Conformal Calibration

A Post-Hoc Method for Robust Deep Q-Learning

Alex Inch[1]

MSc Machine Learning

Supervised by Lorenz Wolf and Mirco Musolesi

Submission date: 8th September 2025

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged

Abstract

The Deep Q-Network (DQN) is a powerful reinforcement learning algorithm which has achieved superhuman performance in domains including game playing, by effectively combining Q-learning with deep neural networks for function approximation. However, DQNs fail in the face of distribution shift due to overoptimism about previously unseen actions. Prior work addresses this via expensive train-time modifications which require re-tuning the entire training process from scratch. In this thesis, we propose and evaluate an alternative approach: conformal calibration. Our method calibrates the value-function of a DQN by observing the agent for as few as 10 episodes, using the apparatus of conformal prediction to derive a theoretically-valid lower bound on the value function. Experiments on classic control show that a calibrated DQN achieves up to 70% higher reward over a range of distribution shifts, with a nearest-neighbour variant achieving 11% improvement on the more challenging Lunar Lander environment.

Acknowledgements

My deepest thanks to my supervisors Lorenz Wolf and Mirco Musolesi, who were extremely active and engaged with this project. Thank you for the thought-provoking conversations and feedback. Thanks also to many friends at UCL who have made this degree and thesis a pleasure: Jeevon Grewal, Jack Lee, and especially Skye Purchase for painstakingly reviewing this thesis on more than one occasion.

Beyond UCL, I have had the good fortune to meet many people who have advised and supported my decision to pause my career and pursue research. My deepest appreciation to Andrew Haynes, Matt Kaminski, and Colin Gilbert at Evident Insights, as well as Tim Gordon, Jim Talbert and Jeevan Vasagar. I owe immense gratitude to Alexandra Mousavizadeh, who took a chance on me years ago, and helped foster my initial interest in machine learning.

Contents

Nomenclature

Conformal Prediction

α	Miscoverage rate
δ	Conformal offset, usually denoted by $\hat{q}$ in CP literature.
$\epsilon(y, \hat{y})$	Score function, usually denoted by s or A in CP literature.
$\mathcal{C}$	Prediction interval
$\mathcal{D}_c$	Calibration set
t_α	Corrected quantile

Reinforcement Learning

$\mathcal{A}$	Action space
β	Conservative Q-Learning parameter
η	Learning rate
γ	Discount factor $\gamma \in [0, 1]$
$\mathcal{D}_r$	Replay buffer
$\mathcal{P}$	Set of possible transition kernels
$\mathcal{M}$	Nominal environment
$\mathcal{M}'$	Test environment with shifted transition kernel
π	Policy
$\mathcal{S}$	State space
$Q_\theta(s, a)$	Action-value function with parameters θ
$\{\mathcal{S}_k\}_{k=1}^K$	Discrete state clusters, with K bins
a_t	Action taken at time step t
P	Nominal transition kernel

P'	Shifted transition kernel
r_t	Reward received at time step t
s_t	State at time step t

Other symbols

$\times$	Cartesian product

Chapter 1

Introduction

Reinforcement learning (RL) agents learn to take decisions in a given environment to maximise their expected positive outcome. This is a powerful, natural approach; unlike supervised and unsupervised learning methods which rely on human-generated data, RL agents learn how to navigate an environment purely via interaction. In this way, RL agents are able to discover strategies and approaches which exceed the efficacy and generality of human-engineered solutions.

One success story of modern RL is the Deep Q-Network (DQN) [48], which paired the Q-learning algorithm [78] with advancements in deep learning following the success of AlexNet in 2012 [31]. In 2015, Mnih et al. [46] showed that a DQN-based agent was able to match or exceed human skill on a wide range of Atari games, without relying on human-generated data or per-game tuning. This breakthrough precipitated a wave of interest in deep RL, and the DQN and its descendants have since been applied to myriad problems including robotics [29], medicine, [41, 80], energy use optimisation [79] and more.

However, one persistent issue with Q-learning and DQN-based agents is perennial *overoptimism* due to a maximisation term in the value-function update [69, 21, 22]. This can impede learning, as an agent mistakenly believes low-value actions to be better than they truly are. In a worst-case scenario, overoptimism in Q-learning can lead to a total collapse in performance, due to the so-called "deadly triad" [74, 67, 23].

One important consequence of overoptimism in DQNs is that it leads to brittle policies in the face of *distribution shift* [32, 49]. Distribution shift occurs when the environment in which an agent acts changes; a robot operating on an oil rig might start to rust and move more slowly, or a self-driving car trained in sunny weather might have to adapt to icy conditions. Distribution shift is a critical challenge in the real-world deployment of RL agents. The world is complex, and it is virtually impossible to account for all possible shifts in the environment during training. Agents which react unpredictably to distribution shift may become unsafe, endangering themselves or humans in their vicinity. For this reason, it is imperative to train *robust* policies which maintain performance in the face of distribution shift.

Most research into robustness works by inducing more pessimistic or conservative policies [49]. Since distribution shift leads agents to out-of-distribution states which they have not experienced during learning, pessimism about unknown outcomes leads agents to stay in-distribution in the face of uncertainty. For example, Conservative Q-Learning (CQL) [32] learns a pessimistic lower bound estimate of the likely reward from a state, leading an agent to discount previously-unseen actions. More broadly, the field of robust RL formalises a pessimistic notion of the environment [27], either by considering the worst-case outcome at each time step, or treating the environment as an adversary to the agent.

One issue with these methods, however, is they are typically expensive training-time modifications. For example, CQL introduces an expensive *logsumexp* term into the value update. Domain randomisation, a popular robust RL method [72], requires training an agent many times, over a range of distribution shifts. Since they modify the training loop, achieving optimal performance with these methods may also require restarting hyperparameter optimisation from scratch, which can be costly and challenging; a standard DQN implementation has at least 13 hyperparameters [57]. In fact, prior work finds that many reported improvements in the deep RL literature can be attributed to insufficient hyperparameter tuning of baselines [2, 25, 26].

In order to address this problem, we investigate the application of conformal prediction (CP) [75] to the RL setting. Conformal prediction is a post-hoc method applied to a learned predictor which allows a practitioner to make finite-sample guarantees about the uncertainty of a prediction without any distributional assumptions. It works by comparing the prediction over new test points to prior predictions on a held-out calibration set. By treating a DQN (or any equivalent value-based method) as a regression function, we can apply conformal prediction to an already-trained agent, calibrating it without training-time modifications. However, conformal prediction relies on the assumption that the training data for a learned function is *exchangeable*—that any shuffled ordering of the training data is consistent with some fixed joint probability. This property, which is often trivial in the supervised learning setting where i.i.d. data is assumed, is violated in RL, as agents experience time-ordered observations in an online manner. In this thesis, therefore, we will explore the following questions:

- How can conformal prediction be adapted to reinforcement learning to mitigate overoptimism in DQN agents?
- Which assumptions of conformal prediction can be relaxed to leverage it in a practical implementation?

By exploring the above questions, we arrive at conformal calibration, a method for lower bounding the value function of a learned DQN to improve robustness. We develop two plug-in variants; a discretised version, CC-Disc (illustrated in Figure 1.1), stores corrections per state–action bin and incurs minimal runtime cost, and CC-NN, a nearest-neighbour version, computes corrections at run-time from previously observed nearby states, trading extra lookup overhead for better memory scaling with state-space dimension.

We will begin by describing relevant background knowledge in reinforcement learning, conformal prediction, and distribution shift in chapter 2. In chapter 3 we proceed to describe prior work which either addresses overoptimism in DQNs or applies conformal prediction to the RL setting. Chapter 4 introduces conformal calibration with a theoretical motivation, accompanied by descriptions of the two different state-aggregation methods used in CC-Disc and CC-NN. We present empirical results in chapter 5, demonstrating that calibration improves robustness, and testing design choices. Limitations of this work and potential future directions are discussed in chapter 6

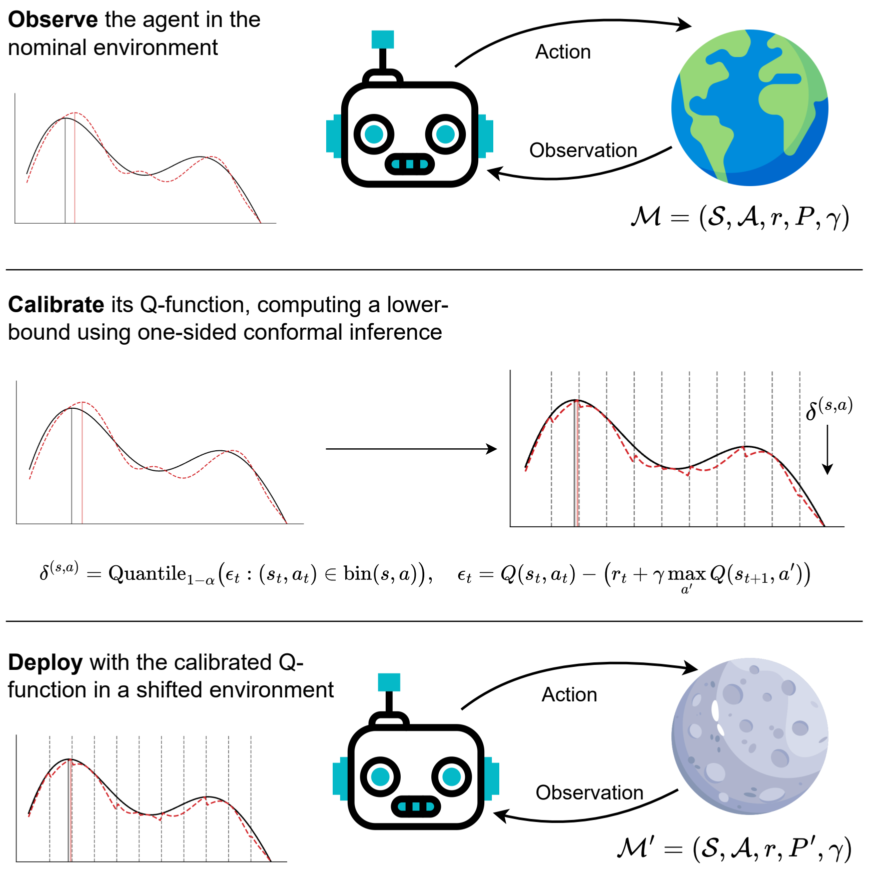
Observe the agent in the nominal environment
Action
Observation
$\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \gamma)$
Calibrate its Q-function, computing a lower-bound using one-sided conformal inference
$\delta^{(s,a)}$
$\delta^{(s,a)} = \text{Quantile}_{1-\alpha}\big(\epsilon_t : (s_t, a_t) \in \text{bin}(s,a)\big), \quad \epsilon_t = Q(s_t,a_t) - \big(r_t + \gamma \max_{a'} Q(s_{t+1}, a')\big)$
Deploy with the calibrated Q-function in a shifted environment
Action
Observation
$\mathcal{M}' = (\mathcal{S}, \mathcal{A}, r, P', \gamma)$

Figure 1.1: Illustration of conformal calibration with the CC-Disc algorithm. Line plots illustrate the true q-function in black, the learned approximation Q_θ in red, and a piecewise lower bounded approximation after calibration (see § 3.1.1). The agent does not leverage information from the test-time environment; calibration is conducted only on the nominal environment. Earth, Moon and robot logos used with permission from Freepik, vectorsmarket15 and Casanova Studio via flaticon.com.

Chapter 2

Background

To explain the ideas underpinning conformal calibration, we first introduce the reinforcement learning (RL) setting, Q-learning and the Deep Q-Network. These are powerful RL algorithms which have been shown to exceed human-level capability in settings such as Atari videogame-playing [46]. We also touch on distribution shift in reinforcement learning and the related field of Robust RL. We introduce conformal prediction, a distribution-free approach to uncertainty estimation with point predictors. Finally, we introduce background concepts around state-space aggregation and nearest neighbour search which will be employed as we move into real implementations.

2.1 Reinforcement Learning

2.1.1 Introduction to Reinforcement Learning

The field of reinforcement learning (RL) examines how agents can learn to act through trial-and-error interactions in an environment, with the goal of maximising long-term reward [67, 28]. Unlike supervised learning, the data distribution is policy-dependent and non-i.i.d.; actions affect future states and information, creating exploration–exploitation trade-offs and temporal credit-assignment challenges. The typical RL setting consists of an agent which takes actions a according to a policy $\pi(a \mid s)$, where s is the state of the environment. Consider the case of a cleaning robot, as paraphrased from Sutton and Barto [67]. The level of battery charge is the state, and the robot can execute two actions; collecting rubbish or charging the battery. Note that s can encode both the internal state of the agent, as well as the external environment. With high battery, a good policy chooses to collect rubbish,

$$\pi(\text{charge} \mid \text{high}) = 0, \quad \pi(\text{collect} \mid \text{high}) = 1.$$

With low battery, the policy instead chooses to charge the robot's battery,

$$\pi(\text{charge} \mid \text{low}) = 1, \quad \pi(\text{collect} \mid \text{low}) = 0.$$

By doing so, the robot will return its battery to a high level of charge, so that it can go on to collect rubbish. The change of state following an action is mediated by the transition kernel $P(\cdot \mid s, a)$, a probability distribution over future states following the agent's action in a given state. In this way, the agent interacts with the environment in a loop, taking actions which alter the state of the environment. This loop can be either episodic, stopping after T timesteps, or infinite-horizon.

The formalism used to describe RL environments is the Markov Decision Process (MDP). In this work we consider a finite, discounted MDP

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \gamma),$$

where $\mathcal{S}$ and $\mathcal{A}$ are measurable state and action spaces, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a bounded reward function, $P(\cdot \mid s, a)$ is the transition kernel which dictates how the state changes after actions, and $\gamma \in [0, 1]$ is the discount factor which trades off near-term vs long-term reward [10, 55]. A policy $\pi(a \mid s)$ induces trajectories $(s_0, a_0, r_1, s_1, a_1, r_2, \dots)$, where $a_t \sim \pi(\cdot \mid s_t)$, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The objective we seek to maximise is the expected discounted return

$$J(\pi) = \mathbb{E}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t}\, r(s_{t'}, a_{t'})\right].$$

We define the value and action–value functions as the expected reward from a given state or state–action pair at timestep t

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) \,\middle|\, s_t = s\right], \qquad Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) \,\middle|\, s_t = s,\ a_t = a\right].$$

They satisfy the Bellman expectation equations

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{a\sim\pi(\cdot|s)}\big[r(s,a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a)} V^{\pi}(s')\big], \\ Q^{\pi}(s,a) &= r(s,a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a),\ a'\sim\pi(\cdot|s')}\big[Q^{\pi}(s',a')\big]. \end{aligned}$$

The optimal values $V^*(s) = \sup_{\pi} V^{\pi}(s)$ and $Q^*(s,a) = \sup_{\pi} Q^{\pi}(s,a)$ obey the Bellman optimality equations

$$\begin{aligned} V^*(s) &= \max_{a\in\mathcal{A}} \Big\{ r(s,a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a)} V^*(s') \Big\}, \\ Q^*(s,a) &= r(s,a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} Q^*(s',a')\right]. \end{aligned}$$

Intuitively, the Bellman optimality equation says that if the optimal value function Q^* is known for all state-action pairs, then the optimal strategy is to select the action a' maximising the expected value of $r + \gamma \max_{a'} Q^*(s,a)$. This strategy is known as a greedy policy $\pi^*(s) \in \arg\max_{a\in\mathcal{A}} Q^*(s,a)$, and if Q^* is optimal then π^* is also optimal [55].

The Bellman optimality operator $\mathcal{T}$

$$(\mathcal{T}Q)(s,a) = r(s,a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} Q(s',a')\right]$$

is a γ-contraction in $\|\cdot\|_{\infty}$, so repeated application converges to its unique fixed point Q^* [10, 12]. These equations underlie value-based RL algorithms like Q-learning; evaluating the value of a policy enables an agent to make decisions based on the expected outcome.

2.1.2 Q-Learning

Q-learning is an RL control algorithm devised by Watkins [78] in 1989. It is tabular—storing an action-value for each state-action pair (s, a). Unlike on-policy methods like Sarsa, which directly follow the policy being optimised [61], Q-learning is an off-policy algorithm which learns from trajectories generated by an arbitrary behaviour policy $\mu(a \mid s) \neq \pi(a \mid s)$. The off-policy nature of Q-learning allows it to consider counterfactual events: "what if the agent had taken a different action?" In contrast to Monte Carlo methods, which update action-values using the full trajectory of rewards generated during an episode, Q-learning bootstraps—it uses only the immediate reward and possible value of next states following an observed transition $(s_t, a_t, r_{t+1}, s_{t+1})$.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \left[r_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s_t, a_t) \right],$$

where s_t, a_t and r_t are the states, actions and rewards observed at timestep t, and η is the learning rate. A Q-learning agent acts using the greedy policy with respect to its learned action-values

$$\pi(a \mid s) = \begin{cases} 1, & \text{if } a = \underset{a' \in \mathcal{A}}{\operatorname{argmax}} \, Q(s, a') \\ 0, & \text{otherwise.} \end{cases}$$

2.1.3 Deep Q-Networks

Q-learning is powerful for classic control problems like gridworld navigation and blackjack [67]. However, as a tabular method, it is subject to the "curse of dimensionality"; if a state-action space has D dimensions, each with n possible values, the number of required table entries grows exponentially with the dimensionality as $N_{\text{entries}} = n^D$. In environments with high-dimensional states, like the pixel-based frame data of an Atari game [8], this leads to more table entries than can be realistically be visited, inhibiting sample efficiency and generalisation. In continuous state spaces, tabular methods do not work at all, requiring discretisation approaches like tile coding to convert the problem into a tabular representation [67].

As a solution, prior work turned to function approximation to output action-values rather than relying on a lookup table. The first notable application of neural network-based function approximation to Q-learning was by Riedmiller [59], who introduced Fitted Q-Iteration (FQI) in 2005. FQI solved the Mountain Car environment in around 300 episodes, compared to 300,000 for a tabular approach over a discretised state space. FQI is a fairly simple extension to Q-learning, implementing batch updates via supervised learning.

Building on FQI, Mnih et al. proposed the Deep Q-Network (DQN) [47, 46], which introduces two significant modifications. The first is experience replay, in which the learner maintains a replay buffer $\mathcal{D}_r$ of prior transitions, and uniformly samples from it during training. This improves sample efficiency as datapoints are reused, as well decorrelating training batches for the neural network (consider the difference between supervised learning on sorted vs shuffled data). The second modification is the introduction of a target network, which uses frozen parameters θ^-. The target

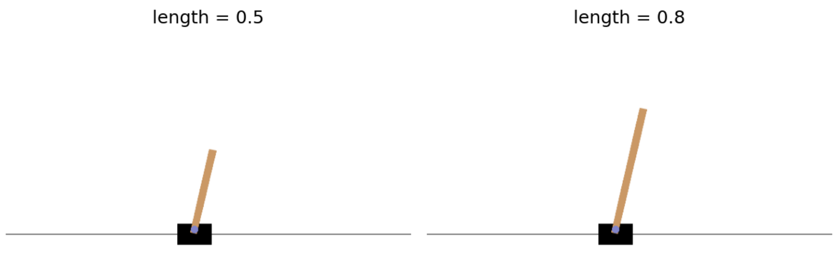

Figure 2.1: Example of distribution shift in the Cart Pole environment, in which the goal is to keep the pole close to vertical. The agent learns with length=0.5, but is evaluated on a different length pole, altering the moment of inertia and hence transition dynamics.

network, which is only updated intermittently, is used for bootstrapping, providing a more stable learning target. The DQN uses backpropagation to update the Q-network, using a squared TD error as loss function (alternatives include a MAE or Huber loss)

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma \max_{a'} Q_{\theta^-}(s', a')) - Q_\theta(s, a)\right)^2\right].$$

One common issue with the DQN is Q-value *overestimation*—the max in the update can lead the learner to overfit to approximation error. Combined with bootstrapping and off-policy learning (so the agent bootstraps off actions which it never actually sees in the training data), this overestimation can compound, impeding learning and leading the agent to take risky actions under distribution shift. We explore this further in § 3.1.

2.1.4 Distribution Shift

A persistent challenge in RL is distribution shift, which occurs when the MDP seen during training differs from the MDP at test-time. Examples could include a self-driving car trained in sunny weather and deployed in rain, or a robotics policy trained in a simulator and deployed in the real world. Distribution shift can affect both the transition kernel $P(\cdot \mid s, a)$ or reward function r, but in this work we focus only on changes to the transition kernel. Agents which maintain performance in the face of distribution shift are said to be robust. This is an attractive quality—robust policies can be safer and reduce the need for retraining as the environment changes [49].

Another form of distribution shift in off-policy methods is the difference in states encountered between a behaviour and target policy. Although we employ off-policy methods in this thesis, we focus on distribution shift due to a change of transition kernel, not due to off-policy methods.

2.1.5 Robust RL

One paradigm concerned with mitigating distribution shift is Robust RL, which addresses train-test discrepancies in the transition kernel, $P \to P'$, by constructing an uncertainty set of possible transition kernels $\mathcal{P} \ni P'$.

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \gamma) \quad \text{(nominal MDP)}, \qquad \mathcal{M}' = (\mathcal{S}, \mathcal{A}, r, P', \gamma) \quad \text{(test MDP)}.$$

It is common to make structural assumptions about $\mathcal{P}$, such as assuming that P' lies within a fixed Kullback-Leibler divergence of the nominal kernel, $D_{KL}(P||P') \leqslant \varepsilon$ [58]. Robust RL agents solve a min-max optimisation problem for the worst case transition kernel. Including the (s, a)-rectangularity assumption (that the worst MDP-level transition kernel is a combination of the worst case for each state-action pair), we arrive at a robust learning objective

$$J(\pi) = \inf_{P' \in \mathcal{P}} \mathbb{E}\left[\sum_{t'=t}^{\infty} \gamma^{t'} r(s_{t'}, a_{t'}) \middle| s_{t'+1} \sim P'(\cdot|s_{t'}, a_{t'})\right], \quad \mathcal{P} = \bigtimes_{(s,a)} \mathcal{P}_{s,a}$$

Although this thesis shares commonalities with Robust RL, we do not make explicit assumptions about $\mathcal{P}$ or adopt Robust RL methods. Instead, we induce robustness via action-value conservatism. The Robust RL literature finds that pessimism improves robustness, but can be limited. Making the most pessimistic assumption possible about $\mathcal{P}$ often leads to overly conservative policies, hampering performance on the nominal environment. Consequently, much work in Robust RL relaxes the assumptions of Robust MDPs to find a reasonable midpoint between nominal performance and conservatism. This kind of tunable conservatism is an inherent strength of the conformal prediction approach.

2.2 Conformal Prediction

Conformal prediction, introduced by Gammerman, Vovk and Vapnik in 1998 [17, 62], is a technique that produces prediction regions for new data points with a guaranteed level of confidence. Unlike most machine learning methods that provide point predictions, conformal prediction provides a set of possible labels for a new object, containing the true label with a certain probability (e.g. 95% of the time).

More formally, conformal prediction allows us to construct a prediction set $\mathcal{C}$ which contains the true value y, with a probability $1 - \alpha$, where α is the user-defined miscoverage rate:

$$Pr[y \in \mathcal{C}] \geqslant 1 - \alpha$$

Notation. There is significant overlap between RL and conformal prediction notation. For example, q signifies action-values and quantile corrections respectively, s indicates a state or nonconformity score, and α a learning rate and miscoverage rate. We use δ for quantile corrections, ϵ for the nonconformity score and retain α for the miscoverage rate, with η for learning rates. Refer to the Nomenclature section for a full listing.

2.2.1 Full Conformal Prediction

Vovk's original paper introduces the idea of *full conformal prediction.* The method assesses how "unusual" a new data point $(x_{n+1}, \hat{y})$ is compared to the existing data $(z_0, \ldots, z_n)$, where $z_i = (x_i, y_i)$. This is done using a nonconformity score $\epsilon : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, where a higher score means the data point is more unusual. Examples of nonconformity scores include absolute residuals $|y - \hat{y}|$ for regression and a likelihood-score $1 - \hat{p}_{\hat{f}}(y \mid x)$ for classification.

The full algorithm proceeds as follows. First, propose a candidate value $y \in \mathcal{Y}$. Next, compute nonconformity scores for all data points z_i in the training set using the candidate value y. To compute ϵ_i, train a predictor f on all other data points z_{-i}

$$\epsilon_i = |y_i - \hat{f}_{-i}(x_i)|, \quad \epsilon_{n+1} = |y - \hat{f}(x_{n+1})|.$$

Next, compute the p-value,

$$p(y) = \frac{1}{n+1}\left(\#\{i : \epsilon_i \geqslant \epsilon_{n+1}\}\right).$$

The p-value is the rank of the residual ϵ_{n+1} with respect to all residuals ϵ_i. Intuitively, a high p-value indicates that the candidate value y would not be unusual compared to the rest of the data.

Finally, iterate over candidate values y to form a prediction interval $\mathcal{C}$. If the data is exchangeable (see § 2.2.2), $\mathcal{C}$ is statistically guaranteed to achieve the specified coverage rate

$$\mathcal{C}(x_{n+1}) = \{y \in \mathcal{Y} : p(y) \geqslant \alpha\}, \quad Pr[y_{\text{true}} \in \mathcal{C}(x_{n+1})] \geqslant 1 - \alpha$$

Without making any assumptions about the underlying distribution, conformal prediction enables us to construct a prediction set with finite-sample guarantees. This is remarkably powerful, but the procedure requires retraining $\mathcal{O}(n)$ times—once for each permutation of the training data—which is computationally infeasible with modern models. As a tractable alternative, we instead turn to split conformal prediction, as outlined in § 2.2.3.

Marginal Coverage Conformal prediction guarantees that prediction intervals will contain the true value y_{true}, subject to some miscoverage rate α. This guarantee applies to the marginal distribution $p(y)$, but not to the conditional distribution $p(y|x)$, ie. in general $\Pr\big[Y_i \in \mathcal{C}(X_i) | X_i = x\big] \not\geqslant 1 - \alpha$. In fact, Lei and Wasserman [36] prove that it is impossible to achieve conditional coverage with non-infinite width prediction intervals. For an example distinguishing marginal and conditional coverage, see Figure 2.4.

2.2.2 Exchangeability

One key assumption leveraged by conformal prediction is *exchangeability*; that all possible orderings of the training data of an algorithm are equally likely under some underlying joint distribution p:

$$p(z_1, z_2, \ldots, z_n) = p(z_{\sigma(1)}, z_{\sigma(2)}, \ldots, z_{\sigma(n)}) \quad \text{for all permutations } \sigma$$

This property, which is weaker than the i.i.d. assumption, can often be assumed in supervised learning. However, exchangeability poses a challenge in reinforcement learning; agents experience time-ordered observations and state transitions. Exchangeability is, therefore, not satisfied by default. We discuss this further in § 4.2.

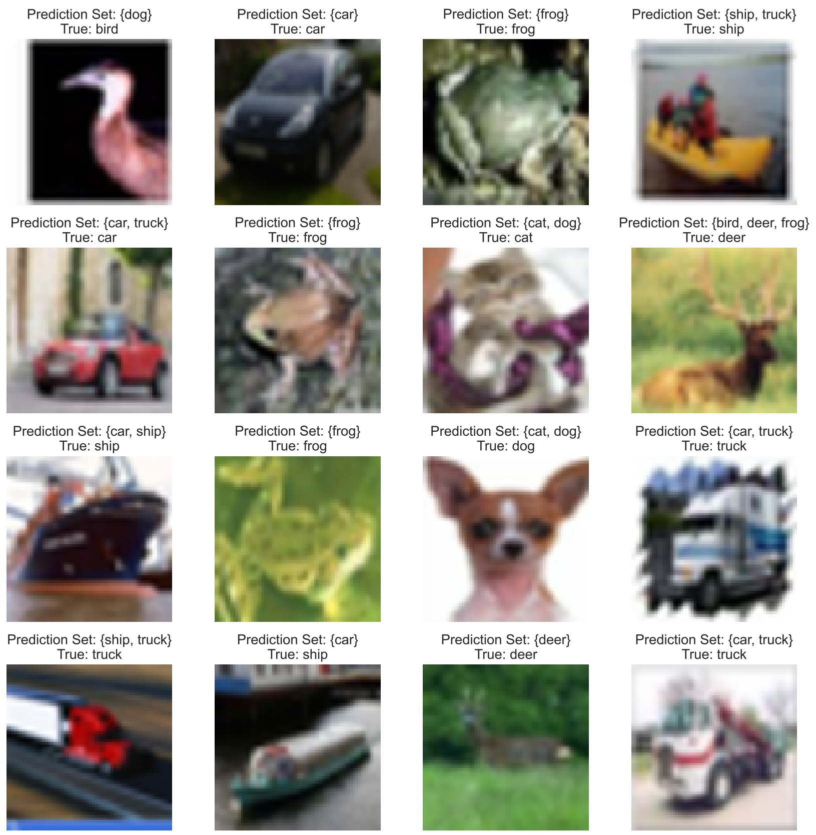

Figure 2.2: Examples of split conformal prediction for classification. Generated using a convnet classifier on the CIFAR10 dataset which achieves 75% accuracy on the test set [34, 30]. Applying conformal prediction with $\alpha = 0.1$, prediction sets contain the true value on 90% of test points.

2.2.3 Split Conformal Prediction

Due to the significant computational demands of full conformal prediction, it is more common to employ *split conformal prediction.* In this paradigm, we partition the available data into a training set $\mathcal{D}_T$ and held-out calibration set $\mathcal{D}_C = \{z_i = (x_i, y_i)\}_{i=1}^m$. We fit a predictor $\hat{f}$ on $\mathcal{D}_T$ once, compute nonconformity scores on $\mathcal{D}_C$, and use a single empirical quantile to calibrate prediction sets for new inputs. The method delivers the same finite-sample, distribution-free marginal coverage as full conformal prediction under exchangeability, at a fraction of the cost [37, 3]. Split conformal prediction proceeds as follows:

1. **Fit the model.** Train $\hat{f}$ on $\mathcal{D}_T$ only.
2. **Calibrate.** Compute calibration scores

$$\epsilon_i \; = \; s\big(\hat{f}(x_i), y_i\big), \quad i = 1, \dots, m,$$

and let $\epsilon_{(1)} \leqslant \cdots \leqslant \epsilon_{(m)}$ be their order statistics. Define the conformal threshold as the value corresponding to the $1-\alpha$ quantile of the residuals. t_α here is a rounding factor which, when the quantile falls between discrete values, breaks ties in favour of the larger value.

$$\delta_{1-\alpha} = \mathrm{Quantile}_{t_\alpha}\left(\{\epsilon_i\}_{i=1}^{m}\right), \qquad t_\alpha \;=\; \frac{\lceil (m+1)(1-\alpha)\rceil}{m}.$$

3. **Predict.** For a new input x_{n+1}, return the split conformal prediction set

$$\mathcal{C}(x_{n+1}) \;=\; \left\{ y \in \mathcal{Y} \,:\, \epsilon\left(y, \hat{f}(x_{n+1})\right) \leqslant \hat{q}_{1-\alpha} \right\}.$$

Compared to full conformal prediction, the split approach replaces $\mathcal{O}(n)$ retrainings with a single fit, making the method computationally tractable for large models or datasets. An example is shown in Figure 2.2.

Coverage with a fixed calibration set. With a fixed calibration and training set, the coverage guarantee of split conformal prediction follows a Beta distribution

$$\Pr\left[Y_{n+1} \in \mathcal{C}(X_{n+1} | \mathcal{D}_C, \mathcal{D}_T)\right] \sim \mathrm{Beta}(mt_\alpha,\; m+1-mt_\alpha).$$

This does not affect theoretical results, which do not condition on the calibration set, but it is relevant when evaluating empirical coverage. Plots of the coverage for different sizes of the calibration set are shown in Figure 2.3.

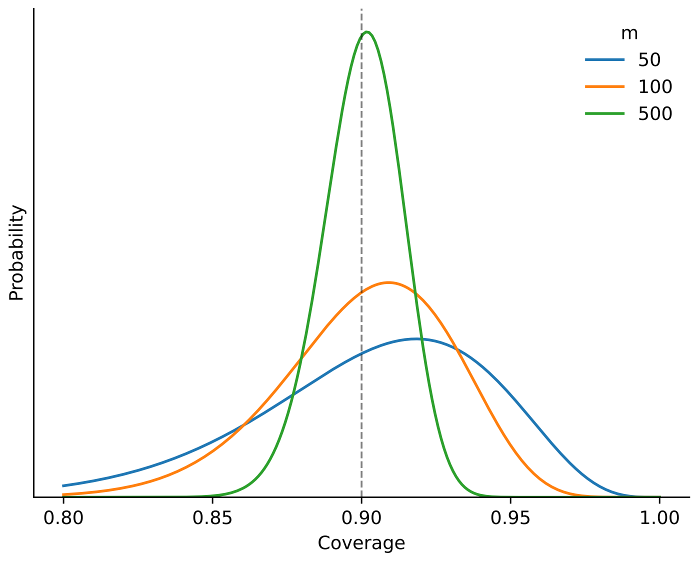

Figure 2.3: When conditioning on a fixed calibration set, the expected coverage follows a Beta distribution, narrowing with more calibration examples. Adapted from a figure by Tibshirani [70].

2.2.4 Conformal Prediction: Regression

Another application is conformal prediction on regression. In the deep RL setting, value functions are examples of common regression functions; outputting typically scalar values for inputs of states or state-action pairs, with the exception of some distributional approaches which output prediction quantiles [9]. In this work, we will employ split conformal prediction in the regression setting for the Q-function of a DQN.

Regression (split conformal). Given a point regressor $\hat{f} : \mathcal{X} \to \mathbb{R}$, compute calibration residuals on $\mathcal{D}_C$,

$$\epsilon_i \;=\; \bigl|y_i - \hat{f}(x_i)\bigr|, \quad i = 1, \ldots, m \;\; (m = |\mathcal{D}_C|),$$

and let $\delta_{1-\alpha}$ be their $(1-\alpha)$-empirical quantile with the standard $\lceil (m+1)(1-\alpha) \rceil$ correction. For a new x, return the symmetric band

$$\mathcal{C}(x) \;=\; \bigl[\, \hat{f}(x) - \delta_{1-\alpha},\; \hat{f}(x) + \delta_{1-\alpha} \,\bigr].$$

This achieves finite-sample *marginal* $1-\alpha$ coverage under exchangeability. Note that this method scales the prediction interval symmetrically. We will use a one-sided approach, discussed in § 4.5.1.

Conformalised quantile regression (CQR). To adapt to heteroskedastic, asymmetric noise, fit quantile regressors $\hat{f}_\ell, \hat{f}_u$ at levels $\alpha/2$ and $1 - \alpha/2$ using a pinball loss. Calibrate one-sided residuals

$$\epsilon_i^\ell = \hat{f}_\ell(x_i) - y_i, \qquad \epsilon_i^u = y_i - \hat{f}_u(x_i),$$

with thresholds δ^ℓ, δ^u from their empirical $(1-\alpha)$-quantiles (using the same $\lceil (m+1)(1-\alpha) \rceil$ index). The prediction set is then

$$\mathcal{C}(x) \;=\; \bigl[\, \hat{f}_\ell(x) - \delta^\ell,\;\; \hat{f}_u(x) + \delta^u \,\bigr],$$

which retains marginal $1-\alpha$ coverage and typically yields tighter, better-targeted intervals than symmetric residual bands [60]. The behaviour is illustrated in Figure 2.4. For compatibility with a standard DQN we do not use quantile regression in this work, but it could be a reasonable extension to combine conformal calibration with the QR-DQN of Dabney et al. [14].

2.2.5 Conformal Inference under Distribution Shift

Prior work investigates conformal prediction under distribution shift. Although we do not use these methods in the default CC-Disc or CC-NN algorithms, we test an adaptive variant of CC-Disc in § 5.3.5. Gibbs and Candès [18] introduces a simple online update to the miscoverage rate

$$\alpha_{t+1} = \alpha_t + \eta\Bigl(\alpha - \mathcal{I}\bigl[\, y_t \notin \mathcal{C}_t(x_t) \,\bigr]\Bigr),$$

where α is the user-specified desired miscoverage rate, $\mathcal{I}$ is the indicator function, and η is the learning rate. See Figure 2.5 for an implementation on a toy problem generating Gaussian noise. Angelopoulos, Barber, and Bates [4] take a similar approach, but rather than modifying

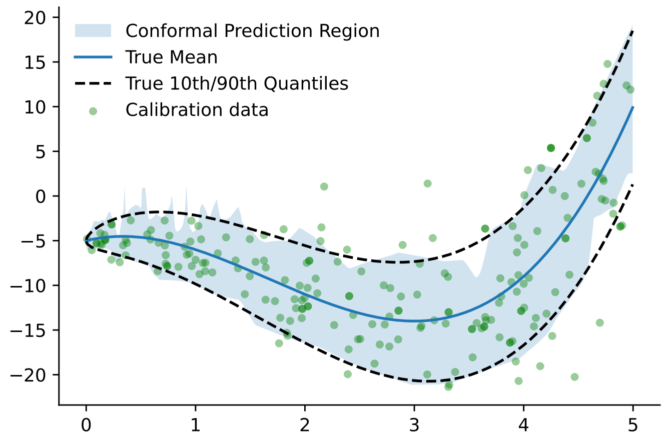

Figure 2.4: Example of conformal prediction for quantile regression. Generated using a neural network quantile regressor with a pinball loss and $\alpha_{lo} = 0.1, \alpha_{hi} = 0.9$. Note that, although the prediction region achieves marginal coverage over all data points, it does not satisfy *conditional coverage*; it over- and under-covers in different regions of the data.

the miscoverage rate, they propose tweaking the interval size itself,

$$\delta_{t+1} = \delta_t + \eta\Big(\mathcal{I}[y_t \notin \mathcal{C}_t(x_t)] - \alpha\Big).$$

2.3 State-Space Discretisation

DQNs are designed for continuous state-spaces, but as we discuss in § 4.2, conformal calibration benefits from conditioning on local state-spaces. One solution, which is common in the RL literature, is state-space discretisation; binning continuous states into discrete-valued features. Discretisation underpins the CC-Disc method we propose. Methods include:

Uniform binning. For each dimension of the state-action space we find an interval containing most of the data—such as [5, 95] quantiles. Split this range evenly. Common numbers of bins per dimension in experiments range from 4-10; higher bin counts harm robustness, due to the loss of generalisation across unseen state-action pairs. This is simple and locality-preserving, but suffers from the curse of dimensionality. If d is the dimensionality of the state-space and $|\mathcal{A}|$ is the number of discrete actions $K = |\mathcal{A}| \cdot (N_{\text{bins}})^d$.

Tile coding. Tile coding [67] overlays N_{tiling} uniform grids ("tilings") with small offsets. A state activates one tile per tiling, yielding a sparse one-hot encoding. Tile coding still suffers from the same fundamental limitations in higher-dimensional spaces $K \sim |\mathcal{A}| N_{\text{tilings}} N_{\text{tiles}}^d$. However, tile coding introduces multiple new parameters that require tuning—not just the number of tiles, but the number of tilings, and how they are formed and offset, which can make it more challenging to tune.

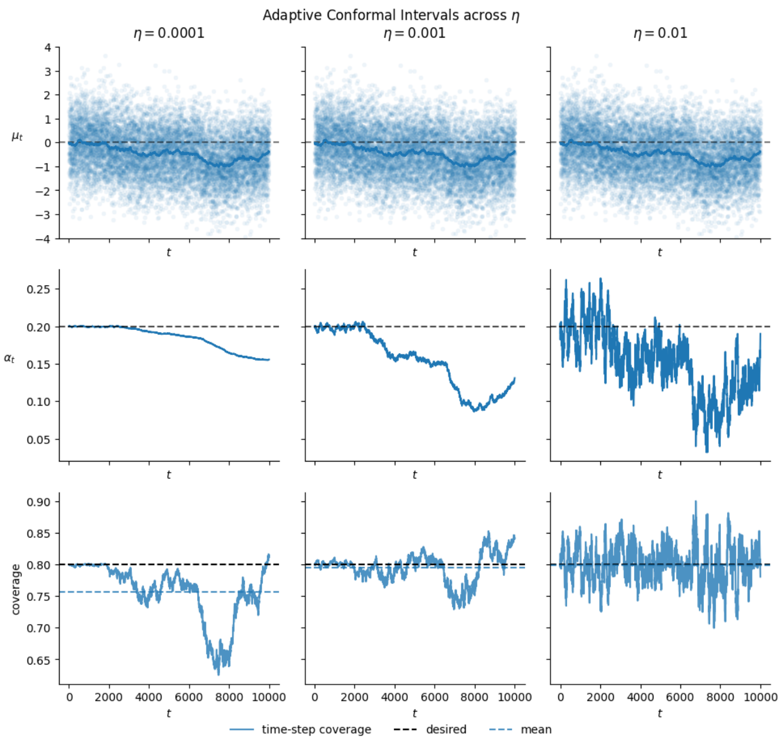

Figure 2.5: Example of adaptive conformal inference [18]. The top row shows data generated by a Gaussian $\mathcal{N}(\mu_t, 1)$. The middle row shows the adaptive miscoverage rate α_t for different values of the responsiveness hyperparameter η. The lowest row shows the true coverage rate, with the true and mean coverage rates shown by the black and blue lines respectively.

Adaptive discretisation. Rather than fixing partitions a priori, tree- or clustering-based schemes refine where needed. A binary partition recursively splits the space to reduce prediction error, yielding nonuniform, data-driven cells [45]. More generally, state aggregation maps many s to a shared index $i = \phi(s)$ via expert rules, PCA+clustering, or vector quantisation (e.g., learned codebooks), improving memory- and sample-efficiency but introducing approximation bias if dissimilar states are merged [67, 45, 51].

2.3.1 Efficient Nearest-Neighbour Search

Discretisation methods either face high-dimensionality limits (binning, tile coding) or are difficult to generalise to arbitrary structures in state-space (PCA, tree, clustering, vector quantisation). As a simple, flexible approach, we will also consider nearest-neighbour search, defining CC-NN.

Nearest neighbours provide a nonparametric representation for continuous $\mathcal{S} \subset \mathbb{R}^d$: given a query s, retrieve the set of k nearest points, then estimate values or dynamics from neighbours. A brute-force scan over all data points scales linearly with dimensionality, $\mathcal{O}(mD)$, so in principle scales more effectively to high-dimensional spaces than simple discretisation techniques. Methods like k-d trees and ball trees further enhance the efficiency of nearest-neighbour search by caching locality in trees or hyperballs, although in spaces with more than 20-30 dimensions query time degrades back towards brute force search [11, 50]. Further extensions to high dimensional

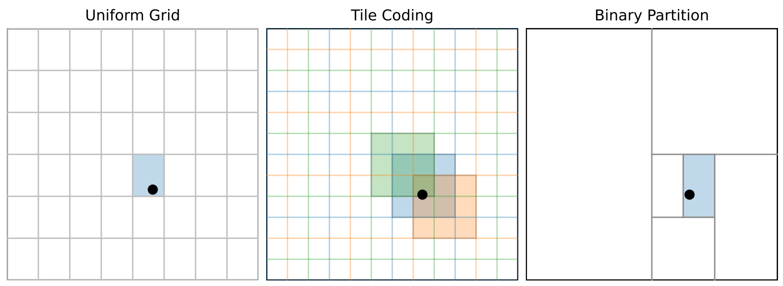

Figure 2.6: Schematic illustrating discretisation schemes. A uniform grid splits the space into fixed-width bins. Tile coding overlays multiple offset grids, where activations are a one-hot-encoding with N_{tilings} active features. A binary partition recursively splits the space in a data-driven manner, with coarse aggregation in unoccupied regions of the space.

spaces require approximate methods like Hierarchical Navigable Small Worlds (as used in vector databases), which have been proven to scale to hundreds of embedding dimensions and millions of data points[42].

2.4 Summary

In this chapter we introduced the core components of conformal calibration: the DQN and conformal prediction. We have described robustness in the face of distribution shift, the goal of conformal calibration. Robust RL motivates the idea of pessimism in the face of distribution shift which we will employ in conformal calibration. In the next section we turn to related work which mitigates overoptimism and prior literature applying conformal prediction to the RL setting.

Chapter 3

Related Work

In this chapter we discuss related work in two strands. The first is prior work addressing the issues of DQN overoptimism in the face of distribution shift, as mentioned in § 2.1.3, namely the Double Deep Q-Network and Conservative Q-Learning. The second is previous work applying conformal prediction to the RL setting. We identify a gaps in the literature around the lack of prior work on post-hoc refinements to address overoptimism

3.1 Overoptimism in DQNs

DQNs can exhibit serious failure modes. The method relies on bootstrapping, off-policy learning and function approximation; the three elements of the "deadly triad." In a worst-case scenario, this can lead the action-value function to diverge, as demonstrated in both classical and deep RL settings [6, 74, 23].

More generally, Q-learning is prone to *overconfidence*. In 1993, Thrun and Schwartz [69] showed that bounded uniform random errors $\sim$ Uniform$(-\varepsilon, \varepsilon)$, for example due to noise or function approximation error, lead to a consistent overestimation of action values by $\gamma\varepsilon\frac{|\mathcal{A}|-1}{|\mathcal{A}|+1}$. Hasselt, Guez, and Silver [22] investigated the same effect in the deep RL setting, and showed that even if value estimates are correct on average, estimation error of any kind can drive action-value estimates away from the true optimal values. Intuitively, this is because the argmax term over action values biases the agent towards overestimated values, as illustrated by Figure 3.1. This overoptimism can be problematic under distribution shift, as the agent pursues out-of-distribution actions which appear beneficial despite never being visited in the training data.

There is substantial work addressing the issue of overestimation as it relates to distribution shift. One approach, typified by methods like batch-constrained RL, BEAR and BRAC, constrains the policy to stay close to the empirical behaviour policy observed in offline data by limiting the actions considered during updates [16, 33, 81]. This shortcuts the deadly triad by preventing an agent from extrapolating too far from the data distribution. Alternatively, REM [1] addresses bootstrapping error by using an ensemble of Q-networks.

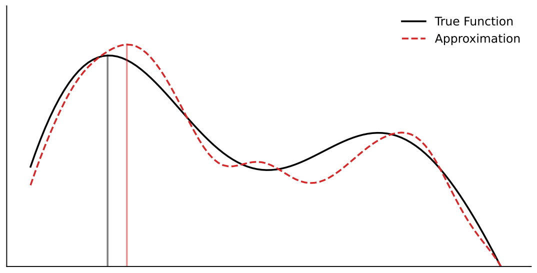

Figure 3.1: Random symmetric approximation error leads to incorrect estimates of the maxima of the action-value function due to the maximisation in the Q-learning update. Vertical lines indicate argmax, which is biased by the approximation error. This figure is inspired by a graphic presented by Sergey Levine at RLC 2024 [39].

3.1.1 Double Deep Q-Networks

Double Deep Q-learning [22], an extension to the tabular Double Q-learning approach proposed by Hasselt [21], directly addresses the overestimation bias inherent in the standard DQN target calculation. The core insight of Double DQN (DDQN) is to decouple the *selection* of the best next action from the *evaluation* of its value. While standard DQN uses the same target network for both operations, DDQN uses the online network $Q(\cdot;\theta)$ to select the action and the target network $Q(\cdot;\theta^{-})$ to evaluate it. The target value Y_t is thus modified from the standard DQN target,

$$J^{\text{DQN}} = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta_t^{-}),$$

to the DDQN target,

$$J^{\text{DDQN}} = r_t + \gamma Q\big(s_{t+1}, \operatorname*{argmax}_{a'} Q(s_{t+1}, a'; \theta_t); \theta_t^{-}\big).$$

By separating selection and evaluation, DDQN avoids the self-reinforcing cycle where an overestimated value is both selected and used for the update. While the online network may still select an action due to overestimation, it is unlikely that the independent target network, starting from a separate initialisation, also overestimates the value of that same action. This simple modification substantially reduces overestimation, leading to more stable training and more accurate final value estimates.

3.1.2 Conservative Q-Learning

The work most closely related to conformal calibration is Conservative Q-Learning (CQL), proposed by Kumar et al. [32]. CQL also attempts to address the issue of overestimation in Q-learning, with the aim of improving robustness to off-policy distribution shift. Instead of trying to learn a precise point estimate of the action-values, like DDQN, CQL modifies the learning objective to lower bound the action-values,

$$L(\theta) = \beta \cdot \underbrace{\Big(\mathbb{E}_{s\sim\mathcal{D},\, a\sim\mu(a|s)}[Q(s,a)] - \mathbb{E}_{s\sim\mathcal{D},\, a\sim\hat{\pi}_\beta(a|s)}[Q(s,a)]\Big)}_{\text{Conservative Q-learning modification}} + \underbrace{\frac{1}{2}\,\mathbb{E}_{s,a,s'\sim\mathcal{D}}\Big[\big(Q(s,a) - \mathcal{T}Q(s,a)\big)^2\Big]}_{\text{Normal Q-learning loss}}. \tag{3.1}$$

Here μ is an arbitrary policy, and $\hat{\pi}_\beta$ is the empirical behaviour policy observed in the training data. The first two terms respectively minimise Q-values while maximising them over actions taken in the dataset. This encourages pessimism about unseen state-action pairs, while maintaining predictions for seen states. The authors derive a variant of the above loss function by choosing the μ which maximises the current Q-iterate, subject to some regulariser $\mathcal{R}(\mu)$,

$$L'(\theta) = \max_\mu L(\theta) + \mathcal{R}(\mu).$$

As a regulariser, they choose the KL divergence against a prior uniform distribution $\rho(a \mid s) = \mathrm{Unif}(a)$, ie. $\mathcal{R}(\mu) = -D_{KL}(\mu \mid\mid \rho)$, the expectation over μ collapses to a logsumexp over actions. This algorithm, which we compare to as a baseline, is referred to as $CQL(\mathcal{H})$,

$$L_\mathcal{H} = \beta \cdot \mathbb{E}_{s\sim\mathcal{D}}\Big[\log\sum_a \exp\big(Q(s,a)\big) - \mathbb{E}_{a\sim\hat{\pi}_\beta(a|s)}[Q(s,a)]\Big] + \frac{1}{2}\,\mathbb{E}_{s,a,s'\sim\mathcal{D}}\Big[\big(Q(s,a) - \mathcal{T}Q(s,a)\big)^2\Big].$$

An important aspect of CQL highlighted by the authors is *gap-expansion*; increasing the difference between Q-values for transitions the agent has seen and those which are out-of-distribution. Gap-expansion discourages the selection of risky values due to Q-function overestimation, leading the agent to stay in-distribution when facing distribution shift.

However, as with the DDQN, CQL is a training-time modification, so can require hyperparameter re-tuning to adapt to an existing workflow, on top of the additional tuning needed for the hyperparameter β.

3.2 Conformal Prediction in RL

Since the resurgence of interest in conformal prediction precipitated by Lei and Wasserman [36] in 2013, there has been a growing body of work applying it to the reinforcement learning setting [43]. Most prior work elides issues with exchangeability by focussing on trajectory-level predictions (see § 2.2.2). These works predominantly focus on the applications of trajectory prediction for multi-agent settings and policy evaluation.

3.2.1 Multi-Agent RL

Substantial prior work applies conformal prediction to planning in multi-agent settings, using conformal prediction to bound the likely position of other agents during a trajectory [40, 24, 65, 64, 35]. Conformal prediction gives hard guarantees about future position, but the flexibility of the user-specified miscoverage rate α can mitigate over-pessimism and freezing. These methods rely on planning from the initial state, recovering exchangeability. Lekeufack et al. [38] extend the ideas of adaptive conformal inference to the trajectory-prediction problem. This approach is capable of updating conformal bounds in an online manner during a trajectory, yielding improved performance in online decision-making. Sheng et al. [63] extends conformal prediction for trajectory planning to the partially observable MDP setting, in which the full environment state is not available to the agent.

CAMMARL [19] also operates in the multi-agent RL setting. Instead of trajectory-planning, it uses conformal prediction to output a prediction set of actions another agent may take at each time-step, which the ego agent can then incorporate into decision-making.

3.2.2 Policy evaluation

Another natural application of conformal prediction to RL is the use of conformal prediction to give guarantees about the outcome of an agent operating in a known environment, known as policy evaluation. Dietterich and Hostetler [15] apply conformal prediction to the policy evaluation problem, proposing an algorithm for whole-trajectory bounds on the reward an agent will likely achieve during episodes in the nominal environment. Taufiq et al. [68] recover exchangeability by considering contextual bandits. Their method, COPP, uses conformal prediction to give prediction intervals in off-policy evaluation, to assess policies before deployment. Since there are no trajectories in the contextual bandit setting, distribution shift due to off-policy methods reduces to covariate shift, a problem addressed in the conformal prediction literature by Tibshirani et al. [71]. Zhang, Shi, and Luo [82] similarly use conformal prediction to output prediction intervals in the off-policy evaluation setting.

3.3 Summary

We identify two large gaps in the literature which conformal calibration addresses. Prior work in RL addresses overoptimism via train-time modifications, necessitating the costly retraining of agents from scratch. Earlier work applying conformal prediction to RL focusses primarily on trajectory prediction and policy evaluation. Instead, conformal calibration is a post-training stage which can be retrofitted to any value-based method. It uses conformal prediction to inform action-selection at inference, inducing pessimism and therefore robustness.

Chapter 4

Approach

We introduce conformal calibration for deep Q-learning. This method uses conformal prediction to estimate a distribution-free lower bound on the Q-function of a DQN, demonstrably improving robustness to distribution shift in multiple RL settings. Additionally, we discuss the theoretical properties of this approach.

We introduce two variants, CC-DISC and CC-NN, which leverage discretisation and nearest neighbour search respectively to localise conformal calibration, mitigating exchangeability violation. CC-DISC is cheap at inference time, but requires state-space discretisation, limiting efficiency in high-dimensional spaces. CC-NN uses nearest-neighbour search at test-time, requiring more inference compute, but scaling to higher dimensional spaces.

4.1 Problem Setup and Notation

We consider a nominal MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \gamma)$ with dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^N$. We train a baseline DQN Q_θ on a training split $\mathcal{D}_\text{T}$, and reserve $\mathcal{D}_\text{C}$ for conformal calibration. We then evaluate the calibrated and uncalibrated DQN on a test MDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, r, P', \gamma)$.

4.2 Assumptions of Conformal Prediction

The key assumption underpinning conformal prediction is *exchangeability*; permutations of training data are equally likely under some joint probability. A significant challenge in applying conformal prediction to RL is exchangeability violation, under which the typical coverage guarantees of conformal prediction no longer hold. We identify two causes of exchangeability violation: time-ordered trajectories and a non-stationary joint distribution during policy learning.

Time-ordering of trajectories. Transitions observed by an agent violate exchangeability since they are ordered by time step. Permutations of a sequence of trajectories experienced by an agent are not, in general, possible under a single consistent MDP. We propose three possible approaches which mitigate this issue:

1. Rather than calibrating over individual transitions, considering full episode trajectories recovers exchangeability. This approach recovers exchangeability because the starting state

distribution is fixed, so any ordering of trajectories is equally likely under the joint MDP and policy. This enables conformal prediction at the trajectory-level, which can inform action selection in the starting state. However, trajectory-level prediction cannot be naively leveraged online during an episode without extensions [38].

2. State-action conditioning under a fixed MDP restores exchangeability, and can be considered a special case of a conformal prediction with a Venn taxonomy (see Ch. 4.6 of Vovk, Gammerman, and Shafer [75]). Since transitions $(s, a) \to s'$ are mediated by a consistent transition kernel $P(\cdot \mid s, a)$, any set of transitions from a given state-action pair are i.i.d. and hence exchangeable, regardless of the policy or time-ordering. To our knowledge, state-action conditioning has not previously been applied to conformal prediction in RL. However, state-action conditioning scales poorly, so requires relaxed assumptions about exchangeability to be applied in high-dimensional or continuous state spaces. We discuss these relaxations in § 4.3.

3. Sampling from a fixed replay buffer during experience replay decorrelates transitions in a batch, breaking time-ordering. We do not explore replay buffer-sampling in this work, but possible methods could utilise conformal prediction when sampling from a replay buffer combined with conformal distribution sampling [76], similar to robustness approaches like EWoK [77].

Online learning. During learning, the policy of an agent changes, altering the joint distribution which generates transitions. Consider a low value state; the agent visits the state during early exploration, but stops visiting it as learning proceeds. Replay buffers or trajectories including this transition therefore violate exchangeability with new test points at inference, as they are not consistent with a fixed MDP. Conformal prediction is thus most naturally applied to a fixed policy.

For these reasons, in this work we choose to apply conformal prediction in concert with state-action conditioning to a frozen DQN.

4.3 Theoretical Framework

To induce conservatism, we wish to apply conformal prediction to lower bound the action-value function of a learned DQN. We show that a Monte Carlo target, applied in the tabular setting, can be used to derive a valid lower bound. Relaxing some assumptions, this theoretical underpinning equips us with the core method of conformal calibration.

Setting. Consider a discounted MDP $(\mathcal{S}, \mathcal{A}, r, P, \gamma)$ with $\gamma \in [0, 1)$ and rewards almost surely bounded as $r_t \in [r_{\min}, r_{\max}]$. For a fixed stochastic target policy π, define the discounted Monte Carlo return from a start pair (s, a):

$$G_\pi^{(s,a)} = \sum_{t=0}^{\infty} \gamma^t r_t, \quad \text{so that} \quad V_{\min} := \frac{r_{\min}}{1-\gamma} \leqslant G_\pi^{(s,a)} \quad \text{a.s.}$$

Let $q^\pi(s, a) := \mathbb{E}[G_\pi^{(s,a)}]$ and $q^\star(s, a) = \sup_\pi q^\pi(s, a)$.

We introduce a *Venn taxonomy* by groups corresponding to state-action pairs $(s,a) \in \mathcal{S} \times \mathcal{A}$. For each group $g = (s,a)$ we assume access to $n_{(s,a)}$ i.i.d. calibration episodes $Y_1^{(s,a)}, \ldots, Y_{n_{(s,a)}}^{(s,a)}$ distributed as $G_\pi^{(s,a)}$ (episodes start from (s,a) and follow π). Let $Q(s,a) \in \mathbb{R}$ be any base predictor. Define one-sided residuals $\epsilon_i^{(s,a)} := Q(s,a) - Y_i^{(s,a)}$, and let

$\delta^{(s,a)}$ be the $k_{(s,a)}$-th order statistic, in ascending order, of $\{\epsilon_i^{(s,a)}\}_{i=1}^{n_{(s,a)}}$, $\quad k_{(s,a)} := \left\lceil (n_{(s,a)}+1)(1-\alpha) \right\rceil$.

Set the groupwise (one-sided) conformal lower predictor

$$L_{(s,a)} \ := \ Q(s,a) - \delta^{(s,a)}.$$

Lemma 1 (One-off conformal coverage)**.** *For every group $g = (s,a)$, the split-conformal construction above satisfies*

$$Pr\left[G_\pi^{(s,a)} \geqslant L_{(s,a)}\right] \geqslant 1 - \alpha,$$

where the probability is over a fresh test episode from the same MDP as the calibration episodes.

Proof. By exchangeability within group g, the rank of the fresh residual $\epsilon_{\text{test}}^{(s,a)} := Q(s,a) - G_\pi^{(s,a)}$ among the set $\{\epsilon_1^{(s,a)}, \ldots, \epsilon_{n_{(s,a)}}^{(s,a)}, \epsilon_{\text{test}}^{(s,a)}\}$ is uniformly distributed on $\{1, \ldots, n_{(s,a)} + 1\}$. Hence

$$\Pr\left[\epsilon_{\text{test}}^{(s,a)} \leqslant \delta^{(s,a)}\right] \ \geqslant \ \frac{k_{(s,a)}}{n_{(s,a)} + 1} \ \geqslant \ 1 - \alpha.$$

The event $\epsilon_{\text{test}}^{(s,a)} \leqslant \delta^{(s,a)}$ is equivalent to $G_\pi^{(s,a)} \geqslant Q(s,a) - \delta^{(s,a)} = L_{(s,a)}$, yielding the claim. □

By consequence of Lemma 1, we have shown that we can condition conformal inference on state-action pairs and derive a probabilistic lower bound on a learned Q-function. We now show that the probabilistic lower bound from conformal inference can be converted into a true lower bound using the Law of Total Expectation.

Theorem 1 (Conformal calibration lower bound on q^π)**.** *Under the assumptions above, for every tuple (s,a),*

$$q^\pi(s,a) \ \geqslant \ (1-\alpha)\, L_{(s,a)} \ + \ \alpha\, V_{\min}.$$

Proof. Fix $g = (s,a)$ and abbreviate $L = L_{(s,a)}$, $G = G_\pi^{(s,a)}$. Since $G \geqslant V_{\min}$ a.s.,

$$\mathbb{E}[G] \ = \ \mathbb{E}\big[G \cdot \mathcal{I}[G \geqslant L]\big] \ + \ \mathbb{E}\big[G \cdot \mathcal{I}[G < L]\big] \ \geqslant \ L \cdot \Pr[G \geqslant L] \ + \ V_{\min} \cdot \Pr[G < L].$$

Where $\mathcal{I}$ is the indicator function. By Lemma 1, $\Pr[G \geqslant L] \geqslant 1 - \alpha$, hence

$$\mathbb{E}[G] \ \geqslant \ (1-\alpha)L \ + \ \alpha V_{\min}.$$

But $\mathbb{E}[G] = q^\pi(s,a)$ by definition. □

Corollary 1 (Lower bound on $q^\star$)**.** *For every (s,a) and any chosen target policy π,*

$$q^\star(s,a) \ \geqslant \ q^\pi(s,a) \ \geqslant \ (1-\alpha)\, L_{(s,a)} \ + \ \alpha\, V_{\min}.$$

Proof. The first inequality is by optimality of $q^\star$. The second is Theorem 1. □

Remark 1 (Choice of π)**.** *To obtain a bound informative for $q^\star$, choose a strong policy (e.g. $\pi = \mathrm{Greedy}(Q)$); Corollary 1 then transfers the bound to $q^\star$.*

Remark 2 (Induced policy π_{LB})**.** *$V_{\min}$ and the scalar factor $(1-\alpha)$ are constant across all state-action pairs. Therefore the greedy policy π_{LB} with respect to the derived lower bound is invariant to their values. Setting $V_{\min} = 0$ and $(1-\alpha) = 1$ gives the estimator $Q(s,a) - \delta^{(s,a)}$, which induces the same policy as the valid lower bound derived above.*

Remark 3 (Tightness and the role of α)**.** *The miscoverage level α introduces a fundamental trade-off between the statistical confidence of the bound and its numerical value (tightness). The parameter α influences the final bound $(1-\alpha)L_{(s,a)} + \alpha V_{\min}$ in two competing ways:*

1. ***Quantile Selection:*** *$L_{(s,a)}$ is determined by the $k_{(s,a)} = \lceil (n_{(s,a)} + 1)(1-\alpha) \rceil$-th order statistic of the residuals. As $\alpha \to 0$, $k_{(s,a)}$ approaches $n_{(s,a)} + 1$, causing $\delta^{(s,a)}$ to approach the maximum observed residual. This makes $L_{(s,a)}$ more conservative (smaller), tending towards the minimum observed return, $\min_i Y_i^{(s,a)}$. Conversely, a larger α leads to a less conservative (larger) $L_{(s,a)}$.*

2. ***Convex Combination:*** *The final bound is a convex combination of the empirical bound $L_{(s,a)}$ and the worst-case bound $V_{\min}$. As $\alpha \to 0$, more weight is placed on $L_{(s,a)}$. As $\alpha \to 1$, the bound degenerates towards $V_{\min}$, becoming maximally loose.*

Consequently, choosing a very small α provides a high-confidence guarantee (e.g., 99.9%), but the resulting lower bound may be overly pessimistic because $L_{(s,a)}$ will be very low. The optimal choice of α must balance the desire for a high-probability statement with the need for a numerically useful lower bound on the action-value.

In practice, we will relax the above assumptions in two ways. Theorem 1 uses a Monte Carlo return, which in practice often suffers from high variance. Instead, we will employ a bootstrapped TD error, for a simpler implementation and lower variance nonconformity scores. We empirically evaluate the two approaches in § 5.3.3. Additionally, Theorem 1 assumes perfect conditioning on (s,a), which is achievable in the tabular case. However, DQNs are designed to be applied to more complex, continuous spaces. We propose multiple schemes for aggregating nearby states; This aggregation violates exchangeability, introducing a gap between the expected and true coverage, but this gap can be reduced by aggregating sufficiently "similar" states (see § 4.5.2).

4.4 State Space Aggregation

One relaxation we make to achieve state-action conditioning is aggregating nearby states. However, this is challenging; the structure of state-action pair visitation is complex, and it varies between policies learned from different random initialisations, shown in Figure 4.1. For low-dimensional state spaces, we evaluate three schemes which enable the discretisation of the state-space into distinct bins, essentially recovering the tabular setting. However, these approaches do not scale to higher-dimensional spaces; instead there we leverage nearest-neighbour search to find nearby points.

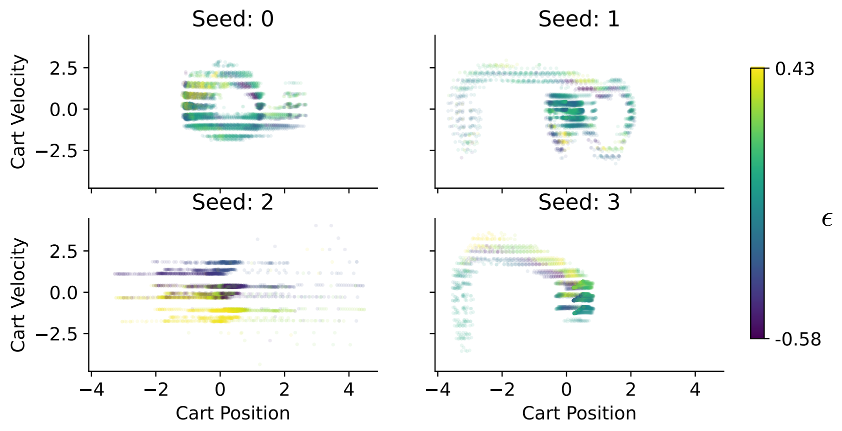

Figure 4.1: States visited across 10,000 transitions in Cart Pole. Subplots show policies trained from different random seeds—state-action visitation varies heavily. Colouring indicates the TD nonconformity score ϵ. The structure is complex, and small movements in the state space can induce large changes in score. Horizontal striations are due to the discrete time steps of the environment.

4.5 Conformal Calibration Framework

4.5.1 One-Sided Split Conformal Inference

In conformal prediction it is typical to use an absolute distance metric (eg. $|\hat{y}_i - y_i|$), which symmetrically grows the interval in response to underestimation *and* overestimation. However, for the purpose of lower bounding a function we only need to penalise overestimation, so employ the signed score ϵ_i to obtain a tighter bound (an empirical comparison is made in § 5.3.2). For transition (s_i, a_i, r_i, s'_i), we define

$$\hat{y}_i = Q_\theta(s_i, a_i), \quad y_i = r_i + \gamma \max_{a'} Q_\theta(s'_i, a'), \quad \epsilon_i = \hat{y}_i - y_i.$$

For bin (s, a), compute the lower-tail conformal offset

$$\delta^{(s,a)} \; = \; \text{Quantile}_{t_\alpha}\big(\{\epsilon_i : (s_i, a_i) \in \text{bin}(s, a)\}\big), \quad t_\alpha = \frac{\lceil (m+1)(1-\alpha) \rceil}{m},$$

where $\text{Quantile}_{t_\alpha}$ is the t_α quantile of the residuals. This corresponds to the one-sided prediction interval

$$\mathcal{C}(s, a) = \big[Q_\theta(s, a) - \delta^{(s,a)}, \; \infty\big).$$

4.5.2 Coverage Gap in Continuous Spaces

In continuous state spaces, we aggregate nearby states into bins. In general this violates exchangeability, so the hard guarantees of conformal prediction no longer hold. However, as shown by Barber et al. [7], the "coverage gap" due to exchangeability violation can be bounded

$$\text{Coverage gap} \; = \; (1 - \alpha) \; - Pr\left[y_{n+1} \in \mathcal{C}^{(s,a)}(x_{n+1})\right],$$

$$\text{Coverage gap} \leqslant \frac{\sum_{i=1}^{n} w_i \, d_{\mathrm{TV}}\big(Z, Z^i\big)}{1 + \sum_{i=1}^{n} w_i}.$$

Where

- $d_{\mathrm{TV}}(\cdot, \cdot)$ is the total variation distance between distributions, the largest possible difference between the probabilities that the two distributions assign to the same event.
- $Z = (z_1, \ldots, z_{n+1})$ is the full calibration dataset sequence with $z_i = (x_i, y_i)$.
- $Z^i = (z_1, \ldots, z_{i-1}, z_{n+1}, z_{i+1}, \ldots, z_n, z_i)$ swaps the test point (x_{n+1}, y_{n+1}) with the i-th training point (x_i, y_i).
- $w_i \in [0, 1]$ are pre-specified weights on data points. Barber uses decaying w_i for distribution shift, but other work uses spatial weights (eg. 1 for nearest neighbours and 0 otherwise)[44].

Intuitively this results means that if states in a bin are similar, the coverage gap will be small. We empirically evaluate coverage in the results section (see section § 5.2.6).

4.5.3 Uncalibrated regions

When a given region of the state space contains few or no calibration examples, $\delta^{(s,a)}$ is inaccurate or undefined. Instead, we choose a global fallback value. Recall that one benefit of Conservative Q-Learning is *gap-expansion*; increasing the difference between the action-values of in- and out-of-distribution actions. To induce gap-expansion we set the correction to the largest of all bins with sufficient calibration data $\delta_{\text{fallback}} = \max_{(s,a)} \delta^{(s,a)}$. An illustration of the fallback behaviour is given in Figure 4.2.

Similarly, when using nearest neighbour search to collect a calibration set in CC-NN, we would like to identify outlier states which are too far from the calibration data for an accurate offset. In this case, we choose a maximum distance for the k^{th} neighbour, beyond which we revert to the fallback measure $\delta_{\text{fallback}} = \max_{(s,a)} \delta^{(s,a)}$. Setting a maximum distance in a multi-dimensional state-space across different coordinate systems (eg. angle, velocity, position) is challenging, so instead of pre-specifying a distance we identify a coverage level $\hat{d} \in [0, 1)$. Denoting the distance from a state-action pair (s_i, a_i) to its k^{th} neighbour as $d_{i \to k}$, the max distance is then

$$d_{\max} = \text{Quantile}_{\hat{d}}\big(\{d_{i \to k}\}\big), \quad (s_i, a_i) \in \mathcal{D}_c.$$

4.6 CC-Disc: Discretised Conformal Calibration

We now begin describing specific instantiations of the conformal calibration framework. First we outline CC-Disc, which uses state-space discretisation, calculated at calibration-time, to compute offsets for each state-action bin. These are then retrieved at inference, giving a relatively lightweight calibration approach.

4.6.1 Partitioning the State-Action Space

To achieve state-action conditioning in continuous spaces we discretise the space. In principle, this means using the calibration data to learn a discretisation function $\textsc{Discretise}(s, a) : \mathcal{S} \times \mathcal{A} \mapsto [K]$, where $[K] = \{1, 2, \ldots, K\}$ is the index set of bins. We test three methods; uniform binning, tile coding and a data-driven binary partition, as described in § 2.3.

4.6.2 Algorithm (CC-Disc)

We present a high-level overview of the steps taken in CC-Disc. Full pseudocode is provided in Appendix A.

1. **Train nominal agent**. Learn Q_θ on $\mathcal{M}$ using standard DQN.
2. **Collect calibration data**. Execute the greedy policy π_Q in $\mathcal{M}$ and store transitions (s, a, r, s', a') in a calibration buffer of size N_c. We save examples in deques with a maximum length greater than the minimum required calibration threshold, so that commonly-visited states have larger calibration sets.
3. **Discretise state-action space**. Construct a $\textsc{Discretise}$ function using $\mathcal{D}_C$. For example, if binning, find the desired quantiles of the data to cover and evenly construct bins spanning the interval.
4. **Compute lower bounds**. Compute residuals ϵ_i and corresponding offsets $\delta^{(s,a)}$ at level α. If a bin has too few calibration examples, instead use the fallback value $\max_{s,a} \delta^{(s,a)}$.
5. **Deploy with lower bounded policy**. At test-time, in shifted environment $\mathcal{M}'$, act greedily with respect to the conformal lower bound

$$a = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \big[Q_\theta(s, a) - \delta^{(s,a)} \big].$$

4.7 CC-NN: Nearest-Neighbour Conformal Calibration

Although state-space discretisation is an intuitive, lightweight approach to state-action conditioning, the number of required grid cells scales exponentially with dimensionality of the state-space, leading to a high, inefficient memory cost. As an alternative, we propose a nearest-neighbour variant, CC-NN. Where CC-Disc computes a discretisation function and offsets during offline calibration step with a cheap lookup at inference, CC-NN constructs the calibration set and computes $\delta^{(s,a)}$ at inference-time, using nearest-neighbour search. This increases the inference cost of calibration, but avoids the exponentially growing memory overhead of CC-Disc (see § 5.2.4 for an evaluation of the incurred cost).

4.7.1 Offline Observation

In CC-NN, we still observe the agent in the nominal environment to build a buffer of previous transitions. We then compute nonconformity scores for each point in the buffer. Lastly we compute a maximum distance as described in § 4.5.3. Beyond this distance, we revert to the fallback value δ_{fallback}.

4.7.2 kNN Quantile at Test Time

At inference time, given state-action pair (s, a), retrieve the k-nearest neighbours and compute the offset,

$$\mathcal{N}_k = \mathrm{kNN}\big(z(s,a); \{z(s_i, a_i)\}_{D_{\mathrm{cal}}}\big),$$

$$\delta^{(s,a)} = \mathrm{Quantile}_{t_\alpha}\big\{\epsilon_i : i \in \mathcal{N}_k\big\}, \quad t_\alpha = \frac{\lceil (k+1)(1-\alpha) \rceil}{k}.$$

As described in § 2.3.1, there are numerous approaches to fast nearest-neighbour search, ranging from exact methods like a k-d tree to approximate methods like HNSW. For the scale of problem CC-NN is applied to, with a 10-dimensional state space and 10,000 transitions in the calibration set, we found a k-d tree was sufficiently performant.

4.7.3 Algorithm (CC-NN)

We present a high-level overview of the steps taken in CC-NN. Note that the first two steps are identical to CC-Disc. Full pseudocode is provided in Appendix A.

1. **Train nominal agent**. Learn Q_θ on $\mathcal{M}$ using standard DQN.

2. **Collect calibration data**. Execute the greedy policy π_Q in $\mathcal{M}$ and store transitions (s, a, r, s', a') in a calibration buffer of size N_c.

3. **Compute residuals**. Compute residuals ϵ_i and corresponding offsets $\delta^{(s,a)}$ at level α for each transition in buffer. Compute the maximum distance as the 90th quantile distance of the kth neighbour.

4. **Compute correction at test-time**. Given a state-action pair, find the k-nearest neighbours, and compute offset $\delta^{(s,a)}$ at level α. If the kth neighbour is further than the maximum distance, use the fallback $\delta_{\mathrm{fallback}}$. Act greedily with respect to the conformal lower bound

$$a = \operatorname*{argmax}_{a \in \mathcal{A}} \big[Q_\theta(s,a) - \delta^{(s,a)}\big].$$

4.8 Summary

In this chapter we motivated and introduced conformal calibration, alongside two variant algorithms CC-Disc and CC-NN. CC-Disc relies on an explicit state-space discretisation function which is constructed after the observation phase. At inference, it only requires a cheap lookup to fetch precomputed corrections. CC-NN, on the other hand, has no explicit discretisation scheme. It constructs a calibration set per-decision at inference-time using nearest-neighbour search. An illustration of the two methods is shown in Figure 4.2.

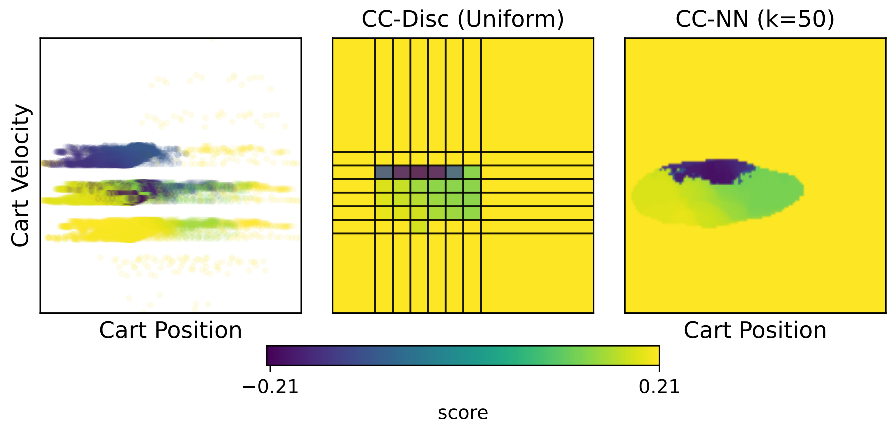

Figure 4.2: A 2D view of the scores of CC-Disc and CC-NN in state-space. The left plot shows state-action visits by a Cart Pole agent, coloured by the nonconformity score. The CC-Disc grid boundaries are selected to cover 90% of the observed transitions. Far from the distribution, the offset is the constant value δ_{fallback}, inducing *gap-expansion*.

Chapter 5

Results

In this section we describe the experimental setup of our empirical evaluations of CC-Disc and CC-NN, and provide empirical results from conformal calibration. We split the results section into two halves; one evaluates the performance impacts of calibration and compares to baselines, and the other evaluates design choices and the effect of hyperparameters.

5.1 Experimental Setup

Environments. All environments use Gymnasium (an up-to-date fork of OpenAI Gym) running locally on a Macbook M3 [13, 73]. We use Stable Baselines3 (SB3) for its vanilla DQN implementation, using a PyTorch backend. SB3-Zoo hyperparameters are used for all environments (see Appendix B) [57, 56, 52]. We test on the 3 classic control environments with discrete action spaces (as required by a DQN): Car tPole, Mountain Car and Acrobot. Additionally, for a slightly larger state space we also test Lunar Lander. For all environments we use default settings. Qualitative testing suggested that the behaviour of conformal calibration was fairly consistent across parameters in the same environment, so for simplicity we focus on one shift parameter to vary: pole length in Cart Pole, gravity in Mountain Car, link length in Acrobot and gravity in Lunar Lander.

Baselines. There are no standard baselines for comparing different robust RL baselines [49], conformal calibration is a novel post-training approach, precluding direct comparison with prior work. Instead, therefore, we compare to similar methods for addressing overestimation in DQNs; DDQN and a Conservative Q-Learning DQN (CQL-DQN). We also evaluate the effect of combining conformal calibration with DDQN and CQL-DQN agents. Implementations of Conservative Q-Learning and DDQN are written in PyTorch and scaffolded around SB3's DQN implementation (code is available at `https://github.com/foty-hub/msc_thesis`). Each baseline was tuned on the nominal environment to surpass a standard reward threshold, but no other robustness-aware tuning was done. Reward thresholds are listed in Appendix B.3.

Discretisation. PyFixedReps is used for uniform binning and tile coding implementations [53]. Scikit-learn is used for its implementation of a k-d tree nearest neighbour algorithm [54] for fast nearest-neighbour search in CC-NN.

Evaluation Protocol. For each environment and algorithm, we train from scratch on a fixed number of episodes, for 25 random seeds. To ensure that training dynamics don't bias results, we discard any agents which fail to meet a sufficient reward threshold on the training environment (thresholds are listed in Appendix B). To test robustness to distribution shift we evaluate agent mean reward over 250 episodes over a range of parameter values (eg. pole length in Cart Pole). For final plots we report the interquartile mean, with 95% bootstrapped confidence intervals as recommended by Agarwal et al. [2]. We use the same hyperparameters for conformal calibration across environments (also listed in Appendix B), with the aim of demonstrating performance without access to shifted environments against which to tune.

Score functions. We evaluate two score functions; the signed and unsigned scores $\epsilon_{\text{signed}}(\hat{y}, y) = \hat{y} - y$ and $\epsilon_{\text{unsigned}} = |\hat{y} - y|$ respectively. Intuitively, the unsigned score penalises both over- and under-estimation, while the signed score only corrects for overestimation. By default we used the signed score. We therefore expect the signed score to give a tighter lower bound and yield better performance. Conformal prediction only leverages order statistics, so squared distances would yield the same prediction sets as the L1 norm. Results are given in § 5.3.2.

Metrics. Evaluating the robustness of a policy is nuanced, and the Robust RL literature lacks a single consistent metric [49]. Common choices include a worst-case return over an uncertainty set of transition kernels $R_{\text{wc}} = \inf_{P' \in \mathcal{P}} J_{P'}(\pi)$, a worst-case performance ratio $\text{Rat}_{\text{wc}} = \inf_{P' \in \mathcal{P}}(J_{P'}(\pi)/J_{P'}(\pi^*_{P'}))$ and conditional value at risk (CVaR), which measures the average reward across episodes with a cumulative reward in the lowest quantiles of the distribution (eg. average reward across the worst 5% of episodes). We argue that, since conformal calibration includes no explicit assumptions about an uncertainty set of transition kernels, these metrics are arbitrary in our setting; we have no strong prior over possible scales of distribution shift, and the range of the uncertainty set can be expanded or narrowed ad hoc to achieve essentially any desired value. The worst-case performance ratio is also challenging to compute and reproduce; it requires training optimal policies for every value of distribution shift evaluated, but it is not simple a priori to evaluate whether a policy is optimal for a given value of shift.

Instead, we follow other work, such as Wang et al. [77], primarily presenting curves over a range of shifts to give a nuanced view of the robustness of calibrated policies. We present a range of values which sufficiently demonstrates the decay from a performant to suboptimal policy. As a summary metric, we report a mean over this range of values, representing an uninformative uniform prior.

The primary metric we consider is the per-seed ratio of the calibrated to uncalibrated mean reward over 250 evaluation episodes. In our setting, this is a natural choice which enables the calculation of meaningful confidence intervals; the robustness of a calibrated policy is dependent on the robustness of the original learned policy (see Figure 5.1). For this reason, the variance of calibrated performance is not a meaningful indicator of the average improvement due to calibration: suppose the calibrated policy were consistently 10% more robust, but the uncalibrated policy exhibited 20% variance across random seeds. Since the calibrated policy "inherits" the variance of uncalibrated policies across seeds, a plot with confidence intervals calculated across seeds would obscure consistent improvement. This motivates a novel metric which averages the improvement due to calibration on a per-seed basis. The expression for the improvement, at a single value of the shift parameter, is given by

$$x = \frac{\mu_{\mathrm{c}}^{(i)} - r_{\min}}{\mu_{\mathrm{u}}^{(i)} - r_{\min}}.$$

Here $\mu_c^{(i)}, \mu_u^{(i)}$ are the mean reward over evaluation episodes of the calibrated and uncalibrated agents respectively. $r_{\min}$ is the minimum possible reward for an episode; the subtraction gives a meaningful ratio on environments with negative rewards (to see why, consider that -50 is a better outcome than -100, but the ratio $\frac{-50}{-100} = 0.5$, which would imply a performance degradation). We then compute the IQM and 95% CI over seeds using 50,000 bootstrap samples. When reporting a single figure to illustrate improvement due to calibration, we report this metric averaged across a range of shift values.

5.2 Results: Performance Evaluation

We first report results illustrating the efficacy of conformal calibration. We explore the behaviour of calibration on the fairly simple Cart Pole environment. Next, we extend the analysis to consider other environments, showing improvements on the Mountain Car and Lunar Lander environments, but no improvement on Acrobot. Lastly, we assess the computational overhead of each method.

5.2.1 Cart Pole

We first evaluate conformal calibration on the Cart Pole environment. Both algorithms increase the average reward over the expected reward, with CC-Disc performing significantly better. Results over multiple seeds in Figure 5.1 show that the calibrated policy is heavily dependent on the uncalibrated policy. Additionally, the robustness of the underlying uncalibrated DQN policy varies significantly with random seed. In some sense this is unsurprising; it is common for the performance of Deep RL algorithms to vary strongly with random seed [2, 26, 25]. Although conformal calibration is deterministic, the dependence on the underlying policy and environmental stochasticity during the observation step means it inherits significant variance.

An interesting behaviour can be observed on seed 4 in Figure 5.1. The uncalibrated policy is suboptimal on the nominal environment, but calibration "corrects" the policy. Both calibration methods boost performance, with CC-NN achieving optimal reward on the nominal environment. Due to the focus on distribution shift we do not focus on this behaviour further in this work, but believe it warrants further investigation. Seed 4 also shows that the calibrated policy can sometimes reduce robustness on specific parameter values.

Calculating the performance ratio and averaging across seeds shows that the robustness gains of CC-Disc increase with greater distribution shift, shown in Figure 5.2. Averaging across shift values, CC-Disc achieves a 69.6 (29.5–250.1)% increase in reward, where the first number is the interquartile mean and the range in brackets indicates an asymmetric bootstrapped 95% confidence interval. CC-NN achieves a 7.4 (-1.3–37.2)% improvement.

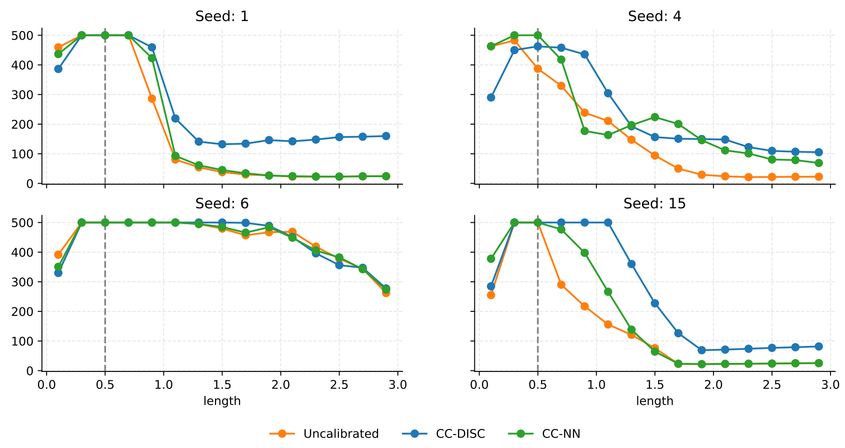

Figure 5.1: Plots of mean reward across DQN agents trained from different random seeds. The vertical dotted line indicates the length of the pole in the nominal environment. The variation in robustness is high, and calibrated policies depend strongly on underlying uncalibrated policies. In some cases, such as the top-right subplot, calibration corrects a suboptimal policy, even on the nominal environment.

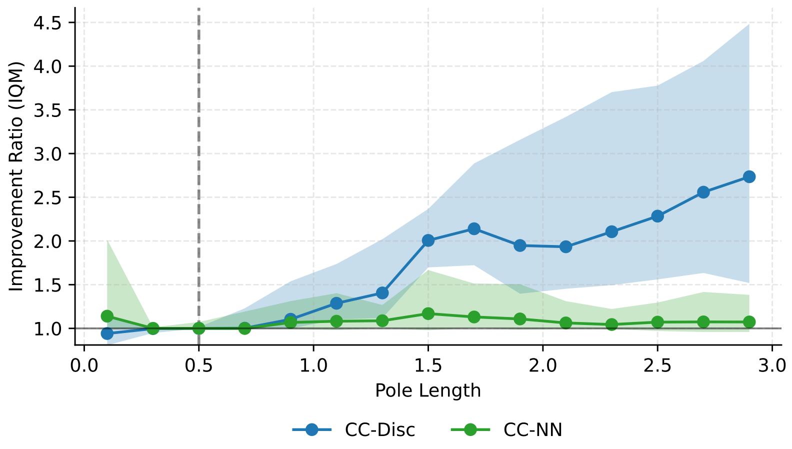

Figure 5.2: On average, the CC-Disc agent achieves a 70% higher reward across a range of pole lengths on Cart Pole. CC-NN achieves a 7% average improvement.

5.2.2 Comparison to Baselines

Comparing baselines, the most notable finding is that the CQL-DQN significantly boosts robustness compared to any other methods. In some respects this is unsurprising—CQL-DQN leverages significant additional computation at train-time, and explicitly introduces conservatism as a training objective. In addition, CQL-DQN required twice as many training steps to reliably achieve optimal performance on the nominal environment, so we might expect that the agent encountered more state-action pairs and learned an accurate Q-function across a broader range of states than the unmodified DQN. We also find that the DDQN does not improve robustness in the face of domain shift, and in fact is less robust than the vanilla DQN.

Comparing uncalibrated to calibrated policies, however, we find that calibration, at least marginally, improves robustness of all three methods, even improving the CQL-DQN on extreme values of the pole length.

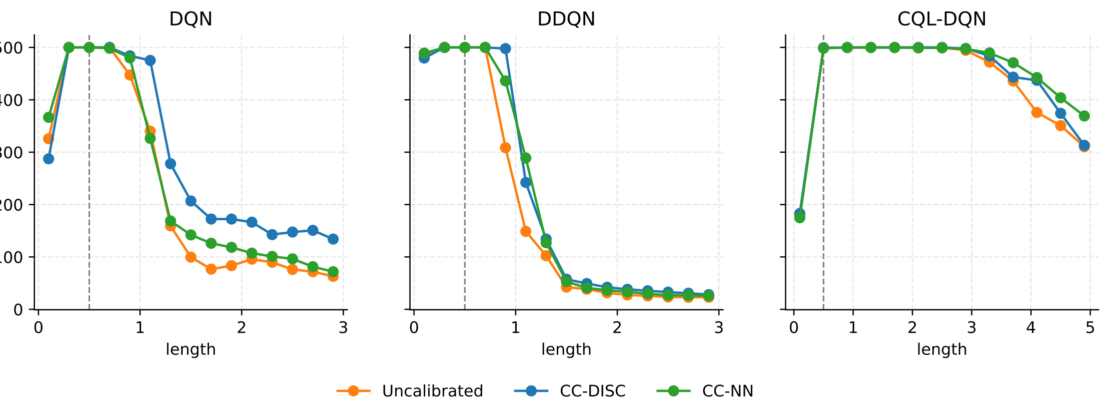

Figure 5.3: Median reward across 25 seeds for each baseline, with and without conformal calibration. CQL-DQN is significantly more robust than DQN and DDQN, and hence is evaluated over a wider range of pole lengths.

5.2.3 Classic Control and Lunar Lander

To assess whether the findings on Cart Pole are consistent across environments, we evaluate calibrated agents on the Acrobot and Mountain Car environments. On Mountain Car CC-Disc improves mean reward by 22.6 (2.9–52.7)%, and CC-NN 7.5 (0.7–24.2)%. On Mountain Car the goal is to pilot drive a vehicle out of a valley, up a hill to a goal. Therefore it is encouraging that calibration mostly improves the policy for higher levels of gravity, for which the environment is more difficult to solve.

In the Acrobot environment, calibration confers no benefits. CC-Disc achieves 0.5 (-0.6–1.7)% improvement. CC-NN slightly reduces robustness, with a change of -5.7 (-9.3—2.4)%. We explore this further in § 5.2.5.

Evaluating CC-Disc for a single seed on the Lunar Lander environment, we observe the consequences of the "curse of dimensionality," as shown in Figure 5.5. With an overly coarse grid the discrete algorithm over-generalises. Robustness improves as the grid becomes more fine. However, this leads to large grids containing millions of features, most of which are never visited

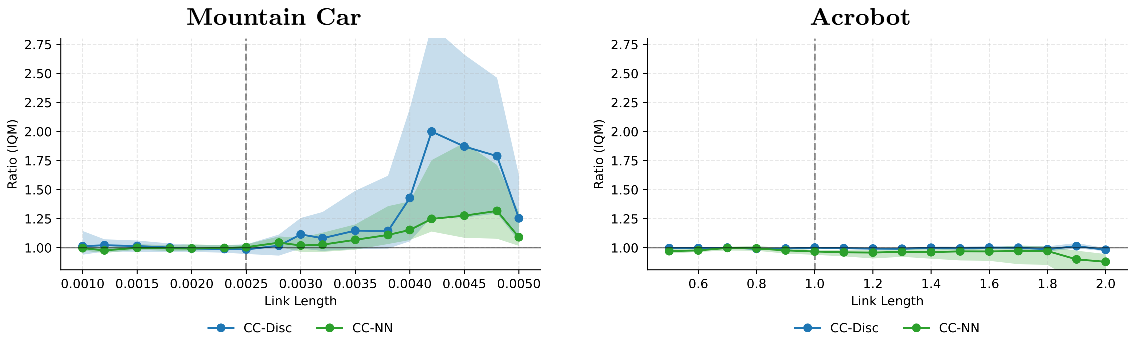

Figure 5.4: Improvement ratios for Mountain Car (left) and Acrobot (right). While the Mountain Car environment benefits as gravity increases, Acrobot sees no benefit over a range of link lengths, with CC-NN slightly *reducing* robustness.

during calibration, leading to a large, inefficient memory overhead.

This motivates the use of CC-NN, which leverages a nearest-neighbours search instead of an explicit state-space discretisation. This method achieves an average improvement of 11 (-13–75)% over the DQN baseline, shown in Figure 5.6.

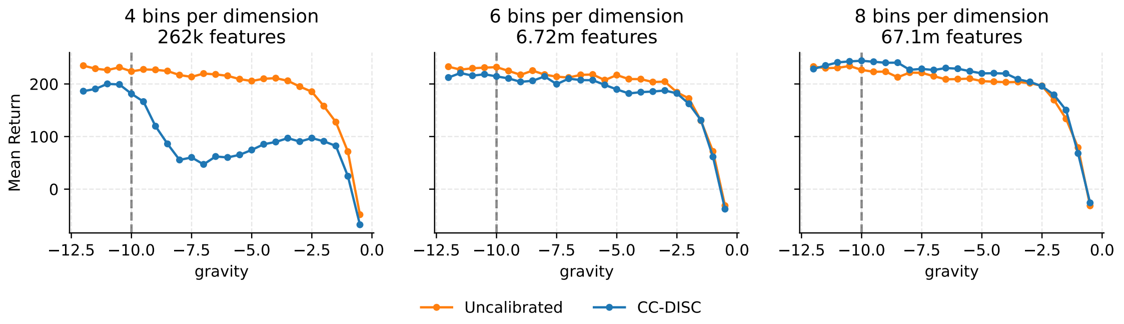

Figure 5.5: Comparing a fixed seed, increasingly fine grids improve the performance of CC-Disc on the Lunar Lander environment. However, since Lunar Lander has an 8-dimensional state space, the required number of grid spaces, and hence memory overhead, explodes to unreasonably large counts.

5.2.4 Computational Overhead

As inference-time modifications to action selection, both CC-Disc and CC-NN incur overhead during action selection. To evaluate this, we run each method for 50 episodes and profile the time spent in action selection, with results shown in Figure 5.7. On average, an unmodified DQN—which has 67,331-69,124 active parameters at inference depending on the input state-space dimension—took 33.8 μs to perform action selection. By comparison, CC-Disc with 6 bins per dim takes 41.4 μs, a 22% increase. This overhead is due to the additional steps of state-action discretisation and indexing into the precomputed offset array. Note that, as discussed in § 5.2.3, the bin count grows exponentially with the state-space dimension, so the memory overhead of CC-Disc grows even though the compute overhead is stable.

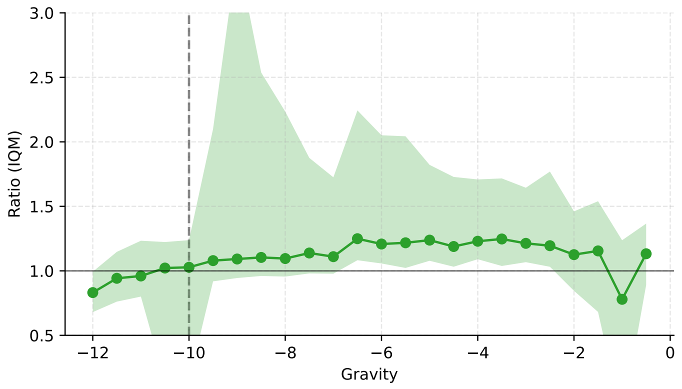

Figure 5.6: CC-NN improvement ratio on Lunar Lander, evaluated over 25 seeds, with 95% confidence intervals per-value. Across all parameter values, the average improvement is 10.8%.

CC-NN, by comparison, incurs significant overhead, taking between 150% to 380% longer for action selection. This is because it constructs a calibration set at inference-time by querying a k-d tree, which regresses to a brute-force search as the dimensionality of the state space increases [5].

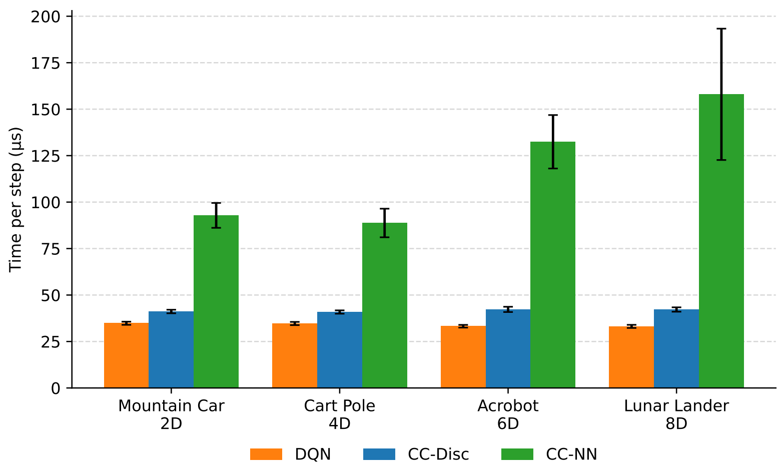

Figure 5.7: Inference compute cost of action selection, comparing a vanilla DQN to calibrated versions using CC-Disc and CC-NN. Tick labels indicate the dimensionality of the state space for each environment. CC-Disc adds a 22% overhead. CC-NN increases inference cost significantly, scaling with state-space dimensionality.

5.2.5 Limited effect on the Acrobot environment

Evaluating all the baselines and calibration algorithms on the Acrobot environment shows that none achieve improved robustness in the face of distribution shift (see Figure 5.8). This suggests that the lack of improved robustness is consistent—no amount of in-distribution conservatism and out-of-distribution scepticism increases robustness. One possible explanation is that the Acrobot environment is based on a double-pendulum, a *chaotic* physical system; small changes to the

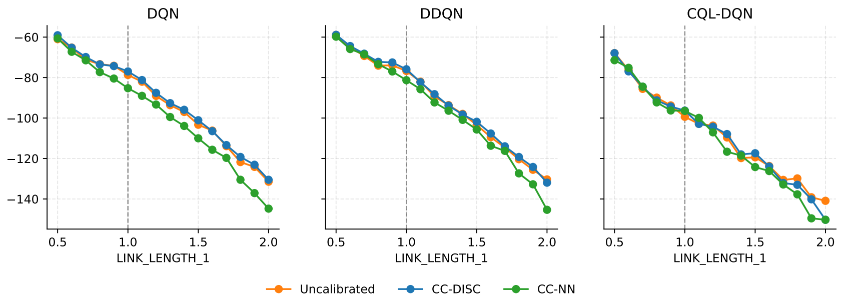

Figure 5.8: On the Acrobot environment, none of the calibration methods or baselines induce significantly more robust policies.

dynamics induce large changes to the trajectories observed. In this case, small distribution shifts move the policy far from the training support, where a constant offset δ_{fallback} across actions fails to modify action-selection. Due to time constraints we were unable to validate this hypothesis, but believe it warrants further research; understanding why calibration fails on Acrobot could yield a deeper understanding of how it induces robustness.

5.2.6 Coverage

To adapt state-action conditioning to continuous state spaces we relaxed the exchangeability assumption, aggregating nearby states (see § 4.3). In theory this exchangeability violation nullifies the coverage guarantees of conformal prediction. To assess the empirical impact of this relaxation, we measure the coverage gap of a calibrated agent on Cart Pole by the following procedure; we apply conformal calibration, then observe an agent on the nominal environment and track how often the observed TD target $r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a')$ is above the calibrated value function (which should be a lower bound with probability $1 - \alpha$). We run the observation stage of conformal calibration for ten million transitions to ensure that almost all grid cells visited have sufficient calibration data. In Figure 5.9 we show an occupancy-weighted histogram where one count represents a single visit to a state-action pair, run over 15 random seeds with $\alpha = 0.1$.

On average, coverage holds roughly, with an average of 89.7% empirical coverage. Most grid spaces cluster around the specified coverage rate, although we see heavier tails than the expected Beta distribution (discussed in § 2.2.3). A small number of grid spaces are severely undercovered, which may be due to grid spaces spanning the score-heterogeneous regions which can be seen in Figure 4.2. It is worth noting that in typical conformal calibration, with the 10,000 calibration transitions used as standard in this thesis, the empirical coverage rate is *higher* than specified, since more grid spaces are covered by the fallback value.

5.3 Results: Design Choices and Hyperparameters

We now present a series of evaluations validating the design choices of the CC-Disc and CC-NN algorithms. We also test the influence of various hyperparameters, illustrating considerations such as the generalisation-vs-locality tradeoff in state-space aggregation.

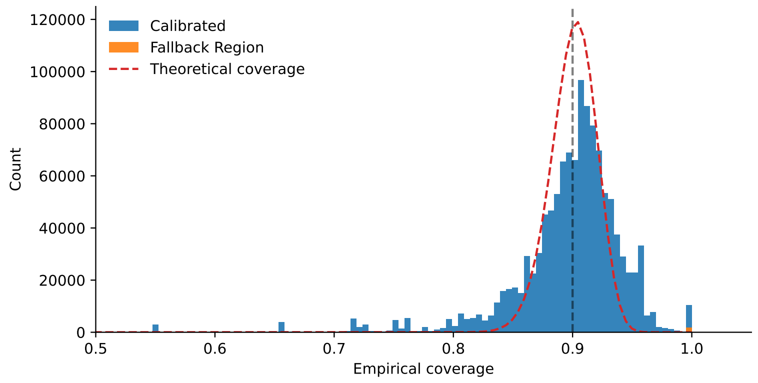

Figure 5.9: Occupancy-weighted histogram of coverage over a uniform grid CC-Disc agent on Cart Pole. The small orange segment with 100% coverage indicates grid spaces with insufficient calibration data which instead use the fallback correction. CC-Disc typically uses variable length calibration sets with a minimum threshold, but we fix them to 250 examples for this experiment.

5.3.1 Discretisation Methods

The defining feature of CC-Disc is state-space discretisation, which enables the application of conformal prediction to continuous state-spaces. State-space discretisation is challenging—even on the same environment, different policies can have radically different state-space visitation patterns, as shown in Figure 4.1. The simplest aggregation scheme is a uniform grid, but this is deeply inefficient; In the Lunar Lander environment a uniform grid has 6.7 million features with only 6 bins per dimension. Given that we typically observe only 10,000 transitions during calibration, almost all of these grid cells are never visited and revert to the constant fallback value.

We would like to find more efficient schemes for state-space discretisation. We explore the use of tile-coding and a data-driven binary partition, as described in § 2.3. The results on Acrobot, Cartpole and Mountain Car are shown in Figure 5.10. For each discretisation method, we vary the following quantities:

- Grid: bins per state dimension
- Tile coding: bins per state dimension, and number of offset grids
- Tree/Binary Partition: maximum depth, the minimum numbers of items per leaf, and the splitting criterion (impurity-based or median).

On average, a simple uniform grid appears to be the most effective, achieving competitive performance on Mountain Car, the highest performance on Cart Pole, and no negative effect on Acrobot. The tree (binary partition) method is the least effective approach, likely due to a propensity to overfit to data, even with a small number of leaves. The tree harms robustness on Acrobot and Mountaincar, with only moderate improvements on Cart Pole, and higher sensitivity to hyperparameter choice than other methods. Tile coding achieves more reliable performance than the tree, but does not substantially improve on the uniform grid.

One reason that coarse grids may outperform fine grids is that broad grid spaces generalise to unseen state-action pairs, where high resolution approaches revert to the fallback value in the

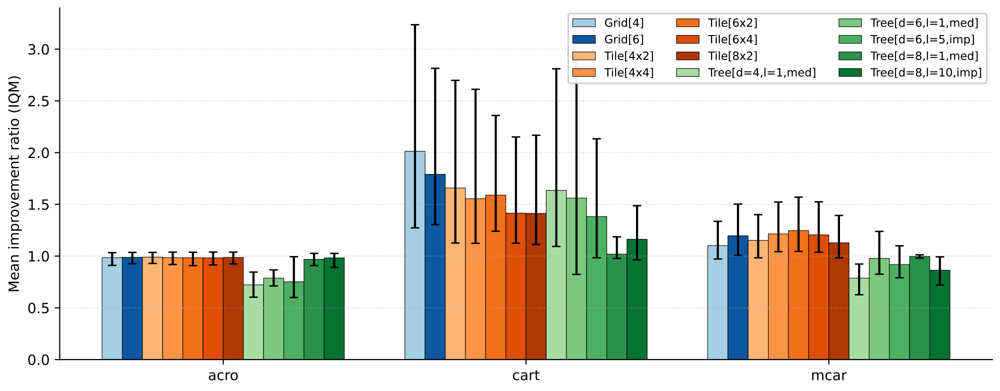

Figure 5.10: Comparison of discretisation methods. Colours indicate the type of discretisation (grid, tile coding, tree), and darker hues indicate more finely-discretised grid spaces.

face of distribution shift. This appears to be a feature of the environment and learned Q-function; the more homogeneous residuals are in state-space, the more a coarse grid will enable useful generalisation.

One interesting note from this experiment is that, on Cart Pole, a four-bin grid is the most performant solution, achieving a 101 (27–224)% performance improvement over the uncalibrated DQN baseline. This is higher than the 70% improvement we achieve using the default 6 bins per dimension. Per-environment tuning of conformal calibration is a promising way to further boost robustness per-environment, although we prioritise default settings to demonstrate the applicability of conformal calibration without significant tuning.

5.3.2 Score function comparison

As discussed in § 4.5.1, we use a signed distance score to lower bound the Q-function without widening prediction sets where the Q-function is an underestimate. To validate this decision, we compare signed and unsigned scores using CC-Disc on Cart Pole. On average the signed score exhibits a higher improvement, although there is significant overlap in confidence intervals, as shown in Figure 5.11. Given the theoretical motivation, we believe the signed score is preferable, but further experiments to certify this with high statistical significance would be valuable.

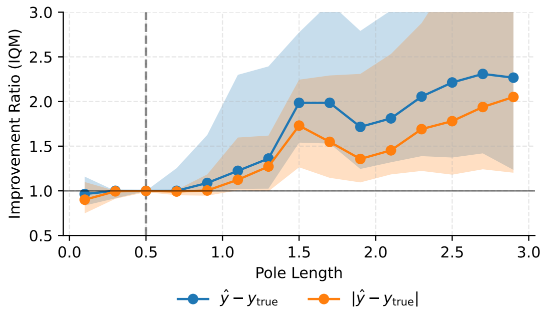

Figure 5.11: Comparing score functions on Cart Pole, the signed score is better on average, but confidence intervals overlap significantly.

5.3.3 Monte Carlo vs TD Return

As discussed in § 4.3, we relax the assumptions of Theorem 1 and use a TD return in the calculation of the nonconformity score ϵ. This simplifies the implementation, but the approximation error due to bootstrapping means we can no longer guarantee that the calibrated policy π_{LB} is induced with respect to a valid lower bound. A comparison of the interquartile mean improvement shows the use of a TD return has no negative impact on performance, shown in Figure 5.12. On the Mountain Car, Acrobot and Cart Pole environments, both targets yield similar results. On Lunar Lander the Monte Carlo return harms robustness—this may be because Lunar Lander is a more complex environment with longer episodes, leading to higher variance Monte Carlo returns, and CC-NN is more vulnerable to the variance in residuals due to the small, fixed-size calibration set.

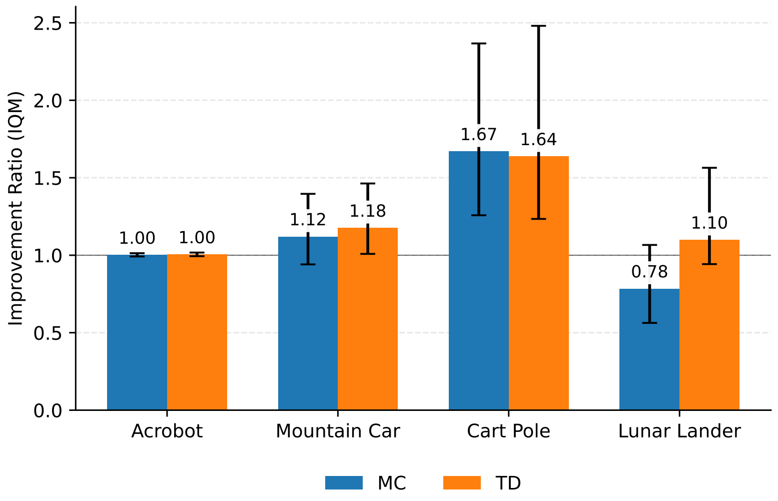

Figure 5.12: Comparison of a Monte Carlo vs TD target for the nonconformity score, on all four environments. The TD target yields consistent gains across environments, where Monte Carlo falters on Lunar Lander. Evaluated using CC-Disc for the Acrobot, Mountain Car and Cart Pole environments, and CC-NN for Lunar Lander.

5.3.4 Effect of α

One of the key hyperparameters in conformal prediction is α, the miscoverage rate. Higher values α give tighter prediction intervals. In conformal calibration, however, the value of α has a limited effect on the robustness improvements, shown in Figure 5.13. The exception is a peak at $\alpha = 0.01$. Possible causes for this low sensitivity could be near-uniform residuals within bins or limited calibration examples, so that different values of α select for similar values of the correction $\delta^{(s,a)}$. Due to time constraints we were not able to explore this behaviour more deeply.

5.3.5 Test-time adaptation

One natural approach to distribution shift is to keep learning by fine-tuning the agent to the test environment. Relatedly, we might ask whether conformal calibration can incorporate information

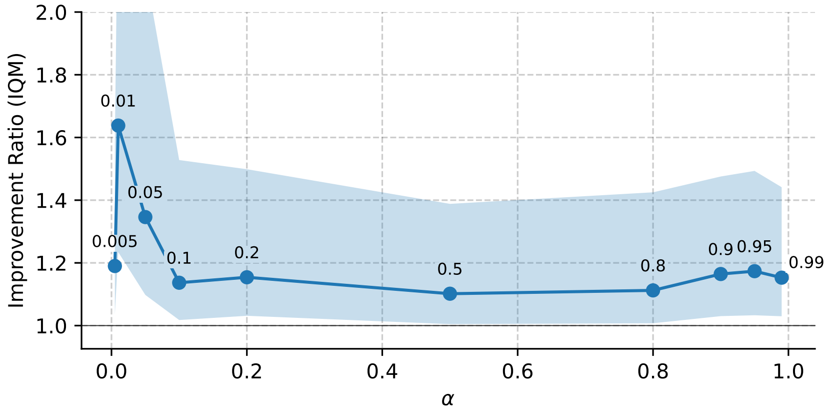

Figure 5.13: Mean improvement ratio on Cart Pole for different values of the miscoverage rate α. Except for a sharp peak at $\alpha = 0.01$, the choice of α does not significantly alter the robustness boost of calibration.

from the test-environment to improve robustness. To investigate this, we test the update proposed by Angelopoulos, Barber, and Bates [4] in CC-Disc, updating the pre-computed corrections in response to test-time observations

$$\delta_{t+1}^{(s,a)} = \delta_t^{(s,a)} + \eta_t\Big(\mathcal{I}\big[r_t + \gamma \max_{a'} Q_\theta(s',a') \leqslant Q_\theta(s,a) - \delta_t^{(s,a)}\big] - \alpha\Big).$$

Testing on Cart Pole and Acrobot demonstrates no clear benefit, as shown in Figure 5.14. The variance of the test-time method appears to be significantly higher, especially for extreme distribution shift on the Acrobot environment. The test-time method does outperform the uncalibrated policy on values close to the nominal environment on Acrobot, which may warrant further research. This is a surprising result; information from the test environment ought to be valuable in devising a more effective policy. A possible explanation is that bootstrapping from unreliable out-of-distribution action-value estimates could lead to spurious updates, but testing with a Monte Carlo return showed that it harmed performance of the test-time method significantly, reducing Cart Pole mean improvement from 70% to 42%.

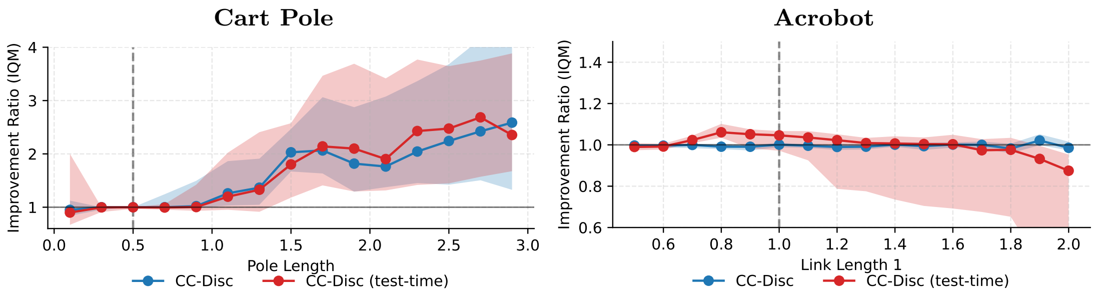

Figure 5.14: Improvement ratios with and without test-time adaptation, with test-time learning rate $\eta = 0.01$.

5.4 Summary

In this chapter we presented empirical results on a mixture of Gymnasium environments including classic control settings and Lunar Lander [73, 13]. We found that, using standard parameters across all environments, CC-Disc boosts robustness on the Cart Pole and Mountain Car environments. We also found that calibration is amenable to combination with conservative approaches like DDQN and CQL-DQN, benefitting robustness. Evaluation on Acrobot shows that no pro-conservatism method reliably boosts robustness. An extension to the Lunar Lander environment, with a larger state-space, demonstrates the shortcomings of the state-space discretisation approach of CC-Disc, motivating the use of CC-NN, which achieves an 11% improvement. However, analysis of compute overhead shows while CC-Disc is fairly cheap at inference, CC-NN is expensive, taking up to 4 times longer on larger state-spaces.

Empirical analysis of the coverage and TD return show that the relaxations we make to the theoretical lower bound derived in Theorem 1 do not significantly affect validity. Assessment of different discretisation approaches for CC-Disc suggests that a simple uniform grid is the most effective choice on the low-dimensional environments tested. Lastly, an exploration of test-time adaptation using adaptive correction scaling shows no clear benefits compared to static pre-computed corrections.

Chapter 6

Conclusion

In this final chapter, we highlight limitations of the work presented and worthwhile avenues for future work, as well as reflecting on the content presented in this thesis.

6.1 Limitations

As a novel method, conformal calibration offers myriad directions for improvement or exploration. In this work, we priotised empirical evaluation and a small number of environments, evaluating across many random seeds to establish that results were not due to random chance. However, as a result there are significant limitations we were unable to address, including:

Dimensionality. Conformal calibration was only tested on fairly small environments with fewer than ten dimensions. Real-world agents have to navigate significantly larger, more complex environments. Although this work demonstrates the impact of calibration on small spaces, it may require significant modifications to scale effectively to these larger spaces.

Theoretical grounding. The gains due to conformal calibration are inconsistent across environments. We see substantial gains on Cart Pole, moderate gains on Lunar Lander and Mountain Car, and no benefit on Acrobot. A more principled understanding of how calibration benefits robustness could boost its reliability on more challenging environments.

Scope of compatible methods. This study focused on value-based methods with discrete action spaces. The results do not yet extend directly to actor–critic or continuous-action settings without additional design.

Hyperparameters and heuristics. Performance is sensitive to design choices such as the miscoverage rate α, discretisation density or k in kNN, distance metrics, and thresholds for out-of-distribution detection. These choices affect both robustness and nominal performance, and poor tuning can reduce the benefits of calibration.

Computational and memory cost. CC-Disc is efficient at inference but memory overhead scales poorly with state dimensionality, while CC-NN scales better with memory but incurs additional lookup time and memory usage. Neither variant is yet optimised for demanding real-time applications.

6.2 Future Work

Adaptive calibration. We found that test-time adaptation yielded no substantial benefit, but this was an unintuitive result, and may be due to the specific choice of adaptation mechanism. Therefore we feel this is an avenue deserving of significantly more research.

Scaling to richer state spaces. More expressive ways of capturing locality could extend calibration to higher dimensions. Learned embeddings, approximate nearest neighbour search, or discretisations based on clustering and representation learning may provide more efficient and flexible alternatives. At present, the method has only been evaluated on relatively small and simple environments, so assessing its effectiveness in larger, more complex domains remains an important next step.

Diagnostics and automation. Developing tools to assess whether calibration is likely to help in a given environment—for example, based on state coverage or sensitivity to distribution shift—could make the method more practical. In a similar vein, calibration improves the robustness of some policies more than others depending on random seed. A deeper theoretical understanding could identify which aspects of an agent's policy make it amenable to calibration.

Additional baselines. We demonstrate that calibration benefits DDQN and Conservative Q-Learning. Further testing with other robustness approaches such as domain randomisation could clarify whether calibration is broadly complementary to other methods, and contribute more generally to a "recipe" for robust agents.

Application to nominal environment. As mentioned in § 5.2.1, calibration can occasionally "correct" suboptimal policies on the nominal environment. This effect is beyond the scope of this work, but may warrant further research.

6.3 Conclusion

This thesis introduced conformal calibration, a post-hoc method for improving the robustness of Deep Q-Learning agents under distribution shift. By adapting conformal prediction to the reinforcement learning setting, we develop a principled way to lower bound the Q-function of a trained DQN without retraining the underlying policy. Two variants are proposed: CC-Disc, which discretises the state–action space, and CC-NN, which applies nearest-neighbour search across the state-space. Both methods demonstrated measurable gains in robustness on classic control benchmarks and Lunar Lander, with CC-Disc excelling in low-dimensional settings and CC-NN providing scalability to richer state spaces.

The theoretical framework grounded these methods in one-sided conformal inference, while empirical results confirmed that calibrated pessimism can mitigate overestimation and improve decision-time reliability. Importantly, these gains were achieved without the computational cost of retraining or extensive hyperparameter search, highlighting post-hoc calibration as a lightweight complement to existing methods such as Double DQN or Conservative Q-Learning.

More broadly, this thesis contributes to an alternative avenue; that reinforcement learning objectives need not be engineered solely through training-time modifications. Instead, post-training calibration offers a practical alternative for retrofitting an existing agent. Recent years have brought the advent of foundation models, large-scale training and compute-intensive meta-learning approaches. In contrast to the typical *tabula rasa* RL paradigm, the ability to tune pre-trained generalist policies may become increasingly important in effective deployment. We hope this work is a small step in this direction.

Bibliography

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. *An Optimistic Perspective on Offline Reinforcement Learning.* 2020. arXiv: `1907.04543 [cs.LG]`. URL: `https://arxiv.org/abs/1907.04543` (cit. on p. 16).

[2] Rishabh Agarwal et al. "Deep Reinforcement Learning at the Edge of the Statistical Precipice". In: *Advances in Neural Information Processing Systems.* Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 29304–29320. URL: `https://proceedings.neurips.cc/paper_files/paper/2021/file/f514cec81cb148559cf475e7426eed5e-Paper.pdf` (cit. on pp. 2, 30, 31, 57).

[3] Anastasios N. Angelopoulos and Stephen Bates. *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification.* 2022. arXiv: `2107.07511 [cs.LG]`. URL: `https://arxiv.org/abs/2107.07511` (cit. on p. 10).

[4] Anastasios Nikolas Angelopoulos, Rina Barber, and Stephen Bates. "Online conformal prediction with decaying step sizes". In: *Proceedings of the 41st International Conference on Machine Learning.* Ed. by Ruslan Salakhutdinov et al. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 1616–1630. URL: `https://proceedings.mlr.press/v235/angelopoulos24a.html` (cit. on pp. 12, 40).

[5] Sunil Arya, David M. Mount, and Onuttom Narayan. "Accounting for boundary effects in nearest neighbor searching". In: *Proceedings of the Eleventh Annual Symposium on Computational Geometry.* SCG '95. Vancouver, British Columbia, Canada: Association for Computing Machinery, 1995, pp. 336–344. ISBN: 0897917243. DOI: `10.1145/220279.220315`. URL: `https://doi.org/10.1145/220279.220315` (cit. on p. 35).

[6] Leemon C. Baird. "Residual Algorithms: Reinforcement Learning with Function Approximation". In: *Proceedings of the Twelfth International Conference on Machine Learning.* Ed. by Armand Prieditis and Stuart J. Russell. San Francisco, CA: Morgan Kaufmann, 1995, pp. 30–37. DOI: `10.1016/B978-1-55860-377-6.50013-X` (cit. on p. 16).

[7] Rina Barber et al. "Conformal prediction beyond exchangeability". In: *The Annals of Statistics* 51 (Apr. 2023). DOI: `10.1214/23-AOS2276` (cit. on p. 24).

[8] M. G. Bellemare et al. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *Journal of Artificial Intelligence Research* 47 (June 2013), pp. 253–279. ISSN: 1076-9757. DOI: `10.1613/jair.3912`. URL: `http://dx.doi.org/10.1613/jair.3912` (cit. on p. 6).

[9] Marc G. Bellemare, Will Dabney, and Rémi Munos. “A Distributional Perspective on Reinforcement Learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 449–458. URL: `https://proceedings.mlr.press/v70/bellemare17a.html` (cit. on p. 12).

[10] Richard Bellman. “A Markovian Decision Process”. In: *Journal of Mathematics and Mechanics* 6.5 (1957), pp. 679–684. ISSN: 00959057, 19435274. URL: `http://www.jstor.org/stable/24900506` (cit. on p. 5).

[11] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782. DOI: `10.1145/361002.361007`. URL: `https://doi.org/10.1145/361002.361007` (cit. on p. 14).

[12] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*. Vol. 3. Optimization and neural computation series. Athena Scientific, 1996, pp. I–XIII, 1–491. ISBN: 1886529108 (cit. on p. 5).

[13] Greg Brockman et al. “OpenAI Gym”. In: (June 2016). DOI: `10.48550/arXiv.1606.01540` (cit. on pp. 29, 41).

[14] Will Dabney et al. “Distributional reinforcement learning with quantile regression”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI’18/IAAI’18/EAAI’18. New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8 (cit. on p. 12).

[15] Thomas G. Dietterich and Jesse Hostetler. *Conformal Prediction Intervals for Markov Decision Process Trajectories*. 2022. arXiv: `2206.04860 [cs.LG]`. URL: `https://arxiv.org/abs/2206.04860` (cit. on p. 19).

[16] Scott Fujimoto, David Meger, and Doina Precup. “Off-Policy Deep Reinforcement Learning without Exploration”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2052–2062. URL: `https://proceedings.mlr.press/v97/fujimoto19a.html` (cit. on p. 16).

[17] A. Gammerman, V. Vovk, and V. Vapnik. “Learning by transduction”. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. UAI’98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., 1998, pp. 148–155. ISBN: 155860555X (cit. on p. 8).

[18] Isaac Gibbs and Emmanuel J. Candès. “Adaptive conformal inference under distribution shift”. In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS ’21. Red Hook, NY, USA: Curran Associates Inc., 2021. ISBN: 9781713845393 (cit. on pp. 12, 14).

[19] Nikunj Gupta, Somjit Nath, and Samira Ebrahimi Kahou. *CAMMARL: Conformal Action Modeling in Multi Agent Reinforcement Learning*. 2024. arXiv: `2306.11128 [cs.LG]`. URL: `https://arxiv.org/abs/2306.11128` (cit. on p. 19).

[20] Danijar Hafner et al. *Mastering Diverse Domains through World Models*. 2024. arXiv: 2301.04104 [cs.AI]. URL: https://arxiv.org/abs/2301.04104 (cit. on p. 56).

[21] Hado Hasselt. “Double Q-learning”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty et al. Vol. 23. Curran Associates, Inc., 2010. URL: https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf (cit. on pp. 1, 17).

[22] Hado van Hasselt, Arthur Guez, and David Silver. *Deep Reinforcement Learning with Double Q-learning*. 2015. arXiv: 1509.06461 [cs.LG]. URL: https://arxiv.org/abs/1509.06461 (cit. on pp. 1, 16, 17).

[23] Hado van Hasselt et al. *Deep Reinforcement Learning and the Deadly Triad*. 2018. arXiv: 1812.02648 [cs.AI]. URL: https://arxiv.org/abs/1812.02648 (cit. on pp. 1, 16).

[24] Sihong He et al. *Robust Multi-Agent Reinforcement Learning with State Uncertainty*. 2023. arXiv: 2307.16212 [cs.LG]. URL: https://arxiv.org/abs/2307.16212 (cit. on p. 18).

[25] Peter Henderson et al. *Deep Reinforcement Learning that Matters*. 2019. arXiv: 1709.06560 [cs.LG]. URL: https://arxiv.org/abs/1709.06560 (cit. on pp. 2, 31).

[26] Riashat Islam et al. *Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control*. 2017. arXiv: 1708.04133 [cs.LG]. URL: https://arxiv.org/abs/1708.04133 (cit. on pp. 2, 31).

[27] Garud Iyengar. “Robust Dynamic Programming”. In: *Math. Oper. Res.* 30 (May 2005), pp. 257–280. DOI: 10.1287/moor.1040.0129 (cit. on p. 2).

[28] L. P. Kaelbling, M. L. Littman, and A. W. Moore. *Reinforcement Learning: A Survey*. 1996. arXiv: cs/9605103 [cs.AI]. URL: https://arxiv.org/abs/cs/9605103 (cit. on p. 4).

[29] Dmitry Kalashnikov et al. *QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation*. 2018. arXiv: 1806.10293 [cs.LG]. URL: https://arxiv.org/abs/1806.10293 (cit. on p. 1).

[30] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: (2009), pp. 32–33. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (cit. on p. 10).

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386 (cit. on p. 1).

[32] Aviral Kumar et al. “Conservative Q-Learning for Offline Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1179–1191. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf (cit. on pp. 1, 17).

[33] Aviral Kumar et al. “Stabilizing off-policy Q-learning via bootstrapping error reduction”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019 (cit. on p. 16).

[34] Yann LeCun et al. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE*. Vol. 86. 11. 1998, pp. 2278–2324. URL: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665` (cit. on p. 10).

[35] Brian Lee and Nikolai Matni. "Single Trajectory Conformal Prediction". In: *2024 IEEE 63rd Conference on Decision and Control (CDC)*. 2024, pp. 3019–3024. DOI: `10.1109/CDC56724.2024.10886644` (cit. on p. 18).

[36] Jing Lei and Larry Wasserman. "Distribution-free Prediction Bands for Non-parametric Regression". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 76.1 (July 2013), pp. 71–96. ISSN: 1369-7412. DOI: `10.1111/rssb.12021`. eprint: `https://academic.oup.com/jrsssb/article-pdf/76/1/71/49514328/jrsssb\_76\_1\_71.pdf`. URL: `https://doi.org/10.1111/rssb.12021` (cit. on pp. 9, 18).

[37] Jing Lei et al. *Distribution-Free Predictive Inference For Regression*. 2017. arXiv: `1604.04173 [stat.ME]`. URL: `https://arxiv.org/abs/1604.04173` (cit. on p. 10).

[38] Jordan Lekeufack et al. *Conformal Decision Theory: Safe Autonomous Decisions from Imperfect Predictions*. 2024. arXiv: `2310.05921 [stat.ML]`. URL: `https://arxiv.org/abs/2310.05921` (cit. on pp. 18, 21).

[39] Sergey Levine. *Reinforcement Learning in the Age of Foundation Models*. Keynote talk, Reinforcement Learning Conference (RLC 2024). Keynote given Aug 10, 2024; video recording available on YouTube. University of Massachusetts Amherst, Amherst, MA, USA: Reinforcement Learning Conference (RLC), Aug. 2024. URL: `https://youtu.be/Az5BoT7lCYo?si=S8Oi13Qb4OkpDirO&t=1272` (cit. on p. 17).

[40] Lars Lindemann et al. *Safe Planning in Dynamic Environments using Conformal Prediction*. 2023. arXiv: `2210.10254 [cs.RO]`. URL: `https://arxiv.org/abs/2210.10254` (cit. on p. 18).

[41] Ning Liu et al. "Learning the Dynamic Treatment Regimes from Medical Registry Data through Deep Q-network". In: *Scientific Reports* 9.1 (2019), p. 1495. DOI: `10.1038/s41598-018-37142-0` (cit. on p. 1).

[42] Yu. A. Malkov and D. A. Yashunin. *Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs*. 2018. arXiv: `1603.09320 [cs.DS]`. URL: `https://arxiv.org/abs/1603.09320` (cit. on p. 15).

[43] Valery Manokhin. *Awesome Conformal Prediction*. Version v1.0.0. "If you use Awesome Conformal Prediction, please cite it as below.". Apr. 2022. DOI: `10.5281/zenodo.6467205`. URL: `https://doi.org/10.5281/zenodo.6467205` (cit. on p. 18).

[44] Huiying Mao, Ryan Martin, and Brian J. Reich. "Valid Model-Free Spatial Prediction". In: *Journal of the American Statistical Association* 119.546 (Jan. 2023), pp. 904–914. ISSN: 1537-274X. DOI: `10.1080/01621459.2022.2147531`. URL: `http://dx.doi.org/10.1080/01621459.2022.2147531` (cit. on p. 25).

[45] Andrew Kachites Mccallum and Dana Ballard. "Reinforcement learning with selective perception and hidden state". AAI9618237. PhD thesis. 1996 (cit. on p. 14).

[46] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 00280836. URL: `http://dx.doi.org/10.1038/nature14236` (cit. on pp. 1, 4, 6).

[47] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning.* 2013. arXiv: `1312.5602 [cs.LG]`. URL: `https://arxiv.org/abs/1312.5602` (cit. on p. 6).

[48] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). arXiv: `1312.5602`. URL: `http://arxiv.org/abs/1312.5602` (cit. on p. 1).

[49] Janosch Moos et al. “Robust Reinforcement Learning: A Review of Foundations and Recent Advances”. In: *Machine Learning and Knowledge Extraction* 4.1 (2022), pp. 276–315. ISSN: 2504-4990. DOI: `10.3390/make4010013`. URL: `https://www.mdpi.com/2504-4990/4/1/13` (cit. on pp. 1, 7, 29, 30).

[50] Stephen M. Omohundro. “Five Balltree Construction Algorithms”. In: 2009. URL: `https://api.semanticscholar.org/CorpusID:61067117` (cit. on p. 14).

[51] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. *Neural Discrete Representation Learning.* 2018. arXiv: `1711.00937 [cs.LG]`. URL: `https://arxiv.org/abs/1711.00937` (cit. on p. 14).

[52] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library.* 2019. arXiv: `1912.01703 [cs.LG]`. URL: `https://arxiv.org/abs/1912.01703` (cit. on p. 29).

[53] Andy Patterson. *PyFixedReps: Python Fixed Representations.* `https://github.com/andnp/PyFixedReps`. Version accessed August 2025. 2025 (cit. on p. 29).

[54] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 29).

[55] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley Series in Probability and Statistics. Wiley, 1994. ISBN: 978-0-47161977-2. DOI: `10.1002/9780470316887`. URL: `https://doi.org/10.1002/9780470316887` (cit. on p. 5).

[56] Antonin Raffin. *RL Baselines3 Zoo.* `https://github.com/DLR-RM/rl-baselines3-zoo`. 2020 (cit. on pp. 29, 54).

[57] Antonin Raffin et al. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: `http://jmlr.org/papers/v22/20-1364.html` (cit. on pp. 2, 29).

[58] Shyam Sundhar Ramesh et al. *Distributionally Robust Model-based Reinforcement Learning with Large State Spaces.* 2023. arXiv: `2309.02236 [cs.LG]`. URL: `https://arxiv.org/abs/2309.02236` (cit. on p. 8).

[59] Martin Riedmiller. “Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method”. In: *Machine Learning: ECML 2005.* Ed. by João Gama et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 317–328. ISBN: 978-3-540-31692-3 (cit. on p. 6).

[60] Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. *Conformalized Quantile Regression*. 2019. arXiv: 1905.03222 [stat.ME]. URL: https://arxiv.org/abs/1905.03222 (cit. on p. 12).

[61] G. Rummery and Mahesan Niranjan. "On-Line Q-Learning Using Connectionist Systems". In: *Technical Report CUED/F-INFENG/TR 166* (Nov. 1994) (cit. on p. 6).

[62] Glenn Shafer and Vladimir Vovk. *A tutorial on conformal prediction*. 2007. arXiv: 0706.3188 [cs.LG]. URL: https://arxiv.org/abs/0706.3188 (cit. on p. 8).

[63] S Sheng et al. "Safe POMDP online planning among dynamic agents via adaptive conformal prediction". In: *IEEE Robotics and Automation Letters* 9.11 (2024), pp. 9946–9953 (cit. on p. 19).

[64] Kegan J. Strawn, Nora Ayanian, and Lars Lindemann. "Conformal Predictive Safety Filter for RL Controllers in Dynamic Environments". In: *IEEE Robotics and Automation Letters* 8.11 (Nov. 2023), pp. 7833–7840. ISSN: 2377-3774. DOI: 10.1109/lra.2023.3322644. URL: http://dx.doi.org/10.1109/LRA.2023.3322644 (cit. on p. 18).

[65] Jiankai Sun et al. "Conformal Prediction for Uncertainty-Aware Planning with Diffusion Dynamics Model". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 80324–80337. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/fe318a2b6c699808019a456b706cd845-Paper-Conference.pdf (cit. on p. 18).

[66] Richard S. Sutton. "Dyna, an integrated architecture for learning, planning, and reacting". In: *SIGART Bull.* 2.4 (July 1991), pp. 160–163. ISSN: 0163-5719. DOI: 10.1145/122344.122377. URL: https://doi.org/10.1145/122344.122377 (cit. on p. 57).

[67] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html (cit. on pp. 1, 4, 6, 13, 14).

[68] Muhammad Faaiz Taufiq et al. *Conformal Off-Policy Prediction in Contextual Bandits*. 2022. arXiv: 2206.04405 [stat.ML]. URL: https://arxiv.org/abs/2206.04405 (cit. on p. 19).

[69] Sebastian Thrun and Anton Schwartz. "Issues in Using Function Approximation for Reinforcement Learning". In: Oct. 1993 (cit. on pp. 1, 16).

[70] Ryan Tibshirani. *Conformal Prediction*. Lecture notes, Advanced Topics in Statistical Learning, Spring 2023. 2023 (cit. on p. 11).

[71] Ryan J. Tibshirani et al. *Conformal Prediction Under Covariate Shift*. 2020. arXiv: 1904.06019 [stat.ME]. URL: https://arxiv.org/abs/1904.06019 (cit. on p. 19).

[72] Josh Tobin et al. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*. 2017. arXiv: 1703.06907 [cs.RO]. URL: https://arxiv.org/abs/1703.06907 (cit. on p. 2).

[73] Mark Towers et al. *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. 2024. arXiv: 2407.17032 [cs.LG]. URL: https://arxiv.org/abs/2407.17032 (cit. on pp. 29, 41).

[74] John N. Tsitsiklis and Benjamin Van Roy. "An Analysis of Temporal-Difference Learning with Function Approximation". In: *IEEE Transactions on Automatic Control* 42.5 (1997), pp. 674–690. DOI: 10.1109/9.580874 (cit. on pp. 1, 16).

[75] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Berlin, Heidelberg: Springer-Verlag, 2005. ISBN: 0387001522 (cit. on pp. 2, 21).

[76] Vladimir Vovk et al. "Nonparametric predictive distributions based on conformal prediction". In: *Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications*. Ed. by Alex Gammerman et al. Vol. 60. Proceedings of Machine Learning Research. PMLR, 13–16 Jun 2017, pp. 82–102. URL: https://proceedings.mlr.press/v60/vovk17a.html (cit. on p. 21).

[77] Kaixin Wang et al. *Bring Your Own (Non-Robust) Algorithm to Solve Robust MDPs by Estimating The Worst Kernel*. 2024. arXiv: 2306.05859 [cs.LG]. URL: https://arxiv.org/abs/2306.05859 (cit. on pp. 21, 30).

[78] C. J. C. H. Watkins. "Learning from Delayed Rewards". PhD thesis. King's College, Oxford, 1989 (cit. on pp. 1, 6).

[79] Tianshu Wei, Yanzhi Wang, and Qi Zhu. "Deep Reinforcement Learning for Building HVAC Control". In: *Proceedings of the 54th Annual Design Automation Conference 2017*. DAC '17. Austin, TX, USA: Association for Computing Machinery, 2017. ISBN: 9781450349277. DOI: 10.1145/3061639.3062224. URL: https://doi.org/10.1145/3061639.3062224 (cit. on p. 1).

[80] XiaoDan Wu et al. "A value-based deep reinforcement learning model with human expertise in optimal treatment of sepsis". In: *NPJ Digital Medicine* 6.1 (2023), p. 15. DOI: 10.1038/s41746-023-00755-5 (cit. on p. 1).

[81] Yifan Wu, George Tucker, and Ofir Nachum. *Behavior Regularized Offline Reinforcement Learning*. 2019. arXiv: 1911.11361 [cs.LG]. URL: https://arxiv.org/abs/1911.11361 (cit. on p. 16).

[82] Yingying Zhang, Chengchun Shi, and Shikai Luo. "Conformal Off-Policy Prediction". In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. Ed. by Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent. Vol. 206. Proceedings of Machine Learning Research. PMLR, 25–27 Apr 2023, pp. 2751–2768. URL: https://proceedings.mlr.press/v206/zhang23c.html (cit. on p. 19).

Appendix A

Pseudocode

We present pseudocode for both CC-DISC and CC-NN. The full code is available at `https://github.com/foty-hub/msc_thesis`.

Algorithm 1 CC-DISC: Discretised Conformal Calibration

```
    Calibration
Require: $\alpha \in [0, 1)$, $n_{\text{calib}}$, $\min_{\text{calib}}$, learned DQN $Q_\theta$, discount factor $\gamma$
 1: Observe $n_{\text{calib}}$ transitions on nominal environment and fill replay buffer $\mathcal{D}_C$
 2: Define DISCRETISE function and n_features                    ▷ eg. bins, tile coding, tree
 3: Initialize array Scores[1..n_features] with empty deques, maxlen $\geqslant \min_{\text{calib}}$
 4: for transition $(s, a, r, s') \in \mathcal{D}_C$ do                          ▷ Build calibration sets
 5:     $\epsilon \leftarrow Q_\theta(s, a) - \big(r + \gamma \max_{a'} Q_\theta(s', a')\big)$
 6:     i ← DISCRETISE$(s, a)$
 7:     Append $\epsilon$ to Scores[i]
 8: for i in n_features do                         ▷ Compute offsets per discretised state
 9:     $m$ = len(Scores[i])
10:     if $m \geqslant \min_{\text{calib}}$ then
11:         $t_\alpha \leftarrow \lceil (m + 1)(1 - \alpha) \rceil / m$
12:         $\delta^{(s,a)} \leftarrow \text{Quantile}_{t_\alpha}\big($Scores[i]$\big)$
13: Set undervisited bin offsets to $\delta_{\text{fallback}} = \max_{s,a} \delta^{(s,a)}$
    Inference - Action Selection
 1: Receive state $s$ from environment
 2: $a = \text{argmax}_{a'} Q_\theta(s, a') - \delta^{(s,a')}$
```

Algorithm 2 CC-NN: Nearest-Neighbour Conformal Calibration

```
    **Calibration**
Require: $\alpha \in [0, 1)$, $n_{\text{calib}}$, learned DQN $Q_\theta$, discount factor $\gamma$, $k$, max distance quantile $\hat{d} \in [0, 1)$
 1: $t_\alpha \leftarrow \lceil (k+1)(1-\alpha) \rceil / k$
 2: Observe $n_{\text{calib}}$ transitions on nominal environment and fill replay buffer $\mathcal{D}_C$
 3: Initialize array Scores[1..n_calib] as zeros
 4: for transition $z_i = (s, a, r, s') \in \mathcal{D}_C$ do          ▷ Score each observed state-action pair
 5:     Scores[i] $\leftarrow Q_\theta(s, a) - \big(r + \gamma \max_{a'} Q_\theta(s', a')\big)$
 6: max_distance $\leftarrow \text{Quantile}_{\hat{d}}\big\{(d_{i \to k})_{i=1}^{n_{\text{calib}}}\big\}$          ▷ $d_{i \to k}$ is distance to the $k^{\text{th}}$ neighbour
 7: Compute $\delta_{\text{fallback}} = \max_{s,a} \delta^{(s,a)}$
    **Inference**
 1: Get state $s$ from environment
 2: for $a \in \mathcal{A}$ do
 3:     distances, indices $\leftarrow \text{ne}_k(s, a)$          ▷ Get k-nearest neighbours
 4:     if max(distances) $>$ max_distance then          ▷ Out-of-distribution
 5:         $\delta^{(s,a)} = \delta_{\text{fallback}}$
 6:     else
 7:         $\delta^{(s,a)} \leftarrow \text{Quantile}_{t_\alpha}\big(\text{Scores[indices]}\big)$          ▷ Compute correction
 8: $a = \text{argmax}_{a'} Q_\theta(s, a') - \delta^{(s,a')}$
```

Appendix B

Experimental Hyperparameters

B.1 DQN hyperparameters

All hyperparameters for experiments are displayed in Table B.1. Hyperparameters are sourced from Stable Baselines Zoo [56]. A learning rate of 3e-4 was used for all DDQN and CQL-DQN runs to achieve more stable learning. The CQL weight parameter β is listed in Table B.1.

Hyperparameter	Acrobot	Cart Pole	Lunar Lander	Mountain Car
training_steps	100,000	50,000	100,000	120,000
learning_rate	6.3e-4	3.0e-4	6.3e-4	4.0e-3
batch_size	128	64	128	128
buffer_size	50000	100000	50000	10000
learning_starts	0	1000	0	1000
gamma	0.99	0.99	0.99	0.98
target_update_interval	250	10	250	600
train_freq	4	256	4	16
gradient_steps	-1	128	1	8
exploration_fraction	0.12	0.16	0.12	0.20
exploration_final_eps	0.10	0.04	0.10	0.07
net_arch	[256, 256]	[256, 256]	[256, 256]	[256, 256]
CQL-β	1.0	1.0	0.1	2.0

Table B.1: Stable baseline 3 DQN hyperparameters used for each environment. Hyperparameters are sourced from Raffin [56].

B.2 Conformal Calibration hyperparameters

Parameters for conformal calibration were kept consistent across environments. A full list of parameters is presented in Table B.3

Hyperparameter	CC-Disc	CC-NN
α	0.01	0.1
Calibration transitions	$10,000$	$10,000$
Score function	Signed	Signed
Return type	TD	TD
Bins per dimension	6	–
Min. calibration examples	50	–
Max. calibration examples	500	–
k	–	50
Max. distance quantile	–	0.9

Table B.2: List of hyperparameters used in experiments for conformal calibration. Hyperparameters were kept consistent across experiments and environments unless noted otherwise. A uniform grid was used as the default discretisation method for CC-Disc.

B.3 Reward Thresholds

In order to identify which seeds to include or exclude when evaluating policies, we use reward thresholds: if an agent achieves a mean reward above this threshold on the nominal environment is it considered to solve the environment.

Gym Environment	Reward Threshold	$r_{\min}$
Acrobot-v1	-100	-200
MountainCar-v0	-110	-200
CartPole-v1	490	0
LunarLander-v3	200	0

Table B.3: Reward thresholds for the different Gym environments used in evaluation.

Appendix C

Conformal Prediction for World Models

One early research direction was the application of conformal prediction for a learned dynamics model, treating next-state predictions as classification over a discrete state space. Using conformal prediction, we calibrated next-state predictions to produce a set containing the true next state with a specified miscoverage rate. This could then be used to induce conservatism by acting with respect to the worst-case outcomes in the prediction set. However we discontinued this research direction for two reasons; due to the variance in calibration set formation, the true coverage rate varies, which is problematic if using a greedy policy with respect to world model predictions—a minor miscalibration leads to extreme conservatism and suboptimal policies. Additionally, most modern work in world modelling, such as Dreamer [20], uses probabilistic world models which output a distribution over next states. These are better suited to sampling predictions than conformal prediction sets which, as a distribution-free method, provide no method for sampling from them. We found that, when evaluated on the gridworld problem FrozenLake, the conformal prediction world model was substantially improved by taking a simple expectation over all next states, rather than acting with respect to the worst outcome in the prediction set. This finding held on both the nominal environment (see Figure C.1), and the test environment after distribution shift.

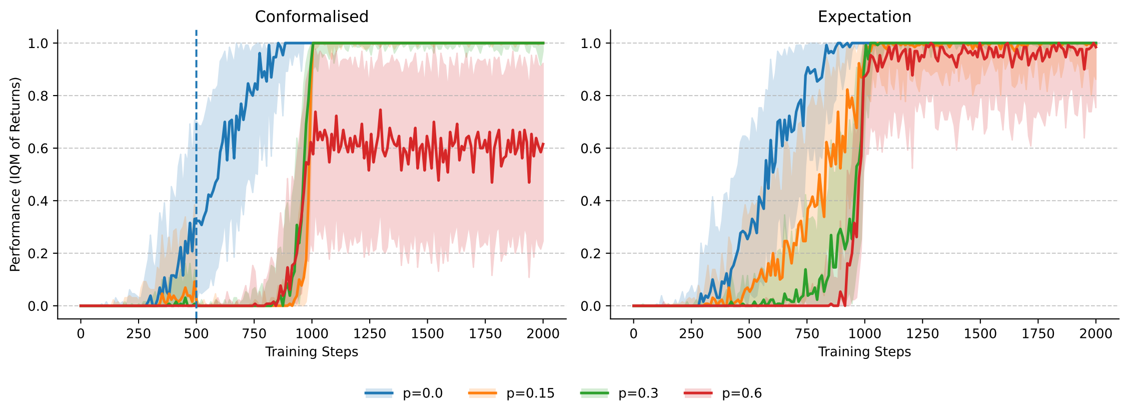

Figure C.1: Training dynamics of a Dyna-V agent [66] using a one-step world model for action selection, evaluated on the FrozenLake environment with varying slip probabilities. The *conformalised* agent uses conformal prediction to pessimistically predict, and avoid, the worst-case outcome. The *expectation* agent takes an expectation over world model predictions. Plots indicate the inter-quartile mean of returns with 95% bootstrapped confidence intervals, as recommended by Agarwal et al. [2].

Appendix D

Generative AI Disclosure

Generative AI was used in this project for the following purposes:

- OpenAI o3, Ai2 Paper Finder and Gemini 2.5-Pro were employed to find papers and related methods alongside normal literature review.
- OpenAI GPT-5 and Codex CLI were used for ancillary coding tasks such as styling plots and adding command-line parsing. Core algorithmic implementations, including baselines, were written without AI assistance.
- OpenAI GPT-5 and Gemini 2.5-Pro were employed to critique drafts of the written thesis and contribute a skeleton structure early. The content itself was written and edited by hand (or by keyboard at least).
- OpenAI GPT-5 was used for some LaTeX formatting.