC): EWC is a regularization-based technique proposed to alleviate catastrophic forgetting by selectively reducing the plasticity of weights drawing inspiration from Bayesian methods (Kirkpatrick et al., 2017). EWC uses Fisher information matrix as a surrogate for the importance of learned weights and uses that for gradient updates. Offline EWC uses one Fisher matrix per task, therefore the number of regularization terms increases linearly with the number of tasks whereas Online EWC (Schwarz et al., 2018) uses only one Fisher matrix that is computed based on all the previous tasks. We consider both these variants in our benchmark.

Stable naive/ER/A-GEM/EWC: Mirzadeh et al. (2020) show that catastrophic forgetting can be mitigated through careful design of training regimes such as learning rate decay, batch size, dropout, and optimizer that can widen the tasks' local minima. The resulting model with these optimal choices is referred to as "stable". In particular, we consider if using larger batches, exponential learning rate decay, dropout (either in feed-forward or recurrent layers in R2D2), and SGD optimizer help improve continual training, thereby leading to better final performance as well as generalization to unseen agents.

Multi-Task Learning (MTL): In this setting, there's a common replay buffer that contains the experiences of the *learner* interacting with all its partners. Mini-batches sampled from this common replay buffer are used for training the *learner*. This serves as an upper-bound on the achievable performance in our benchmark.

We can observe from Figure 3 and Table 1 that online EWC with Adam has the best average score in both the zero-shot and few-shot setting among the LLL algorithms, while the forgetting is least for ER with SGD. Table 1 also shows the effect of the optimizer on different LLL algorithms. LLL algorithms with SGD tend to have comparatively less forgetting and better zero-shot performance on average. We can infer from Figure 4 that using Adam helps in fast adaptation to current task albeit at the expense of greater forgetting.

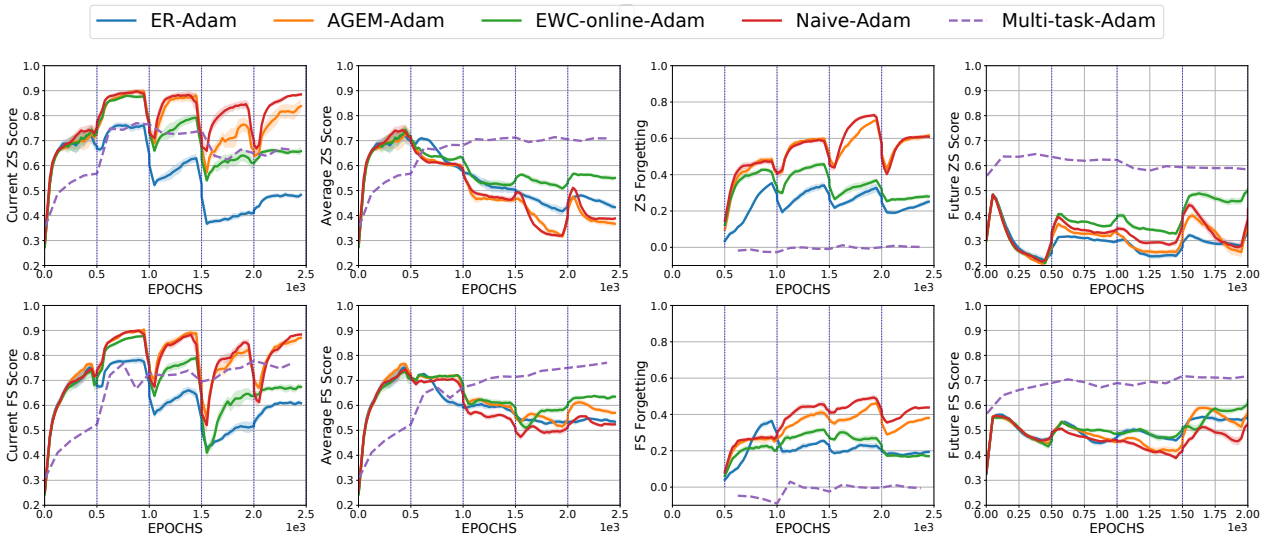


Figure 3. Zero-shot (top row) and Few-shot (bottom row) performance of different LLL algorithms with Adam optimizer on *hard* task. From left to right: current score (↑), average score (↑), forgetting (↓), and average future score (↑) respectively. (↑ = higher better, ↓ = lower better).

Table 1. Benchmarking LLL methods on Hanabi. Average accuracy and forgetting of LLL algorithms on *hard* task averaged over 5 runs with 5000 games. (↑ = higher better, ↓ = lower better)

Method	Zero-shot		Few-shot	
	A_T ↑	F_T ↓	A_T ↑	F_T ↓
Naive Adam	0.39	0.60	0.52	0.44
EWC off. Adam	0.45	0.45	0.60	0.27
EWC on. Adam	**0.55**	0.28	**0.63**	0.17
ER Adam	0.44	0.26	0.53	0.20
AGEM Adam	0.38	0.62	0.57	0.38
Naive SGD	0.52	0.15	0.52	0.14
EWC off. SGD	0.49	0.12	0.50	0.11
EWC on. SGD	0.47	0.12	0.48	0.11
ER SGD	0.50	**0.05**	0.50	**0.04**
AGEM SGD	0.50	0.13	0.51	0.12
Multi-Task Adam	0.70	0	0.77	0
Multi-Task SGD	0.50	0	0.51	0

This effect can also be seen in higher average few shot scores (Table 1). As one might expect, MTL with Adam achieves the highest average score in both zero-shot and few-shot.

From the last column of Figure 3, we can observe that when the *learner* start playing with a new partner, there is an increase in average future score suggesting it has learned some useful skills from previous tasks that is transferable to the other partners. However, with more training, the *learner* possibly overfits to its current partner, leading to a drop in average future scores.

To explore the effect of different training regimes on continual training, we study the effect of larger batches (128 vs 32), learning rate decay with high initial rates (0.2/0.02), and dropout. Our experiments suggest that *Lifelong Hanabi* does not benefit greatly from the use of large batches as the gain in scores is negligible. This observation aligns with the "stable" networks (Mirzadeh et al., 2020) that suggests using small batches. In the case of EWC, we find that there is a stark difference in performance with different λ values (the weight assigned to the Fischer term). Our experiments indicate larger λ is beneficial. Appendix- C contains the training curves of LLL algorithms for other settings — *hard* task with SGD as well as *easy* task with both Adam and SGD.

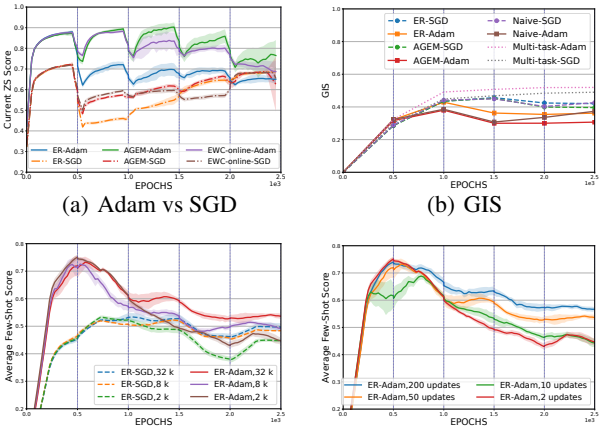
5.2. Lifelong learning under constrained memory and compute:

5.2.1. Episodic memory size

In order to understand the effect of episodic memory on the performance of memory-based LLL algorithms, we vary the episodic memory size ($\{2k, 8k, 32k\} \times$ number of tasks) in the case of ER with both SGD and Adam as shown in Figure 4. In both these cases, a larger episodic memory size results in a better final performance.

5.2.2. Gradient updates for few-shot evaluation

To better understand the ability of the *learner* trained with different LLL algorithms to adapt quickly to all its partners, we vary the number of gradient steps used to update the *learner* in the few-shot evaluation scenario. As it can be seen from Figure 4, there is a considerable difference between 10



(a) Adam vs SGD (b) GIS

(c) Episodic memory size (d) Few-shot gradient steps

Figure 4. More experiments: Generalization score with Inter-CP agents at the end of every task during continual training. LLL algorithms using SGD as an optimizer have better generalization performance compared to Adam. ER SGD has the highest GIS.

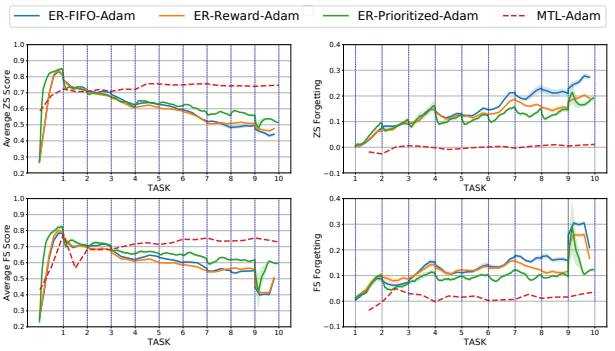


Figure 5. Zero-shot (top row) and Few-shot (bottom row) performance of ER methods with different types of episodic memory designed for lifelong RL (Isele & Cosgun, 2018) with Adam optimizer on 10 tasks.

and 50 gradient updates on the final performance, whereas the benefit reaped beyond 50 updates is minimal.

5.3. Lifelong RL methods:

(Isele & Cosgun, 2018) propose some strategies for storing experiences in the replay-buffer that has been shown to reduce catastrophic forgetting in RL. All our methods use prioritized replay buffer that resembles the *surprise* strategy (Isele & Cosgun, 2018). In addition, we also compare this with *FIFO* and *Reward* strategies. For *FIFO*, we set the prioritization exponent α to 0 (Schaul et al., 2015), which is equivalent to uniformly sampling. In case of *Reward*, we do prioritized sampling that favors experiences based on the absolute value of reward instead of TD-error as done in our default case. As can be seen in Figure 5, ER with prioritized sampling performs best compared to *Reward* and *FIFO* strategies in terms of both average score and average forgetting. Implementing other sampling strategies such as *Global Distribution Matching* and *Coverage Maximization* are left for future work.

5.4. Zero-shot coordination:

We compare our best-performing LLL algorithms with recent MARL methods that have shown good performance on Hanabi (Table 2). In addition to reporting self-play evaluation scores, we evaluate each training method with two sets of unseen partners under zero-shot coordination scenario: (1) *Intra-CP* - a set consisting of agents that are trained with the same MARL method as the training method. For example, the SAD+OP agent is evaluated with other SAD+OP agents only, but with different architectures and seeds. Similarly, in order to evaluate agents trained in our setup, we evaluate them with other agents that are trained with the same MARL method as the *learner*, (2) *Inter-CP* - a set containing 20 agents across all the MARL methods we consider.

As we can observe from Table 2, although recent MARL methods can achieve good scores in SP and *Intra-CP* evaluations, they fail to achieve high scores in *Inter-CP* highlighting their inability to coordinate effectively with other MARL methods in the zero-shot scenario. We can observe that agents trained in our setup have significant improvement in both *Inter-CP* and *Intra-CP* compared to the agent at the start of continual learning, however, their SP scores are lower than at the start. The difference in scores due to continual training is indicated in brackets. It is also worth mentioning that IQL+AUX+ER achieve a better *Inter-CP* score than even other MARL methods, although this comes at a cost of slight reduction to *Intra-CP* score.

All the training methods in Table 2 have some limitations that we highlight now. During training, SAD allows agents to have access to the greedy action of their team mates in addition to the actual exploratory action chosen (GA). AUX refers to having an auxiliary task that predicts the *learner*'s own hand and hence requires ground truth labels for this (L). OP requires symmetries of the game to be known beforehand (SYM). IQL+ER and IQL+AUX+ER require pre-trained agents in sequence for LLL (P), while IQL + Multi-Task requires access to all pre-trained agents simultaneously (UP). Figure 4 shows the progression in generalization performance (*Inter-CP*) after every task during continual training for several LLL algorithms. MTL (with Adam) and ER (with SGD) have the best scores with the *Inter-CP* agents at the end of continual training. However, MTL needs to interact with all the partners' at the same time which is not always a realistic assumption.

Table 2. Comparison with other MARL algorithms on self-play (SP), cross-play evaluation scores within method (*Intra-CP*), and across different methods (*Inter-CP*). C: centralized training, GA: agents share their greedy action along with their standard action, L: true labels of cards needed, SYM: symmetries of the game needed upfront, P: require access to some pre-trained agents in sequence, UP: Having access to all the fixed pre-trained agents at the same time. (↑ / ↓ = Difference in score after continual training, red: pre-trained with MARL method, blue: trained continually with LLL method)

Training Method	SP	Intra-CP	Inter-CP	Limitations
SAD	23.85 ± 0.03	7.70 ± 0.69	14.60 ± 0.24	C + GA
SAD + AUX	23.57 ± 0.03	20.97 ± 0.80	18.51 ± 0.23	C + GA + L
SAD + OP	24.14 ± 0.03	10.10 ± 0.87	16.09 ± 0.25	C + Sym + GA
SAD + AUX + OP	23.40 ± 0.04	21.23 ± 0.25	17.77 ± 0.23	C + Sym + L + GA
IQL + ER	20.91 ± 0.05 (↓ 2.98)	15.73 ± 0.39 (↑ **7.06**)	16.32 ± 0.21 (↑ **8.09**)	P
IQL + AUX + ER	22.34 ± 0.06 (↓ 1.46)	20.90 ± 0.06 (↓ 0.15)	$\mathbf{19.17 \pm 0.22}$ (↑ **1.33**)	L + P
IQL + Multi-task	20.93 ± 0.09 (↓ 2.96)	16.05 ± 0.30 (↑ **7.38**)	$\mathbf{17.88 \pm 0.17}$ (↑ **9.65**)	UP

6. Conclusions and future work

In this work, we proposed *Lifelong Hanabi* as a new challenging benchmark for lifelong RL. The non-stationarity in our benchmark was introduced through agents having different strategies instead of synthetic modifications to the environment or agent, while cross-play score served as an easy metric to quantify the similarity between tasks. We analyzed the performance of some well-known LLL algorithms on this benchmark. We also showed that an IQL agent continually trained in our setup can zero-shot coordinate effectively with unseen agents. We hope that the lifelong RL community adopt this as a standard benchmark for evaluating algorithmic advances due to its ease of use.

Lifelong Hanabi aims to facilitate development of novel algorithms for lifelong learning specific to RL (i.e. lifelong RL). This framework also serves as a step towards thinking beyond centralized training in MARL. Some interesting future directions are to understand the kind of policies learned by the agents trained in our setup through policy visualization to see what kind of conventions (if any) emerges. It is also valuable to evaluate our trained agents with humans, as developing artificial agents capable of coordinating effectively with humans is an important long-term goal of modern AI. Moreover, exploiting recent advances in Meta-RL such as (Zintgraf et al., 2019) for faster adaptation in the *few-shot* evaluation setting, instead of naively fine-tuning can lead to agents that adapt well in the *ad-hoc* scenario. We believe studying the effect of the order of *partners* that the *learner* encounters and its effect on final performance is an interesting next step. Currently, the non-stationarity across different strategies is what we only exploit to design LLL tasks. This already resulted in an interesting trade-off for the learner between adapting to new partners and not forgetting to coordinate well with previous partners. Our preliminary experiments suggest that extending our framework to learning partners is extremely difficult for current methods, however, this could be an exciting future research direction.

Acknowledgment We would like to thank Gabriele Prato, Mahta Ramezanian, Khimya Khetarpal, Matthew Riemer, Ankit Vani, Moksh Jain, and Salem Lahlou for reviewing the paper and their valuable feedback. We are also grateful to Hengyuan Hu for patiently answering queries regarding the Hanabi SAD code repository, Darshan Patil and Rodrigo Chavez Zavaleta for reviewing our code and Olexa Bilaniuk for helping us with engineering issues internally at Mila. We would like to acknowledge Compute Canada and Calcul Quebec for providing computing resources used in this work. SC is supported by a Canada CIFAR AI Chair and an NSERC Discovery Grant.

References

Abdelsalam, M., Faramarzi, M., Sodhani, S., and Chandar, S. Iirc: Incremental implicitly-refined classification. *arXiv preprint arXiv:2012.12477*, 2020.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

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)*, pp. 139–154, 2018.

Antoniou, A., Patacchiola, M., Ochal, M., and Storkey, A. Defining benchmarks for continual few-shot learning. *arXiv preprint arXiv:2004.11967*, 2020.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.