Attention Over The Past for Data Efficiency in RL

Allen Zheng

CMU-CS-26-114
May 2026

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:

Geoffrey J. Gordon, Chair
Jeff Schneider

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Keywords: Reinforcement Learning, Policy Gradient Methods, Advantage Estimation, Sample Efficiency

Dedicated to my cat Apple

Abstract

We introduce a framework that incorporates *attention over past states* into reinforcement learning (RL) in two complementary ways. First, we use an attention mechanism over a trajectory buffer of previously visited states to construct a *history-aware critic*, replacing the standard neural network critic with an estimate computed as an attention-weighted average over stored values. Second, we replace GAE, which is computed through a single trajectory, with an *attention-weighted advantage* that, in addition to stepping forward through time, also steps according to similar states. States more similar to the current one contribute more to the return estimate, providing a smooth, similarity-weighted alternative to the sequential rollout. Together, these two mechanisms reduce variance by pooling signals across similar states rather than relying on a single trajectory.

Table of Contents

List of Figures

Chapter 1

Introduction

Reinforcement learning (RL) is a general framework for sequential decision-making in which an agent learns to act by interacting with an environment and receiving scalar reward signals [9]. Unlike supervised learning, RL agents receive delayed and often sparse feedback about the consequences of their actions. This combination of exploration, delayed credit assignment, and non-stationary data distributions makes RL challenging.

The field has achieved remarkable empirical successes in recent years. Deep RL agents have learned to play Atari video games at superhuman levels directly from raw pixels [10], mastered the game of Go by combining self-play with tree search [11], and solved complex continuous control tasks in simulated robotics [5, 8]. Beyond games, RL has demonstrated practical impact across a diverse range of domains: robotic manipulation and locomotion [12], chip floor-planning [13], and most notably in recent time, the fine-tuning of large language models via human feedback through horizon 1 RL[14]. The generality of the RL framework has made it a versatile tool.

Despite these successes, deep RL still has some challenges. One is sample inefficiency: deep RL algorithms typically require orders of magnitude more environment interactions than humans to achieve comparable performance. They must discover reward-relevant behaviour from scratch and propagate credit backwards through long sequences of actions. Another notable one is variance in the learning signal: policy gradient methods in particular rely on Monte Carlo estimates of the advantage function that can have extremely high variance, slowing convergence and destabilizing training.

This thesis will addresses both challenges through *attention over past states*. Rather than treating the replay buffer merely as a source of data for updates, we use it as memory that the agent can query at inference time. By attending over stored state embeddings using the Gaussian kernel, the agent can pool value estimates from similar past experiences when

computing its critic, and can replace the single-trajectory rollout of GAE with a similarity-weighted aggregation across the buffer when computing its advantage. We reduce variance by exploiting the fact that states with similar embeddings tend to have similar values and similar returns, effectively averaging over many past experiences rather than relying on any single trajectory or transition.

The remainder of this thesis is structured as follows. Section **2** introduces the necessary background on MDPs, value functions, temporal difference learning, GAE, and policy gradient methods. Section **3** surveys related work on attention in RL, sequence modeling approaches, smoothing kernels for value estimation, and episodic memory. Section **4** presents our method, including the attention-based critic and attention-based advantage estimator. Section **5** describes the experimental setup, and Section **6** presents results. We conclude with a discussion of limitations and future directions.

Chapter 2

Preliminaries

2.1 Markov Decision Processes

We consider RL as an agent learning a policy over an MDP.

Definition 1 (MDP)**.**

A *Markov Decision Process* is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, where:

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition kernel, with $P(s' \mid s, a)$ denoting the probability of transitioning to s' from s after taking action a,
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function,
- $\gamma \in [0, 1)$ is the discount factor, and
- ρ_0 is the initial state distribution.

At each time step t the agent observes state $s_t \in \mathcal{S}$, selects action $a_t \in \mathcal{A}$ according to a *policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, receives reward $r_t = r(s_t, a_t)$, and transitions to $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The interaction produces a *trajectory* $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$.

2.2 Returns and Value Functions

Definition 2 (Discounted Return)**.** *The* discounted return *from time step t is*

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}.$$

For finite-horizon episodes of length T, the sum terminates at $k = T - t$.

Definition 3 (State-Value Function)**.** *The* state-value function *under policy* π *is*

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \;\middle|\; s_t = s\right].$$

Definition 4 (Action-Value Function)**.** *The* action-value function *(Q-function) under policy* π *is*

$$Q^\pi(s,a) = \mathbb{E}_\pi[G_t \mid s_t = s,\, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \;\middle|\; s_t = s,\, a_t = a\right].$$

Definition 5 (Advantage Function)**.** *The* advantage function *measures how much better action* a *is than the average action in state* s*:*

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s).$$

The value functions satisfy the *Bellman equations*:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi,\, s'\sim P}[r(s,a) + \gamma V^\pi(s')]\,,$$
$$Q^\pi(s,a) = r(s,a) + \gamma\, \mathbb{E}_{s'\sim P,\, a'\sim\pi}[Q^\pi(s',a')]\,.$$

The agent's goal is to find a policy π_θ maximising $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[G_0]$.

2.3 Temporal Difference Learning

TD methods learn V^π online by bootstrapping rather than waiting for the full return. **TD(0)** uses the single-step TD error $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ to form the update $V(s_t) \leftarrow V(s_t) + \alpha\,\delta_t$. At the other extreme, **TD(1)** (Monte Carlo) uses the full observed return $G_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$, which is unbiased but high-variance. Between these, **n-step returns** $G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V(s_{t+n})$ mix observed rewards with a bootstrap, and **TD(λ)** takes an exponentially weighted average over all n-step returns via $\lambda \in [0,1]$.

2.4 Policy Gradient Methods

Policy gradient methods solve MDPs by parameterizing the policy π_θ and optimize θ by ascending the gradient of the expected return $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[G_0]$. This makes them naturally applicable to continuous action spaces, stochastic policies, and settings where the policy has structure that is easier to parameterize directly than to recover from a value function.

The Policy Gradient Theorem. The central result underpinning all policy gradient methods is the policy gradient theorem [1], which provides an expression for $\nabla_\theta J(\theta)$ that does not require differentiating through the environment dynamics:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, Q^{\pi_\theta}(s_t, a_t)\right].$$

The term $\nabla_\theta \log \pi_\theta(a_t \mid s_t)$ is the *score function* of the policy: it points in the direction in parameter space that increases the log-probability of action a_t in state s_t. The Q-function weights this direction by the long-run value of the action, so the gradient increases the probability of actions that lead to high returns and decreases the probability of actions that lead to low returns.

Subtracting a *baseline* $b(s_t)$ from the Q-function leaves the gradient unbiased, since

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, b(s_t)] = 0,$$

while potentially reducing variance significantly. Taking $b(s_t) = V^{\pi_\theta}(s_t)$ yields the advantage function $A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)$, which centers the gradient signal around zero and is common in practice:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \hat{A}(s_t, a_t)\right].$$

REINFORCE. The simplest instantiation of the policy gradient theorem is REINFORCE [2], which replaces $Q^{\pi_\theta}(s_t, a_t)$ with the Monte Carlo return G_t sampled from a complete trajectory. The update rule for a trajectory τ of length T is

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, (G_t - b),$$

where b is a scalar baseline, often the mean return over a batch of trajectories. REINFORCE is unbiased, it estimates the true policy gradient in expectation, but suffers from high variance because G_t depends on the entire sequence of stochastic transitions and rewards following t. In practice, REINFORCE can require very large numbers of samples to achieve reliable gradient estimates, particularly in environments with long horizons or sparse rewards where the variance of G_t is large. This motivated the development of actor-critic methods, which trade a controlled amount of bias for substantially lower variance.

Actor-Critic. Actor-critic methods [3] introduce a *critic*, a learned approximation $V_\phi(s)$ of the value function, to replace the high variance Monte Carlo return with a lower variance bootstrapped estimate. The actor π_θ is updated via the policy gradient, and the critic V_ϕ is trained to minimize the mean-squared TD error. The critic can be used to calculate a number of quantities that slot into the policy gradient. One common one is advantage, which is estimated using the one-step TD residual:

$$\hat{A}_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) = \delta_t.$$

This estimate has much lower variance than the Monte Carlo return because it depends only on the single transition (s_t, a_t, r_t, s_{t+1}) rather than the entire future trajectory. The cost is bias: δ_t is only an unbiased estimate of the true advantage when $V_\phi = V^{\pi_\theta}$, which is rarely the case in practice. The actor and critic are often represented by neural networks that share a common trunk with separate output heads, allowing them to share learned features while maintaining separate objectives.

The actor-critic framework is the foundation for most modern deep policy gradient algorithms. A key practical challenge is that the two networks must be updated in a coordinated way: if the critic is inaccurate, the actor gradient is biased. If the actor changes too rapidly, the critic's value estimates become stale. Both GAE (Section **2.5**) and our attention-based advantage estimator operate within this framework, replacing the one-step TD residual with richer advantage estimates that better balance bias and variance.

Proximal Policy Optimisation. A fundamental difficulty in policy gradient methods is determining the appropriate step size for policy updates. Large steps can catastrophically degrade performance because the policy gradient is a local approximation that may not hold far from the current parameters. On the other hand small steps waste samples. Trust Region Policy Optimisation (TRPO) [5] addressed this by constraining updates to a trust region defined by a KL-divergence bound:

$$\max_\theta \; \mathbb{E}_t\Big[\rho_t(\theta)\,\hat{A}_t\Big] \quad \text{subject to} \quad \mathbb{E}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,\|\, \pi_\theta(\cdot \mid s_t)]] \le \delta,$$

where $\rho_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ is the importance ratio between the new and old policy. This provides strong monotonic improvement guarantees but requires computing second-order natural gradient steps via conjugate gradient, which is computationally expensive and difficult to implement correctly.

Proximal Policy Optimisation (PPO) [8] retains the spirit of TRPO while drastically simplifying the implementation. Rather than enforcing the KL constraint via second-order optimization, PPO clips the importance ratio directly in the objective:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\Big(\rho_t(\theta)\,\hat{A}_t,\; \text{clip}(\rho_t(\theta),\, 1-\epsilon,\, 1+\epsilon)\,\hat{A}_t\Big)\Big],$$

where ϵ is a small hyperparameter, typically in the range $[0.1, 0.2]$. The clipping has an intuitive interpretation. If $\hat{A}_t > 0$, the update tries to increase ρ_t (making the action more likely), but isn't incentivized to increase it beyond $1+\epsilon$. If $\hat{A}_t < 0$, the update tries to decrease ρ_t but is also not incentivized from decreasing it below $1-\epsilon$. In both cases, the clip removes the incentive for excessively large policy changes, attempting to keep the new policy within a trust region of the old one without requiring an explicit constraint.

PPO collects a batch of experience under the current policy $\pi_{\theta_{\text{old}}}$, computes advantage estimates for all transitions in the batch, and then performs multiple epochs of minibatch gradient ascent on $\mathcal{L}^{\text{CLIP}}$ before discarding the data and collecting a fresh batch. The total loss also includes an entropy bonus to encourage local exploration:

$$\mathcal{L}^{\text{PPO}}(\theta, \phi) = \mathcal{L}^{\text{CLIP}}(\theta) + c\,\mathbb{E}_t[H[\pi_\theta(\cdot \mid s_t)]]\,,$$

where c is a weighting coefficient and H denotes the entropy of the policy. PPO combines strong empirical performance with simplicity, stability, and low computational overhead, and has become the de facto standard policy gradient algorithm across continuous control, game playing, and large language model fine-tuning.

2.5 Generalized Advantage Estimation

The central challenge in policy gradient estimation is controlling the bias-variance trade-off in the advantage estimate. Schulman et al. [7] address this by giving a new advantage estimate. Using an approximate value function $V \approx V^{\pi,\gamma}$, they define the *TD residual*

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t),$$

which is an unbiased estimate of $A^{\pi,\gamma}(s_t, a_t)$ when $V = V^{\pi,\gamma}$.

k-step advantage estimators. The k-step return gives a family of advantage estimators of increasing horizon:

$$\begin{aligned}
\hat{A}_t^{(1)} &= \delta_t^V = -V(s_t) + r_t + \gamma V(s_{t+1}),\\
\hat{A}_t^{(2)} &= \delta_t^V + \gamma\delta_{t+1}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}),\\
&\vdots\\
\hat{A}_t^{(k)} &= \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}).
\end{aligned}$$

Shorter horizons ($k = 1$) have low variance but high bias when V is inaccurate while longer horizons reduce bias at the cost of higher variance.

Generalized Advantage Estimator. GAE takes an exponentially weighted average over all k-step estimators, controlled by $\lambda \in [0, 1]$:

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = (1-\lambda)\sum_{k=1}^{\infty} \lambda^{k-1} \hat{A}_t^{(k)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l\, \delta_{t+l}^V.$$

Special cases. Setting $\lambda = 0$ recovers the one-step TD advantage (low variance, potentially high bias):

$$\hat{A}_t^{\mathrm{GAE}(\gamma,0)} = \delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t).$$

Setting $\lambda = 1$ gives the full Monte Carlo advantage (unbiased, high variance), since the sum telescopes:

$$\begin{aligned}\hat{A}_t^{\mathrm{GAE}(\gamma,1)} &= \sum_{l=0}^{\infty} \gamma^l \, \delta_{t+l}^V \\ &= \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) = G_t - V(s_t).\end{aligned}$$

Values of $\lambda \in (0, 1)$ smoothly interpolate between these extremes. GAE as an advantage estimate can be slotted into PPO to give a common general algorithm to tackle many MDPs.

Chapter 3

Related Work

3.1 Attention Mechanisms in Reinforcement Learning

Early work incorporating attention into deep RL focused on spatial attention over visual observations. These methods demonstrate the use of attention to process states can lead to improved efficiency and generalization capacity, though the benefits can be inconsistent and varies among environments. Sorokin et al. [15] introduced the Deep Attention Recurrent Q-Network (DARQN), which augments DQN with soft and hard attention mechanisms applied to CNN feature maps before being consumed by an LSTM. The attention is used to focus on relevant spatial regions of the game screen, and its visualisation reveals that the agent learns to track game-relevant objects such as ball trajectories and enemy positions. While DARQN demonstrated improvements over DQN on several Atari games, the gains were inconsistent, which the authors attributed to the limited context window of the recurrent component. Manchin et al. [16] argued that prior failures to make attention work in RL stemmed from hand-crafted attention targets that imposed preconceived notions of importance. Their method instead applies self-attention directly over stacked frames, exploiting the Markovian structure inherent in the state input without supervised signals. This self-supervised formulation produced the first consistent improvements from attention in the Arcade Learning Environment, including several new state-of-the-art results, and yielded interpretable visualizations of the agent's multiple simultaneous foci of attention. Similarly, Zambaldi et al. [17] proposed Relational Deep RL, which applies multi-head self-attention over a set of spatial entities extracted from the current observation, enabling the agent to reason about pairwise relations between objects in a scene.

A parallel line of work recasts RL as a sequence modeling problem, leveraging the Transformer architecture [6] and its capacity for long-range attention. Chen et al. [18] introduced the Decision Transformer, which abstracts the RL problem as conditional sequence model-

ing: a causally masked GPT-style Transformer is conditioned on desired return-to-go, past states, and past actions, and autoregressively predicts the next action. By framing the problem this way, the Decision Transformer bypasses explicit value function estimation and policy gradient computation entirely. Despite this simplicity, it matches or exceeds state-of-the-art offline RL baselines on Atari and OpenAI Gym tasks. Concurrently, Janner et al. [19] proposed the Trajectory Transformer, which takes a more model-based approach: a Transformer is trained to jointly model distributions over states, actions, and rewards, and beam search over this model serves as the planning algorithm. The Trajectory Transformer demonstrates the flexibility of the sequence modeling perspective, performing well across long-horizon dynamics prediction, goal-conditioned RL, and offline RL. These approaches use the Transformer's attention mechanism globally over an entire trajectory context for sequence-level prediction.

3.2 Kernel Smoothing for Value Function Approximation

Domingues et al. [20] study the exploration-exploitation dilemma in finite-horizon reinforcement learning over continuous state-action spaces endowed with a metric, providing the first finite-time regret analysis for kernel-based RL with smoothing kernels. They introduce Kernel-UCBVI, a model-based optimistic algorithm that constructs non-parametric estimators of both the reward function and the transition kernel from collected experience, then runs value iteration on the resulting estimated MDP while adding an upper confidence bonus to drive exploration of poorly covered regions of the state-action space. Under Lipschitz continuity of rewards and transitions, and for problems with K episodes of horizon H, they prove a regret bound of $O(H^3 K^{2d/(2d+1)})$, where d is the covering dimension of the joint state-action space. They also demonstrate the effectiveness of Kernel-UCBVI experimentally, showcasing the power of smoothing kernels for MDPs.

3.3 Nearest Neighbours and Episodic Memory in RL

A related family of methods uses nearest-neighbour lookup over a memory of past experiences to improve sample efficiency. These methods, broadly termed *episodic control*, are motivated by the hippocampal episodic memory system, which enables rapid one-shot learning in biological agents. Blundell et al. [21] introduced Model-Free Episodic Control (MFEC), which maintains a non-parametric table of the best return observed from each state-action pair, indexed by a compressed state representation. At execution time, the agent selects actions by performing a k-nearest-neighbour lookup in this table. MFEC attains strong performance

significantly faster than DQN on several Atari tasks, demonstrating that fast retrieval of past experiences can compensate for the slow parametric learning of standard deep RL.

Pritzel et al. [22] extended this with Neural Episodic Control (NEC), replacing the random-projection state embeddings of MFEC with a learned convolutional encoder, and using a differentiable neural dictionary (DND) to enable end-to-end gradient-based training. The DND stores state embeddings as keys and Q-value estimates as values, retrieved via a weighted k-NN query. NEC achieves faster learning than DQN and A3C, and represents a step toward making episodic retrieval fully differentiable.

Lin et al. [23] proposed Episodic Memory Deep Q-Networks (EMDQN), which uses episodic memory as a supervisory signal during DQN training rather than as the primary policy. An episodic return is computed by k-NN lookup and used as a regression target to regularise the Q-network, stabilising training and reducing overestimation. EMDQN achieves comparable or superior performance to DQN with only one-fifth of the interactions.

Shen and Yang [24] provided theoretical grounding for nearest-neighbour value estimation, proposing the Nearest Neighbour Actor-Critic (NNAC). They prove finite-sample regret bounds for a nearest-neighbour function approximator under Lipschitz continuity assumptions, and show empirically that plugging the NN estimator into existing deep RL methods accelerates training.

Chapter 4

Method

4.1 Overview

We start from the standard actor-critic framework and propose two modifications, both built on attention over a shared experience buffer:

1. **Attention-based Critic** — a value function that attends over a buffer of previously encountered states to produce a richer baseline.
2. **Attention-based Return and Advantage** — a similarity-weighted return that replaces the sequential trajectory rollout of GAE with a weighted sum over past states, where each state's reward contributes proportionally to its similarity to the current state.

4.2 Overall Architecture

Our system maintains two learned components:

(a) a **policy network** $\pi_\theta(a \mid s)$, and

(b) a **shared state encoder** $\phi_\psi : \mathcal{S} \to \mathbb{R}^d$, used by both the critic and the return estimator.

A rolling *state buffer* $\mathcal{B} = \{(s_i, a_i, r_i, s'_i, v_i)\}_{i=1}^{N}$ stores the N most recent transitions together with their associated scalar values v_i. The critic $\hat{V}^{\text{att}}$ and the attention-based advantage $\hat{A}_\lambda^{\text{att}}$ are both computed by attending over $\mathcal{B}$ using embeddings from ϕ_ψ. No separate value network exists.

4.3 Attention-Based Advantage

Definition.
All states are embedded by the shared encoder ϕ_ψ:

$$\mathbf{e}_i = \phi_\psi(s_i) \quad \forall\, s_i \in \mathcal{B}.$$

Similarity is measured by the RBF kernel

$$k(\mathbf{e}_i,\, \mathbf{e}_j) = \exp\!\big(-\beta_{ker}\, \|\mathbf{e}_i - \mathbf{e}_j\|^2\big)\,,$$

Let $\mathbf{e}'_i = \phi_\psi(s'_i)$ denote the embedding of the next state stored in the buffer. Transition attention weights are computed as

$$w'_{i,j} = \frac{\exp(\beta_{ret}\, k(\mathbf{e}'_i, \mathbf{e}_j))}{\sum_{j=1}^{N} \exp(\beta_{ret}\, k(\mathbf{e}'_i, \mathbf{e}_j))},$$

i.e. each next state s'_i attends over the buffer to produce a soft-transition to similar states. As $\beta_{ret} \to \infty$ the weights collapse to hard next-state transitions assuming the next states are not in the buffer more than once.
The n-step attention-based advantages are then

$$\begin{aligned}
\hat{A}^1(s_t, a_t) &= -V(s_t) + r_t + \gamma \sum_i w'_{t,i}\, V(s'_i), \\
\hat{A}^2(s_t, a_t) &= -V(s_t) + r_t + \gamma \sum_i w'_{t,i} \left(r_i + \gamma \sum_j w'_{i,j}\, V(s'_j) \right), \\
\hat{A}^3(s_t, a_t) &= -V(s_t) + r_t + \gamma \sum_i w'_{t,i} \left(r_i + \gamma \sum_j w'_{i,j} \left(r_j + \gamma \sum_k w'_{j,k}\, V(s'_k) \right) \right), \\
&\vdots
\end{aligned}$$

with the general recursion $\hat{A}^n(s_t, a_t) = -V(s_t) + r_t + \gamma \sum_i w'_{t,i} \Big(V(s'_i) + \hat{A}^{n-1}(s_i, a_i) \Big)$.
Attention-GAE. Taking an exponentially weighted moving average over the n-step estimates gives the attention-based analogue of GAE:

$$
\begin{aligned}
\hat{A}^{\lambda}(s_t, a_t) &= (1-\lambda)\left(\hat{A}^1(s_t, a_t) + \lambda \hat{A}^2(s_t, a_t) + \lambda^2 \hat{A}^2(s_t, a_t) + \dots\right) \\
&= (1-\lambda)\left(-(V(s_t) + r_t)(1 + \lambda + \lambda^2 + \dots) + \gamma \sum_i w'_{t,i}\right. \\
&\quad \left. + \left(r_i(\lambda + \lambda^2 + \dots)\gamma^2 \sum_j w'_{i,j}(\lambda^2 + \lambda^3 + \dots)\right)\right) \\
&= -V(s_t) + r_t + \lambda\gamma \sum_i w'_{t,i}\left(r_i + \lambda\gamma \sum_j w'_{i,j}\left(r_j + \lambda\gamma \sum_k w'_{j,k}(\cdots)\right)\right),
\end{aligned}
$$

computed efficiently via the recursion $\hat{A}^{\lambda}(s_t, a_t) = \delta_t^{\text{att}} + \lambda\gamma\, \hat{A}^{\lambda}(s_{t+1}, a_{t+1})$, where $\delta_t^{\text{att}} = r_t + \gamma \sum_i w'_{t,i} V(s'_i) - V(s_t)$ is the attention-weighted TD error. When $w'_{t,i}$ recovers hard transitions, this reduces exactly to standard GAE.

Attention-returns.

A similar transformation can also be applied to the Monte Carlo returns,

$$
r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots
$$

giving

$$
\hat{Q}(s_t, a_t) = r_t + \gamma \sum_i w'_{t,i}\left(r_i + \gamma \sum_j w'_{i,j}\,(r_j + \dots)\right)
$$

Since the transition probabilities are weighted by next state, we consider only deterministic MDPs for the rest of this thesis. Non-deterministic MDPs can also be considered with proper representation over state-action pairs (s_t, a_t), though for deterministic MDPs this is unnecessary as s_{t+1} is a representation for state-action pairs.

Implicit MDP

Consider an MDP that replaces the transition function P with $\tilde{P}$ where instead of transitioning s_t, a_t to s_{t+1} it transitions to each state s'_i with probability $w'_{t,i}$. That is

$$
\tilde{P}(s'|s_t, a_t) = \sum_i w'_{t,i} \mathbb{1}_{s'}(s_i)
$$

The attention-GAE/returns are simply the corresponding values over this implicit MDP. Notice this introduces some bias into the system for lower variance in the estimates. β_{ret} gives control over this trade-off, as $\beta_{ret} \to \infty$ the bias is reduced for more variance.

4.4 Attention-Based Critic

Architecture. For a given state s_t not in the buffer, encode $\mathbf{e}_t$ as

$$\mathbf{e}_t = \phi_\psi(s_t).$$

and similar to the attention-based advantage, use a softmax over the RBF kernel with temperature β_{critic} to get the attention weights:

$$w_{t,i} = \frac{\exp(\beta_{critic}\, k(\mathbf{e}_t, \mathbf{e}_i))}{\sum_{j=1}^{N} \exp(\beta_{critic}\, k(\mathbf{e}_t, \mathbf{e}_j))}.$$

β_{critic} and β_{ret} may want to be different since for the attention-based advantage the next state is in the buffer with it, while here the given state s_t is almost never exactly in the buffer in continuous environments. The critic is then the attention-weighted sum of the scalar values stored in the buffer:

$$V^{\text{att}}(s_t) = \sum_{i=1}^{N} w_{t,i}\, v_i,$$

where v_i can represent any return estimate (Monte Carlo, GAE, etc.) stored at collection time. No value network is used. ϕ_ψ is the only learned component of the critic.

Training. The encoder is trained by minimizing

$$\mathcal{L}_{\text{critic}} = \mathbb{E}\Big[\big(\hat{y}_t - V^{\text{att}}(s_t)\big)^2\Big],$$

where $\hat{y}_t$ is the regression target (e.g. Monte Carlo return or GAE estimate).

Note this overall technique is similar to the Differentiable Neural Dictionary used in NEC [22]. Due to the lower dimension of the MDPs we evaluate over, we don't require a nearest neighbour tree. In addition, we use a Gaussian kernel to ensure the representation learned is compatible with the attention-based advantage. We primarily use the attention critic to learn the representation for attention GAE, as it is the only source of a gradient signal.

These two ideas are compatible with any policy gradient algorithm. To evaluate them concretely, we present a modification of PPO incorporating both the attention-based critic and the attention-based advantage estimate.

4.5 Full Algorithm

Algorithm 1 Attention-Augmented PPO

Initialize policy π_θ, shared encoder ϕ_ψ, buffer $\mathcal{B} \leftarrow \emptyset$
for each iteration **do**
    Generate trajectories $\mathcal{T}$ by rolling out π_θ in the environment
    for each $s_t \in \mathcal{T}$ **do**
        Embed: $\mathbf{e}_t = \phi_\psi(s_t)$, $\mathbf{e}_i = \phi_\psi(s_i)$ for all $s_i \in \mathcal{B}$
        Compute $V^{\text{att}}(s_t) = \sum_i w_{t,i}\, v_i$ *(Section 4.4)*
        Compute $\hat{A}_t^\lambda$ via attention-GAE recursion *(Section 4.3)*
        Append $(s_t, a_t, r_t, s_{t+1}, v_t)$ to $\mathcal{B}$ (evict oldest if full)
    end for
    Update θ with PPO clipped objective using $\hat{A}_t^\lambda$
    Update ψ by minimising $\mathcal{L}_{\text{critic}}$
end for

Chapter 5

Experimental Setup

5.1 Environments

Experiments are conducted on the MuJoCo continuous control suite, working over 9 different MDPs with observation space dimensions ranging from $4 - 348$. Each environment requires a robot to complete some tasks to obtain a reward.

5.2 Evaluation Protocol

Each configuration is run with 30 random seeds. Performance is reported as mean $\pm$ standard error of episode return. Policies are trained on 100,000 total time steps, with each update based on 2000 steps. The replay buffer is capped to a size of 6000 transitions, for a total of 6000 possible values to query for the critic and 8000 possible values for attention-based advantage. We compare the results of Attention-Augmented PPO with base PPO.

5.3 Hyperparameters

We kept $\beta_{ret} = 100$ and $\beta_{critic} = 10$ over all environments. Although the optimal temperature coefficients likely differ depending on the structure of the state distribution and policy value function, we kept it the same to showcase the flexibility for fixed parameters.

Chapter 6

Results

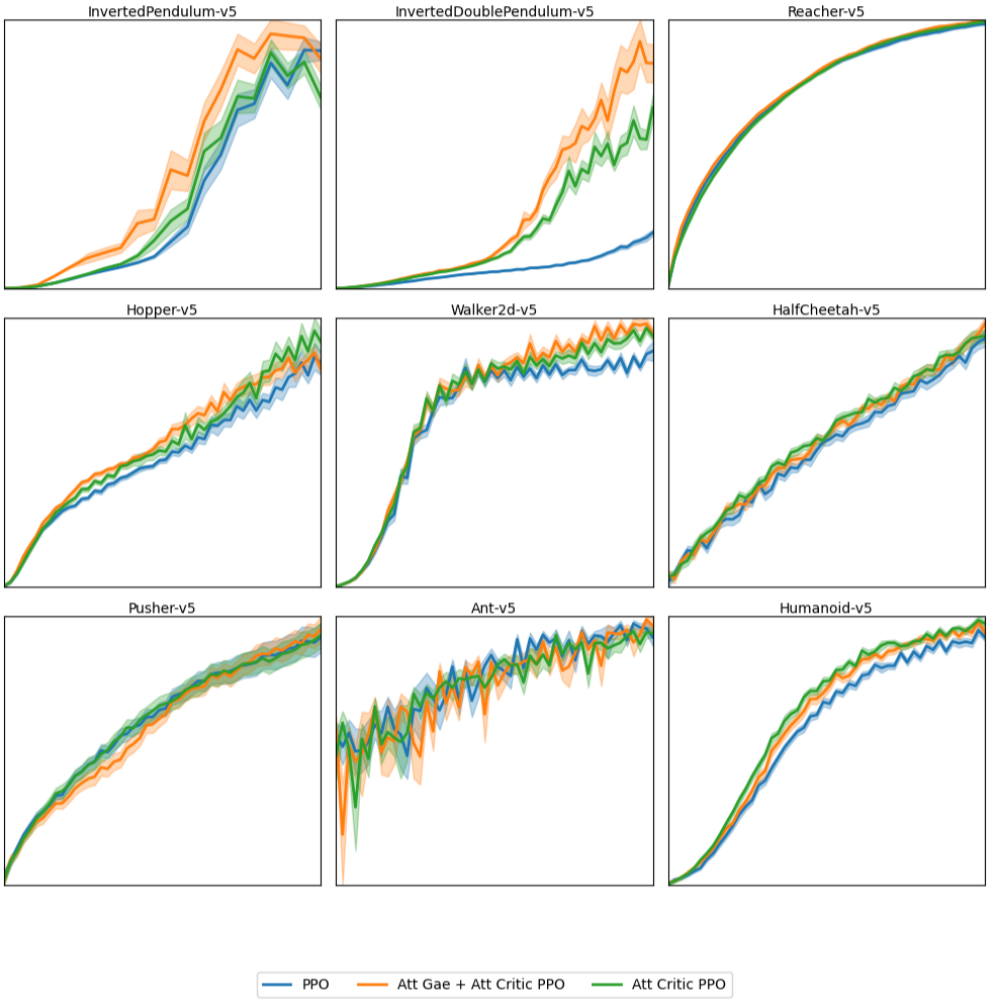

Figure 6.1. Learning curves for OpenAI Gym MuJoCo continuous control tasks. The shaded area denotes one standard error over 30 trials. Curves are not smoothed.

Results are shown in **Figure 6.1**. Here we compare the orange line (Att Gae + Att Critic PPO) to the blue line (base PPO). We see that the Attention PPO (both Gae and Critic)

performs better on most environments, notably having a significant improvement over the *Inverted Pendulum* MDPs. It performs slightly worse at the start in *Pusher* and has no notable differences in *Ant* or *Reacher*, with slight improvements in the rest.

Using only the attention critic (green line) performs worse than using the full system with both critic and GAE improvements over *Inverted Pendulum* MDPs, though closer over the others. The environments where removing attention GAE hurts are relatively low in dimension (4-9) and also low in exploration, which gives two possible explanations for this performance difference. Overall, theses additions can provide significant benefits over certain environments with small performance increases over others but never a notable performance decrease.

6.1 Analysis

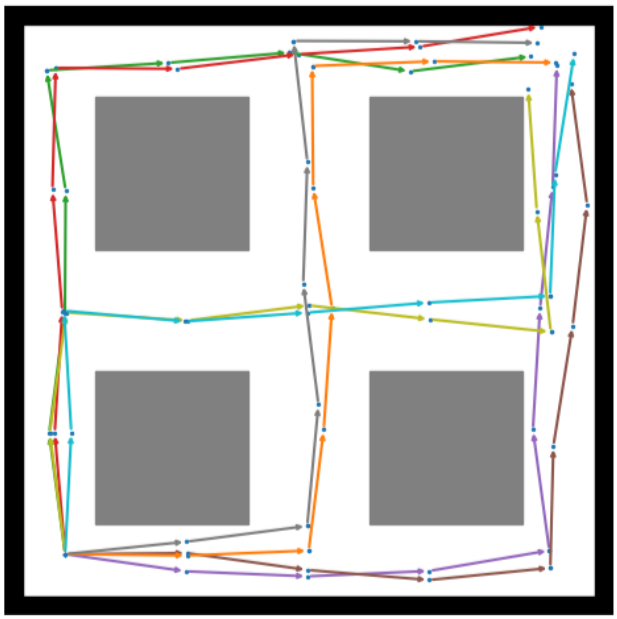

Figure 6.2. 8 Trajectories in toy environment

To illustrate the stitching property of the attention-based returns we consider a small 2D environment with continuous states. The world has 4 large rocks in the middle that block paths, with starting states at the south west corner and a goal at the north east corner. We ran a policy that, when at a crossroad picked either north or east with approximately even probability. In a corridor, the policy kept going in that direction. We ran this policy to collect 8 trajectories, with **Figure 6.2** showing all 8 trajectories in different colors. **Figure 6.3**

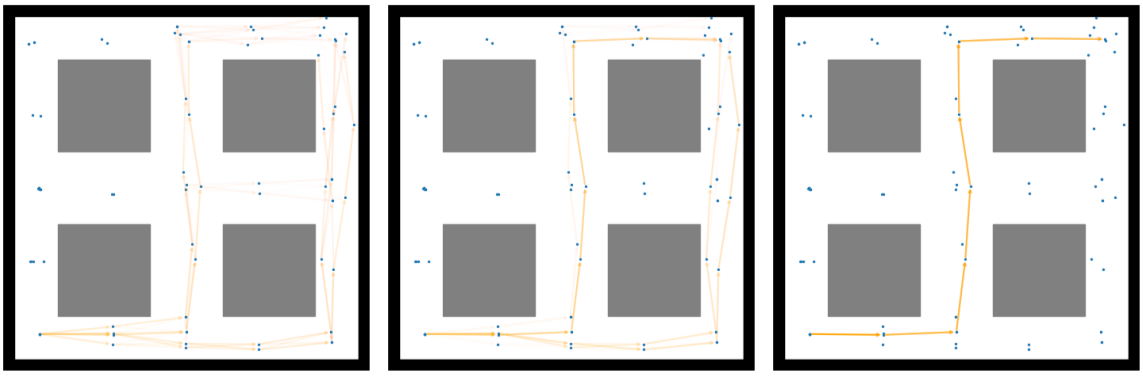

Figure 6.3. Attention return trees for beta = 10, 30, 1000 respectively

shows the resulting tree structure from the attention-based returns. Over **Figure 6.2** we focus on the first step of the orange trajectory, with all others potential attention targets. With high beta the return depends only on the original trajectory. With a medium sized beta it also pays attentions to nearby trajectories that have followed a different path. With a low beta it can stitch together different trajectories. Notably it sees a path that goes east, north, east, north that the policy had not actually experienced. It accurately notes that as a possibility according to the policy (though puts a low weight on it). To test the amount

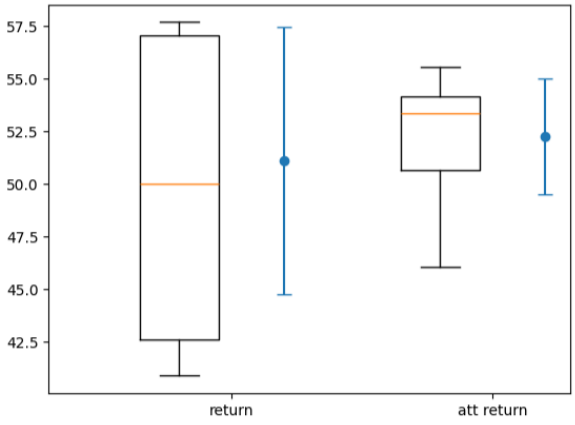

Figure 6.4. Distribution of returns against attention returns in toy environment

of bias introduced for the amount of variance reduced we took a transition (s_t, a_t, s_{t+1}, r_t) and found the distribution on the return G_t and the distribution on the returns based on attention. The return G_t was found by running the policy starting at s_{t+1} 1000 times. The distribution on the returns based on attention is found by running the policy starting from the initial state 7 times, and starting at s_t 1 time. Then calculating the returns based on

attention using these 8 total trajectories. This was also done 1000 times. **Figure 6.4** plots the two side by side with both a box plot and plotting mean $\pm$ std. We notice that although the attention-based returns does introduce bias, it significantly reduces the variance. The 1 standard deviation interval for attention-based return is contained within the interval for the return.

6.2 Bounds on Bias

Overall the attention-based returns gives another hyperparameter for the bias-variance trade-off. We characterize the bias introduced under the attention weighted transitions. The main benefit of the attention-based returns comes when similar states give similar returns. For states far away, although the returns will differ more, the probability weight on them will also be lower. We bound the error introduced by averaging over the value function, then propagate through the Bellman operator to give a final bound.

Theorem 1. *For a policy π, let $\mathcal{B}$ be the buffer with size n collected under π. Let $V(s)$ be the value function of the original MDP and let $\tilde{V}(s)$ be the value function of the derived MDP by replacing the transitions with $\tilde{P}(s'|s_t, a_t) = \sum_i w_{t+1,i} \mathbb{1}_{s'}(s_i)$ where*

$$w_{t,i} = \frac{e^{\beta e^{-d(s_t,s_i)^2}}}{\sum_j e^{\beta e^{-d(s_t,s_j)^2}}}$$

Assume the state norms are bounded by D and the policy value function is L-Lipschitz. Then for any state $s_i \in \mathcal{B}$

$$|V(s_i) - \tilde{V}(s_i)| \leq \frac{\gamma LDn}{(1-\gamma)\sqrt{2e\beta}}$$

Lemma 1. *For states $s_0, s_1, \ldots, s_n$ then*

$$\left| V(s_0) - \sum_i w_{0,i} V(s_i) \right| < \frac{LDn}{\sqrt{2e\beta}}$$

Proof. Let $d_i = d(s_0, s_i)$. From L-Lipschitz we get $|V(s_i) - V(s)| \leq L d_i$ Notice that

$$w_{0,i} \leq \frac{e^{\beta e^{-d_i^2}}}{e^{\beta}} = e^{\beta(e^{-d_i^2} - 1)}$$

since s_0 is in the buffer. Using Jensen's over the function $e^{-x} - 1$ we get that for any value $x \in [0, N]$ then

$$\frac{N-x}{N}(e^0 - 1) + \frac{x}{N}(e^{-N} - 1) = \frac{x}{N}(e^{-N} - 1) \geq e^{-x} - 1$$

Substituting $x = d_i^2$ and $D^2 = N$ gives

$$e^{-d_i^2} - 1 \leq -d_i^2(1 - e^{-D^2})/D^2 < -d_i^2/D^2$$

giving

$$w_{0,i} \leq e^{-\beta d_i^2/D^2}$$

for

$$w_{0,i}|V(s_0) - V(s_i)| \leq Ld_i e^{-\beta d_i^2/D^2}$$

Taking a derivative over the expression $d_i e^{-\beta d_i^2/D^2}$ gives that is maximized at $d_i = \frac{1}{\sqrt{2\beta/D^2}}$ for a value of $\frac{D}{2e\beta}$ giving

$$w_{0,i}|V(s_0) - V(s_i)| \leq \frac{LD}{\sqrt{2e\beta}}$$

Thus summing all n values together gives $|V(s_0) - \sum_i w_{0,i}V(s_i)| < \frac{LDn}{\sqrt{2e\beta}}$. □

Returning to Thm 1: The value functions satisfy the Bellmen equations

$$V(s_i) = r(s_i) + \gamma V(s_{i+1})$$

$$\tilde{V}(s_i) = r(s_i) + \gamma \sum_j w_{i+1,j}\tilde{V}(s_j)$$

Thus taking their difference

$$\begin{aligned}
|V(s_i) - \tilde{V}(s_i)| &= |\gamma V(s_{i+1}) - \gamma \sum_j w_{i+1,j}\tilde{V}(s_j)| \\
&= \left|\gamma\left(V(s_{i+1}) - \sum_j w_{i+1,j}V(s_j)\right) + \gamma\left(\sum_j w_{i+1,j}(V(s_j) - \tilde{V}(s_j))\right)\right| \\
&\leq \frac{\gamma LDn}{\sqrt{2e\beta}} + \gamma\left(\sum_j w_{i+1,j}|V(s_j) - \tilde{V}(s_j)|\right)
\end{aligned}$$

Vectorization gives $\|V(s) - \tilde{V}(s)\|_\infty \leq \frac{\gamma LDn}{\sqrt{2e\beta}} + \|\gamma P(V(s) - \tilde{V}(s))\|_\infty$ where P is the transition matrix. Since the transition matrix is stochastic then $\|P\|_\infty = 1$ so

$$\|V(s) - \tilde{V}(s)\|_\infty \leq \frac{\gamma LDn}{\sqrt{2e\beta}} + \gamma\|P\|_\infty\|V(s) - \tilde{V}(s)\|_\infty = \frac{\gamma LDn}{\sqrt{2e\beta}} + \gamma\|V(s) - \tilde{V}(s)\|_\infty$$

giving

$$|V(s_i) - \tilde{V}(s_i)| \leq \frac{\gamma LDn}{(1-\gamma)\sqrt{2e\beta}}$$

The bound is inversely related to $\sqrt{\beta}$ ensuring that for a given policy over an MDP, a large enough beta can limit the amount of bias introduced to any ϵ. For smoother value functions, the bound decreases as other states are more informative about the current state.

In the worst case having more states in the buffer can increase bias, though in practice most states are in better locations (either far away enough to not matter or close enough to help).

6.3 Limitations

To determine when these modifications give the most benefit, we give some limitations of this method

- Attention over a large buffer can be slow without efficient approximate nearest-neighbour indexing.
- The approach assumes that similar observations correspond to similar value, which may be violated in POMDPs. We can correct for this by using a state, such as a learned state from all past observations though the best method remains an open question.
- A large enough state space can cause attention to fail at capturing similar states. We hope that more advanced state representation learning can mitigate this problem, but investigating this question is future work.

Chapter 7

Conclusion

We proposed two methods of incorporating knowledge from past states into RL based on attention. Attention based GAE averages different trajectories together while the Attention Critic gives an informed value estimate. We found that together they can give a significant boost in performance for environments where attention is efficient at retrieving similar states, such as in low dimension. These techniques improve data efficiency by stitching together the past and re-using the information gathered. We expect that difficulties in higher dimensional spaces can be overcome with better representations, and we hope future work can extend these results to more complex environments.

*

Bibliography

[1] Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12.

[2] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256.

[3] Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12.

[4] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *Proceedings of the International Conference on Machine Learning*, pp. 1928–1937.

[5] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. *Proceedings of the International Conference on Machine Learning*, pp. 1889–1897.

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

[7] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

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

[9] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[10] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[11] Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[12] Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40.

[13] Mirhoseini, A., Goldie, A., Yazgan, M., et al. (2021). A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212.

[14] Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35.

[15] Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., and Ignateva, A. (2015) Deep Attention Recurrent Q-Network arXiv:1512.01693.

[16] Manchin, A., Abbasnejad, E., van den Hengel, Anton. (2019) Reinforcement Learning with Attention that Works: A Self-Supervised Approach arXiv:1904.03367.

[17] Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O., Battaglia, P. (2018) Relational Deep Reinforcement Learning arXiv:1806.01830

[18] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Igor, M (2021) Decision Transformer: Reinforcement Learning via Sequence Modeling arXiv:2106.01345

[19] Janner, M., Li, Q., Levine, Sergey (2021) Offline Reinforcement Learning as One Big Sequence Modeling Problem arXiv:2106.02039

[20] Darwiche Domingues, O,. Ménard, P., Pirotta, M., Kaufmann, E., Valko, M (2020) Kernel-Based Reinforcement Learning: A Finite-Time Analysis arXiv:2004.05599

[21] Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Z Leibo, J., Rae, J., Wierstra, D., Hassabis, D (2016) Model-Free Episodic Control arXiv:1606.04460

[22] Pritzel, A., Uria, B., Srinivasan, A., Puigdomènech, A., Vinyals, O., Hassabis, D., Wierstra, D., Blundell, C (2017) Neural Episodic Control arXiv:1703.01988

[23] Lin, Z., Zhao, T., Yang, G., Zhang, L (2018) Episodic Memory Deep Q-Networks arXiv:1805.07603

[24] Shen, J., F. Yang, L (2021) Theoretically Principled Deep RL Acceleration via Nearest Neighbor Function Approximation arXiv:1805.07603