A Survey of Temporal Credit Assignment in Deep Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

The Credit Assignment Problem (CAP) refers to the longstanding challenge of Reinforcement Learning (RL) agents to associate actions with their long-term consequences. Solving the CAP is a crucial step towards the successful deployment of RL in the real world since most decision problems provide feedback that is noisy, delayed, and with little or no information about the causes. These conditions make it hard to distinguish serendipitous outcomes from those caused by informed decision-making. However, the mathematical nature of credit and the CAP remains poorly understood and defined. In this survey, we review the state of the art of Temporal Credit Assignment (CA) in deep RL. We propose a unifying formalism for credit that enables equitable comparisons of state of the art algorithms and improves our understanding of the trade-offs between the various methods. We cast the CAP as the problem of learning the influence of an action over an outcome from a finite amount of experience. We discuss the challenges posed by *delayed effects*, *transpositions*, and a *lack of action influence*, and analyse how existing methods aim to address them. Finally, we survey the protocols to evaluate a credit assignment method, and suggest ways to diagnoses the sources of struggle for different credit assignment methods. Overall, this survey provides an overview of the field for new-entry practitioners and researchers, it offers a coherent perspective for scholars looking to expedite the starting stages of a new study on the CAP, and it suggests potential directions for future research.

1 Introduction

RL is poised to impact many real world problems that require sequential decision making, such as strategy (Silver et al., 2016; 2018; Schrittwieser et al., 2020; Anthony et al., 2020; Vinyals et al., 2019; Perolat et al., 2022) and arcade video games (Mnih et al., 2013; 2015; Badia et al., 2020; Wurman et al., 2022), climate control (Wang & Hong, 2020), energy management (Gao, 2014), car driving (Filos et al., 2020) and stratospheric balloon navigation (Bellemare et al., 2020), designing circuits (Mirhoseini et al., 2020), cybersecurity (Nguyen & Reddi, 2021), robotics (Kormushev et al., 2013), or physics (Degrave et al., 2022). One fundamental mechanism allowing RL agents to succeed in these scenarios is their ability to evaluate the *influence* of their actions over outcomes – e.g., a win, a loss, a particular event, a payoff. Often, these outcomes are consequences of isolated decisions taken in a very remote past: actions can have long-term effects. The problem of learning to associate actions with distant, future outcomes is known as the temporal Credit Assignment Problem (CAP): *to distribute the credit of success among the multitude of decisions involved* (Minsky, 1961). Overall, the *influence* that an action has on an outcome represents *knowledge* in the form of *associations* between actions and outcomes (Sutton et al., 2011; Zhang et al., 2020). These associations constitute the scaffolding that agencies can use to deduce, reason, improve and act to address decision-making problems and ultimately improve on their data efficiency.

Solving the CAP is paramount since most decision problems have two important characteristics: they take a *long time to complete*, and they seldom provide immediate feedback, but often *with delay* and little insight as to which actions caused it. These conditions produce environments where the feedback signal is weak, noisy, or deceiving, and the ability to separate serendipitous outcomes from those caused by informed decision-making becomes a hard challenge. Furthermore, as these environments grow in complexity with the aim to scale to real-world tasks (Rahmandad et al., 2009; Luoma et al., 2017), the actions taken by an agent affect an increasingly vanishing part of the outcome. In these conditions, it becomes challenging to learn value functions that accurately represent the *influence* of an action, and to be able to distinguish and order the relative long-term values of different actions. In fact, canonical Deep Reinforcement Learning (Deep RL) solutions to *control* are often brittle to the hyperparameter choice (Henderson et al., 2018), inelastic to generalise zero-shot to different tasks (Kirk et al., 2023), prone to overfitting (Behzadan & Hsu, 2019; Wang et al., 2022), and sample-inefficient (Ye et al., 2021; Kapturowski et al., 2023). Overall, building a solid foundation of knowledge that can unlock solutions to complex problems beyond those already solved calls for better CA techniques (Mesnard et al., 2021).

In the current state of RL, *action values* are a key proxy for *action influence*. Values *actualise* a return by synthesising statistics of the *future* into properties of the *present*. Recently, the advent of Deep RL (Arulkumaran et al., 2017) granted access to new avenues to express credit through values, either by using memory (Goyal et al., 2019; Hung et al., 2019), associative memory (Hung et al., 2019; Ferret et al., 2021a; Raposo et al., 2021), counterfactuals (Mesnard et al., 2021), planning (Edwards et al., 2018; Goyal et al., 2019; van Hasselt et al., 2021) or by meta-learning (Xu et al., 2018; Houthooft et al., 2018; Oh et al., 2020; Xu et al., 2020; Zahavy et al., 2020). The research on CAP is now fervent, and with a rapidly growing corpus of works.

Motivation. Despite its central role, there is little discussion on the precise mathematical nature of credit. While these proxies are sufficient to unlock solutions to complex tasks, it remains unclear where to draw the line between a generic measure of action influence and *credit*. Existing works focus on partial aspects or sub-problems (Hung et al., 2019; Arjona-Medina et al., 2019; Arumugam et al., 2021) and not all works refer to the CAP explicitly in their text (Andrychowicz et al., 2017; Nota et al., 2021; Goyal et al., 2019), despite their findings providing relevant contributions to address the problem. The resulting literature is fragmented and lacks a space to connect recent works and put their efforts in perspective for the future. The field still holds open questions:

Q1. *What* is the *credit* of an action? How is it different from an *action value*? And what is the CAP? What in words, and what in mathematics?

Q2. *How* do agents learn to *assign* credit? What are the main methods in the literature and how can they be organised?

Q3. How can we *evaluate* whether a method is improving on a challenge? How can we monitor advancements?

Goals. Here, we propose potential answers to these questions and set out to realign the fundamental issue raised by Minsky (1961) to the Deep RL framework. Our main goal is to provide an overview of the field to new-entry practitioners and researchers, and, for scholars looking to develop the field further, to put the heterogeneous set of works into a comprehensive, coherent perspective. Lastly, we aim to reconnect works whose findings are relevant for CAP, but that do not refer to it directly. To the best of our knowledge, the work by Ferret (2022, Chapter 4) is the only effort in this direction, and the literature offers no explicit surveys on the temporal CA problem in Deep RL.

Scope. The survey focuses on temporal CA in single-agent Deep RL, and the problems of *(i)* quantifying the influence of an action mathematically and formalising a mathematical objective for the CA problem *(ii)* defining its challenges, and categorising the existing methods to learn the quantities above, *(iii)* defining a suitable evaluation protocol to monitor the advancement of the field. We do not discuss *structural* CA in Deep Neural Networks (DNNs), that is, the problem of assigning credit or blame to individual parameters of a DNN (Schmidhuber, 2015; Balduzzi et al., 2015). We also do not discuss CA in multi-agent RL, that is, to ascertain which agents are responsible for creating good reinforcement signals (Chang et al., 2003; Foerster et al., 2018). When credit (assignment) is used without any preceding adjective, we always refer to *temporal* credit (assignment). In particular, with the adjective *temporal* we refer to fact that *"each ultimate success is associated with a vast number of internal decisions"*(Minsky, 1961) and that these decisions, together with states and rewards, are arranged to form a *temporal* sequence.

The survey focuses on Deep RL. In surveying existing formalisms and methods, we only look at the Deep RL literature, and when proposing new ones, we tailor them to Deep RL theories and applications. We exclude from the review methods specifically designed to solve decision problems with linear or tabular RL, as they do not bode well for scaling to complex problems.

Outline. We address **Q1.**, **Q2.** and **Q3.** in the three major sections of the manuscript. Respectively:

- **Section 4** addresses **Q1.**, proposing a definition of credit and the CAP and providing a survey of action influence measures.
- **Section 5** and **Section 6** address **Q2.**, respectively discussing the key challenges to solving the CAP and the existing methods to assign credit.
- **Section 7** answers **Q3.**, reviewing the problem setup, the metrics and the evaluation protocols to monitor advancements in the field.

For each question, we contribute by: *(a)* systematising *existing works* into a simpler, coherent space; *(b)* discussing it, and *(c)* synthesising our perspective into a unifying formalism. Table 1 outlines the suggested reading flow according to the type of reader.

Reader type	Suggested Flow
Specialised CA scholar	$1 \to 2 \to 4 \to 5 \to 6 \to 7$
RL researcher	$1 \to 4 \to 5 \to 6 \to 7$
Deep Learning researcher	$1 \to 3 \to 4 \to 5 \to 6 \to 7$
Practitioner (applied researcher)	$6 \to 4.4 \to 3$
Proposing a new CA method	$7 \to 6 \to 2 \to 4$

Table 1: Suggested flow of reading by type of reader to support the outline in Section 1. Numbers represent section numbers.

2 Related Work

Three existing works stand out for proposing a better understanding of the CAP explicitly. Ferret (2022, Chapter 4) designs a conceptual framework to unify and study credit assignment methods. The chapter proposes a general formalism for a range of credit assignment functions and discusses their characteristics and general desiderata. Unlike Ferret (2022, Chapter 4), we survey potential formalisms for a mathematical definition of credit (Section 4); in view of the new formalism, we propose an alternative view of the methods to assign credit (Section 6), and an evaluation protocol to measure future advancements in the field. Arumugam et al. (2021) analyses the CAP from an information theoretic perspective. The work focuses on the notion of *information sparsity* to clarify the role of credit in solving sparse reward problems in RL. Despite the work questioning what credit is mathematically, it does not survey existing material, and it does not provide a framework that can unify existing approaches to represent credit under a single formalism. Harutyunyan et al. (2019) propose a principled method to measure the credit of an action. However, the study does not aim to survey existing methods to *measure* credit, the methods to *assign* credit, and the methods to evaluate a credit assignment method, and does not aim to organise them into a cohesive synthesis.

The literature offers also surveys on related topics. We discuss them in Appendix A to preserve the fluidity of the manuscript.

As a result, none of these works position CAP in a single space that enables thorough discussion, assessment and critique. Instead, we propose a formalism that unifies the existing *quantities* that represent the influence of an action (Section 4). Based on this, we can analyse the advantages and limitations of existing measures of action influence. The resulting framework provides a way to gather the variety of existing *methods* that learn these quantities from experience (Section 6), and to monitor the advancements in solving the CAP.

3 Notation and Background

Here we introduce the notation and background that we will follow in the rest of the paper.

Notations. We use calligraphic characters to denote sets and the corresponding lowercases to denote their elements, for example, $x \in \mathcal{X}$. For a measurable space $(\mathcal{X}, \Sigma)$, we denote the set of probability measures over $\mathcal{X}$ with $\Delta(\mathcal{X})$. We use an uppercase letter X to indicate a random variable, and the notation $\mathbb{P}_X$ to denote its distribution over the sample set $\mathcal{X}$, for example, $\mathbb{P}_X : \mathcal{X} \to \Delta(\mathcal{X})$. When we mention a *random event* X (for example, a *random action*) we refer to a random draw of a specific value $x \in \mathcal{X}$ from its distribution $\mathbb{P}_X$ and we write, $X \sim \mathbb{P}_X$. When a distribution is clear from the context, we omit it from the subscript and write $\mathbb{P}(X)$ instead of $\mathbb{P}_X(X)$. We use $\mathbb{1}_{\mathcal{Y}}(x)$ for the indicator function that maps an element $x \in \mathcal{X}$ to 1 if $x \in \mathcal{Y} \subset \mathcal{X}$ and 0 otherwise. We use $\mathbb{R}$ to denote the set of real numbers and $\mathbb{B} = \{0, 1\}$ to denote the Boolean domain. We use $\ell_\infty(x) = \|x\|_\infty = \sup_i |x_i|$ to denote the ℓ-infinity norm of a vector x of components x_i. We write the Kullback-Leibler divergence between two discrete probability distributions $\mathbb{P}_P(X)$ and $\mathbb{P}_Q(X)$ with sample space $\mathcal{X}$ as: $D_{KL}(\mathbb{P}_P(X)||\mathbb{P}_Q(X)) = \sum_{x \in \mathcal{X}} [\mathbb{P}_P(x) \log(\mathbb{P}_P(x)/\mathbb{P}_Q(x))]$.

Reinforcement Learning. We consider the problem of learning by interacting with an environment. A program (the *agent*) interacts with an *environment* by making decisions (*actions*). The action is the agent's interface with the environment. Before each action, the agent may *observe* part of the environment and take suitable actions. The action changes the state of the environment. After each action, the agent may perceive a feedback signal (the *reward*). The goal of the agent is to learn a rule of behaviour (the *policy*) that maximises the expected sum of rewards.

Markov Decision Processes (MDPs). MDPs formalise decision-making problems. This survey focuses on the most common MDP settings for Deep RL. Formally, a discounted MDP (Howard, 1960; Puterman, 2014) is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, \mu, \gamma)$. $\mathcal{S}$ is a finite set of states (the *state space*) and $\mathcal{A}$ is a finite set of actions (*the action space*). $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is a deterministic, bounded reward function that maps a state-action pair to a scalar reward $r \in [r_{min}, r_{max}] = \mathcal{R}$. $\gamma \in [0, 1]$ is a discount factor and $\mu : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a transition kernel, which maps a state-action pair to probabilities over states. We refer to an arbitrary state $s \in \mathcal{S}$ with s, an action $a \in \mathcal{A}$ with a and a reward $r \in [r_{min}, r_{max}]$ with r. Given a state-action tuple

(s, a), the probability of the next random state S_{t+1} being s' depends on a *state-transition* distribution: $\mathbb{P}_\mu(S_{t+1} = s'|S_t = s, A_t = a) = \mu(s'|s, a), \forall s, s' \in \mathcal{S}$. We refer to S_t as the *random state* at time t. The probability of the action a depends on the agent's policy, which is a stationary mapping $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, from a state to a probability distribution over actions.

These settings give rise to a discrete-time, stateless (Markovian), Random Process (RP) with the additional notions of *actions* to represent decisions and *rewards* for a feedback signal. Given an initial state distribution $\mathbb{P}_{\mu_0}(S_0)$, the process begins with a random state $s_0 \sim \mathbb{P}_{\mu_0}$. Starting from s_0, at each time t the agent interacts with the environment by choosing an action $A_t \sim \mathbb{P}_\pi(\cdot|s_t)$, observing the reward $r_t \sim R_t(S_t, A_t)$ and the next state $s_{t+1} \sim \mathbb{P}_\mu$. If a state s_t is also an *absorbing* state ($s \in \overline{\mathcal{S}} \subset \mathcal{S}$), the MDP transitions to the same state s_t with probability 1 and reward 0, and we say that the episode terminates. We refer to the union of each temporal *transition* (s_t, a_t, r_t, s_{t+1}) as a *trajectory* or *episode* $d = \{s_t, a_t, r_t, : 0 \leq t \leq T\}$, where T is the *horizon* of the episode.

We mostly consider episodic settings where the probability of ending in an absorbing state in finite time is 1, resulting in the random horizon T. We consider discrete action spaces $\mathcal{A} = \{a_i : 1 \leq i \leq n\}$ only.

A trajectory is also a random variable in the space of all trajectories $\mathcal{D} = (\mathcal{S} \times \mathcal{A} \times \mathcal{R})^T$, and its distribution is the joint of all of its components $\mathbb{P}_D(D) = \mathbb{P}_{A,S,R}(s_0, a_1, r_1, \ldots, s_T)$. Given an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, \mu, \gamma)$ and fixing a policy π produces a Markov Process (MP) $\mathcal{M}^\pi$ and induces a distribution over trajectory $\mathbb{P}_{\mu,\pi}(D)$. Therefore, we refer to a random trajectory that starts at k and ends at T also as a sequence of random decisions $D_k = \{X_k, \ldots, X_t, \ldots, X_{T-1}, S_T\}$.

We refer to the *return* random variable Z_t, as the sum of discounted rewards from time t to the end of the episode, $Z_t = \sum_{t=k}^{T} \gamma^{k-t} R(S_k, A_k)$. The *control* objective of an RL problem is to find a policy π^* that maximises the expected return,

$$\pi^* \in \operatorname*{argmax}_\pi \mathbb{E}_{\mu,\pi} \left[\sum_{t=0}^{T} \gamma^t R(S_t, A_t) \right] = \mathbb{E}\left[Z_0\right]. \tag{1}$$

Partially-Observable MDPs (POMDPs). POMDPs are MDPs in which agents do not get to observe a true state of the environment, but only a transformation of it, and are specified with an additional tuple $\langle \mathcal{O}, \mu_O \rangle$, where $\mathcal{O}$ is an observation space, and $\mu_O : \mathcal{S} \to \Delta(\mathcal{O})$ is an observation kernel, that maps the true environment state to observation probabilities. Because transitioning between observations is not Markovian, policies are a mapping from partial *trajectories*, which we denote as *histories*, to actions. Histories are sequences of transitions $h_t = \{O_0\} \cup \{A_k, R_k, O_{k+1} : 0 < k < t-1\} \in (\mathcal{O} \times \mathcal{A} \times \mathcal{R})^t = \mathcal{H}$.

Generalised Policy Iteration (GPI). We now introduce the concept of value functions. The *state value function* of a policy π is the expected return of the policy from state s_t, $v^\pi(s) = \mathbb{E}_{\pi,\mu}[Z_t|S_t = s]$. The action-value function (or Q-function) of a policy π is the expected return of the policy from state s_t if the agent takes a_t, $q^\pi(s, a) = \mathbb{E}_{\pi,\mu}[Z_t|S_t = s, A_t = a]$. Policy Evaluation (PE) is then the process that maps a policy π to its value function. A canonical PE procedure starts from an arbitrary value function V_0 and iteratively applies the Bellman operator, $\mathcal{T}$, such that:

$$\hat{v}^\pi_{k+1}(S_t) = \mathcal{T}^\pi[\hat{v}^\pi_k(S_t)] := \mathbb{E}_{\pi,\mu}\left[R(S_t, A_t) + \gamma \hat{v}_k(S_{t+1})\right], \tag{2}$$

where $\hat{v}_k$ denotes the value approximation at iteration k, $A_t \sim \mathbb{P}_\pi(\cdot|S_t)$, and $S_{t+1} \sim \mathbb{P}_{\pi,\mu}(\cdot|S_t, A_t)$. The Bellman operator is a γ-contraction in the ℓ_∞ and the ℓ_2 norms, and its fixed point is the value of the policy π. Hence, successive applications of the Bellman operator improve prediction accuracy because the current value gets closer to the true value of the policy. We refer to the PE as the *prediction* objective (Sutton & Barto, 2018). Policy improvement maps a policy π to an improved policy:

$$\pi_{k+1}(a|S) = \mathcal{G}[\pi_k, S] = \mathbb{1}_{\{a\}}(\operatorname*{argmax}_{u \in \mathcal{A}} \left[R(S, u) + \gamma v_k(S')\right]) = \mathbb{1}_{\{a\}}(\operatorname*{argmax}_{u \in \mathcal{A}} \left[q_k(S, u)\right]). \tag{3}$$

We refer to GPI as a general method to solve the *control* problem (Sutton & Barto, 2018) deriving from the composition of PE and Policy Improvement (PI). In particular, we refer to the algorithm that alternates

an arbitrary number k of PE steps and a PI step as Modified Policy Iteration (MPI) (Puterman & Shin, 1978; Scherrer et al., 2015). For $k = 1$, MPI recovers Value Iteration, while for $k \to +\infty$, it recovers Policy Iteration. For any value of $k \in [1, +\infty]$, and under mild assumptions, MPI converges to an optimal policy (Puterman, 2014).

In Deep RL we parameterise a policy using a neural network with parameters set θ and denote the distribution over action as $\pi(a|s, \theta)$. We apply the same reasoning for value functions, with parameters set ϕ, which leads to $v(s, \phi)$ and $q(s, a, \phi)$ for the state and action value functions respectively.

4 Quantifying action influences

We start by answering **Q1.**, which aims to address the problem of *what* to measure, when referring to credit. Since Minsky (1961) raised the Credit Assignment Problem (CAP), a multitude of works paraphrased his words:

- "*The problem of how to incorporate knowledge*" and "*given an outcome, how relevant were past decisions?*" (Harutyunyan et al., 2019),
- "*Is concerned with identifying the contribution of past actions on observed future outcomes*" (Arumugam et al., 2021),
- "*The problem of measuring an action's influence on future rewards*" (Mesnard et al., 2021),
- "*An agent must assign credit or blame for the rewards it obtains to past states and actions*" (Chelu et al., 2022),
- "*The challenge of matching observed outcomes in the future to decisions made in the past*" (Venuto et al., 2022),
- "*Given an observed outcome, how much did previous actions contribute to its realization?*" (Ferret, 2022, Chapter 4.1).

These descriptions converge to Minsky's original question and show agreement in the literature on an informal notion of credit. In this introduction, we propose to reflect on the different metrics that exist in literature to quantify it. We generalise the idea of *action value*, which often only refers to q-values, to that of *action influence*, which describes a broader range of metrics used to quantify the credit of an action. While we do not provide a definitive answer on what credit *should* be, we review how different works in the existing RL literature have characterised it. We now start from developing an intuition of the notion of credit.

Consider Figure 1, inspired to both Figure 1 of Harutyunyan et al. (2019) and to the *umbrella* problem in Osband et al. (2020). The action taken at x_0 determines the return of the episode by itself. From the point of view of *control*, any policy that always takes a' in x_0 (i.e., $\pi^* \in \Pi^* : \pi^*(a'|x_0) = 1$), and then any other action afterwards, is an optimal policy. From the CAP point of view, some optimal actions, namely those after the first one, do not *actually* contribute to optimal returns. Indeed, alternative actions still produce optimal returns and contribute equally to each other to achieve the goal, so their credit is equal. We can see that, in addition to optimality, credit not only identifies optimal actions but informs them of how *necessary* they are to achieve an outcome of interest.

From the example, we can deduce that credit evaluates actions for their potential to influence an outcome. The resulting CAP is the problem of **estimating** the *influence* of an action over an outcome from experimental data and describes a pure association between them.

Why solving the CAP? Action evaluation is a cornerstone of RL. In fact, solving a control problem often involves running a GPI scheme. Here, the influence of an action drives learning for it suggests a possible direction to improve the policy. For example, the action-value plays that role in Equation (3). It follows that the quality of the measure of influence fundamentally impacts the quality of the policy improvement. Low quality evaluations can lead the policy to diverge from the optimal one, hinder learning, and slow down progress (Sutton & Barto, 2018; van Hasselt et al., 2018). On the contrary, high quality evaluations provide

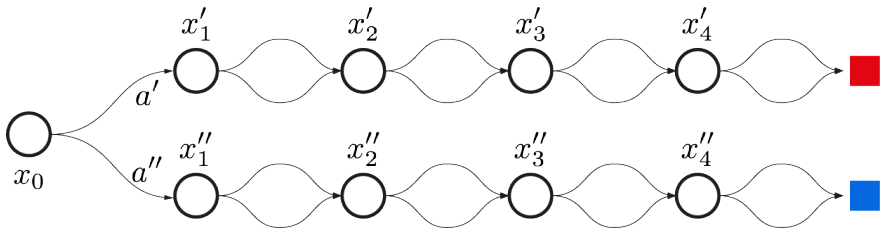


Figure 1: A simplified MDP to develop an intuition of credit. The agent starts at x_0, and can choose between two actions, a' and a'' in each state; the reward is 1 when reaching the upper, solid red square, and 0 otherwise. The first action determines the outcome alone.

accurate, robust and reliable signals that foster convergence, sample-efficiency and low variance. While simple evaluations are enough for specialised experiments, the real world is a complex blend of multiple, sometimes hierarchical tasks. In these cases, the optimal value changes from one task to another and these simple evaluations do not bode well to adapt to general problem solving. Yet, the causal structure that underlies the real word is shared among all tasks and the modularity of its causal mechanisms is often a valuable property to incorporate. In these conditions, learning to assign credit in one environment becomes a lever to assign credit in another (Ferret et al., 2021a), and ultimately makes learning faster, more accurate and more efficient. For these reasons, and because an optimal policy only requires discovering one single optimal trajectory, credit stores knowledge beyond that expressed by optimal behaviours alone, and solving the control problem is not sufficient to solve the CAP, with the former being an underspecification of the latter.

4.1 Are all *action values, credit*?

As we stated earlier, most Deep RL algorithms use some form of *action influence* to evaluate the impacts of an action on an outcome. This is a fundamental requirement to rank actions and select the optimal one to solve complex tasks. For example, many model-free methods use the *state-action value* function $q^\pi(s, a)$ to evaluate actions (Mnih et al., 2015; van Hasselt et al., 2016), where actions contribute as much as the expected return they achieve at termination of the episode. Advantage Learning (AL) (Baird, 1999; Mnih et al., 2016; Wang et al., 2016b, Chapter 5) uses the *advantage* function $A^\pi(s_t, a_t) = q^\pi(s_t, a_t) - v^\pi(s_t)$ [1] to measure credit, while other works study the effects of the *action-gap* (Farahmand, 2011; Bellemare et al., 2016; Vieillard et al., 2020b) on it, that is the relative difference between the expected return of the best action and that of another action, usually the second best. Action influence is also a key ingredient of actor-critic and policy gradient methods (Lillicrap et al., 2015; Mnih et al., 2016; Wang et al., 2016a), where the policy gradient is proportional to $\mathbb{E}_{\mu,\pi}[A^\pi(s, a)\nabla \log \pi(A|s)]$, with $A^\pi(s, a)$ estimating the influence of the action A.

These proxies are sufficient to select optimal actions and unlock solutions to complex tasks (Silver et al., 2018; Wang et al., 2016b; Kapturowski et al., 2019; Badia et al., 2020; Ferret et al., 2021b). However, while these works explicitly refer to the action influence as a measure of credit, the term is not formally defined and it remains unclear where to draw the line between *credit* and other quantities. Key questions arise: *What is the difference between these quantities and credit? Do they actually represent credit as originally formulated by Minsky (1961)? If so, under what conditions do they do?* Without a clear definition of *what* to measure, we do not have an appropriate quantity to target when designing an algorithm to solve the CAP. More importantly, we do not have an appropriate quantity to use as a single source of truth and term of reference to measure the accuracy of other metrics of action influence, and how well they approximate credit. To fill this gap, we proceed as follows:

- Section 4.2 formalises what is a *goal* or an *outcome*: what we evaluate the action for;
- Section 4.3 unifies existing functions under the same formalism;

[1] To be consistent with the RL literature we abuse notation and denote the advantage with a capital letter A^π despite not being random and being the same symbol of the action A_t.

- Section 4.4 formalises the CAP following this definition.
- Section 4.5 analyses how different works interpreted and quantified *action influences* and reviews them.
- Section 4.6 distils the properties that existing measure of action influence.

We suggest the reader only interested in the final formalism to directly skip to Section 4.4, and to come back to the next sections to understand the motivation behind it.

4.2 What is a *goal*?

Because credit measures the influence of an action upon achieving a certain goal, to define credit formally we must be able to describe *goals* formally, and without a clear understanding of what constitutes one, an agent cannot construct a learning signal to evaluate its actions. *Goal* is a synonym for *purpose*, which we can informally describe as a performance to meet or a prescription to follow. Defining a goal rigorously allows to make the relationship between the action and the goal explicit (Ferret, 2022, Chapter 4) and enables the agent to decompose complex behaviour into elementary ones in a compositional (Sutton et al., 1999; Bacon et al., 2017), and possibly hierarchical way (Flet-Berliac, 2019; Pateria et al., 2021; Hafner et al., 2022). This idea is at the foundation of many CA methods (Sutton et al., 1999; 2011; Schaul et al., 2015a; Andrychowicz et al., 2017; Harutyunyan et al., 2019; Bacon et al., 2017; Smith et al., 2018; Riemer et al., 2018; Bagaria & Konidaris, 2019; Harutyunyan et al., 2018; Klissarov & Precup, 2021). We proceed with a formal definition of *goals* in the next paragraph, and review how these goals are *represented* in seminal works on CA in the one after. This will lay the foundation for a unifying notion of credit later in Sections 4.3.

Defining goals. To define goals formally we adopt the *reward hypothesis*, which posits:

> *That all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward).* (Sutton, 2004).

Here, the goal is defined as the *behaviour* that results from the process of maximising the return. The reward hypothesis has been further advanced by later studies (Abel et al., 2021; Pitis, 2019; Shakerinava & Ravanbakhsh, 2022; Bowling et al., 2023). In the following text, we employ the goal definition in Bowling et al. (2023), which we report hereafter:

Definition 1 (Goal)**.** *Given a distribution of finite histories $\mathbb{P}(H), \forall H \in \mathcal{H}$, we define a* goal *as a partial ordering over $\mathbb{P}(H)$, and for all $h, h' \in \mathcal{H}$ we write $h \succsim h'$ to indicate that h is preferred to h' or that the two are indifferently preferred.*

Here, H is a random history in the set of all histories $\mathcal{H}$ as described in Section 3, and $\mathbb{P}(H)$ is an unknown distribution over histories, different from that induced by the policy and the environment. An agent behaviour and an environment then induce a new distribution over histories and we obtain $\mathbb{P}_{\mu,\pi}(H)$ as described in Section 3. This in turn allows to define a partial ordering over policies, rather than histories, and we write analogously $\pi \succsim \pi'$ to indicate the preference. For the *Markov Reward Theorem* (Bowling et al., 2023, Theorem 4.1) and under mild conditions (Bowling et al., 2023), there exist a deterministic, Markov reward function[2] $R : \mathcal{O} \times \mathcal{A} \to [0, 1]$ such that the maximisation of the expected sum of rewards is consistent with the preference relation over policies.

Subjective and objective goals. The *Markov Reward Theorem* holds both in the case of the preferences being defined *internally* by the agent itself – this is the case of *intrinsic motivation* (Piaget et al., 1952; Chentanez et al., 2004; Barto et al., 2004; Singh et al., 2009; Barto, 2013; Colas et al., 2022) – and in case they originate from an *external* entity, such as agent-designer. In the first case, the agent doing the

[2]We omit the transition dependent discounting for the sake of conciseness and because not relevant to our problem. The reader can consult Pitis (2019); White (2017) for details.

maximising is the same as the one holding the ordering over policies, and we refer to the corresponding goal as a *subjective goal*. In the second case, an *agent designer* or an unknown, non-observable entity holds the ordering and a separate *learning agent* is the one pursuing the optimisation process. We refer to a goal as an *objective goal* in this latter case. These settings usually correspond to the distinction between goals and sub-goals in the literature (Liu et al., 2022).

Outcomes. A particularly interesting use of goals for CA is in hindsight (Andrychowicz et al., 2017). Here the agent acts with a goal in mind, but it evaluates a trajectory as if *a* reward function – one different from the original one – was maximised in the current trajectory. We discuss the benefits of these methods in Section 6.4. When this is the case, we use the term *outcome* to indicate a realised goal in hindsight. In particular, given a history $H \sim \mathbb{P}_{\mu,\pi}(H)$, there exist a deterministic, Markov reward function R that is maximal in H. We refer to the corresponding H as an outcome. For example, consider a trajectory h that ends in a certain state s. There exist a Markov reward function that outputs always 0 and 1 only when the s is the final state of h. We refer to h as an *outcome*.

In other words, this way of defining goals or outcomes corresponds to defining a task to solve, which in turn can be expressed through a reward function with the characteristics described above. Vice-versa, the reward function can *encode* a task. When credit is assigned with respect to a particular goal or outcome, it then evaluates the influence of an action to solving a particular task. As discussed above, this is key to decompose and recompose complex behaviours and the definition aligns with that of other disciplines, such as psychology where *a goal . . . is a cognitive representation of something that is possible in the future* (Elliot & Fryer, 2008) or philosophy, where representations do not merely read the world as it is but they express *preferences* over something that is possible in the future (Hoffman, 2016; Prakash et al., 2021; Le Lan et al., 2022).

Representing goals and outcomes. However, expressing the relation between actions and goals explicitly, that is, when the function that returns the credit of an action has a goal as an input, raises the problem of how to *represent* a goal for computational purposes. This is important because among the CA methods that define goals explicitly (Sutton et al., 2011; Schaul et al., 2015a; Andrychowicz et al., 2017; Rauber et al., 2019; Harutyunyan et al., 2019; Tang & Kucukelbir, 2021; Arulkumaran et al., 2022; Chen et al., 2021), not many use the rigour of a general-purpose definition of goal such as that in Bowling et al. (2023). In these works, the *goal-representation space*, which we denote as $\psi \in \Psi$, is arbitrarily chosen to represent specific features of a trajectory. It denotes an *object*, rather than a performance or a prescription to meet. For example, a *goal-representation* ψ can be a state (Sutton et al., 2011; Andrychowicz et al., 2017) and $\psi \in \Psi = \mathcal{S}$. It can be a specific observation (Nair et al., 2018) with $\psi \in \Psi = \mathcal{O}$. Alternatively, it can be an abstract feature vector (Mesnard et al., 2021) that reports on some characteristics of a history, and we have $\psi \in \Psi = \mathbb{R}^d$ with d is the dimensionality of the vector. Even, a goal can be represented by a natural language instruction (Luketina et al., 2019) and $\psi \in \Psi = \mathbb{R}^d$ be the embedding of that piece of text. A goal can be represented by a scalar $\psi \in \Psi = \mathbb{R}$ (Chen et al., 2021) that indicates a specific return to achieve, or even a full command (Schmidhuber, 2019), that is a return to achieve is a specific window of time.

While these representations are all useful heuristics, they lack formal rigour and leave space to ambiguities. For example, saying that the goal is a state might mean that *visiting the state at the end of the trajectory* is the goal or that visiting it in the *middle* of it is the goal. This is often not formally defined and what is the reward function that corresponds to that specific representation of a goal is not always clear. In the following text, when surveying a method or a metric that specifies a goal, we refer to the specific goal representation used in the work and make an effort to detail what is the reward function that underpins that goal representation.

4.3 What is an *assignment*?

Having established a formalism for goals and outcomes we are now ready to describe *credit* formally and we proceed with a formalism that unifies the existing measures of action influence. We first describe a generic function that generalises most CAs, and then proceed to formalise the CAP. Overall, this formulation provides a term of reference for the quantities described in Section 4.5. We now formalise an *assignment*:

Definition 2 (Assignment)**.** *Consider an action $a \in \mathcal{A}$, a goal $g \in \mathcal{G}$, and a context $c \in \mathcal{C}$ representing some experience data. We use the term* assignment function *or simply* assignment *to denote a function $\mathcal{K}$ that maps a context, an action and an outcome to a quantity $y \in \mathcal{Y}$, which we refer to as the* ***influence*** *of the action:*

$$K : \mathcal{C} \times \mathcal{A} \times \mathcal{G} \to \mathcal{Y}. \tag{4}$$

Here, a context $c \in \mathcal{C}$ represents some input data and can be arbitrarily chosen depending on the assignment in question. A context must hold information about the present, for example, the current state or the current observation; it may contain information about the past, for example, the sequence of past decisions occurred until now for a POMDP; to evaluate the current action, it must contain information about what future actions will be taken *in-potentia*, for example by specifying a policy to follow when $a \in \mathcal{A}$ is not taken, or a fixed trajectory, in which case the current action is evaluated in hindsight (Andrychowicz et al., 2017). We provide further details to contexts in Appendix B.

In the general case, the action influence is a random variable $Y \in \mathcal{Y} \subset \mathbb{R}^d$. This is the case, for example, of the action-value distribution (Bellemare et al., 2017) as described in Equation 10, where the action influence is defined over the full distribution of returns. However, most methods extract some scalar measures of the full influence distribution, such as expectations (Watkins, 1989), and the action influence becomes a scalar $y \in \mathbb{R}$. In the following text, we mostly consider scalar forms of the influence $\mathcal{Y} = \mathbb{R}$ as these represent the majority of the existing formulations.

In practice, an *assignment* provides a single mathematical form to talk about the multitude of ways to quantify action influence that are used in the literature. It takes an action $a \in \mathcal{A}$, some contextual data $c \in \mathcal{C}$ and a goal $g \in \mathcal{G}$ and maps it to some measure of *action influence*. While maintaining the same mathematical form, different assignments can return different values of action influence and steer the improvement in different directions.

Equation (4) also resembles the General Value Function (GVF) (Sutton et al., 2011), where the influence $y = q^\pi(s, a, g)$ is the expected return of the policy π when taking action a in state s, with respect a goal g. However, in GVFs: *(i)* y is an *action value* and does not generalise other forms of action influence; the goal is an MDP state $g \in \mathcal{S}$ and does not generalise to our notion of goals in Section 4.2; the function only considers forward predictions and does not generalise to evaluating an action in hindsight (Andrychowicz et al., 2017). Table 2 page 11 contains further details on the comparison and further specifies the relationship between the most common functions and their corresponding assignment.

4.4 The credit assignment problem

The generality of the assignment formalism reflects the great heterogeneity of action influence metrics, which we review later in Section 4.5. This heterogeneity shows that, even if most studies agree on an intuitive notion of credit, they diverge in practice on how to quantify credit mathematically. Having unified the existing assignments in the previous section, we now proceed to formalise the CAP analogously. This allows us to put the existing methods into a coherent perspective as a guarantee for a fair comparison, and to maintain the heterogeneity of the existing measures of action influence.

We cast the CAP as the problem of approximating a measure of action influence from experience. We assume standard model-free, Deep RL settings and consider an assignment represented as a neural network $k : \mathcal{C} \times \mathcal{A} \times \mathcal{G} \times \Phi \to \mathbb{R}$ with parameters $\phi \in \Phi = \mathbb{R}^n$ that can be used to approximate the credit of the actions. This usually represents the critic, or the value function of a RL algorithm. In addition, we admit a stochastic function to represent the policy, also in the form of a neural network $f : \mathcal{S} \times \Theta \to \Delta(\mathcal{A})$, with parameters set $\theta \in \Theta = \mathbb{R}^m$. We assume that $n \ll |\mathcal{S}| \times |\mathcal{A}|$ and $m \ll |\mathcal{S}| \times |\mathcal{A}|$ and note that often subsets of parameters are shared among the two functions.

We further assume that the agent has access to a set of experiences $\mathcal{D}$ and that it can sample from it according to a distribution $D \sim \mathbb{P}_C$. This can be a pre-compiled set of external demonstrations, where $\mathbb{P}_C(D) = \mathcal{U}(D)$, or an MDP, where $\mathbb{P}_C = \mathbb{P}_{\mu,\pi}(D)$, or even a fictitious model of an MDP $\mathbb{P}_C = \mathbb{P}_{\widetilde{\mu},\pi}(D)$, where $\widetilde{\mu}$ is a function internal to the agent, of the same form of μ. These are also mild assumptions as they

Assignment	Action influence	Context	Action	Goal
State-action-value	$q^\pi(s,a)$	$s \in \mathcal{S}$	$a \in \mathcal{A}$	$g \in \mathbb{R}$
Advantage	$q^\pi(s,a) - v(s)$	$s \in \mathcal{S}$	$a \in \mathcal{A}$	$g \in \mathbb{R}$
General q-value function	$q^\pi(s,a,g)$	$s \in \mathcal{S}$	$a \in \mathcal{A}$	$g \in \mathcal{S}$
Distributional action-value	$Q^\pi(s,a)$	$s \in \mathcal{S}$	$a \in \mathcal{A}$	$g \in \{0,\dots,n\}$
Distributional advantage	$D_{KL}(Q^\pi(s,a)\|\|V^\pi(s,a))$	$s \in \mathcal{S}$	$a \in \mathcal{A}$	$g \in \{0,\dots,n\}$
Hindsight advantage	$1 - \frac{\pi(A_t\|s)}{\mathbb{P}_D(A_t\|s_t,Z_t)} Z_t$	$s \in \mathcal{S}, h_T \in \mathcal{H}$	$a \in h$	$g \in \mathbb{R}$
Counterfactual advantage	$\mathbb{P}_D(A_t = a\|S_t = s, F_t = f)q(s,a,f)$	$s \in \mathcal{S}$	$a \in h$	$g \in \mathbb{R}$
Posterior value	$\sum_{t=0}^{T} \mathbb{P}_{\mu,\pi}(U_t = u\|h_t)v^\pi(o_t,x_t)$	$o \in \mathcal{O}, b \in \mathbb{R}^d, \pi$	$A \sim \pi$	$g \in \mathbb{R}$
Policy-conditioned value	$q(s,a,\pi)$	$s \in \mathcal{S}, \pi \in \Pi$	$a \in \mathcal{A}$	$g \in \mathbb{R}$

Table 2: A list of the most common action-*influences* and their assignment functions in the Deep RL literature analysed in this survey. For each function, the table specifies the influence, the context representation, the action, and the goal representation of the corresponding assignment function $K \in \mathcal{K}$.

correspond to, respectively, offline settings, online settings, and model-based settings where the model is learned. We detail these settings in Appendix B. We now define the CAP formally.

Definition 3 (The credit assignment problem)**.** *Consider an MDP $\mathcal{M}$, a goal $g \in \mathcal{G}$, and a set of experience $c \in \mathcal{C}$. Consider an arbitrary assignment $K \in \mathcal{K}$ as described in Equation* (4)*. Given a parameterised function $\widetilde{K} : \mathcal{C} \times \mathcal{A} \times \mathcal{G} \times \Phi \to \mathbb{R}$ with parameters set $\phi \in \Phi \subset \mathcal{R}^n$, we refer to the* Credit Assignment Problem *as the problem of finding the set of parameters $\phi \in \Phi$ such that:*

$$\widetilde{K}(c,a,g,\phi) = K(c,a,g), \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \tag{5}$$

Different choices of action influence have a great impact on the hardness of the problem. In particular, there is a trade-off between:

(a) how effective the chosen measure of influence is to inform the direction of the policy improvement,

(b) how easy it is to learn that function from experience.

For example, using *causal influence* (Janzing et al., 2013) as a measure of action influence makes the CAP hard to solve in practice. The reason is that discovering causal mechanisms from associations alone is notoriously challenging (Pearl, 2009; Bareinboim et al., 2022) and pure causal relationships are rarely observed in nature (Pearl et al., 2000) but in specific experimental conditions. However, causal knowledge is reliable, robust to changes in the experience collected and effective, and causal mechanisms can be invariant to changes in the goal. On the contrary, q-values are easier to learn as they represent a measure of statistical correlation between state-actions and outcomes, but their knowledge is limited to the bare minimum necessary to solve a control problem. Which quantity to use in each specific instance or each specific problem is still subject of investigation in the literature. Ideally, we should aim for the most effective measure of influence that can be learned with the least amount of experience.

4.5 Existing assignment functions

We now survey the most important assignment functions from the literature and their corresponding measure of action influence. The following list is not exhaustive, but rather it is representative of the limitations of existing credit formalisms. For brevity, and without loss of generality, we omit functions that do not explicitly evaluate actions (for example, state-values), but we notice that it is still possible to reinterpret an assignment to a state as an assignment to a set of actions for it affects all the actions that led to that state.

State-action values (Shannon, 1950; Schultz, 1967; Michie, 1963; Watkins, 1989) are a hallmark of RL, and are described by the following expression:

$$q^\pi(s,a) = \mathbb{E}_{\mu,\pi}[Z_t | S_t = s, A_t = a]. \tag{6}$$

Here, the context c is a state $s \in \mathcal{S}$ in the case of MDPs or a history $h \in \mathcal{H}$ for a POMDP. The q-function quantifies the credit of an action by the expected return of the action in the context. q-values are among the simplest way to quantify credit and offer a basic mechanism to solve control problems. However, while q-functions offer solid theoretical guarantees in tabular RL, they can be unstable in Deep RL. When paired with bootstrapping and off-policy learning, q-values are well known to diverge from the optimal solution (Sutton & Barto, 2018). van Hasselt et al. (2018) provides empirical evidence of the phenomenon, investigating the relationship between divergence and performance, and how different variables affect divergence. In particular, the work showed that the Deep Q-Network (DQN) (Mnih et al., 2015) is not guaranteed to converge to the optimal q-function. The divergence rate on both evaluation and control problems increases depending on specific mechanisms, such as the amount of bootstrapping, or the amount of prioritisation of updates (Schaul et al., 2015b). An additional problem arises in GPI schemes used to solve control problems. While during evaluation the policy is fixed, here the policy continuously changes, and it becomes more challenging to track the target of the update while converging to it, as the change of policy make the problem appear non-stationary from the point of view of the value estimation. This is because the policy changes, but there is no signal that informs the policy evaluation about the change. To mitigate the issue, many methods either use a fixed network as an evaluation target (Mnih et al., 2015), they perform Polyak averaging of the target network (Haarnoja et al., 2018), or they clip the gradient update to a maximum cap (Schulman et al., 2017). To further support the idea, theoretical and empirical evidence (Bellemare et al., 2016) shows that the q-function is *inconsistent*: for any suboptimal action a, the optimal value function $q^*(s, a)$ describes the value of a *non-stationary* policy, which selects a different action $\pi^*(s)$ (rather than a) at each visit of s. The inconsistency of q-values for suboptimal actions has also been shown empirically. Schaul et al. (2022) measured the per-state *policy change* $W(\pi, \pi'|s) = \sum_{a \in \mathcal{A}} |\pi(a|s) - \pi'(a|s)|$ for several Atari 2600 games Arcade Learning Environment (ALE) (Bellemare et al., 2013), and showed that the action-gap (the difference of value between the best and the second best action) undergoes brutal changes despite the agent maintaining a constant value of expected returns. In practice, Deep RL algorithms often use q-targets to approximate the q-value, for example, n-step targets (Sutton & Barto, 2018, Chapter 7), or λ-returns (Watkins, 1989; Jaakkola et al., 1993; Sutton & Barto, 2018, Chapter 12). However, we consider them as *methods*, rather than quantities to measure credit, since the q-value is the quantity to which the function approximator converges. For this reason we discuss them in Section 6.1.

Advantage (Baird, 1999) measures, in a given state, the difference between the q-value of an action and the value of its state

$$A^\pi(s, a) = q^\pi(s, a) - v^\pi(s). \tag{7}$$

Here, the context c is the same as in Equation (6). Because $v^\pi(s) = \sum_{a \in \mathcal{A}} q(s, a)\mathbb{P}_\pi(a)$ and $A^\pi(s, a) = q^\pi(s, a) - \mathbb{E}_\pi[q^\pi(s, a)]$, the advantage function measures action influence by the amount an action is better than average, that is, if $A^\pi(s, a) > 0$. As also shown in Bellemare et al. (2016), using the advantage to quantify credit can increase the *action-gap*, the value difference between the optimal and the second-best action. Empirical evidence has shown the consistent benefits of advantage over q-values (Baird, 1999; Wang et al., 2016b; Bellemare et al., 2016; Schulman et al., 2016), most probably due to its regularisation effects (Vieillard et al., 2020b;a; Ferret et al., 2021a). On the other hand, when estimated directly and not by composing state and state-action values, for example in Pan et al. (2022), advantage does not permit bootstrapping. This is because advantage lacks an absolute measure of action influence, and only maintains one that is relative to the other possible actions. Overall, in canonical benchmarks for both evaluation (Wang et al., 2016b) and control (Bellemare et al., 2013), advantage has been shown to improve over q-values (Wang et al., 2016b). In particular, policy evaluation experiences faster convergence in large action spaces because the state-value $v^\pi(s)$ can hold information that is shared between multiple actions. For control, it improves the score over several Atari 2600 games compared to both double q-learning (van Hasselt et al., 2016) and prioritised experience replay (Schaul et al., 2015b).

GVFs (Sutton et al., 2011; Schaul et al., 2015a) are a set of q-value functions that predict returns with respect to multiple reward functions:

$$q^{\pi,R}(s,a) = \{\forall R \in \mathcal{R} : \mathbb{E}_{\mu,\pi}\left[\sum_{t}^{T} R(S_t, A_t)|S_t = s, A_t = a\right]\}, \tag{8}$$

where R is a pseudo-reward function and $\mathcal{R}$ is an arbitrary, pre-defined set of reward functions. Notice that we omit the pseudo-termination and pseudo-discounting terms that appear in their original formulation (Sutton et al., 2011) to maintain the focus on credit assignment. The context c is the same of q-values and advantage, and the goal that the pseudo-reward represents is to reach a specific state $g = s \in \mathcal{S}$. When first introduced (Sutton et al., 2011), the idea of GVFs stemmed from the observation that canonical value functions are limit to only a single task at the same time. Solving a new task would require learning a values function *ex-novo*. By maintaining multiple assignment functions at the same time, one for each goal, GVFs can instantly quantify the influence of an action with respect to multiple goals simultaneously. However, while GVFs maintain multiple assignments, the goal is still not an explicit input of the value function. Instead, it is left implicit and each assignment serves the ultimate goal to maximise a different pseudo-reward function (Sutton et al., 2011).

Universal Value Functions Approximators (UVFAs) (Schaul et al., 2015a) scale GVFs to Deep RL and advance their idea further by conflating these multiple assignment functions into a single one, represented as a deep neural network. Here, unlike for state-action values and GVFs, the goal is an explicit input of the assignment:

$$q^{\pi}(s,a,g) = \mathbb{E}_{\mu,\pi}[Z_t|S_t = s, A_t = a, G_t = g], \tag{9}$$

The action influence here is measured with respect to a goal explicitly. This allows to leverage the generalisation capacity of deep neural networks and to generalise not only over space of states but also over that of goals.

Distributional values (Jaquette, 1973; Sobel, 1982; White, 1988; Bellemare et al., 2017) consider the full return distribution Z_t instead of the expected value:

$$Q^{\pi}(s,a) = \mathbb{P}_{\mu,\pi}(Z_t|S_t = s, A_t = a), \tag{10}$$

where $\mathbb{P}_{\mu,\pi}(Z_t)$ is the probability of achieving a certain return $\mathbb{P}_{\mu,\pi}(Z_t = z)$ with $z \in \mathbb{R}$ and $Q^{\pi}(s,a)$ maps a state-action pair to the distribution over returns. Notice that we use uppercase Q to denote the value distribution and the lowercase q for its expectation (Equation (6)). To translate the idea into a practical algorithm, Bellemare et al. (2017) proposes a discretised version of the value distribution by projecting $\mathbb{P}_{\mu,\pi}(Z_t)$ on a finite support $\mathcal{C} = \{0 \leq i \leq C\}$. The discretised value distribution then becomes $Q^{\pi}(s,a) = \mathbb{P}_C(Z_t|S_t = s, A_t = a)$, where $\mathbb{P}_C$ is a categorical Bernoulli denoting the quantised version of the value distribution Z_t in the sample space $\mathcal{C}$, and describes the probability that a return $\mathbb{P}(Z_t = c), \forall c \in \mathcal{C}$ happens. Here, the context c is the current MDP state and the goal is to achieve a policy that maximises the value distribution under the Wasserstein metric (Bellemare et al., 2017). Notice that while the optimal expected value function $q^*(s,a)$ is unique, in general there are many optimal value distributions. Experimental evidence (Bellemare et al., 2017) suggests that distributional values provide a better quantification of the action influence, leading to superior results in well known benchmarks for control (Bellemare et al., 2013). However, it is yet not clear why distributional values improve over their expected counterparts. One hypothesis is that predicting for multiple goals works as an auxiliary task (Jaderberg et al., 2017), which often lead to better performance. Another hypothesis, is that the distributional Bellman optimality operator proposed in Bellemare et al. (2017) produces a smoother optimisation problem, but the evidence remains weak or inconclusive (Sun et al., 2022).

Distributional advantage (Arumugam et al., 2021) proposes a probabilistic equivalent of the expected advantage:

$$A^{\pi}(s,a) = D_{KL}(Q^{\pi}(s,a)||V^{\pi}(s)), \tag{11}$$

and borrows the properties of both distributional values and the expected advantage. Intuitively, Equation (11) measures how much the value distribution for a given state-action pair changes relative to the distribution for the particular state only, marginalising over all actions. The relationship between the two distributions can then be interpreted as the distributional analogue of Equation (7), where the two quantities appear in their expectation instead. The biggest drawback of this measure of action influence is that it is only treated in theory and there is no empirical evidence that supports distributional advantage as a useful proxy for credit in practice.

Hindsight advantage (Harutyunyan et al., 2019) stems from conditioning the action influence on future states or returns. The return-conditional hindsight advantage function can be written as follows:

$$A^\pi(s,a,z) = 1 - \frac{\mathbb{P}_\pi(A_t = a|S_t = s)}{\mathbb{P}_D(A_t = a|S_t = s, Z_t = z)} z. \tag{12}$$

Here $A^\pi(s,a,z)$ denotes the return-conditional advantage and $\mathbb{P}_D(a_t|S_t = s, Z_t = z)$ is the return-conditional *hindsight distribution* that describes the probability to take action a in s, given than we observe return z at the end of the episode. The context c is the same for q-values and advantage and the goal is a specific value of the return $Z = z$. The idea of *hindsight* – initially presented in Andrychowicz et al. (2017) – is that even if the trajectory does not provide useful information for the main goal, it can be revisited as if the goal was the outcomes just achieved. Hindsight advantage brings this idea to the extreme and rather than evaluating only for a pre-defined set of goals such as in Andrychowicz et al. (2017), it evaluates for every experienced state or return. Here, the action influence is quantified by that proportion of return determined by the ratio in Equation (12). To develop an intuition of it, if the action a leads to the return z with probability 1 such that $\mathbb{P}_D(A_t = a|S_t = s, Z_t = z) = 1$, but the behaviour policy π takes a with probability 0, the credit of the action a is 0. There exist also a state-conditional formulation rather than a return-conditional one, and we refer to Harutyunyan et al. (2019) for details on it to keep the description concise.

Future-conditional advantage (Mesnard et al., 2021) generalises hindsight advantage to use an arbitrary property of the future:

$$A^\pi(s,a,f) = \mathbb{P}_D(A_t = a|S_t = s, F_t = f)q(s,a,f), \tag{13}$$

with $F_t = \psi(d_t)$ being some random property of the future trajectory d_t that starts at time t and ends at the random horizon T; $q(s,a,f) = \mathbb{E}_{\mu,\pi}[Z_t|S_t = s, F_t = f, A_t = a]$ denotes the future-conditioned state-action value function. Notice that you can derive the hindsight advantage by setting $\psi = z$.

Counterfactual advantage (Mesnard et al., 2021) proposes a specific choice of F such that F is independent from the current action. This produces a future-conditional advantage that factorises the influence of an action in two components: the contribution deriving from the intervention itself (the action) and the luck represented by all the components not under control of the agent at the time t, such as fortuitous outcomes of the state-transition dynamics, exogenous reward noise, or future actions. The form is the same that in Equation 13, with the additional condition that $A_t \perp F_t$ and we have $\mathbb{E}_U[D_{KL}(\mathbb{P}(A_t|S_t = s)||\mathbb{P}(A_t|S_t = s, F_t = f)] = 0$. The main intuition behind *counterfactual advantage* is the following. While to compute counterfactuals we need access to a model of the environment, in model-free settings we can still compute all the relevant information u that does not depend on this model. Once learned, a model of u can then represents a valid baseline to compute counterfactuals in a model-free way. To stay in the scope of this section, we detail how to learn this quantity in Section 6.4.

Posterior value functions (Nota et al., 2021) reflects on partial-observability and proposes a characterisation of hindsight advantage bespoke to POMDPs. The intuition behind Posterior Value Functions (PVFs) is that the evaluated action accounts only for a small portion of the variance of returns. The majority of the variance is often due to the part of the trajectory that still has to happen. For this reason, incorporating in the baseline information usually unavailable after the time when the action was taken could have a greater impact in reducing the variance of the policy gradient estimator. PVFs focus on the variance of a future-conditional baseline (Mesnard et al., 2021) caused by the partial observability. Nota et al. (2021) factorises

Name	Explicitness	Recursivity	Future-dependent	Causality
State-action value	○	●	○	○
Advantage	○	●	○	○
GVFs/UVFAs	●	●	○	○
Distributional action-value	◒	●	○	○
Distributional advantage	◒	○	○	●
Hindsight advantage	◒	○	◒	○
Counterfactual advantage	◒	○	◒	●
Posterior value	○	○	●	○
Observation-action value	○	○	○	○
Policy-conditioned value	○	●	●	○

Table 3: A list of the most common action-*influences* and their assignment functions in the Deep RL literature analysed in this survey, and the properties they respect. Respectively, empty circles, half circles and bullets indicate that the property is not respected, that it is only partially respected, and it is fully respected. See Sections 4.5 and 4.6 for details.

a state s into an observable component o and an non-observable one u, and formalises the PVF as follows:

$$v_t^\pi(h_t) = \sum_{u \in \mathcal{U}} \mathbb{P}_{\mu,\pi}(U_t = u | h_t) v^\pi(o_t, u_t) \tag{14}$$

where $u \in \mathcal{U}$ is the non-observable component of s_t such that $s = \{u, o\}$. Notice that this method is not taking actions into account. However, it is trivial to derive the corresponding Posterior Action-Value Function (PAVF) as: $q_t^\pi(h_t, a) = R(s_t, a_t) + v_t^\pi(h_t)$.

Policy-conditioned values (Harb et al., 2020; Faccio et al., 2021) are value functions that include the policy as an input. For example, a policy-conditioned state-action value has the form:

$$q(s, \pi, a) = \mathbb{E}_{\mu,\pi}[Z_t | S_t = s, \pi_i = \pi, A_t = a], \tag{15}$$

where $\pi_i \in \Pi$ denotes the policy from which to sample the actions that follow A_t. Here, the context c is union of the current MDP state and the policy π, and the goal is to maximise the MDP return, which is unknown to the agent. The main difference with state-action values is that, all else being equal, $q(s, \pi, a, g)$ produces different values *instantly* when π varies, since π is now an explicit input, where $q^\pi(s, a, g)$ requires a full PE procedure instead. Using the policy as an input raises the problem of *representing* a policy in a way that can be fed to a neural network. Harb et al. (2020) and Faccio et al. (2021) propose two methods to represent a policy. To keep our attention on the CAP, we refer to their works for further details on possible ways to represent a policy (Harb et al., 2020; Faccio et al., 2021). Here we limit to convey that the problem of representing a policy has been already raised in the literature.

4.6 Discussion

The sheer variety of assignment functions described above leads to an equally broad range of metrics to quantify action influence and what is the best assignment function for a specific problem remains an open question. While we do not provide a definitive answer to the question of which properties are necessary or sufficient for an assignment function to output a satisfactory measure of credit, we set out to draw attention to the problem by abstracting out some of the properties that the metrics above share or lack. We identify the following properties of an assignment function and summarise our analysis in Table 3.

Explicitness. We use the term *explicitness* when the goal appears as an explicit input of the assignment and it is not left implicit or inferred from experience. Using the goal as an input allows assigning credit for multiple goals at the same time. The decision problem can then more easily be broken down into subroutines that are both independent from each other and independently useful to achieve some superior goal g. Overall, explicitness allows incorporating more knowledge because the assignment spans each goals

without losing information about others. This is the case, for example, of UVFAs, hindsight advantages, and future conditional advantages. As discussed in the previous section, *distributional values* can also be interpreted as explicitly assigning credit for each atom of the quantised return distribution, which is why we only partially consider them having this property in Table 3. Likewise, hindsight and future-conditional advantage, while not conditioning on a goal explicitly, can be interpreted as conditioning the influence on sub-goals that are states or returns, and future statistics, respectively. For this reason we consider them as partially being explicit assignments.

Recursivity. We use the term *Recursivity* to characterise the ability of an assignment function to support *bootstrapping* (Sutton & Barto, 2018). When an assignment is Markovian, it also respects a relationship of the type: $K(c_{t+1}, a_{t+1}, g) = f(K(c_t, a_t, g))$, where f projects the influence from the time t to $t+1$. For example, q-values can be written as: $q^\pi(s_{t+1}, a_{t+1}, g) = R(s_t, a_t) + \gamma q^\pi(s_t, a_t, g)$. The possibility to learn an action influence by bootstrapping provides key advantages. Theoretically, bootstrapping reduces the variance of the estimation at the cost of a bias (Sutton & Barto, 2018). In practice, bootstrapping is often necessary in Deep RL when the length of the episode for certain environments makes full Monte-Carlo estimations intractable due to computational and memory constraints. Often, when an assignment supports bootstrapping it also provides an absolute measure of influence of the action, opposed to a relative one. For example, the advantage produces a measure of influence that is relative to all the other possible actions. Indeed we can write $A^\pi(s, a) = q^\pi(s, a) - v^\pi(s)$ or $A^\pi(s, a) = q^\pi(s, a) - \mathbb{E}_{a' \sim \pi}[q^\pi(s, a')]$. In fact, when estimated directly (Pan et al., 2022) and not as the difference between $q(s, a)$ and $v(s)$, it cannot be learnt via bootstrapping, and one must obtain complete episodes to have unbiased samples of the return. This is often not advised as it increases the variance of the estimate of the return. At the same time, advantage also produces a measure of influence that is relative to all the other possible actions. On the other hand, q-values, which provide a measure of influence that does not vary if that of other actions in the same state do, support bootstrapping. Overall, both approaches to quantify influence have their pros and cons, and the main benefit of recursivity is to allow bootstrapping.

Future-dependent. We use the term *future-dependent* for assignments that take as input information about what actions will be or have been taken *after* the time t at which the action A_t is evaluated. This is a key ingredient of many evaluations because the influence of the current action over a goal depends also on what happens *after* the action. For example, picking up a key is not meaningful if the policy does not lead to open the door afterwards, and the action to grab the key becomes irrelevant. Actions can be specified *in-potentia*, for example by specifying a policy to follow after the action. This is the case of policy-conditioned value function, whose benefit is to explicitly condition the value function on the policy such that, if the policy changes, but the action remains the same, the influence of the action changes *instantly*. Actions can also be specified *in realisation*. This is the case, for example, of hindsight evaluations (Andrychowicz et al., 2017) such as the hindsight advantage, the counterfactual advantage, and the PVF where actions are evaluated considering the full trajectory just collected. However, these functions only consider *features* of the future: the hindsight advantage considers only the final state or the final return of a trajectory; the counterfactual advantage considers some action-independent features of the future; the posterior value function considers only the non-observable components. Because futures are not considered fully, we consider these functions as only partially specifying the future. Furthermore, while state-action value functions, the advantage and their distributional counterparts specify a policy in principle, that information is not an explicit input of the assignment, but only left implicit. In practice, in Deep RL, if the policy changes, these assignment would not change their estimates.

Causality. We refer to a *causal* assignment when the influence that it produces is also a measure of causal influence (Janzing et al., 2013). For example, the counterfactual advantage proposes an interpretation closer to causality of the action influence by factorising the influence of an action in two. The first factor includes only the non-controllable components of the trajectory (e.g., exogenous reward noise, stochasticity of the state-transition dynamics, stochasticity in the observation kernel), or those not under direct control of the agent at time t, such as future actions. The second factor includes only the effects of the action alone. The interpretation is that, while the latter is due causation, the former is only due to fortuitous correlations. This vicinity to causality theory exists despite the counterfactual advantage not being a satisfactory measure

of causal influence as described in Janzing et al. (2013). Distributional advantage in Equation 11 can also be interpreted as containing elements of causality. In fact, we have that the expectation of the advantage over states and actions is the Conditional Mutual Information (CMI) between the policy and the return, conditioned on the state-transition dynamics: $\mathbb{E}_{\mu,\pi}[D_{KL}(Q^\pi(s,a)||V^\pi(s))] = \mathcal{I}(\mathbb{P}_\pi(A|S=s); Z|\mathbb{P}_\mu(S))$. The CMI (with its limitations (Janzing et al., 2013)) is a known measure of causal influence.

Overall, these properties define some characteristics of an assignment, each one bringing positive and negative aspects. Explicitness allows to maintain the influence of an action with respect to multiple goals at the same time, promoting the reuse of information and a compositional onset of behaviour. Recursivity ensures that the influence can be learned via bootstrapping, an essential component of many RL methods. Future-dependency separates assignments by whether they include information about the future actions. Finally, causality has the benefit to filter out the spurious correlation and provides clearer signals to the policy improvement.

4.7 Summary

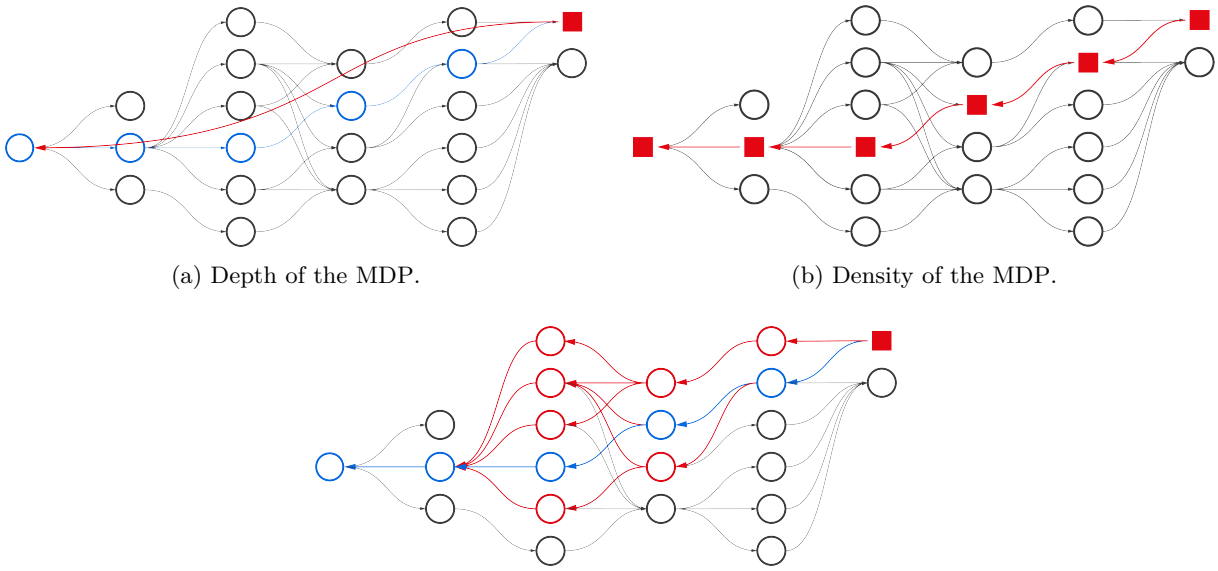
In this section, we addressed **Q1.** and discussed the problem to quantify action influences. In Section 4.1 we formalised our question: "*How do different works quantify action influences?*" and "*Are these quantities satisfactory measures of credit?*". We proceeded to answer the questions. In Section 4.2 we formalised the concept of *outcome* as some arbitrary function of a given history. In Section 4.3 we defined the assignment function as a function that returns a measure of action influence. In Section 4.4 we used this definition to formalise the CAP as the problem of learning a measure of action influence from experience. We refer to the set of protocols of this learning process as a credit assignment *method*. In Section 4.5 we surveyed existing measures of action influence from literature, detailed the intuition behind them, their advantages and drawbacks. Finally, in Section 4.6 we discussed how these measures of action influence relate to each other, the properties that they share and those that are more rare in literature, but still promising for future advancements. In the next section, we proceed to address **Q2.**, by describing the challenges that arise from solving the CAP in Section 5 and surveying the methods to solve the CAP in Section 6.

5 The challenges to assign credit in Deep RL

Having clarified what are the measure of action influence available in literature, we now look at the challenges that arise to learn them and, together with Section 6, answer **Q2.**. The challenges proposed below provide a perspective to understand the principal directions of development of the methods to assign credit and to classify the sheer variety of methods that have been proposed. These challenges are often independent of the choice of action influence and apply to all of them. However, solving the CAP with a measure of influence or another will impact the prominence of each challenge. For example, hindsight methods deal better with sparsity compared to q-values, and so do GVFs. Overall, the following challenges help identify the major, outstanding research questions around devising a method to assign credit.

The current literature identifies the following sub-problems to assign credit: *(a)* **delayed rewards** (Raposo et al., 2021; Hung et al., 2019; Arjona-Medina et al., 2019; Chelu et al., 2022): reward collection happens long after the action that determined it, causing its influence to be perceived as faint; *(b)* **sparse rewards** (Arjona-Medina et al., 2019; Seo et al., 2019; Chen & Lin, 2020; Chelu et al., 2022): the reward function is zero everywhere, and rarely spikes, causing uninformative Temporal Difference (TD) errors; *(c)* **partial observability** (Harutyunyan et al., 2019): where the agent does not hold perfect information about the current state; *(d)* **high variance** (Harutyunyan et al., 2019; Mesnard et al., 2021; van Hasselt et al., 2021) of the optimisation process; *(e)* the resort to **time as a heuristic** to determine the credit of an action (Harutyunyan et al., 2019; Raposo et al., 2021): *(f)* the lack of **counterfactual** CA (Harutyunyan et al., 2019; Foerster et al., 2018; Mesnard et al., 2021; Buesing et al., 2019; van Hasselt et al., 2021); *(g)* **slow convergence** (Arjona-Medina et al., 2019).

While these issues are all very relevant to CAP, their classification is also tailored to control problems. Some of these are described by the use of a particular solution, such as *(e)*, or the lack thereof, like *(f)*, rather than by a characteristic of the decision or of the optimisation problem. Here, we systematise these issues



(a) Depth of the MDP.

(b) Density of the MDP.

(c) Breadth of the MDP.

Figure 2: Visual intuition of the three challenges to temporal CA and their respective set of solutions, using the graph analogy. Nodes and arrows represent, respectively, MDP states and actions. Blue nodes and arrows denote the current episode. Black ones show states that could have potentially been visited, but have not. Square nodes denote goals. Forward arrows (pointing right) represent environment interactions, whereas backward arrows (pointing left) denote credit propagation via state-action back-ups. From top left: **(a)** the temporal distance between the accountable action and the target state requires propagating credit deep back in time; **(b)** considering any state as a target increases the density of possible associations and reduces information sparsity; and finally, **(c)** the breadth of possible pathways leading to the target state.

and transfer them to the CAP. We identify three principal characteristics of MDPs, which we refer to as *dimensions* of the MDP: **depth**, **density** and **breadth** (see Figure 2). Challenges to CA emerge when pathological conditions on depth, density, and breadth produce specific phenomena that mask the learning signal to be unreliable, inaccurate, or insufficient to correctly reinforce an action. We now detail these three dimensions and the corresponding challenges that arise.

5.1 Delayed effects due to high MDP depth

We refer to the *depth* of an MDP as the number of temporal steps that intervene between an highly influential action and an outcome. When this happens, we refer to the action as a *remote* action, and the outcome as a *delayed* outcome. When outcomes are delayed, the increase of temporal distance often corresponds to a combinatorial increase of possible alternative futures and the paths to get to them. In these conditions, recognising which action was responsible for the outcome is harder since the space of possible associations is very large. We identify two main reasons for an outcome to be delayed, depending on whether the decision after the remote action influences the outcome or not.

The first is that the success of the action is not immediate but requires a sequence of actions to be performed *afterwards*, which causes the causal chain to be long. This issue originates from the typical hierarchical structure of many MDPs, where the agent must first perform a sequence of actions to reach a subjective sub-goal, and then perform another sequence to reach another. These behaviours can then be composed to reach the final, objective goal and solve the assigned task. When this happens, agents must be able to assign credit to the individual actions that are responsible for the objective goal, while still being able to select sub-goals along the way and assign credit to action for their ability to reach the subjective sub-goal. The key-to-door task (Hung et al., 2019) is a good example of this phenomenon, where the agent must first

collect a key, to be able to open a door later. Here, one decision (opening the door) is contingent upon a previous one (collecting the key), and the credit of the latter is delayed until the former is performed.

The second reason why outcomes can be delayed is that they might only be *observed* after a long time horizon since the decisions taken *after* the remote action do not influence the outcome significantly. This issue originates from behavioural psychology and is known as the *delayed reinforcement* problem (Lattal, 2010),

> *Reinforcement is delayed whenever there is a period of time between the response producing the reinforcer and its subsequent delivery.* (Lattal, 2010)

One can find references to the same phenomenon in RL as *long-term CA* (Ma et al., 2021; Raposo et al., 2021; Hung et al., 2019) or *long-term consequences* (Barto, 1997; Vinyals et al., 2017; Hung et al., 2019), *long-horizon tasks* (Gupta et al., 2019; Arumugam et al., 2021), or *delayed rewards* (Hung et al., 2019; Arjona-Medina et al., 2019). The main challenge with *delayed reinforcements* is in being able to ignore the series of irrelevant decisions that are encountered between the remote action and the delayed outcome, focus on the actions that are responsible for the outcome, and assign credit accordingly. This is a key requirement because most CA methods rely on temporal recency as a heuristic to assign credit (Klopf, 1972; Sutton, 1988; Mahmood et al., 2015; Sutton et al., 2016; Jiang et al., 2021a). When this is the case, the actions in the proximity of achieving the goal are reinforced, even if not actually being responsible for the outcome (only the remote action is), but only happen to be temporally close to the outcome.

In practice, the prevalence of delayed effects can manifest as a lack of progress in training, but it is often hard to isolate the impact of delayed effects from the other features of the environment that hinder learning without appropriate experimental conditions being set. For example, consider the key-to-door environments introduced above, where the agent has to collect a key that opens a door and then navigate to a certain position of a grid. The effects of picking up the key are delayed until the target square is reached, which, in turn, is contingent upon opening the door. In these conditions it is hard to disentangle the problem of *exploring* the right combination with that of learning that the event of grabbing the key is necessary to obtain that combination. We discuss more in detail how to diagnose this in Section 7 and the relationship between CAP and exploration in Section 5.4. To solve the challenge, a subset of studies on the CAP has focused on this issue, and have proposed methods that can deal specifically with delayed effects either by using memory (Hung et al., 2019; Arjona-Medina et al., 2019; Ferret et al., 2021a; Ren et al., 2022; Raposo et al., 2021), re-weighing updates (Sutton et al., 2016; Chelu et al., 2022), or meta-learning (Xu et al., 2018; Badia et al., 2020; Kapturowski et al., 2022; Flennerhag et al., 2021; Oh et al., 2020).

5.2 Low action influence due to low MDP density

If delayed effects are characterised by a large temporal distance between an action and the outcome it causes, MDP sparsity derives from a *lack of influence* between them. This is substantially different from delayed effects, where actions can cause outcomes very frequently, except with delay. Here, actions have little impact on the probability to achieve a given goal neither now, nor far in the future, and it does not matter what the agent does, the outcome will still be the same. We identify two main reasons why this happens.

The first one is a high stochasticity of the state-transition distribution, which is characterised by a high entropy of the state-transition distribution $\mathcal{H}(\mathbb{P}_\mu)$ and/or the reward function $\mathcal{H}(\mathbb{P}(R))$. When this happens, actions hardly affect the future states of the trajectory. The agent is unable to make predictions with high confidence, and therefore cannot select actions that are likely to lead to the goal.

The second reason is the low goal density. This is the canonical case of reward sparsity in RL, where the goal is only achievable in a small subset of the state space, or for a specific sequence of actions.

Formally, we can measure the *lack of influence* using the notion of information sparsity (Arumugam et al., 2021) of an MDP.

Definition 4 (MDP sparsity)**.** *An MDP is ε-information sparse if:*

$$\max_{\pi \in \Pi} \mathbb{E}_{\mu,\pi}[D_{KL}(P_{\pi,\mu}(Z|s,a)||P_{\pi,\mu}(Z|s))] \leq \varepsilon, \tag{16}$$

where $\mathbb{E}_{\mu,\pi}$ denotes the expectation over the stationary distribution induced by the policy and the state-transition dynamics. The information sparsity of an MDP is the maximum information gain that can be obtained by an agent, represented by a policy π, by knowing which immediate action is played. When the information gain is low everywhere, and only concentrated in a small subset of decisions, CA methods often struggle to assign credit, because the probability of the outcome occurring is low, and there is rarely a signal to propagate. Here, exploration also plays a key role. Indeed, to acquire knowledge (the CAP), the underlying set of associations between actions an outcome must be discovered first (exploration). Often, this is faced by artificially improving the learning signal, either by reward shaping (Ng et al., 1999; Zou et al., 2019; Hu et al., 2020), using auxiliary goals (Sutton et al., 2011; Schaul et al., 2015a), or selecting goals in hindsight (Rauber et al., 2019; Andrychowicz et al., 2017; Harutyunyan et al., 2019; Tang & Kucukelbir, 2021).

5.3 Low action influence due to high MDP breadth

We refer to the *breadth* of an MDP by the expected number of possible alternative routes that can lead to a given outcome. To provide an intuition of how it affects CA we borrow that of *transpositions* from game theory, in particular from chess. *A transposition is an alternative sequence of actions and states that results in the same final result.* In RL, given a trajectory h, we call transposition another trajectory h' that produces the same outcome $\psi(h) = \psi(h')$. We formalise the concept using the notion of the *null space* of a policy (Schaul et al., 2022). Given a policy π, its null space is the subspace of policies with the same expected value:

$$\text{Null}(\pi) := \overline{\Pi} \subseteq \Pi : v(\overline{\pi}) = v(\pi), \quad \forall \pi, \overline{\pi}_2 \in \overline{\Pi}. \tag{17}$$

Definition 5 (Transposition)**.** *Given a reference policy π, and its null space $Null(\pi)$, consider a random sample $\overline{\pi} \in Null(\pi)$. We refer to* transposition *as any unique trajectory drawn following $\overline{\pi}$.*

An *optimal set of transpositions* is then the set of transpositions for the set of optimal policies Π^*.

We refer to this as *transposition* because it is a form of *permutation* of the decisions in the trajectory, which still produce the same results. Because many optimal pathways exist, there is no one key pathway that is responsible for the outcome, and not *bottleneck* decision that the agent has to make necessarily to achieve the goal. When this happens, the influence of these actions is low because credit is diluted over many alternative routes.

This challenge occurs when there exist a high number of combinations of states and actions that can lead to the same outcome. When only one pathway is found, its credit is confounded as high, when in fact it is not, because many other combinations lead to the same outcome. Deep RL algorithms often struggle to find all transpositions, because they often stop exploring already when a single solution (one optimal pathway) is found. To mitigate the problem, different studies experimented with propagating credit to experiences beyond the current trajectory using memory (van Hasselt et al., 2021; Jiang et al., 2021c), world models (Chelu et al., 2020) that imagine proceeding backwards (Edwards et al., 2018; Goyal et al., 2019; Nair et al., 2020; Buesing et al., 2019; Lu et al., 2020; Zhang et al., 2020; Lai et al., 2020), or that plan forward (Sutton, 1990).

5.4 Relationship with the exploration problem

The challenges above refer to the CAP alone and try to isolate independent components of an MDP that affect the CAP. Before concluding the section, we discuss the relationship between these three challenges and the exploration problem (Amin et al., 2021).

Exploration and CA are two cornerstones of RL. Exploration is the problem of *discovering* temporal sequences of states, actions and rewards with the purpose of acquiring new information and expanding the pool of viable ways to act in an unknown environment (Jiang et al., 2023). For example, consider a key-to-door environment, where the agent need to pick up a key, which opens a door, behind which lies a reward. In this environment, Exploration *discovers* the combination of actions and states that visits the key, grabs it, goes to the door and opens it. The discovery is often *not* the result of informed decision-making, the agent

does not know that the key opens a door and this happens often only by chance[3]. On the other hand, CA is tasked with consuming the full set of experiences acquired by Exploration, with the purpose of *associating* elements of it: an state-action and an outcome. Associations remain alive for a certain period of time until superseded by others or due to extinction (Thorndike, 1898; Pavlov, 1927; Skinner, 1937). In the example above, one can think of CA as *remembering* – *learning* – that the key opens a door, in order to reuse this association in the future. Unlike with exploration, this behaviour is not the result of chance anymore, but of informed-decision making powered by ability to forecast the effects of the action (Sutton et al., 2011).

While exploration and CA can be studied independently under particular experimental conditions (see Section 7), they depend on each other when solving control problems, where the agent relies on a combination of both. The relationship between the two is particularly evident in sparse MDP, where the action influence is low. In these conditions, from the CA point of view, actions struggle to have any influence on the outcome. From the point of view of exploration, this is often described in different terms as the sparse reward problem (Ladosz et al., 2022). Indeed, when rewards are particularly sparse (Ladosz et al., 2022), exploration struggles to discover rewards. In support of this connection, studies on CA often start from sparse rewards setups, for example, in Andrychowicz et al. (2017); Arumugam et al. (2021); Edwards et al. (2018). On one hand, this connection makes it often hard to disentangle the impacts of CA and exploration on solving the overall RL problem. On the other hand, it highlights that the two problems are interdependent in the most common control settings and making claims on one or the other requires particular care.

To conclude, while CA and exploration are orthogonal problems and can be studied independently, what data to use to learn credit is fundamental to solve control problem, and that is where the two problems connect.

5.5 Summary

In this section, we have identified the challenges that arise from solving the CAP. These challenges include delayed rewards, sparse rewards, partial observability, high variance, the resort to time as a heuristic, lack of counterfactual CA, and sample efficiency. We have systematised these issues as challenges that emerge from specific properties of the decision problem, which we refer to as dimensions of the MDP: depth, density, and breadth. Challenges to CA emerge when pathological conditions on these dimensions produce specific phenomena that mask the learning signal to be unreliable, inaccurate, or insufficient to correctly reinforce an action. We have provided an intuition of this classification with the aid of graphs and proceeded to detail each challenge. Finally, we discussed the connection between the CAP and the exploration problem, encouraging to take particular care to disentangle the contribution of each of them when making claims on one or the other.

With these challenges in mind, we now proceed to review the state of the art in CA, and discuss the methods that have been proposed to address them.

6 Methods to assign credit in Deep RL

Following the definition of CAP in Section 4.4 a *credit assignment method* is an algorithm to approximate action influence from a finite amount of experience. In this section, we present a list of credit assignment methods that focuses on Deep RL. Our classification aims to identify the principal directions of development around credit assignment algorithms, that is, to minimise the intersection between each class of methods. Our intent is to understand the density around each set of approaches, to locate the branches suggesting the most promising results, and to draw a trend of the latest findings. This can be helpful to both the researchers on the CAP who want to have a bigger picture of the current state of the art, to general RL practitioners and research engineers to identify the most promising methods to use in their applications, and to the part of the scientific community that focuses on different problems, but that can benefit from the insights on CA. We define a CA method according to how it specifies three elements.

[3]Or, rather, by the laws of the exploration algorithm.

(a) The measure of action influence, thus the assignment function K. This usual an approximation of one of the quantities discussed in Section 4.5, for example, an n-step return target in place of the full target.

(b) The protocol that the method uses to approximate K from the experience $\mathcal{D}$.

(c) The mechanism it uses to collect the experience, which we refer to as the *contextual distribution* (see Appendix B for more details).[4]

This provides consistency with the framework just proposed, and allows categorising each method by the heuristics that it uses to assign credit. Therefore, for each method, we report the three elements described above. We identify the following categories:

1. Methods using **time contiguity** as a heuristic (Section 6.1).
2. Those **decomposing returns** into per-timestep utilities (Section 6.2).
3. Those conditioning on **predefined goals** explicitly (Section 6.3).
4. Methods conditioning the present on **future outcomes in hindsight** (Section 6.4).
5. Modelling trajectories as **sequences** (Section 6.5).
6. Those **planning or learning backwards** from an outcome (Section 6.6).
7. **Meta-learning** different proxies for credit (Section 6.7).

Note that, we do not claim that this list of methods is exhaustive. Rather, as for Section 4.5, this taxonomy is representative of the main approaches to assign credit, and a tool to understand the current state of the art in the field. We are keen to receive feedback on missing methods from the list to improve further revisions of the manuscript.

To simplify the reading, we group the classes into subsections, and format each method into its own paragraph. Each method contains a brief description of the intuition, how it is employed to assign credit, and a specification of the context, the action value it measures, and the way it learns that quantity from experience. We now proceed to describe the methods, which we also summarise in 4.

6.1 Time as a heuristic

One common way to assign credit is to use time contiguity as a proxy for causality: an action is as influential as it is temporally close to the outcome. This means that, regardless of the action being an actual cause of the outcome, if the action and the outcome appear temporally close in the same trajectory, the action is assigned high credit. At their foundation, there is TD learning (Sutton, 1988), which we describe below.

TD learning (Sutton, 1984; 1988; Sutton & Barto, 2018) iteratively updates an initial guess of the value function according to the differences between expected and observed outcomes. More specifically, the agent starts with an initial guess of values, acts in the environment, observes returns, and aligns the current guess with the observed return. The difference between the expected return and the observed one is the TD error.

When the temporal distance between the goal and the action is high – a premise at the base of the CAP – it is often impractical to observe very far rewards. As time grows, so does the variance of the observed outcome, due to intrinsic stochasticity in the environment dynamics, the reward function, or the policy. To mitigate the issue, TD methods often replace the theoretical measure of influence with a an approximation: the *TD target*. Instead of updating the current guess on the observed return, these methods use the sum of discounted rewards observed for arbitrary n steps, plus their current value estimate at the last step observed.

[4]To enhance the flow of the manuscript, we formalise *contextual distributions* in Appendix B, and since they are intuitive concepts, we describe them in words when surveying the methods.

Publication	Method	Class	Depth	Density	Breadth
Klopf (1972)	ET	Time	●	○	○
Sutton et al. (2016)	ETD	Time	●	○	○
Baird (1999)	AL	Time	○	○	●
Pan et al. (2022)	DAE	Time	○	○	●
Ferret et al. (2021b)	SAIL	Time	○	●	●
Hung et al. (2019)	TVT	Return decomposition	●	○	○
Arjona-Medina et al. (2019)	RUDDER	Return decomposition	●	○	○
Ferret et al. (2021a)	SECRET	Return decomposition	●	●	○
Ren et al. (2022)	RRD	Return decomposition	●	○	○
Raposo et al. (2021)	SR	Return decomposition	●	○	○
Sutton et al. (2011)	GVF	Auxiliary goals	○	●	○
Schaul et al. (2015a)	UVFA	Auxiliary goals	○	●	○
Andrychowicz et al. (2017)	HER	Future-conditioning	○	●	○
Rauber et al. (2019)	HPG	Future-conditioning	○	●	○
Harutyunyan et al. (2019)	HCA	Future-conditioning	○	●	○
Schmidhuber (2019)	UDRL	Future-conditioning	○	●	○
Mesnard et al. (2021)	CCA	Future-conditioning	○	●	●
Nota et al. (2021)	PPG	Future-conditioning	○	●	●
Venuto et al. (2022)	PGIF	Future-conditioning	○	●	○
Buesing et al. (2019)	CBPS	Future-conditioning	○	●	●
Janner et al. (2021)	TT	Sequence modelling	○	●	○
Chen et al. (2021)	DT	Sequence modelling	○	●	○
Zheng et al. (2022)	ODT	Sequence modelling	○	●	○
Furuta et al. (2022)	GDT	Sequence modelling	○	●	○
Goyal et al. (2019)	Recall traces	Backward planning	○	●	●
Edwards et al. (2018)	FBRL	Backward planning	○	●	●
Nair et al. (2020)	TRASS	Backward planning	○	●	●
Wang et al. (2021)	ROMI	Backward planning	○	●	●
Lai et al. (2020)	BMPO	Backward planning	○	●	●
van Hasselt et al. (2021)	ET(λ)	Learning predecessors	●	○	●
Xu et al. (2018)	MG	Meta-Learning	●	○	○
Xu et al. (2020)	FRODO	Meta-Learning	●	○	○
Yin et al. (2023)	Distr. MG	Meta-Learning	●	○	○

Table 4: List of the most representative algorithms for CA classified by the CA challenge they aim to address. For each method we report the publication that proposed it, the class we assigned to it, and whether it is designed to address each challenge described in Section 5. Hollow circles mean that the method does not address the challenge and the full circle represents the opposite.

This is referred to as *boostrapping* (Sutton & Barto, 2018) and allows to write the current action influence as a function of a future one. As a consequence, the *TD target* is what drives the learning process. In GPI schemes, the value function is updated to approximate the *target*, and not the theoretical action influence measure behind it, even if eventually it may converge to it (Sutton & Barto, 2018; van Hasselt et al., 2018, Chapter 11.3). Since policy improvement uses the current approximation of the value to update the policy, future behaviour are shaped according to it.

We separate the methods in this category in three subgroups: those specifically designed around the advantage function, those re-weighing updates, and those assigning credit to sets of temporally extended courses of actions.

6.1.1 Advantage-based approaches

The first subset of methods use some form of *advantage* (see Section 4.5) as a measure of action influence but still uses time as a heuristic to learn it.

Policy Gradient (PG) and Actor-Critic (AC) methods with a baseline function (Sutton & Barto, 2018, Chapter 13) approximate advantage to measure action influence when using the value function as a baseline. In fact, the policy gradient is proportional to $\mathbb{E}_{\mu,\pi}[(Q^\pi(s,a) - b(s))\nabla \log \pi(A|s)]$ and if we choose $v(s)$ as our baseline $b(s)$, we get $\mathbb{E}_{\mu,\pi}[(A^\pi(s,a))\nabla \log \pi(A|s)]$ because $q^\pi(s,a) - v^\pi(s,a) = A^\pi(s,a)$. The use an action-independent baseline function usually helps to reduce the variance of the policy gradients, while maintaining an unbiased estimate of it (Sutton & Barto, 2018). What function to use as a baseline is the subject of major studies and different choices of baselines often yield methods that go beyond using time as a heuristic (Harutyunyan et al., 2019; Mesnard et al., 2021; Nota et al., 2021; Mesnard et al., 2023).

Advantage Learning (AL) (Baird, 1999) also uses time as a proxy for causality. However, rather than using q-values as a proxy for credit, AL uses the advantage $A_t = q^\pi(s_t, a_t) - v(s_t)$, which, despite not being a measure of causal influence, it improves on the q-value by providing a measure of the relative importance of the action. There are many instances of AL in the Deep RL literature. Duelling Deep Q-Network (DQN) (Wang et al., 2016b) improves on DQN by replacing the q-value with the advantage as a proxy for credit. In these methods the action influence is measured by the advantage:

$$K(c, a, g) = A^\pi(s, a). \tag{18}$$

The context c is an MDP state $c = s \in \mathcal{S}$, the action is the greedy action with respect to the current advantage estimation, and the goal is the maximum expected return of an optimal policy $g = z^* \in \mathbb{R}$. The advantage, and its effects of using it as a proxy for credit, has been further investigated by measuring the action-gap (Bellemare et al., 2016), the difference between the highest and the second-highest action value.

Direct Advantage Estimation (DAE) (Pan et al., 2022) exploits the idea that $\mathbb{E}_\pi[A^\pi(s,a)] = 0$ to estimate the advantage directly from data and not as the usual difference $A^\pi(s,a) = q^\pi(s,a) - v^\pi(s)$. Because this method does not approximate q or v as an intermediate step to $A\pi$, DAE is limited to full Monte-Carlo returns. This way of estimating the advantage has the further drawback of not allowing bootstrapping anymore. In the canonical way of estimating advantage, $q^\pi(s,a)$ or $v^\pi(s)$ can tell how valuable the current state is, which is necessary to bootstrap the following values. However, since we do not have estimations of neither $q^\pi(s,a)$ or $v^\pi(s)$, but only of $A^\pi(s,a)$ directly, we cannot learn the advantage function with bootstrapping. The influence of an action in DAE is the same as for canonical advantage learning.

Self-Imitation Advantage Estimation (SAIL) (Ferret et al., 2021b) combines AL with self-imitation learning into an off-policy algorithm that increases the action-gap (Bellemare et al., 2016). As for DAE, it differs from previous methods by the protocol it uses to estimate the advantage from experience and combines advantage learning with self-imitation learning (Oh et al., 2018).

6.1.2 Re-weighing updates and compound targets

The second subset of methods in this category re-weighs temporal updates according to some heuristics. Re-weighing updates can be useful to emphasise or de-emphasise important states or actions.

Eligibility Traces (ET) (Klopf, 1972; Singh & Sutton, 1996; Precup, 2000a; Geist et al., 2014; Mousavi et al., 2017) credit the long-term impact of actions on future rewards by keeping track of the influence of past actions on the agent's future reward. Specifically, an eligibility trace (Sutton & Barto, 2018, Chapter 12) is a function that assigns a weight to each state-action pair, based on the recency of the state-action pair. A *trace* spikes every time a state-action is visited and decays exponentially over time until the next visit. There are several types of eligibility traces, depending on the law of decay of the trace, for example, Klopf (1972); Singh & Sutton (1996). Overall, they often admit two equivalent implementations: the *forward* and the *backward* view, which differ both in the context and the action value that they measure. For the reasons described in Section 1, we focus on their Deep RL formulation, which mostly implements their backward view. Deep Q(λ)-Network (DQ(λ)N) (Mousavi et al., 2017) implement eligibility traces on top of a DQN (Mnih et al., 2015). In ETs with deep function approximation, the eligibility trace is a vector $e \in \mathbb{R}^d$ with the same number of components d as the parameters of the DNN, and the action influence is measured by q-value with parameters set $\theta \in \mathbb{R}^d$:

$$K(c, a, g) = q^\pi(s, a, \theta). \tag{19}$$

The context c is an MDP state $c = s \in \mathcal{S}$, the action is either sampled from the policy for on-policy methods (e.g., SARSA(λ)) or arbitrarily chosen for off-policy methods (e.g., Q(λ)), and the goal is the maximum expected return of an optimal policy $g = z^* \in \mathbb{R}$. The ET information is embedded in the parameters θ since they are updated according to $\theta \leftarrow \theta + \delta e$. δ is a TD error and e is the eligibility trace, incremented at each update by the value gradient (Sutton & Barto, 2018, Chapter 12): $e \leftarrow \gamma\lambda e + \nabla_\theta q^\pi(s, a)$. Notice the dependence on Z_t, which underlines that the method is a *backward* method. Successive works advanced on the idea of ETs, and proposed different updates for the eligibility vector (Singh & Sutton, 1996; van Hasselt & Sutton, 2015; Precup, 2000a).

Emphatic Temporal Differences (ETDs) (Sutton et al., 2016; Mahmood et al., 2015; Jiang et al., 2021b) improves on the instabilities of ET by re-weighing state-action updates. The re-weighing is based on the *emphatic trace*, a per-parameter value that encodes the degree of bootstrapping of a state. The intuition behind ETDs is that states with high (low) uncertainty – states whose estimates result from heavy (soft) bootstrapping – are less (more) reliable. The main adaptation of the algorithm to Deep RL is by Jiang et al. (2021b), who propose the Windowed Emphatic TD(λ) (WETD) algorithm that measures the emphatic trace in temporal window of n steps. The influence of an action in WETD is the same as for any other ET, but the trace itself is different, and measures the amount of bootstrapping of the current estimate. ETDs provide an additional mechanism to re-weigh updates, the interest function $i : \mathcal{S} \to [0, \infty)$. By emphasising or de-emphasising the interest on a state, the interest function can be a helpful tool to encode the influence of the actions that had led to that state. Because hand-crafting an interest function requires to important human effort, and the result might be suboptimal,Klissarov et al. (2022) proposes a method to learn and adapt the interest function at each update using meta-gradients. Improvements on both discrete control, such as ALE, and on continuous control problems, such as MuJoCo (Todorov et al., 2012), suggests that the interest function can be helpful to assign credit faster and more accurately.

Selective Credit Assignment (SCA) (Chelu et al., 2022) generalises the idea of re-weighting TD updates to a generic weighting function.

6.1.3 Assigning credit to temporally extended actions

The third and last subset of methods in this category assigns credit to temporally extended actions rather than an action primitive.

The option framework (Sutton et al., 1999; Precup, 2000b) stems from the intuition that it is often convenient to engage into strategical choices at a higher level of abstractions. In fact, *options* (or *skills* in

another branch of literature (Haarnoja et al., 2017; Eysenbach et al., 2018)) generalise the concept of action. An option represents temporally extended courses of actions that an agent can select, and assign credit to, to produce a specific behaviour. Formally, an option is a triple $(\pi, \beta, \mathcal{S}^\pi)$ where π is a policy, $\beta : \mathcal{H} \to \mathbb{B}$ is a *termination condition function* indicating when to cease using the option, and $\mathcal{S}^\pi \subset \mathcal{S}$ is an *initiation set* that determines when the option is available if $s \in \mathcal{S}^\pi$ or not available for use otherwise. The initiation set is often relaxed to the whole state space and $\mathcal{S}^\pi = \mathcal{S}$. For example, in a key-to-door environment, such as MiniGrid (Chevalier-Boisvert et al., 2018) or MiniHack (Samvelyan et al., 2021) the agent might select the option *pick up the key*, followed by *open the door*. Each of this macro-action requires a policy to be executed. For example, *pick up the key* requires to select the actions that lead to have key in front before grabbing it. We refer to Sutton et al. (1999, Section 2) for more details on the framework and the execution of an option in an MDP.

Yet, the biggest obstacle to learning options is how to choose the option set. The majority of works has focused on finding *sub-goals* (Liu et al., 2022) and learning sub-policies to achieve them. Often in literature these or options themselves are pre-specified, which is inflexible as they have to be specified for each single task. For this reason, most advancements of the option framework to Deep RL focus on how to discover the options set from experimental data. Overall, with the option framework, credit is assigned at two levels: at the level of the sub-policy (often referred to as intra-option level) and at the extra-option level, that is to choose which option to follow. We review works about learning options next, and dedicate a separate section to auxiliary goal-conditioning in Section 6.3.

The option-critic architecture (Bacon et al., 2017) scales options to Deep RL and mirrors the actor-critic architecture but with options rather than actions. The option-critic architecture allows to learn both at the intra-option level together with the corresponding termination function, and the policy over them simultaneously. The option executes using the *call-and-return* model. Starting from a state s, the agent picks an option ω according to its policy over options π_Ω. This option then determines the primitive action selection process through the intra-policy π_ω until the option termination function β signals to stop. Learning options, and assigning credit to its actions, is then possible using the *intra-option policy gradient* and the *termination gradient* theorems (Bacon et al., 2017), which define the gradient (thus the corresponding update) for all three elements of the learning process: the option $\omega \in \Omega$, their termination function $\beta(s)$ and the policy over options π_Ω. Here, the context is a state $s \in \mathcal{S}$, the actions to assign credit to are both the intra-option action $a \in \mathcal{A}$ and the option $\omega \in \Omega$, and the goal is to maximise the return. Overall, learning with the option-critic architecture does not require a special training methodology but allows any method used for actor critics.

Hierarchical option-critics (Riemer et al., 2018) extend the previous results from learning options at only two levels (intra and extra options) to learning options at multiple hierarchical levels of resolution. However, the *hierarchical option-critic* architecture only generalises to a fixed number of hierarchical levels, which cannot be changed during or after training. Since Riemer et al. (2018) generalises Bacon et al. (2017), they propose a generalisation of both the *intra-option policy gradient* and the *termination gradient*.

Flexible option learning (Klissarov & Precup, 2021) improves on the previous methods by assigning credit to all options *simultaneously*, rather than a single option at a time – the one currently used – while learning in hierarchical settings. The main contribution of the work to discovering/learning options is the theoretical formulation that makes the above possible. The intuition stems from the way importance sampling (Hesterberg, 1995; Sutton et al., 2014; Precup, 2000a) re-weighs off-policy actions using the probabilities of the action under the behavioural and that under the target policy. Credit is then assigned proportionally to the probability of choosing an option that has not been taken.

Summary. Overall, the methods that use time as a heuristic directly stem from transferring RL concepts to Deep RL, and represent a baseline for the methods that we review in the next sections. These methods use either q-values or advantages as a measure of action influence by actively interacting with an environment (see Section B). Time contiguity is the strong bias that distinguishes them from others. Recency is used as a proxy for credit and a substitute for the causal strength of the relationship between actions and outcomes. While this is a reasonable assumption in many cases and allows solving quite complex problems, this is

not always true. In fact, one of the reasons for which the CAP has not taken-off until recent years is that environments were quite simple to solve from the CA point of view. This was necessary because the CAP would have overcast the others and it wouldn't have been possible to solve them. Indeed, CA methods do not usually shine in classical benchmark but they also do not degrade learning.

Especially in conditions of delayed effects, either of a hierarchical nature of in the case (key-door) or plain temporal delay (distractors), these methods are not the best choice, either because they do not scale well to Deep RL, or because they don't deal with the challenge directly. They are also not designed to handle the sparsity of the action influence and do not perform as well as those that address these challenges directly. Except AL, which uses the action-gap as a proxy for credit, these methods do not incentivise the agent to find multiple pathways to the same goal. Today, the research on these methods is not very active, as they have been superseded by methods that address the challenges of CAP directly.

6.2 Decomposing return contributions

A line of research that aims to overcome the limitation of time-based methods in addressing the delayed effect challenge focuses on decomposing returns into per-timestep contributions. These methods are heavily based on the idea of *reward shaping* (Ng et al., 1999), and often construct a dual decision problem in which the future expected reward is 0 because there are no delayed rewards. This reward function then acts as a measure of action influence. They interpret the CAP as a *redistribution* problem. Given an observed return at termination, the value of each action in the trajectory is re-distributed to the time-steps that it influenced based on what already happened, and not on what will happen, such as in the case of forward methods. While introducing a new mechanism to learn how to assign credit, these methods still use TD errors, $\delta_t = q^\pi(s_t, a_t) - q^\pi(s_{t-1}, a_{t-1})$, and an action is as creditable as the difference in expected returns between two contiguous time steps. We now review the main methods in this category.

Temporal Value Transport (TVT) (Hung et al., 2019) uses an external memory system to contain the loss of action influence due to time. The memory mechanism is based on the Differentiable Neural Computer (DNC) (Grefenstette et al., 2015; Graves et al., 2016), a neural network then reads and writes events to an external memory matrix. To write, state-action-reward triples are projected to a lower dimensional space, and processed by the DNC. During training, this works as a trigger: when a specific state-action pair is read from memory, it is associated with the current one, transporting the value – credit – from the present to the remote state. To read, the state-action-reward is reconstructed from the latent code. During inference, this act as a proxy for credit, and by pointing to past state-action-reward triple that is highly correlated with the current return, it measures the influence of the action.

Return Decomposition for Delayer Rewards (RUDDER) (Arjona-Medina et al., 2019) stems from the intuition that, if we can construct a reward function that *redistributes* the rewards collected in a trajectory such that the expected future reward is zero, we obtain an instantaneous signal that immediately informs the agent about future rewards. In practice, a function $g(s, a)$ outputs the sum of discounter rewards of a trajectory d, including the past, present and future rewards. The difference between its output at two consecutive time steps represents the influence of the action.

$$K(c, a, g) = g(s_{t-1}, a_{t-1}) - g(s_t, a_t) = R^*(s, a), \tag{20}$$

where $R^*(s, a)$ is the reward function of $\mathcal{M}^*$. The context c is a history $h = \{o_t, a_t, r_t : 0 \leq t \leq T\}$ from the assigned MDP, the action is an action from the trajectory $a \in h$, and the goal is the maximum expected return of an optimal policy $g = z^* \in \mathbb{R}$.

Self-Attentional Credit Assignment for Transfer (SECRET) (Ferret et al., 2021a) uses a causal Transformer-like architecture (Vaswani et al., 2017) with a self-attention mechanism (Lin et al., 2017) in the standalone supervised task of reconstructing the sequence of rewards from observations and actions. It then views attention weights over past state-action pairs as credit for the generated rewards. This was shown to help with long-term CA in a way that transfers to novel tasks when trained over a distribution of tasks. We

can write its measure of action influence as follows:

$$K(c, a, g) = \sum_{t=1}^{T} \mathbb{1}\{S_t = s, A_t = a\} \sum_{i=t}^{T} \alpha_{t \leftarrow i} R(s_i, a_i). \tag{21}$$

Here, $\alpha_{t \leftarrow i}$ is the attention weight on (o_i, a_i) when predicting the reward r_j. Also here, the context is a history h, the action is an action from the trajectory $a \in h$, and the goal is the maximum expected return of an optimal policy $g = z^* \in \mathbb{R}$.

Randomised Return Decomposition (RRD) (Ren et al., 2022) advances the idea of return decomposition presented by Arjona-Medina et al. (2019) further. The method assumes that a small set of subsequences that compose a trajectory is responsible for the terminal reward (or the return). A reward model is then trained to predict the episodic return, given the subset of transitions that are randomly sampled from the trajectory. Because this method proposes the same formulation as Arjona-Medina et al. (2019), its action influence is $R^*(s, a)$, with the difference that Randomised Return Decomposition (RRD) directly optimises for the reward function.

Synthetic returns (SR) (Raposo et al., 2021) assume only one state-action to be responsible for the terminal reward. They propose a form of state pairs association where the earlier state (the *operant*) is a leading indicator of the reward obtained in the later one (the *reinforcer*). The association model is learned using a form of episodic memory. Each entry in the memory buffer, which holds the states visited in the current episode, is associated with a reward – the *synthetic* reward – via supervised learning. At training time, this allows propagating credit *directly* from the reinforcer to the operant at-a-distance: without local temporal difference. At inference time, when this reward model is accurately learned, each time the operant is observed, the synthetic reward model spikes, indicating a creditable state-action pair. Here the synthetic reward acts as a measure of causal influence, and we write:

$$K(c, a, g) = q^{\pi}(s, a) + c(s). \tag{22}$$

Here $c(s)$ is the synthetic reward function and it is trained with value regression on the loss $||r_t - g(s_t) \sum_{k=0}^{t-1} c(s_t) - b(s_t)||^2$, where $g(s_t)$ and $b(s_t)$ are auxiliary neural networks optimised together with c. As for Arjona-Medina et al. (2019), the context c is a history h from the assigned MDP, the action is an action from the trajectory $a \in h$, and the goal is the maximum expected return of an optimal policy $g = z^* \in \mathbb{R}$. This method is, however, stable only within a narrow range of hyperparameters and assumes that only one single action is to be credited.

Summary. This set of work arises from the direct interpretation of the CAP as a redistribution problem, and they are the first to report this connection explicitly. These methods assign credit *backward*, that is, based on what has already happened, rather than on what they predict will happen, as in the case of forward methods. Despite their measure of action influence not being a satisfactory quantification of credit (see Section 4.3), the resulting methods are explicitly designed to address the delayed effect challenge, and therefore perform more robustly in tasks that require long-term CA. One key drawback of these methods is their lack of a foundational theory. Today, research to improve on the depth of CA is still ongoing, with the temporal coherence of behaviour over long time-spans, which humans conceptualise as *strategies*, being the main target. These models are still slow to learn and often unstable to different environments and hyperparameters.

6.3 Conditioning on a predefined set of goals

If in the previous section we saw advancements on the distributivity aspect, the methods in this category are the first methods to evaluate actions for their ability to achieve multiple goals explicitly. They do so by conditioning the value function on a goal and then using the resulting value function to evaluate actions. The intuition behind them is that the agent's knowledge about the future can be decomposed into more elementary associations between states and goals. What distinguishes these methods from the ones that follow is that the set of goals they consider is objective and thus predefined, and the agent is not allowed

to choose a subjective one and learn from it. We now describe the two most influential methods in this category.

General Value Functions (GVFs) (Sutton et al., 2011) stem from the idea that knowledge about the world can be expressed in the form of predictions and decomposed into independent ones. These predictions can then be organised hierarchically to solve more complex problems. While GVFs carry many modifications to the canonical value, we focus on its goal-conditioning for the purpose of this review, which is also its foundational idea. As described in Section 4.5, GVFs conditions the action value on a goal to express the expected return with respect to the reward function that the goal induces. In their original formulation (Sutton et al., 2011), GVFs are a set of value functions, one for each goal. The goal is any object in a predefined goal set of MDP states $g \in \mathcal{S}$, and the resulting measure of action influence is the following:

$$K(c, a, g) = q^{\pi}(s, a, g), \tag{23}$$

that is the q-function with respect to the goal-conditioned reward function $R(s, a, g)$, which is 0 everywhere, and 1 when the goal is achieved $\psi(d) = g$. Because GVFs evaluate an action for what it is going to happen in the future, GVFs are forward methods, and interpret the CAP as a prediction problem, "What is the expected return of this action, given that I am going to achieve this goal?".

Universal Value Functions Approximators (UVFAs) (Schaul et al., 2015a) scale-up the idea of GVFs to a large set of goals, by using a single value function to amortise all of them. One major benefit of UVFAs over GVFs is that they are readily applicable to Deep RL by simply adding the goal as an input to the value function approximator. This allows the agent to learn end-to-end with bootstrapping and allows for exploiting a shared prediction structure across different states and goals. Since they derive from GVFs, UVFA share most of their characteristics. The context is an MDP state $s \in \mathcal{S}$; the goal is still any object in a predefined goal set of states, $g \in \mathcal{S}$, and the credit of an action is the expected return of the reward function induced by the goal (see Equation (23)).

Summary. Both GVFs and UVFAs are forward methods that condition the value function on a goal to express the expected return of the reward function that the goal induces. Their interpretation of credit is still linked to the idea of temporal contiguity described in Section 6.1, and they still suffer from the same drawbacks and limitations. In particular, they do not explicitly handle the delayed effect challenge, and their performance in the corresponding tasks suffers the same correlation problems as the ones described in Section 6.1. However, by conditioning the value function on a goal, they provide a way to extract some signal from the environment even when the action influence is low. As described in Section 4.2, using the goal as an explicit input to the assignment allows to: *(a)* maintain knowledge about multiple goals at the same time and *(b)* to be aware of the objective, which in turn sets out to eventually build an internal metric describing the distance from achieving the goal. For example, if the agent's goal is to achieve a minimum return of 1,it becomes possible to verify if and when that is achieved. This is not a possibility when goals are implicit, for example, to maximise the return. Which is the maximum return? Without an answer to the question it is hard to verify that the goal is achieved. Finally, as for the previous categories, most of these methods are value-based methods with a greedy actor.

6.4 Conditioning in hindsight

The methods in this category are characterised by the idea of re-evaluating the action influence according to what the agent achieved, rather than what it was supposed to achieve. They exploit the richness of the temporal stream and flexibility of the definition of outcomes to produce a higher number of decision-outcome pairs, which results in a higher number of associations and overall a denser signal to learn credit for. To produce more associations, their credit formulation is inherently goal-explicit, and they learn credit in hindsight. However, unlike goal-conditional values (Section 6.3), they do not act on a pre-defined objective set of goals. Rather, they collect a trajectory and re-examine it with a different goal in mind. The goal they consider is an outcome in the trajectory just collected, and therefore the success distribution, albeit still sparse when considering a single goal, is denser when considering all possible goals, allowing for a richer learning signal. In practice, this translates into additional training data for the neural network that learns

the credit model. For these reasons, these methods work backward and adopt the interpretation of the CAP as a redistribution problem. Notice, however, that there is a difference between *backward* and *hindsight*. The former refers to the fact that the credit is assigned based on what happened in the past after the action has been taken. The latter refers to the fact that the interactive experience is re-purposed: while a goal was the original intent of the agent, credit is assigned to one that is actually experienced in the trajectory.

We separate the methods in this category into three subgroups. Those that re-label past experience under a different perspective, such as achieving a different goal than the one the agent started; this increases the amount of data to learn. Those that condition the action evaluation with statistics of the future during training, which becomes an explicit performance request at inference time. Those that use hindsight to expose actions irrelevant to achieving the goal via counterfactual reasoning.

6.4.1 Relabelling experience

Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) stems from the problem of learning in sparse rewards environments, which is an example of low action influence. The method develops on the following intuition: despite a trajectory not being optimal, there is some useful information in that experience to enhance the learning process. Hindsight Experience Replay (HER) brings together the UVFA approach to assign credit and the experience replay technique (Lin, 1992) to re-examine trajectories. After collecting a set of trajectories from the environment, the agent stores not only the return for the goal that was originally pursued but also for a subset of other pre-defined goals. This is usually described as a process of relabelling the experience with respect to different goals. We refer to this process of re-examining a trajectory collected with a prior goal in mind and evaluating it according to the actually realised outcome as *hindsight conditioning*, which is also the main innovation that HER brings to the CAP. Notice that the original goal is important because the trajectory is collected with a policy that aims to maximise the return for that specific goal. However, in HER the goal set is still predefined, which is a key insight exploited by other methods, such as Hindsight Credit Assignment (HCA) and Upside-Down RL (UDRL) to increase the autonomy of the agent. HER uses the goal-conditioned q-values described in Section 6.3 to measure action influence:

$$K(c, a, g) = q^{\pi}(s, a, g). \tag{24}$$

Here the context c is a history $h = \{o_t, a_t, r_t : 0 \leq t \leq T\}$ from the assigned MDP, the action is an action from the trajectory $a \in h$, and the outcome is the final state of the trajectory s_T.

Hindsight Policy Gradient (HPG) (Rauber et al., 2019) transfers the findings of HER to policy gradient setting. Since HER is limited to off-policy learning with experience replay (Lin, 1992), this effectively allows to extend the concept of hindsight to iterative settings. Instead of updating the policy based on the actual reward received, Hindsight Policy Gradient (HPG) updates the policy based on the hindsight reward, which is calculated based on the new goals that were defined using HER. The main difference with HER is that in HPGs, both the critic and the actor are conditioned on the additional goal. This results in a goal-conditioned policy $\pi(A|S = s, G = g)$, describing the probability of taking an action, given the current state and a realised outcome. The action influence is the advantage formulation of the hindsight policy gradients:

$$K(c, a, g) = q^{\pi}(s, a, g) - v^{\pi}(s, g), \tag{25}$$

where $q^{\pi}(s, a, g)$ and $v^{\pi}(s, g)$ are the goal-conditioned value functions. Here the context c is a history $h = \{o_t, a_t, r_t : 0 \leq t \leq T\}$, the goal is arbitrarily sampled from a goal set, $g \in \mathcal{G}$. Like HER, HPG is tailored to tasks with low action influence and it is shown to be effective in sparse reward settings. Overall, HER and HPG are the first completed work to talk about *hindsight* as the re-examination of outcomes for CA. Their solution is not particularly interesting for the CAP as they do not cast their problem as a connect the finding to the CAP explicitly. However, they are key precursors of the methods that we review next, which instead provide novel and reusable developments for CAP specifically.

6.4.2 Conditioning on the future

Hindsight Credit Assignment (HCA) Traditional reinforcement learning algorithms often struggle with credit assignment as they rely solely on foresight. These methods operate under the assumption that we lack knowledge of what occurs beyond a given time step, making accurate credit assignment challenging, especially in intricate environments. (Harutyunyan et al., 2019), on the other hand, centres on utilising hindsight information, acknowledging that credit assignment and learning typically take place after the agent completes its current trajectory. This approach enables us to leverage this additional data to refine the learning of critical variables necessary for credit assignment.

(Harutyunyan et al., 2019) introduces a new family of algorithms known as "Hindsight Credit Assignment" (HCA). HCA algorithms explicitly assign credit to past actions based on the likelihood of those actions leading to the observed outcome. This is achieved by comparing a learned *hindsight distribution* over actions, conditioned by a future state or return, with the policy that generated the trajectory.

More precisely, the *hindsight distribution*, $h(a|s_t, \pi, g)$ is the likelihood of an action a, given the outcome g experienced in the trajectory $d \sim \mathbb{P}_{\mu,\pi}(D|S_0 = s, a_t \sim \pi)$. In practice, Harutyunyan et al. (2019) consider two classes of outcomes: states and returns. We refer to the algorithms that derive from these two classes of goals as *state-HCA* and *return-HCA*. For state-HCA, the context c is the current state s_t at time t; the outcome is a future state in the trajectory $s_{t'} \in d$ where $t' > t$; the credit is the ratio between the state-conditional hindsight distribution and the policy $\frac{h_t(a|s_t,s'_t)}{\pi(a|s_t)}$. For return-HCA, the context c is the current state s_t at time t; the outcome is the observed return Z_t; the credit is the ratio between the return-conditional hindsight distribution and the policy $1 - \frac{\pi(a|s_t)}{h_t(a|s_t,Z_t)}$.

For example, return-HCA measures the influence of an action with the *hindsight advantage* described in Section 4:

$$K(c, a, g) = 1 - \frac{\pi(A_t|S_t = s_t)}{\mathbb{P}_{\mu,\pi}(A_t|S_t = s_t, Z_t = z_t)} z_t. \tag{26}$$

The resulting ratio provides a measure of how crucial a particular action was in achieving the outcome. A ratio deviating further from 1 indicates a greater impact (positive or negative) of that action on the outcome.

To compute the *hindsight distribution*, HCA algorithms employ a technique related to importance sampling. Importance sampling estimates the expected value of a function under one distribution (the *hindsight distribution*) using samples from another distribution (the policy distribution). In the context of HCA, importance sampling weights are determined based on the likelihood of the agent taking each action in the trajectory, given the hindsight state compared to the likelihood of the policy for that same action. Once the hindsight distribution is computed, HCA algorithms can be used to update the agent's policy and value function. One approach involves using the hindsight distribution to reweight the agent's experience. This means the agent will learn more from actions that were more likely to have contributed to the observed outcome.

Besides advancing the idea of hindsight, (Harutyunyan et al., 2019) carries one novelty:the possibility to drop the typical policy evaluation settings, where the goal is to learn a value function by the repeated application of the Bellman expectation backup. Instead, action values are defined as a measure of the likelihood that the action and the outcome appear together in the trajectory, and are a precursor of the sequence modelling techniques described in the next section (Section 6.5).

Upside-Down RL (UDRL) (Schmidhuber, 2019; Srivastava et al., 2019; Ashley et al., 2022; Štrupl et al., 2022) is another implementation of the idea to condition on the properties of the future. The intuition behind UDRL is that rather than conditioning returns on actions, which is the case of the methods in Section 6.1, we can invert the dependency and condition actions on returns instead. This allows using returns as input, and inferring the action distribution that would achieve that return. The action distribution is approximated using a neural network, the *behaviour policy*, that is trained via maximum likelihood estimation using trajectories collected online from the environment. In UDRL the context is a completed trajectory d; the outcome is a command that achieves the return Z_k in $H = T - k$ time-steps, which we denote as $g = (Z_k, H)$; the credit of an action a is its probability according to the behaviour function, $\pi(a|s, g)$. In

addition to HCA, UDRL also conditions the return to be achieved in a specific time span. The goal that results from achieving a desired return in a specific time span is called *command* (Schmidhuber, 2019).

Posterior Policy Gradients (PPGs) (Nota et al., 2021) further the idea of hindsight to provide lower-variance, future-conditioned baselines for policy gradient methods. At the base of PPG there is a novel value estimator, the PVF. The intuition behind PVFs is that in POMDPs the state value is not a valid baseline because the true state is hidden from the agent, and the observation cannot provide as a sufficient statistic for the return. However, after a full episode, the agent has more information to calculate a better, *a posteriori* guess of the state value at earlier states in the trajectory. Nota et al. (2021) refers to the family of possible *a posteriori* estimations of the state value as the PVF. Formally, a PVF decomposes a state into its current observation o_t, and some hidden state that is not observable and typically unknown b_t. The value of a state can then be written as the expected observation-action value function over the possible non-observable states $u_T \in \mathcal{U} = \mathbb{R}^d$. The action influence of a PPG is quantified by the expression:

$$K(c, a, g) = \mathop{\mathbb{E}}_{u \in \mathcal{U}} \left[\mathbb{P}(u_t = u | h_t) v(o_t, u_t) \right]. \tag{27}$$

Here, the context is a history h_t, the action is the current action and the goal is the return distribution Z_t. In practice, PVF advances HCA by learning which statistics of the trajectory, $\psi(d)$, are useful to assign credit, rather than specifying it objectively as a state or a return.

Policy Gradient Incorporating the Future (PGIF) (Venuto et al., 2022) implements the same idea of PVFs, but proposes a different method to learn the posterior. They propose an information bottleneck (a variational autoencoder) between the prior and the posterior value function to approximate the posterior. This encourages the agent to learn only the useful non-observable features. Finally, they theoretically link their formulation to the idea of teacher forcing (Williams & Zipser, 1989; Lamb et al., 2016; Goyal et al., 2017) in recurrent neural networks.

6.4.3 Exposing irrelevant factors

Future-Conditional Policy Gradient (FC-PG) For being data efficient, credit assignment methods need to disentangle the effects of a given action of the agent from the effects of external factors and subsequent actions. External factors in reinforcement learning are any factors that affect the state of the environment or the agent's reward, but are outside of the agent's control. This can include things like the actions of other agents in the environment, changes in the environment state due to natural processes or events. These factors can make credit assignment difficult because they can obscure the relationship between the agent's actions and its rewards.

Mesnard et al. (2021) proposes to get inspiration from the counterfactuals from causality theory to improve credit assignment in model-free reinforcement learning. The key idea is to condition value functions on future events, and learn to extract relevant information from a trajectory. Relevant information here corresponds to all information that is predictive of the return while being independent of the agent's action at time t. This allows the agent to separate the effect of its own actions, *the skills*, from the effect of external factors and subsequent actions, the *luck*, which will enable refined credit assignment and therefore faster and more stable learning.

It shows that these algorithms are provably lower variance than vanilla policy gradient, and develops valid, practical variants that avoid the potential bias from conditioning on future information. One variant explicitly tries to remove information from the hindsight conditioning that depends on the current action while the second variant avoids the potential bias from conditioning on future information thanks to a technique related to important sampling. Counterfactual Credit Assignment (CCA) currently provides some of the best results on the CAP.

Counterfactually-Guided Policy Search (CGPS) (Buesing et al., 2019) is a precursor of Mesnard et al. (2021) in a model-based setting, where the hindsight statistic is known *a priori*, rather than learned from experience. These are represented as a Structural Causal Model (SCM) and actual experience is combined with counterfactual queries to perform off-policy evaluation.

Summary. While this line of research brings many independent novelties, the most relevant for the scope of this section is the idea of hindsight conditioning, which can be summarised by the intuition to revisit past estimations with additional information about the future. The evidence provided by these studies suggests that hindsight conditioning provides great benefit in terms of CA, while the only requirement on the overall RL strategy is to employ an actor-critic algorithm, which we consider a mild assumption. Overall, learning in hindsight improves on delayed effects, especially those induced by the hierarchical structure of the decision process, despite not targeting it directly. The methods improve the ability to solve decision problems where the action influence is overall very low. Finally, some of these methods (Mesnard et al., 2021; Buesing et al., 2019) also incentivise the discovery of multiple pathways to the same goal, by identifying decisions that are irrelevant to the outcome, resulting in the fact that any of them can be taken without affecting the outcome.

6.5 Modelling transitions as sequences

The methods in this category are based on the observation that RL can be seen as a sequence modelling problem. Their main idea is to transfer the successes of sequence modelling in Natural Language Processing (NLP) to improve RL. On a high level they all share the same assumption that a sequence in RL is a sequence of transitions (s, a, r), and they differ in either how to model the sequence, the problem they solve, or the specific method they transfer from NLP. Another common characteristic is that they often learn from offline datasets, which is a limitation that is not shared by the other methods in this section.

Trajectory Transformers (TTs) (Janner et al., 2021) implement a decoder-only (Radford et al., 2018; 2019) transformer (Vaswani et al., 2017) to model the sequence of transitions. TTs learn from an observational stream of data, composed of expert demonstrations resulting in an offline RL training protocol. The main idea of TTs is to model the next token in the sequence, which is a composed by next state, the next action, and the resulting reward. This enables planning, which TTs exploit to plan via beam search. Notice that, for any of these paradigms, if the sequence model is autoregressive – the next prediction depends only on the past history, but since a full episode is available, the future-conditioned probabilities are still well-defined, and also TTs can condition on the future. In TTs the action influence is the product between the action probability according to the demonstration dataset and its q-value:

$$K(c, a, g) = \mathbb{P}_\theta(A_t = a_t | Z_t = z_t) q^\pi(s, a). \tag{28}$$

Here, the context c is an MDP state $c = s \in \mathcal{S}$, the action is arbitrarily selected, and the goal is the return distribution $\mathbb{P}(Z)$.

Decision Transformers (DTs) (Chen et al., 2021) proceed on the same lines as TTs but ground the problem in learning, rather than planning. DTs interpret a sequence as a list of (s_t, a_t, Z_t) triples, where Z_t is the discounted sum of rewards from t to the end of the episode. They then use a decoder-only transformer to learn a model of the actor that takes the current state and the return as input and outputs a distribution over actions. In addition, they optionally learn a model of the critic as well, which takes the current state and each action in the distribution to output the value of each action. The sequences are sampled from expert or semi-expert demonstrations, and the model is trained to maximise the likelihood of the actions taken by the expert. From the perspective of CA, TTs are equivalent to DTs, and they share the same limitation in that they struggle to assign credit accurately to experience beyond that of the offline dataset. Furthermore, like HCA (Harutyunyan et al., 2019), DTs bring more than one novelty to RL. Besides modelling the likelihood of the next token, they also use returns as input to the model, resulting in a form of future conditioning. However, for CA and this section, we are only interested in their idea of sequence modelling, and we will not discuss the other novelties.

Online Decision Transformers (ODTs) (Zheng et al., 2022) extend DT to fine-tuning from experience collected online, to overcome the limitations of learning from offline data only. To adapt the notion of exploration to sequence modelling, they propose to pre-train on the offline dataset, and then employ maximum entropy exploration and collect data outside the support of the offline dataset. Finally, they learn off-policy using a replay buffer composed of full trajectories. From the perspective of CAP, the only difference with DTs is that they learn from a different source of experience. While DTs collect contextual data from a static

dataset of demonstrations, ODTs do it by interacting with an environment (see also Appendix B) at the stage of fine-tuning. Fine-tuning online allows to increase the scale of the data available to learn from, and later evidence (Lee et al., 2022) shows that this brings to better performance, and to suggests promising scaling laws.

Generalised Decision Transformers (GDTs) (Furuta et al., 2022) generalise DT to model quantities beyond the return, the same way FC-PG generalises HCA to model quantities beyond the states and returns. Furuta et al. (2022) introduce the idea of *hindsight information*, defined as the information content of a function of the trajectory, that is, the self-information of an outcome. In theory, this formulation is similar to that of PVF, and the two threads of research arrive at similar conclusions from different perspectives. However, in practice, this degenerates to using states or returns. To measure action influence, the difference with its DT predecessor is that the goal g us now self-information of an arbitrary goal.

Summary. Sequence modelling in RL transfers the advances of sequence modelling in NLP to the RL setting. The main idea for the purpose of CA is to measure credit by estimating the probability of the next action (or the next token), conditioned on the context and the goal, according to an offline dataset of expert trajectories (Chen et al., 2021; Janner et al., 2021), with online fine-tuning (Lee et al., 2022). Their development has a similar pattern to that of hindsight methods and progressively generalises to more complex settings, such as online learning (Zheng et al., 2022) and more general outcomes (Furuta et al., 2022). Overall, they are a promising direction for CA, especially for their ability to scale to large datasets. It is not clear how these methods position with respect to the CA challenges described in Section 5, for the lack of experimentation on tasks that explicitly stress the agent's ability to assign credit. However, for their vicinity to future-conditioned methods, they bear some of the same advantages and also share some limitations. In particular, for their ability to define outcomes in hindsight, regardless of an objective learning signal, they bode well in tasks with low action influence.

6.6 Planning and learning backwards

The methods in this category extend CA to potential predecessor decisions that have not been taken, but could have led to the same outcome (Chelu et al., 2020). The main intuition behind these methods is that, in environments with low action influence, influential actions are rare, and when a goal is achieved the agent should use that event to extract as much information as possible to assign credit to relevant decisions. We divide the section into two major sub-categories, depending on whether the agent identifies predecessor states by planning with an inverse model, or by learning relevant statistics without it.

6.6.1 Planning backwards

Recall traces (Goyal et al., 2019) combine model-free updates from Section 6.1 with learning a backward model of the environment. A backward model $\mu^{-1}(S_{t-1}|S_t = s, A_{t-1} = a)$ describes the probability of a state S_{t-1} being the predecessor of another state S_t, given that the backward action $A_{t=1}$ was taken. The backward action is sampled from a *backward policy*, $\pi_b(a_{t-1}|s_t)$, which predicts the previous action, and a *backward dynamics*. By autoregressively sampling from the backward policy and dynamics, the agent can cross the MDP backwards, starting from a final state, S_T, up until a starting state, S_0 to produce a new history, *recall trace*. This allows the agent to collect experience that always leads to a certain state, s_T, but that does so from different starting points, discovering multiple pathways to the same goal. By performing behaviour cloning on the recall trace, Formally, the agent alternates between steps of GPI via model-free updates and steps of behaviour cloning on trajectories collected via the backward model (trajectories are reversed to match the forward arrow of time before cloning). This is a key step towards solving the CAP as it allows propagating credit to decisions that have not been taken but could have led to the same outcome without interacting with the environment directly. Recall-traces measure the influence of an action by its q-value, but differ from any other method using the same action influence because the contextual data is produced via backward crossing. The goal is the expected return.

Forward-Backward RL (FBRL) Edwards et al. (2018) is a concurrent work to Goyal et al. (2019) that also learns a backward model to train on imagined reversed trajectories, and provides further evidence of the method's benefits.

Time-Reversal as Self Supervision (TRASS) (Nair et al., 2020) follows on the same idea of learning a backward state-transition dynamics model. However, compared to FBRL and recall traces, TRASS require the ability to access a simulator of the environment that can reset to a subset of high reward states. This is a fairly restrictive assumption compared to FBRL and recall traces, which resort to the more canonical GPI to find the states of interest. Overall, Nair et al. (2020) provide additional evidence that, if a goal is achieved, it is beneficial to assign credit to state-action pairs discovered by crossing the MDP backwards from the time the goal is achieved.

Reverse Offline Model-based Imagination (ROMI) (Wang et al., 2021) proceeds on the same line but focuses on offline settings, emphasising that, all else being equal, backward models enable better generalisation than forward ones, as they start directly from high-value states from which the return can only diminish.

Bidirectional Model-based Policy Optimisation (BMPO) (Lai et al., 2020) learns both a forward and a backward model of the state-transition dynamics and proves that BMPO achieves a lower return error bound than pure forward or pure backward methods. Overall, this method allows both to assign credit directly via direct policy search.

The relationship between forward and backward planning is the subject of further investigations (van Hasselt et al., 2019; Chelu et al., 2020). van Hasselt et al. (2019) provides empirical evidence suggesting that assigning credit from hypothetical transitions, that is, when trajectories are sampled from a forward model, improves the overall efficiency in control problems. This remarks the difference between assigning credit in a model-free fashion, where the model is only used to generate fictitious trajectories and assigning credit via search, where the model is used to look-ahead or look-behind. Chelu et al. (2020) and van Hasselt et al. (2019) further show that backward planning provides even greater benefits than forward planning when the state-transition dynamics are stochastic.

6.6.2 Learning predecessors

Expected Eligibility Trace (ET(λ)) (van Hasselt et al., 2021) provide a model-free alternative to backward planning that assigns credit to potential predecessors decisions of the outcome: decisions that have been taken in the past, but have not in the last episode. The main idea is to weight the action value by its expected eligibility trace, that is, the instantaneous trace (see Section 6.1), but in expectation over the random trajectory, defined by the policy and the state-transition dynamics. The Deep RL implementation of ET(λ) considers the expected trace upon the action value representation – usually the last layer of a neural network value approximator. Like for other ETs algorithms, ET(λ) measures action influence using the q-value of the decision and encodes the information of the trace in the parameters of the function approximator. In this case the authors interpret the value network as a composition of a non-linear representation function $\phi(s)$ and a linear value function $v(s_t) = w\top\phi(s)$. The expected trace $e(s) = E\phi(s)$ is then the result of applying a second linear operator E on the representation. $e(s)$ is then trained to minimise the expected ℓ_2 norm between the current estimation of $e(s)$ and the instantaneous trace.

Summary. In this section, we have reviewed the main methods that enhance CA by extending it to decisions that have not been taken, but could have led to the same outcome. The intuition behind them is that, in tasks where the action influence is low, creditable actions are rare findings, and when this happens the agent can use that occurrence to extract as much information as possible. One set of methods does so by learning inverse models of the state-transition dynamics and walking backwards from the outcome. Chelu et al. (2020); van Hasselt et al. (2019) further analyse the conditions in which backward planning is beneficial. Another set of methods exploits the idea of eligibility traces and keeps a measure of the marginal state-action probability to assign credit to actions that could have led to the same outcome. Overall, these

methods are designed to thrive in tasks where the action influence is low. Also, for their ability to start from a high-value state, backward planning methods can find a higher number of optimal transpositions, and therefore provide a less biased estimate of the credit of a state-action pair. On the other hand, none of these methods addresses delayed effects directly, and, albeit not explicitly tested on these, they are not the best choice in those circumstances.

6.7 Meta-learning proxies for credit

Often, control methods show brittleness to the choice of hyperparameters of the RL problem, for example, the number of steps to look-ahead in bootstrapping, what discount factor to use, or meta-parameters specific to the method at hand. How to select these meta-parameters is an accurate balance that depends on the task, the algorithm, and the objective of the agent. Because a policy improvement step is as valuable as accurate is the current estimation of action influence, these issues also transfer to CA. The methods in this category assign credit by meta-learning these meta-parameters, which changes the measure of action influence, for example, by optimising the λ of ET. The nature of these methods is quite different from that of the others we surveyed earlier. For this reason, it is sometimes difficult to analyse using the usual framework, and we present them differently, by describing their main idea, and the way they are implemented in Deep RL.

Meta Gradient (MG) RL (Xu et al., 2018) remarks how different CA methods underlie different choices of the target, and proposes to answer the question: "*Which target results in the best performance?*". The method interprets the target as a parametric, differentiable function that can be used and modified by the agent to guide its behaviour to achieve the highest returns. In particular, *Meta-Gradients* consider the λ-return (Sutton, 1988) target, for it can generalise the choice of many targets (Schulman et al., 2016), and learn its meta-parameters, the bootstrapping parameter λ and the discount factor γ, via online cross validation (Sutton, 1992). In this review, we are interested in their formulation of the evaluation problem with differentiable, non-linear function approximators. After collecting a batch of trajectories, the agent computes the λ-return for each trajectory with the current guess of parameters and meta-parameters. One can then calculate either the gradient of the parameters, or the meta-gradient of the meta-parameters, and perform updates accordingly. In Meta-gradient RL, the action influence is akin to that of the methods based on the ET, but the hyperparameters of the TD(λ) are meta-optimised. Zheng et al. (2018) also propose to learn the target online but they consider the value of the learned target as an intrinsic reward. Since their introduction, meta-gradients have been applied to different meta-parameters.

Flexible Reinforcement Objective Discovered Online (FRODO) (Xu et al., 2020) scales up meta-gradients by parameterising the target with a neural network and learning its meta-parameters online.

Distributional meta-gradient (Yin et al., 2023) combines the idea of meta-gradients with distributional RL (Bellemare et al., 2017), and learn the meta-parameters of the target by meta-learning the distribution of the returns.

Zheng et al. (2020) meta-learn intrinsic rewards across multiple lifetimes, where it is clearer that the intrinsic reward itself is a proxy for credit.

Summary. Overall these methods assign credit to actions by applying canonical RL algorithms to a meta-learning goal. The goal can come in the form of an update target (Xu et al., 2018; Zheng et al., 2018; Xu et al., 2020), a full return distribution (Yin et al., 2023), or a reward function (Zheng et al., 2020). Since these methods are not explicitly designed for CA, it is not clear what their performance is with respect the challenges described in Section 5. However, despite the lack of diagnostic experiments that stress specific aspects of credit assignment, these methods have been shown to perform well in complex RL tasks, such as the ALE (Bellemare et al., 2013).

7 Evaluating credit

The aim of this section is to survey the state of the art in the metrics, the tasks and the evaluation protocols to evaluate a CA algorithm. Like accurate evaluation is fundamental to RL agents to improve their policy, evaluating a method is fundamental to our research to monitor if and how the field is advancing. We discuss the main components of the evaluation procedure, the tasks, the performance metrics and the protocol in the three subsections, respectively. We start with the metrics and protocol altogether, as they should be as agnostic as possible to the considered task.

7.1 Metrics

We categorise the existing metrics to evaluate a CA method in two main classes. The first class uses the metrics already used for control problems. These mostly aim to assess the agent's ability to make optimal decisions, but they do not explicitly measure the accuracy of the action influence. The second class aims to assess the quality of an assignment directly, and usually aggregates metrics over the course of the RL training procedure. We now proceed to describe the two classes of metrics.

7.1.1 Metrics borrowed from control

Bias, variance and contraction rate. The first, intuitive, obvious proxy to assess the quality of credit assignment methods is the performance in suitable *control* problems. Formally, we refer to the bias, variance and contraction rate of the policy improvement operator described by Rowland et al. (2020), which we now recall. For the evaluation operator described in (2), we can specify these quantities as follows. Notice that these metrics are not applicable to all the methods, either some of the variables cannot be accessed or because operators they act on are not formally defined for the method in question.

$$\Gamma = \sup_{s \in \mathcal{S}} \frac{||\mathcal{T}V^{\pi}(s) - \mathcal{T}V'^{\pi}(s)||_{\infty}}{||V^{\pi}(s) - V'^{\pi}(s)||_{\infty}} \tag{29}$$

is the contraction rate and describes, how fast the operator converges to its fixed point, if it does so, and thus how efficient it is. Here $V^{\pi}(s)$ and $V'^{\pi}(s)$ are two estimates of the value of a state.

If $\mathcal{T}$ is contractive, that is, if $\forall \mathcal{T} : \Gamma < 1$, there exist a fixed-point bias of $\mathcal{T}$ is given by:

$$\xi = ||V^{\pi}(s) - V^{*\pi}(s)||_2, \tag{30}$$

where $\hat{V}^{\pi}(s)$ is the true, unique fixed point of $\mathcal{T}$, whose existence is guaranteed by $\Gamma < 1$. For every evaluation operator $\mathcal{T}$ there is an update rule $\Lambda : \mathbb{R}^{|\mathcal{S}|} \times \mathcal{D} \rightarrow \mathbb{R}$ that takes as input a value estimate and a trajectory and outputs an update for the value. Λ has a variance

$$\nu = \mathbb{E}_{\mu,\pi}[||\Lambda[V(s), D] - \mathcal{T}V(s)||_2^2]. \tag{31}$$

These three quantities are usually in a trade-off (Rowland et al., 2020). Indeed, many (if not all) studies on credit assignment (Hung et al., 2019; Mesnard et al., 2021; Ren et al., 2022; Raposo et al., 2021) report the empirical return and its variance. Because the contraction rate is often harder to calculate, an alternative metric is the time-to-performance, which evaluates the number of interactions necessary to reach a given performance. These mostly aim at showing improvement in sample efficiency and/or asymptotical performance. While useful, this is often not enough to assess the quality of credit assignment, as superior returns can be the result of better exploration, better optimisation, better representation learning, luck (as per the environment dynamics' stochasticity) or of a combination of such factors. Nonetheless, when the only difference between two RL algorithms lies in how credit is assigned, and this is not confounded by factors aforementioned, it is generally safe to attribute improvements to superior credit, given that the improvements are statistically significant (Henderson et al., 2018; Agarwal et al., 2021). Notice that this metrics can only be applied to measure of influence that result from the fixed point iterations.

Task completion rate. A related, but more precise set of metrics is the task completion rate. Given a budget of trials, the task completion rate measures the frequency of success, that is the number of times the task was solved over the total number of trials. Considering completion rates instead of bias, variance and trade-off is useful as it alleviates another issue of these performance metrics: there is no distinction between easy-to-optimise rewards and hard-to-optimise rewards. We will illustrate that on the Key-to-Apple-to-Door task. Due to the stochasticity from the Apple phase, it is generally impossible to distinguish performance on apple picking (easy-to-optimise rewards) and door opening (hard-to-optimise rewards, that superior credit assignment methods should make easier to obtain over time). However, notice that this clarity in reporting credit comes at a cost. In fact, these kind of metrics necessitate expert knowledge about the task at hand, but are more precise than performance metrics. They often suffer from the same confounders as performance metrics.

Value error. As the value (resp. action-value) function is at the heart of many credit assignment methods, another proxy to the quality of the credit is the quality of value estimation, which can be estimated from the distribution of TD errors (Andrychowicz et al., 2017; Rauber et al., 2019; Arjona-Medina et al., 2019). A drawback of the expected TD error is that it can be misleading. When an algorithm does not fully converge, for example, because of a sparse reward function, it can happen that the value error is very low. This is because the current policy never visits a state with reward different from zero, and the value function collapses to always return zero. This only applies to CA methods that do not circumvent the value function altogether.

7.1.2 Bespoke metrics for credit assignments

We now review metrics that measure the quality of individual credit assignments, that is how well actions are mapped to corresponding outcomes, or how well outcomes are redistributed to past actions. Usually, these metrics are calculated in hindsight, after outcomes have been observed.

Using knowledge about the causal structure. Given expert knowledge about the causal structure of the task at hand, that is which actions cause which outcomes, one can leverage it and compare it to the corresponding credit assignments, which approximate such cause and effect relationships. We give several examples from the literature. In Delayed Catch, Raposo et al. (2021) look at the estimated credit for actions that lead to catches and the end-of-episode reward that only depends on catches. They do the same on the Atari game Skiing, which is a more complex environment but is similar in the sense that only getting between poles grants rewards at the end of episode. Ferret et al. (2021a) adopt a similar approach and look at the estimated credit given to actions responsible for trigger switches in the Triggers environment, which contribute alone to the end-of-episode reward. Arjona-Medina et al. (2019) look at redistributions of RUDDER on several tasks including the Atari 2600 game Bowling.

Counterfactual simulation. A natural approach, which is nonetheless seldom explored in the literature, is counterfactual simulation. On a high-level, it consists in asking what would have happened if actions that are credited for particular outcomes had been replaced by another action. This is close to the notion of hindsight advantage.

Comparing to actual values of the estimated quantity. This only applies to methods whose credit assignments are mathematically grounded, in the sense that they are the empirical approximations of well-defined quantities. In general, one can leverage extra compute and the ability to reset a simulator to arbitrary states to obtain accurate estimations of the underlying quantity, and compare it to the actual, resource-constrained quantity used as credit assignment.

7.2 Tasks

In what follows, we present environments that we think are most relevant to evaluate credit assignment methods and individual credit assignments. The most significant tasks are those that present all the three challenges to assign credit: delayed rewards, transpositions and sparsity of the influence. This often corre-

sponds to experiments that have reward delay, high marginal entropy of the reward, and partial observability. To benchmark explicit credit assignment methods, we additionally need to be able to recover the ground truth influence of actions w.r.t. given outcomes, or we can use our knowledge of the environment and develop more subjective measures.

7.2.1 Diagnostic tasks

Diagnostic tasks are useful as sanity checks for RL agents and present the advantage to be ran rather quickly, compared to complex environments with visual input that may imply several millions of samples before agents manage to solve the task at hand. Notice that these tasks may not be representative of the performance of the method at scale, but provide a useful signal to diagnose the behaviour of the algorithm in the challenges described in Section 5. Sometimes, the same environment can represent both a diagnostic task and an experiment at scale simply by changing the space of the observations or the action space.

We first present chain-like environments, that can be represented graphically by a chain (environments **a** to **c**), and then a grid-like environment (environment **d**), that has more natural grid representations for both the environment and the state.

a) Aliasing chain. The aliasing chain (introduced in Harutyunyan et al. (2019) as Delayed Effect) is an environment whose outcome depends only on the first action. A series of perceptually aliased and zero-reward states follow this first action and an outcome is observed at the end of the chain ($+1$ or -1 depending on the binary first action).

b) Discounting chain. The discounting chain (Osband et al., 2020) is an environment in which a first action leads to a series of states with inconsequential decisions with a final reward that is either 1 or $1 + \epsilon$, and a variable length. It highlights issues with the discounting horizon.

c) Ambiguous bandit. The ambiguous bandit (Harutyunyan et al., 2019) is a variant of a two-armed bandit problem. The agent is given two actions: one that transitions to a state with a slightly more advantageous Gaussian distribution over rewards with probability $1 - \epsilon$, and another that does so with probability ϵ.

d) Triggers. Triggers (Ferret et al., 2021a) is a family of environments and corresponding discrete control tasks that are suited for the quantitative analysis of the credit assignment abilities of RL algorithms. Each environment is a bounded square-shaped 2D gridworld where the agent collects rewards that are conditioned on the previous activation of all the triggers of the map. Collecting all triggers turns the negative value of rewards into positive and this knowledge can be exploited to assess proper credit assignment: the actions of collecting triggers appear natural to be credited. The environments are procedurally generated: when requesting a new environment, a random layout is drawn according to the input specifications.

7.2.2 Tasks at scale

In the following, we present higher-dimension benchmarks for agents equipped with credit assignment capabilities.

Atari. The Arcade Learning Environment (Bellemare et al., 2013) (ALE) is an emulator in which RL agents compete to reach the highest scores on 56 classic Atari games. We list the ones we deem interesting for temporal credit assignment assessment due to delayed rewards, which were first highlighted by Arjona-Medina et al. (2019). **Bowling**: like in real-life bowling, the agent must throw a bowling ball at pins, while ideally curving the ball so that it can clear all pins in one throw. The agent experiences rewards with a high delay, at the end of all rolls (between 2 and 4 depending on the number of strikes achieved). **Venture**: the agent must enter a room, collect a treasure and shoot monsters. Shooting monsters only give rewards after the treasure was collected, and there is no in-game reward for collecting it. **Seaquest**: the agent controls a submarine and must sink enemy submarines. To reach higher scores, the agent has to additionally rescue divers that only provide reward once the submarine lacks oxygen and surfaces to replenish it. **Solaris**: the

agent controls a spaceship that earns points by hunting enemy spaceships. These shooting phases are followed by the choice of the next zone to explore on a high-level map, which conditions future reward. **Skiing**: the agent controls a skier that has to go between poles while going down the slope. The agent gets no reward until reaching the bottom of the slope, at which time it receives a reward proportional to the pairs of poles it went through, which makes for long-term credit assignment.

VizDoom. VizDoom (Kempka et al., 2016) is a suite of partially observable 3D tasks based on the classical Doom video game, a first-person shooter. As mentioned before, it is an interesting sandbox for credit assignment because it optionally provides high-level information such as labelled game objects, depth as well as a top-view minimap representation; all of which can be used for approximate optimally efficient credit assignment algorithms.

BoxWorld. BoxWorld (Zambaldi et al., 2018) is a family of environments that shares similarities with Triggers, while being more challenging. Environments are also procedurally-generated square-shaped 2D gridworlds with discrete controls. The goal is to reach a gem, which requires going through a series of boxes protected by locks that can only be opened with keys of the same colour while avoiding distractor boxes. The relations between keys and locks can be utilised to assess assigned credit since the completion of the task (as well as intermediate rewards for opening locks) depends on the collection of the right keys.

Sokoban. Sokoban (Racanière et al., 2017) is a family of environments that is similar to the two previous ones. The agent must push boxes to intended positions on the grid while avoiding dead-end situations (for instance, if a block is stuck against walls on two sides, it cannot be moved anymore). While there is no definite criterion to identify decisive actions, actions that lead to dead-ends are known and can be exploited to assess the quality of credit assignment.

DeepMind Lab. DeepMind Lab (Beattie et al., 2016) (DMLab) is a suite of partially observable 3D tasks with rich visual input. We identify several tasks that might be of interest to assess credit assignment capabilities, some of which were used in recent work. **Keys-Doors**: the agent navigates to keys that open doors (identified by their shared colour) so that it can get to an absorbing state represented by a cake. Ferret et al. (2021a) consider a harder variant of the task where collecting keys is not directly rewarded anymore and feedback is delayed until opening doors. **Keys-Apples-Doors**: Hung et al. (2019) consider an extended version of the previous task. The agent still has to collect a key, but after a fixed duration a distractor phase begins in which it can only collect small rewards from apples, and finally the agent must find and open a door with the key it got in the initial phase. To solve the task, the agent has to learn the correlation or causation link between the key and the door, which is made hard because of the extended temporal distance between the two events and of the distractor phase. **Deferred Effects**: the agent navigates between two rooms, the first one of which contains apples that give low rewards, while the other contains cakes that give high rewards but is entirely in the dark. The agent can turn the light on by reaching the switch in the first room, but it gets an immediate negative reward for it. In the end, the most successful policy is to activate the switch regardless of the immediate cost so that a maximum number of cakes can be collected in the second room before the time limit.

7.3 Protocol

Online evaluation. The most standard approach is to evaluate the quality of credit assignment methods and individual credit assignments along the RL training procedure. As the policy changes, the credit assignments change since the effect of actions depends on subsequent actions (which are dictated by the policy). One can dynamically track the quality of credit assignments and that of the credit assignment method using the metrics developed in the previous section. For the credit assignment method, since it requires a dataset of interaction, one can consider using the most trajectories produced by the agent. An advantage of this approach is that it allows evaluating the evolution of the credit assignment quality along the RL training, with an evolving policy and resulting dynamics. Also, since the goal of credit assignment is to help turn feedback into improvements, it makes sense to evaluate it in the context of said improvements. While natural, online evaluation means one has little control over the data distribution of the evaluation. This

is problematic because it is generally hard to disentangle credit quality from the nature of the trajectories it is evaluated on. A corollary is that outcomes that necessitate precise exploration (which can be the outcomes for which agents would benefit most from accurate credit assignment) might not be explored.

Offline evaluation. An alternative is to consider offline evaluation. It requires a dataset of interactions, either collected before or during the RL training. Credit assignments and the credit assignment method then use the parameters learned during the RL training while being evaluated on the offline data. As the policy in the offline data is generally not the latest policy from the online training, offline evaluation is better suited for policy-conditioned credit assignment or (to some extent) trajectory-conditioned credit assignment. Indeed, other forms of credit assignment are specific to a single policy, and evaluating these on data generated from another policy would not be accurate. An important advantage of offline evaluation is that it alleviates the impact of exploration, as one controls the data distribution credit is evaluated on.

8 Closing, discussion and open challenges

The CAP is the problem to approximate the causal influence of an action from a finite amount of experience, and it is of critical importance to deploy RL agents into the real world that are effective, general, safe and interpretable. However, there is a misalignment in the current literature on what credit means in words and how it is formalised. In this survey, we put the basis to reconcile this gap by reviewing the state of the art of the temporal CAP in Deep RL, focusing on three major questions.

8.1 Summary

Overall, we observed tree major fronts of development around the CAP.

The first front concerns the problems of *how to quantify action influence*. In **Section 4** we addressed **Q1.** and analysed the quantities that the existing works use to represent the influence of an action. In **Section 4.1** we unified these measures of action influence in the *assignment class*. In Sections 4.5 and 4.3 we showed that the existing literature agrees on an intuition of credit as causal influence, but that it does not translate that well into mathematics and none of the current quantities are satisfactory measures of causal influence. As a consequence, we proposed a set of principles that we suggest a measure of action influence should respect to represent credit.

The second front aims to address the question of *how to learn an action influence from experience* and concerns the *methods* to assign credit. In **Section 5** we looked at the challenges that arise from learn these measures of action influence and, together with **Section 6**, answered **Q2.**. We first reviewed the most common obstacles to learning already identified in literature and realigned them to our newly developed formalism. We identified three dimensions of an MDP, depth, breadth and density and described pathological conditions on each of them that hinder the CA. In Section 6 we defined a CA as an algorithm whose aim is to approximate a measure of action influence from a finite amount of experience. We categorised methods into those that: *(i)* use temporal contiguity as a proxy for causal influence; *(ii)* decompose the total return into smaller per-timestep contributions; *(iii)* condition the present on information about the future using the idea of hindsight; *(iv)* use sequence modelling and represent action influence as the likelihood of action to follow a state and predict an outcome; *(v)* learn to imagine backward transitions that always start at a key state and propagate back to the state that could generate them; *(vi)* meta-learn action influence measures.

Finally, the third research front deals with *how to evaluate quantities and methods* to assign credit and aims to provide an unbiased estimation of the progress in the field. In **Section 7** we addressed **Q3.** and analysed how current methods evaluate their performance and how we can monitor future advancements. We highlighted that current benchmarks are not fit for the purpose. Diagnostic benchmarks do not isolate the specific CAP challenges identified in Section 5: delayed effects, transpositions and sparsity. We explained that benchmarks at scale often cannot disentangle the CAP from the exploration problem and it becomes hard to understand whether a method is advancing one problem or another.

8.2 Discussion and open challenges

As this survey suggest, the work in the field is now fervent and the number of studies in a bullish trend, with many works showing substantial gains in control problems only by – to the best of our current knowledge – advancing on the CAP alone (Bellemare et al., 2017; van Hasselt et al., 2021; Edwards et al., 2018; Mesnard et al., 2021; 2023). We observed that the take-off of CA research in the broader area of RL research is only recent. Most probably the reason of this only recent take-off is to be found in the hierarchy of problems in the broader RL field. The tasks considered in earlier Deep RL research were simple from the CA point of view because otherwise it would have not been possible to solve them. Using tasks where assigning credit is hard would have – and probably still do, e.g., Küttler et al. (2020) – obfuscate other problems that it was necessary to solve before solving the CAP. For example, adding the CAP on the top of scaling RL to high-dimensional observations (Arulkumaran et al., 2017) or dealing with large action spaces (Dulac-Arnold et al., 2015; van Hasselt & Wiering, 2009) would have, most likely, concealed any evidence of progress for the underlying challenges. This is also why CA methods do not usually shine in classical benchmark (Bellemare et al., 2013) and peer reviews are often hard on these works.

On this background, the CAP still hold open question and there is still much discussion required to consider the problem solved. In particular, the following observations describe our positions with respect to this survey.

Aligning future works to a common problem definition. The lack of a review since its conception (Minsky, 1961) and the rapid advancements produced a fragmented landscape of definitions for action influence, an ambiguity in the meaning of *credit assignment*, a misalignment between the general intuition and its practical quantification, and a general lack of coherence in the principal directions of the works. While this diversity being beneficial for the diversification of the research, it is also detrimental to comparing the same works, as their starting point and their aim only intersect in a few places. Future works aiming to propose a new CA method should clarify these preliminary concepts. Answers to "What is the choice of the measure of action influence? Why the choice? What is the method to learn that influence from experience? How is it evaluated?" would be good a starting point.

Characterising credit. "*What is the* minimum *set of properties that a measure of action influence should respect to inform control? What the more desirable ones?*". This question remains unanswered, with some ideas in Ferret (2022, Chapter 4), and we still need to understand what characterises a proper measure of credit.

Causality. The relationship between CA and causality is underexplored but in a small subset of works (Mesnard et al., 2021; Pitis et al., 2020; Buesing et al., 2019). The literature lacks a clear and complete formalisms that casts the CAP as a problem of causal discovery. Investigating this connection and formalising a measure of action influence that is also a satisfactory measure of causal influence would help better understand the effects of choosing a measure of action influence over another. Overall, we need a better understanding of the connection between CA and causality: what happens when credit is a strict measure of causal influence? How do current algorithms perform with respect to this measure? Can we devise an algorithm that exploits a causal measure of influence?

Optimal credit. Many works refer to *optimal credit* or to *assigning credit optimally* but it is unclear what that formally means. "*When is credit optimal?*" remains unanswered.

Combining benefits from different methods. Method conditioning on the future currently show superior results with respect to methods in other categories. These promising methods includes hindsight (Section 6.4), sequence modelling (Section 6.5) and backward learning and planning methods (Section 6.6). However, while hindsight methods are advancing fast, sequence modelling and backward planning methods are underinvestigated. We need a better understanding of the connection between these two worlds, which could potentially lead to even better ways of assigning credit. Could there be a connection between these methods? What are the effects of combining backward planning methods with more satisfactory measures of influence, for example, with CCA?

Benchmarking. The benchmarks currently used review a CA method (Chevalier-Boisvert et al., 2018; Bellemare et al., 2013; Samvelyan et al., 2021) (see Section 7.2) are often borrowed from *control* problems. This creates two problems. On one hand, these benchmarks cannot isolate the issues caused by each of the challenges reviewed in Section 5. This makes it hard to understand which challenge of the CAP the method is improving, and it is not clear which methods to combine to achieve a unison advancement. On the other hand, to acquire knowledge (the CAP), the underlying set of associations between actions an outcome must be first discovered (see Section 5.4). Because solving control requires to solve both the underlying CAP and exploration problem, it is not always clear if one or the other is responsible for an improvement. On a complementary note, CA methods are often evaluated in actor-critic settings (Harutyunyan et al., 2019; Mesnard et al., 2021), which adds additional layers of complexity. This, together with other accessories unnecessary to validate a new algorithm, can obfuscate the contribution of the credit mechanism to the overall RL success. As a consequence, literature lacks a fair comparison among all the methods and it is not clear how all the methods in Section 6 behave with respect to each other against the same set of benchmarks. Overall, the lack of a comprehensive understanding of the state of the art leads to a poor signal to direct future research. We call for new, community-driven single set of benchmarks that disentangle the CAP from the exploration problem and isolate the challenges described in Section 5. How to disentangle the CAP and the exploration problem? How to isolate each challenge? Shall we evaluate in value-based settings and would the raking between the methods be consistent with an evaluation in actor-critic settings? These questions are still unanswered.

Reproducibility. Many works propose open source code, but experiments are often not reproducible, their code is hard to read, hard to run and hard to understand. Making code public is not enough and cannot be considered open-source if it is not easily usable. Other than public, open source code should be accessible, documented, easy to run, and accompanied by continuous support to questions and issues that may arise from its later usage. We need future research to acquire more rigour in the way to publish, present, and support the code that accompanies scientific publications. In particular, we need *(i)* a formalised, shared and broadly agreed standard that is not necessarily a *new* standard; *(ii)* for new studies to adhere to this standard, and *(iii)* for publishers to review accompanying code at least as thoroughly as when reviewing scientific manuscripts.

Monitoring advancements. The community lacks a database containing comprehensive, curated results of each baseline. Currently, baselines are often re-run when a new method is proposed. This can potentially lead to comparisons that are unfair both because the baselines could be suboptimal (e.g., in the hyperparameters choice, training regime) and their reproduction could be not faithful (e.g., in translating the mathematics into code). When these conditions are not met, it is not clear whether a new method is advancing the field because it assigns credit better or because of misaligned baselines. We call for new, community-driven database holding the latest evaluations of each baseline. The evaluation should be driven by the authors and the authors be responsible for its results. When such a database will be available, new publications should run be tested against the same benchmarks but they not re-run previous baselines and rather refer to the curated results stored in the database.

Peer reviewing CA works. As a consequence of the issues identified above, and because CA methods do not usually shine in classical benchmarks (Bellemare et al., 2013), peer reviews often do not have the tools to capture the novelties of a method and its improvements. On one hand, we need a clear evaluation protocol, including a shared benchmark and leaderboard to facilitate peer reviews. On the other hand, peer reviews must steer away from using tools and metrics that would be used for control, and use those appropriate for the CAP instead.

Lack of priors and foundation models. Most of the CA methods start to learn credit from scratch, without any prior knowledge but the one held by the initialisation pattern of its underlying network. This represents a main obstacle to making CA efficient because at each new learning phase even elementary associations must be learned from scratch. In contrasts, when facing a new tasks, humans often rely on their prior knowledge to determine the influence of an action. In the current state of the art, the use of priors to assign credit more efficiently is overlooked. Viceversa, the relevance of the CAP and the use of more

advanced methods for CA (Mesnard et al., 2021; 2023; Edwards et al., 2018; van Hasselt et al., 2021) is often underestimated for the development of foundation models in RL.

8.3 Conclusions

To conclude, in this survey, we have set out to formally settle the CAP in Deep RL. The resulting material does not aim to solve the CAP, but rather proposes a unifying framework that enables a fair comparison among the methods that assign credit and organises existing material to expedite the starting stages of new studies. Where the literature lacks answers, we identify the gaps and organise them in a list of challenges. We kindly encourage the research community to join in solving these challenges in a shared effort and we hope that the material collected in this manuscript can be a helpful resource to both inform future advancements in the field and inspire new applications in the real world.

Acknowledgments

E.P. was supported by the Engineering and Physical Sciences Research Council (EPSRC) [grant number: EP/R513143/1]. The authors thank Sephora Madjiheurem for the insightful discussions in the early stages of the manuscript.

References

David Abel, Will Dabney, Anna Harutyunyan, Mark K Ho, Michael Littman, Doina Precup, and Satinder Singh. On the expressivity of markov reward. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Mostafa Al-Emran. Hierarchical reinforcement learning: a survey. *International journal of computing and digital systems*, 4(02), 2015.

Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, volume 30, 2017. URL `https://proceedings.neurips.cc/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf`.

Thomas Anthony, Tom Eccles, Andrea Tacchetti, János Kramár, Ian Gemp, Thomas Hudson, Nicolas Porcel, Marc Lanctot, Julien Pérolat, Richard Everett, et al. Learning to play no-press diplomacy with best response policy iteration. *Advances in Neural Information Processing Systems*, 33:17987–18003, 2020.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

Kai Arulkumaran, Dylan R Ashley, Jürgen Schmidhuber, and Rupesh K Srivastava. All you need is supervised learning: From imitation learning to meta-rl with upside down rl. *arXiv preprint arXiv:2202.11960*, 2022.

Dilip Arumugam, Peter Henderson, and Pierre-Luc Bacon. An information-theoretic perspective on credit assignment in reinforcement learning. *CoRR*, abs/2103.06224, 2021. URL `https://arxiv.org/abs/2103.06224`.

Dylan R Ashley, Kai Arulkumaran, Jürgen Schmidhuber, and Rupesh Kumar Srivastava. Learning relative return policies with upside-down reinforcement learning. *arXiv preprint arXiv:2202.12742*, 2022.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pp. 507–517. PMLR, 2020.

Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2019.

Leemon C III Baird. *Reinforcement Learning Through Gradient Descent*. PhD thesis, US Air Force Academy, 1999.

David Balduzzi, Hastagiri Vanchinathan, and Joachim Buhmann. Kickback cuts backprop's red-tape: Biologically plausible credit assignment in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Elias Bareinboim, Juan D. Correa, Duligur Ibeling, and Thomas Icard. *On Pearl's Hierarchy and the Foundations of Causal Inference*, pp. 507–556. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022. ISBN 9781450395861. URL `https://doi.org/10.1145/3501714.3501743`.

Andrew G Barto. Reinforcement learning. In *Neural systems for control*, pp. 7–30. Elsevier, 1997.

Andrew G Barto. Intrinsic motivation and reinforcement learning. *Intrinsically motivated learning in natural and artificial systems*, pp. 17–47, 2013.

Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

Andrew G Barto, Satinder Singh, Nuttapong Chentanez, et al. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, volume 112, pp. 19. Citeseer, 2004.

Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Vahid Behzadan and William Hsu. Adversarial exploitation of policy imitation. *arXiv preprint arXiv:1906.01121*, 2019.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458. Proceedings of Machine Learning Research, 2017.

Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

Michael Bowling, John D Martin, David Abel, and Will Dabney. Settling the reward hypothesis. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3003–3020. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/bowling23a.html.

Lars Buesing, Theophane Weber, Yori Zwols, Nicolas Heess, Sebastien Racaniere, Arthur Guez, and Jean-Baptiste Lespiau. Woulda, coulda, shoulda: Counterfactually-guided policy search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJG0voC9YQ.

Yu-Han Chang, Tracey Ho, and Leslie Kaelbling. All learning is local: Multi-agent learning in global reward games. *Advances in neural information processing systems*, 16, 2003.

Veronica Chelu, Doina Precup, and Hado P van Hasselt. Forethought and hindsight in credit assignment. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2270–2281. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/18064d61b6f93dab8681a460779b8429-Paper.pdf.

Veronica Chelu, Diana Borsa, Doina Precup, and Hado van Hasselt. Selective credit assignment. *arXiv preprint arXiv:2202.09699*, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097, 2021. URL https://proceedings.neurips.cc/paper/2021/file/7f489f642a0ddb10272b5c31057f0663-Paper.pdf.

Zhixin Chen and Mengxiang Lin. Self-imitation learning in sparse reward settings. *arXiv preprint arXiv:2010.06962*, 2020.

Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17, 2004.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

Ashley D Edwards, Laura Downs, and James C Davidson. Forward-backward reinforcement learning. *arXiv preprint arXiv:1803.10227*, 2018.

Andrew J Elliot and James W Fryer. The goal construct in psychology. In *Handbook of motivation science*, volume 18, pp. 235–250, 2008.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Francesco Faccio, Louis Kirsch, and Jürgen Schmidhuber. Parameter-based value functions. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tV6oBfuyLTQ.

Amir-massoud Farahmand. Action-gap phenomenon in reinforcement learning. *Advances in Neural Information Processing Systems*, 24, 2011.

Johan Ferret. *On Actions that Matter: Credit Assignment and Interpretability in Reinforcement Learning*. PhD thesis, Université de Lille, 2022.

Johan Ferret, Raphaël Marinier, Matthieu Geist, and Olivier Pietquin. Self-attentional credit assignment for transfer in reinforcement learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021a. ISBN 9780999241165.

Johan Ferret, Olivier Pietquin, and Matthieu Geist. Self-imitation advantage learning. In *AAMAS 2021-20th International Conference on Autonomous Agents and Multiagent Systems*, 2021b.

Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *International Conference on Machine Learning*, pp. 3145–3153. PMLR, 2020.

Sebastian Flennerhag, Yannick Schroecker, Tom Zahavy, Hado van Hasselt, David Silver, and Satinder Singh. Bootstrapped meta-learning. *arXiv preprint arXiv:2109.04504*, 2021.

Yannis Flet-Berliac. The promise of hierarchical reinforcement learning. *The Gradient*, 9, 2019.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=CAjxVodl_v`.

Jim Gao. Machine learning applications for data center optimization, 2014.

Javier García, Fern, and o Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015. URL `http://jmlr.org/papers/v16/garcia15a.html`.

Matthieu Geist, Bruno Scherrer, et al. Off-policy learning with eligibility traces: a survey. *J. Mach. Learn. Res.*, 15(1):289–333, 2014.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. *Advances in neural information processing systems*, 30, 2017.

Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HygsfnR9Ym`.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. *Advances in neural information processing systems*, 28, 2015.

Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Matthieu Geist, et al. There is no turning back: A self-supervised approach for reversibility-aware reinforcement learning. *Advances in Neural Information Processing Systems*, 34:1898–1911, 2021.

Arthur Guez, Fabio Viola, Theophane Weber, Lars Buesing, Steven Kapturowski, Doina Precup, David Silver, and Nicolas Heess. Value-driven hindsight modelling. *Advances in Neural Information Processing Systems*, 33:12499–12509, 2020.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. Proceedings of Machine Learning Research, 2018.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.

Jean Harb, Tom Schaul, Doina Precup, and Pierre-Luc Bacon. Policy evaluation networks. *arXiv preprint arXiv:2002.11833*, 2020.

Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Nowe. Learning with options that terminate off-policy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. *Advances in neural information processing systems*, 32, 2019.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, 1995.

Donald D Hoffman. The interface theory of perception. *Current Directions in Psychological Science*, 25(3): 157–161, 2016.

Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. *Advances in Neural Information Processing Systems*, 31, 2018.

Ronald A Howard. *Dynamic programming and Markov processes.* John Wiley, 1960.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature Communications*, 10(1):5223, 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-13073-w. URL `https://doi.org/10.1038/s41467-019-13073-w`.

Tommi Jaakkola, Michael Jordan, and Satinder Singh. Convergence of stochastic iterative dynamic programming algorithms. *Advances in neural information processing systems*, 6, 1993.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJ6yPD5xg`.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

Dominik Janzing, David Balduzzi, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Quantifying causal influences. *The Annals Of Statistics*, pp. 2324–2358, 2013.

Stratton C Jaquette. Markov decision processes with a new optimality criterion: Discrete time. *The Annals of Statistics*, 1(3):496–505, 1973.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. Proceedings of Machine Learning Research, 2021a.

Minqi Jiang, Tim Rocktäschel, and Edward Grefenstette. General intelligence requires rethinking exploration. *Royal Society Open Science*, 10(6):230539, 2023.

Ray Jiang, Tom Zahavy, Zhongwen Xu, Adam White, Matteo Hessel, Charles Blundell, and Hado van Hasselt. Emphatic algorithms for deep reinforcement learning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5023–5033. PMLR, 18–24 Jul 2021b. URL `https://proceedings.mlr.press/v139/jiang21j.html`.

Zhengyao Jiang, Tianjun Zhang, Robert Kirk, Tim Rocktäschel, and Edward Grefenstette. Graph backup: Data efficient backup exploiting markovian data. In *Deep RL Workshop NeurIPS 2021*, 2021c. URL `https://openreview.net/forum?id=0UQqmPGuL4n`.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2019.

Steven Kapturowski, Víctor Campos, Ray Jiang, Nemanja Rakićević, Hado van Hasselt, Charles Blundell, and Adrià Puigdomènech Badia. Human-level atari 200x faster. *arXiv preprint arXiv:2209.07550*, 2022.

Steven Kapturowski, Víctor Campos, Ray Jiang, Nemanja Rakicevic, Hado van Hasselt, Charles Blundell, and Adria Puigdomenech Badia. Human-level atari 200x faster. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=JtC6yOHRoJJ`.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE conference on computational intelligence and games (CIG)*, pp. 1–8. IEEE, 2016.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

Martin Klissarov and Doina Precup. Flexible option learning. *Advances in Neural Information Processing Systems*, 34:4632–4646, 2021.

Martin Klissarov, Rasool Fakoor, Jonas Mueller, Kavosh Asadi, Taesup Kim, and Alex Smola. Adaptive interest for emphatic reinforcement learning. In *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*, 2022. URL `https://openreview.net/forum?id=ZGi3bDRXkx`.

A Harry Klopf. Brain function and adaptive systems: a heterostatic theory. Technical Report 133, Air Force Cambridge Research Laboratories. Special Reports, Bedford, Massachusets, 1972.

Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu. Bidirectional model-based policy optimization. In *International Conference on Machine Learning*, pp. 5618–5627. PMLR, 2020.

Alex M Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

Kennon A Lattal. Delayed reinforcement of operant behavior. *Journal of the Experimental Analysis of Behavior*, 93(1):129–139, 2010.

Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G Bellemare. On the generalization of representations in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 4132–4157. PMLR, 2022.

Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*, 2017.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

Chaochao Lu, Biwei Huang, Ke Wang, José Miguel Hernández-Lobato, Kun Zhang, and Bernhard Schölkopf. Sample-efficient reinforcement learning via counterfactual-based data augmentation. *arXiv preprint arXiv:2012.09092*, 2020.

J Luketina, N Nardelli, G Farquhar, J Foerster, J Andreas, E Grefenstette, S Whiteson, and T Rocktäschel. A survey of reinforcement learning informed by natural language. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, August 10-16 2019, Macao, China.*, volume 57, pp. 6309–6317. AAAI Press (Association for the Advancement of Artificial Intelligence), 2019.

Jukka Luoma, Sampsa Ruutu, Adelaide Wilcox King, and Henrikki Tikkanen. Time delays, competitive interdependence, and firm performance. *Strategic Management Journal*, 38(3):506–525, 2017.

Michel Ma, Pierluca D'Oro, Yoshua Bengio, and Pierre-Luc Bacon. Long-term credit assignment via model-based temporal shortcuts. In *Deep RL Workshop NeurIPS 2021*, 2021.

A Rupam Mahmood, Huizhen Yu, Martha White, and Richard S Sutton. Emphatic temporal-difference learning. *arXiv preprint arXiv:1507.01569*, 2015.

Matheus RF Mendonca, Artur Ziviani, and André MS Barreto. Graph-based skill acquisition for reinforcement learning. *ACM Computing Surveys (CSUR)*, 52(1):1–26, 2019.

Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Thomas S Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. In *International Conference on Machine Learning*, pp. 7654–7664. Proceedings of Machine Learning Research, 2021.

Thomas Mesnard, Wenqi Chen, Alaa Saade, Yunhao Tang, Mark Rowland, Theophane Weber, Clare Lyle, Audrunas Gruslys, Michal Valko, Will Dabney, Georg Ostrovski, Eric Moulines, and Remi Munos. Quantile credit assignment. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24517–24531. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/mesnard23a.html.

Donald Michie. Experiments on the mechanization of game-learning part i. characterization of the model and its parameters. *The Computer Journal*, 6(3):232–236, 1963.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961. ISSN 00968390. doi: 10.1109/JRPROC.1961.287775.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, Deep Learning Workshop, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Seyed Sajad Mousavi, Michael Schukat, Enda Howley, and Patrick Mannion. Applying q (λ)-learning in deep reinforcement learning to play atari games. In *AAMAS Adaptive Learning Agents (ALA) Workshop*, pp. 1–6, 2017.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

Suraj Nair, Mohammad Babaeizadeh, Chelsea Finn, Sergey Levine, and Vikash Kumar. Trass: Time reversal as self-supervision. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 115–121. IEEE, 2020.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Thanh Thi Nguyen and Vijay Janapa Reddi. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Chris Nota, Philip Thomas, and Bruno C. Da Silva. Posterior value functions: Hindsight baselines for policy gradient methods. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8238–8247. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/nota21a.html.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR, 2018.

Junhyuk Oh, Matteo Hessel, Wojciech M. Czarnecki, Zhongwen Xu, Hado P van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1060–1070. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0b96d81f0494fde5428c7aea243c9157-Paper.pdf.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rygf-kSYwH`.

Hsiao-Ru Pan, Nico Gürtler, Alexander Neitz, and Bernhard Schölkopf. Direct advantage estimation. *Advances in Neural Information Processing Systems*, 35:11869–11880, 2022.

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

P. I. Pavlov. *Conditioned Reflexes*. Oxford University Press, London, UK, 1927.

Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

Judea Pearl et al. Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19(2):3, 2000.

Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

Jean Piaget, Margaret Cook, et al. *The origins of intelligence in children*, volume 8. International Universities Press New York, 1952.

Silviu Pitis. Rethinking the discount factor in reinforcement learning: A decision theoretic approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7949–7956, 2019.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. *Advances in Neural Information Processing Systems*, 33:3976–3990, 2020.

Chetan Prakash, Kyle D Stephens, Donald D Hoffman, Manish Singh, and Chris Fields. Fitness beats truth in the evolution of perception. *Acta Biotheoretica*, 69:319–341, 2021.

Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000a.

Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 2000b.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Martin L Puterman and Moon Chirl Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11):1127–1137, 1978.

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Hazhir Rahmandad, Nelson Repenning, and John Sterman. Effects of feedback delay on learning. *System Dynamics Review*, 25(4):309–338, 2009.

David Raposo, Sam Ritter, Adam Santoro, Greg Wayne, Theophane Weber, Matt Botvinick, Hado van Hasselt, and Francis Song. Synthetic returns for long-term credit assignment. *CoRR*, 2021.

Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg2viA5FQ`.

Zhizhou Ren, Ruihan Guo, Yuan Zhou, and Jian Peng. Learning long-term reward redistribution via randomized return decomposition. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=lpkGn3k2YdD`.

Matthew Riemer, Miao Liu, and Gerald Tesauro. Learning abstract options. *Advances in neural information processing systems*, 31, 2018.

Mark Rowland, Will Dabney, and Rémi Munos. Adaptive trade-offs in off-policy learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 34–44. PMLR, 2020.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL `https://openreview.net/forum?id=skFwlyefkWJ`.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015a. PMLR. URL `https://proceedings.mlr.press/v37/schaul15.html`.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015b.

Tom Schaul, André Barreto, John Quan, and Georg Ostrovski. The phenomenon of policy churn. *arXiv preprint arXiv:2206.00730*, 2022.

Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner, and Matthieu Geist. Approximate modified policy iteration and its application to the game of tetris. *Journal of Machine Learning Research*, 16(49):1629–1676, 2015.

Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Donald G Schultz. State functions and linear control systems. *McGraw-Hill Book Company*, 1967.

Minah Seo, Luiz Felipe Vecchietti, Sangkeum Lee, and Dongsoo Har. Rewards prediction-based credit assignment for reinforcement learning with sparse binary rewards. *IEEE Access*, 7:118776–118791, 2019.

Mehran Shakerinava and Siamak Ravanbakhsh. Utility theory for sequential decision making. In *International Conference on Machine Learning*, pp. 19616–19625. PMLR, 2022.

Claude E Shannon. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606. Cognitive Science Society, 2009.

Satinder P Singh and Richard S Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1):123–158, 1996.

Burrhus F Skinner. Two types of conditioned reflex: A reply to konorski and miller. *The Journal of General Psychology*, 16(1):272–279, 1937.

Matthew Smith, Herke Hoof, and Joelle Pineau. An inference-based policy gradient method for learning options. In *International Conference on Machine Learning*, pp. 4703–4712. PMLR, 2018.

Matthew J Sobel. The variance of discounted markov decision processes. *Journal of Applied Probability*, 19 (4):794–802, 1982.

Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaskowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *CoRR*, abs/1912.02877, 2019. URL `http://arxiv.org/abs/1912.02877`.

Miroslav Štrupl, Francesco Faccio, Dylan R Ashley, Jürgen Schmidhuber, and Rupesh Kumar Srivastava. Upside-down reinforcement learning can diverge in stochastic environments with episodic resets. *arXiv preprint arXiv:2205.06595*, 2022.

Ke Sun, Bei Jiang, and Linglong Kong. How does value distribution in distributional reinforcement learning help optimization? *arXiv preprint arXiv:2209.14513*, 2022.

Rich Sutton, Ashique Rupam Mahmood, Doina Precup, and Hado Hasselt. A new q (lambda) with interim forward view and monte carlo equivalence. In *International Conference on Machine Learning*, pp. 568–576. PMLR, 2014.

Richard S Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts, 1984.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pp. 216–224. Elsevier, 1990.

Richard S Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pp. 171–176. Citeseer, 1992.

Richard S. Sutton. The reward hypothesis. `http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html`, 2004.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: an introduction*. MIT Press, 2nd edition, 2018. ISBN 9781626239777. URL `http://incompleteideas.net/book/RLbook2020.pdf`.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, 2011.

Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1):2603–2631, 2016.

Yunhao Tang and Alp Kucukelbir. Hindsight expectation maximization for goal-conditioned reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2863–2871. PMLR, 2021.

Edward Lee Thorndike. Animal intelligence: An experimental study of the associative processes in animals. *American Journal of Psychology*, 2(4), 1898.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Hado van Hasselt and Richard S Sutton. Learning to predict independent of span. *arXiv preprint arXiv:1508.04582*, 2015.

Hado van Hasselt and Marco A Wiering. Using continuous action spaces to solve discrete problems. In *2009 International Joint Conference on Neural Networks*, pp. 1149–1156. IEEE, 2009.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Hado van Hasselt, Sephora Madjiheurem, Matteo Hessel, David Silver, André Barreto, and Diana Borsa. Expected eligibility traces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9997–10005, 2021.

Hado P van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

David Venuto, Elaine Lau, Doina Precup, and Ofir Nachum. Policy gradients incorporating the future. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=EHaUTlm2eHg`.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:12163–12174, 2020a.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4235–4246, 2020b.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021.

Tony Tong Wang, Adam Gleave, Nora Belrose, Tom Tseng, Joseph Miller, Michael D Dennis, Yawen Duan, Viktor Pogrebniak, Sergey Levine, and Stuart Russell. Adversarial policies beat professional-level go ais. *arXiv preprint arXiv:2211.00241*, 2022.

Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2016a.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016b.

Christopher J C H Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, United Kingdom, 1989.

Douglas J White. Mean, variance, and probabilistic criteria in finite markov decision processes: A review. *Journal of Optimization Theory and Applications*, 56:1–29, 1988.

Martha White. Unifying task specification in reinforcement learning. In *International Conference on Machine Learning*, pp. 3742–3750. PMLR, 2017.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Zhongwen Xu, Hado P van Hasselt, Matteo Hessel, Junhyuk Oh, Satinder Singh, and David Silver. Meta-gradient reinforcement learning with an objective discovered online. *Advances in Neural Information Processing Systems*, 33:15254–15264, 2020.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Haiyan Yin, Shuicheng YAN, and Zhongwen Xu. Distributional meta-gradient reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=LGkmUauBUL`.

Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. Self-tuning deep reinforcement learning. *arXiv preprint arXiv:2002.12928*, 2020.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

Shangtong Zhang, Vivek Veeriah, and Shimon Whiteson. Learning retrospective knowledge with reverse reinforcement learning. *Advances in Neural Information Processing Systems*, 33:19976–19987, 2020.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27042–27059. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/zheng22c.html`.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, 31, 2018.

Zeyu Zheng, Junhyuk Oh, Matteo Hessel, Zhongwen Xu, Manuel Kroiss, Hado van Hasselt, David Silver, and Satinder Singh. What can learned intrinsic rewards capture? In *International Conference on Machine Learning*, pp. 11436–11446. PMLR, 2020.

Haosheng Zou, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. Reward shaping via meta-learning. *arXiv preprint arXiv:1901.09330*, 2019.