University of Nevada, Reno

Attention-Enabled Reinforcement Learning for Control of Scalable Multi-Agent Systems

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering

by

Joseph A. Dailey

Dr. Hao Xu / Thesis Advisor

May, 2024

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Joseph Dailey

entitled

Attention-Enabled Reinforcement Learning for Control of Scalable Multi-Agent Systems

be accepted in partial fulfillment of the
requirements for the degree of

Master of Science

Hao Xu, Ph.D.
Advisor

M. Sami Fadali, Ph.D.
Committee Member

Hung La, Ph.D.
Graduate School Representative

Markus Kemmelmeier, Ph.D., Dean
Graduate School

May, 2024

ABSTRACT

Multi-agent reinforcement learning has been the subject of considerable interest and effort for its potential as a means of specifying behavior policies for multi-agent systems. Specifically, on-policy algorithms based on gradient estimation have achieved state-of-the-art performance on end-to-end control problems once thought beyond the scope of machine learning methods.

In seeking to apply the benefits of MARL to practical control of physical autonomous systems, we must begin to account for three factors: (1) the presence of other autonomous elements in the environment configuration space, which may or may not be amenable to coordination; (2) non-idealities in sensing the configuration of the environment (e.g. locality and limited observability); and (3) variability in the number of sensed dynamical elements.

The attention head, a relational ML structure originally designed for extraction of abstract natural language features, is structurally well suited to addressing these challenges. This work presents a systematic argument and framework for the use of attention as an input layer to enable learning of neural policy models in changing multi-agent environments which are not well-suited to other representations.

In benchmark physical simulations, it is shown that such models achieve competitive performance on cooperative and mixed cooperative/competitive MAS control tasks as the agent cohort is arbitrarily changed. Prospective advantages of attention-based architectures

for physical autonomous systems in select applications are discussed, as well as drawbacks associated with explainability and potential for emergent behavior.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1. Background

In the study of autonomous systems, *multi-agent systems* (MAS) are those systems with multiple independent controllable elements ("agents") interacting simultaneously, and in a decentralized fashion, with an uncontrolled environment [1]. At each instant in time, each agent has an *observation* (often a vector) representing some information about the state of the environment, and performs some vector *action* (e.g. a command velocity) to perturb the environment in order to advance a design goal.

MAS designs have proven powerful for solving problems in defense, surveying, logistics and other areas demanding distributed action over large regions, with potential non-local gains for locally suboptimal agent behavior [1]. In recent years, this potential has grown still greater with the advent of multi-agent reinforcement learning (MARL) [2].

The promise of MARL is to apply machine learning methods to the discovery of multi-agent scenarios where effective policies have proven inaccessible in explicit form. Much of the research in this area has targeted algorithm design, leveraging self-play or game-theoretic reward attribution to shape the optimization trajectory [3]. However, there has been comparatively little work on correct architectural design for MARL policy models.

1.2. Problem description

Many of the problems discussed in the MAS/MARL literature to date have the property that the number of control-relevant dynamical elements may vary in the same setting from episode to episode, or from moment to moment ("polyvalence" for convenience). For

example, a polyvalent “dogfight” environment might consist of two opposing teams of fighter pilots in an otherwise empty space, the “red” team having some indefinite number of pilots *m* and the “blue” team having *n*. At any time, any of the pilots may be shot down or involved in a collision, removing it from the environment. Such environments have unique demands for autonomous control policies, namely (1) behavior that is in some sense at least “weakly optimal” for varying numbers of observed states, within reasonable bounds (“variadic”); and (2) invariant to the permutation of these states as represented.

This work concerns the specification of decentralized control policies for agents in such settings, specifically by machine learning to capture efficient group behavior that is not feasible to express analytically. The potential for unexpectedly sophisticated and efficient strategies for underspecified tasks to emerge from such stochastic optimization is illustrated well by works such as [4], [5]. Section 2.1 provides some explanation of the policy gradient methods that enable translation of the classical discrete RL paradigm to continuous control-domain problems. In 2.2, some consideration is given to the subsequent problem of imputing reward to individual agents in a MAS so as to direct the training trajectory toward cooperative behavior, after [6]. 2.3 provides a rationale for attention as a compelling numerically-optimizable structure for capturing polyvalent environment states by analogy to the natural language processing problem. Finally, 2.4 discusses existing applications of this technology in the growing MARL literature to date, and identifies three key areas for contribution: (1) development of flexible monolithic policies that are amenable to taking any (constrained) number of states as inputs, (2) implementation of continuous rather than discretely selected control actions, and (3) full decentralization of the group strategy into individual attention-based policy models.

CHAPTER 2. LITERATURE REVIEW

2.1. Overview of reinforcement learning

Reinforcement learning is typical of machine learning methods in that it concerns the numerical optimization of a model mapping input data to output data. The essential distinctions are merely:

1. The input data represent some state property of an *environment* which is to be acted upon, and;
2. The output data represent an action which an *agent* (i.e. a system under the control of the model) is to exert upon the environment.

The data are thus interactive; that is, the state trajectory of the environment depends on the history of actions. For this reason there is generally little or no *a priori* information about the distribution of environment states. Instead, a *reward* function associates a (possibly random) real number to each state-action pair, which is to be maximized (cf. minimization of loss or error). This quantity should, by design, represent a desirable control outcome; a simple case might be the negative of the Euclidean distance to some goal state.

2.1.1. Discrete RL

If states and actions are discrete and finite, the RL problem reduces to the specification of a Markov decision process. The reward associated with each state-action can be stored and used as inputs to a Bellman equation.

The control policy is then a matter of selecting the action associated with the greatest value of Q for each state. Q-learning is known to converge to the optimal policy when the necessary assumptions are met [7].

However, these constraints are quite stringent, and very rarely apply to the control of physical systems. In particular, if the environment is non-stationary, the value associated with a state-action pair may change. Additionally, the restriction of the control policy to an MDP obviates the possibility of continuous control, making it unsuitable for e.g. vehicular navigation.

2.1.2. Continuous RL

Although the focus of this work is on policy architectures suitable for MARL settings, rather than the learning algorithms themselves, it becomes necessary to establish how we arrive at useful values of the attention matrices. To that end, we first briefly discuss the single-agent case of a *policy gradient* optimization method, and then show how the multi-agent generalization follows as the problem of estimating an optimal attribution of marginal reward to many such agents.

By optimizing "on-policy," we make it our objective to maximize the expectation of reward in the model parameters θ, over some period of interest, i.e.

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{\tau} r_t\right]$$

$$\boldsymbol{a}_t = \pi_\theta \circ \omega(\boldsymbol{s}_t)$$

$$(\boldsymbol{s}_{t+1}, r_t) = \phi(\boldsymbol{s}_t, \boldsymbol{a}_t)$$

where:

- r_t is the reward associated with the state-action pair at time *t;*
- $\boldsymbol{s}_t$ is the environment state vector;
- ω is a mapping from the environment state space onto the agent observation space, which may be the identity, introduce noise, mask unobserved states, etc.;
- $\boldsymbol{a}_t$ is the action taken at time *t;*
- π_θ is the parametric control policy with parameters θ;
- ϕ is a discrete state transition function representing the environment dynamics, which may be abstracted away in model-free learning.

The general gradient ascent optimization step for this treatment of the problem is, ideally,

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}\left[\sum_{t=0}^{\tau} r_t\right]$$

with the learning rate $\alpha \ll 1$ a hyperparameter determined empirically. However, unless the environment dynamics are trivial, neither the expected reward nor its gradient are obtainable in explicit form. Instead, we must rely on a subjective *advantage* $\hat{A}_t$, a measure of the gain in reward associated with the state transition at time *t*, according to some idealized "critic" model. Since $\hat{A}_t$ should be defined to account for future gains due to the present control action (multiplied by some discount scalar), we can rewrite the gradient ascent step as

$$\theta \leftarrow \theta + \alpha \nabla_\theta \widehat{\mathbb{E}}_t\left[\hat{A}_t\right]$$

with $\widehat{\mathbb{E}}_t$ the moving subjective expectation given timesteps $0, \ldots, t$.

Although still a challenge, it is considerably easier to estimate a critic function for $\hat{A}_t$ given several samples of the state-reward history; this is the principal matter of the "proximal policy optimization" algorithm family [8].

This is sufficient for single-agent environments. In MARL, however, we should like to reward each agent (i.e. update its weights) only in proportion to its contribution to the control objective.

2.2. MARL approaches

In 2002, Parsons and Woldridge [9] suggested the following reasonable criteria for decision protocols in multi-agent systems, based on a model of agents as players in a game-theoretic system; here they are presented with some modification of language to the present context:

Table 2.1: Game-theoretic properties of an idealized multi-agent system policy

Criterion	**Explanation**
Guaranteed success	Guaranteed convergence to an action for all players
Maximizing social welfare	Maximization of reward summed over all agents
Pareto efficiency	No deviation in policy will increase one agent's reward without decreasing another's; no net utility is simply "left on the table"
Individual rationality	No deviation benefits any agent individually
Stability	No agent is incentivized to change policy if the others hold constant
Simplicity	Policy is computationally tractable
Distribution	No single points of failure; the behavior of the cohort is well-defined in the absence of information from any one among them

Certain among these are satisfied more-or-less automatically by the structure of the MARL environment: "guaranteed success" equates to correctness of the program implementing the observation/action/reward loop, and "simplicity" exists in proportion to the resources available for the implementation. Others we aim here to determine by experiment: "stability" and "individual rationality," at least in the case where agents have homogeneous policies, are both indicated by the empirical convergence of the reward function over many iterations of the "game." The criterion of "distribution" is what will be addressed by

training over many different cardinalities of the agent cohort, reasoning that the reward function will fail to converge if the policy is ill-defined for any of these.

It is worth noting that maximization of social welfare and Pareto efficiency are considerably more difficult to argue and are indeed outside the scope of this work. Reinforcement learning does not generally treat problems where an optimal outcome (or even a slightly improved one) may be shown to exist analytically, since there usually exist better alternatives. These represent arguments in favor of the continued need for optimal control theory, as well as the relatively new fields of *explainable* and *trustable* artificial intelligence. However, in Section 2.4 a proof is discussed for improved suboptimality bounds from attention on broad classes of offline multi-agent learning problems [10].

It is generally agreed that there is yet no “silver bullet” algorithm or structure to address these conditions, in no small part because the formulation of the MARL problem is so heavily dependent on the structure problem under consideration, the rapidly changing state of relevant physical technology, and the assumptions made in representing the particular problem under consideration [3]. To the extent that at least *locally* optimal MAS policies can be found, progress has been made on determining algorithms for that purpose. The counterfactual multi-agent (COMA) [11] family of policy gradient methods generalize the on-policy optimization algorithms discussed earlier by having the critic model retrospectively estimate the distribution of reward with respect to each agent’s individual actions, conditioned upon the actual actions of each other. Multi-agent posthumous credit assignment (MA-POCA) [6] improves further by using self-attention in the critic function to account for changing agent cardinality; understanding the advantage this entails, and

how it can likewise be applied in the agent control policy itself, requires some discussion of the attention head structure (see 3.1).

2.3. Parallels to natural language processing methods

By way of justifying the use of the attention structure, it is useful to note that the desirable properties mentioned in the above definition of a variadic policy (order invariance, variable shape) have parallels in the natural language processing (NLP) literature. Bahdanau et al. in [12] illustrate and address two classical machine translation conundrums:

1. The proximity and order of multiple tokens in a string are not uniformly a function of their syntactic relationship; and
2. Valid sentences may be of any length, without diminishing the relevance of earlier tokens.

The *attention head* structure was developed in [12] and refined in [13] to address these problems by defining a learnable "query-key-value" mapping from any finite number of vector embeddings onto a single output vector.

Similarly, we may observe that in many MAS settings,

1. The mutual objective relevance of any two agents' observations at a given time generally does not depend on any permutation of the agent set ("permutation-invariant"); and
2. The elements in the environment, including the agent cohort itself, may have various cardinalities, without diminishing the relevance of any single element to the objective.

2.4. Attention in MARL to date

The attention architecture is relatively new, and the extent to which its capabilities scale with size are even more recently appreciated. Practical adoption in reinforcement learning to date, let alone in MARL, is thus fairly limited as of this writing, but the potential of the technology in domain-specific applications is nonetheless already strongly indicated by recent work [14], [15], [16]. Further, it is thought that the analogous ability to attribute varying degrees of attention to different dimensions of stimulus is responsible in part for the phenomenon of reinforcement learning in human neurology [17].

Zhang et al. [10] show by way of the Bellman error function (which describes performance relative to a hypothetical optimum) that if a transformer network converges on a model-free offline RL problem, it does so with a Bellman error independent of the number of agents *N*, even if *N* falls far short of justifying the mean-field assumption; and further, that for model-based offline reinforcement learning, the suboptimality gap grows relatively slowly, in $O(\sqrt{\log n}$. By way of demonstration, they compare a transformer-based model to a multi-layer perceptron, deep set, and graph-convolutional network on the same MPE "Spread" environment discussed in Chapter 5 for *N* ranging from 3 to 30, showing substantial gains in final performance. However, it is not clear whether the authors trained a single variadic policy on a mixture of the environment configurations, or specified an individual attention-based policy for each based on the same architecture.

Liu et al. [18] leverage the attention head's relational structure to construct a communication graph of the agents, describing whether two agents have a meaningful interaction at each time step of the simulation (either Traffic Junction or Predator-Prey);

their states are mutually observable if and only if they are connected in this graph. The authors then implement a graph neural network learning algorithm to determine the control policy for each agent. This represents a significant step in the use of attention to facilitate group behavior in a MAS, but it is limited in two respects: (1) the attention neural network is used only once, in a centralized fashion, to construct the communication graph; and (2) the environment dynamics and cardinality are held constant during training, limiting the extent to which the resulting policies may be generalized.

More recently, Hu et al. [19] demonstrate that an transformer-based architecture may be deployed on many cooperating agents concurrently, decoupling their policy learning while maintaining a variadic representation of the environment. They identify several key advantages, namely that the transformer architecture enables arbitrary dimension of both the input *and* output spaces of the model, and that the activations within the attention head comprise an intuitive representation of the highest-weighted inputs for a given action, which aids interpretability. They show that this architecture provides for transfer learning from low-valence configurations to higher ones. However, this architecture is only demonstrated for discrete actions representing the selection of one observed entity, and it is further shown in the same paper that when the action-observation correspondence is not preserved, performance is considerably worse than more standard recurrent architectures. Thus, while promising, this does not yet represent generalized continuous control.

It would therefore seem that the areas most in need of contribution are (1) specifying monolithic attention-based policies that provide efficient control for *any* number of sensed states within constraints; (2) adapting the transformer/attention architecture to continuous

control outputs and showing robustness of performance; and (3) fully decentralizing the attention-based agent policies from any kind of central planner. The experiments discussed in Chapters 4, 5 and 6 are designed to accomplish these objectives.

CHAPTER 3. ATTENTION THEORY

The scaled dot-product attention head, as described in [13], is defined by

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

$$Q: n_q \times d_k$$
$$K: n_v \times d_k$$
$$V: n_v \times d_v$$

where Q is a matrix whose rows represent "query" vectors, K is a matrix whose rows represent "key" vectors to be compared to those of Q, and V is a matrix of "value" row vectors associated to the rows of K. This is by way of analogy to a relational database structure.

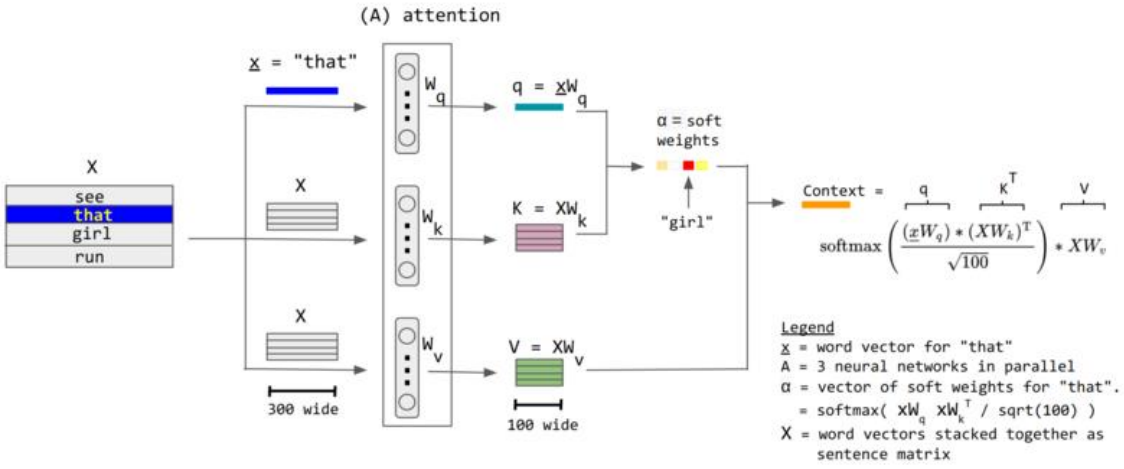

Figure 3.1: Calculation flow through a single attention head [20]

Intuitively, this represents a kind of "soft search" for a convex combination of the value vectors given by the rows of V, where the weights are defined by the similarity of the rows of Q to the columns of K.

The problem of attention, then, is to find useful assignments of *Q*, *K* and *V*. In general, *K* and *V* are some learned mappings of the same set of inputs; in natural language processing, these inputs are usually semantic vector embeddings of each of the words in a sentence, often masked or concatenated with a positional encoding, e.g. [21].

Self-attention, the case of interest to this work, is the case where *Q* also maps this same set (thus $n_q = n_v$); it associates a scalar measure of "relatedness," under some learned relation represented by the combination of *Q* and *K*, to each pair of inputs, and these become the weights for each row of *V*.

In RL (or indeed more general dynamical control problems), we can replace the semantic-space vectors from the natural language setting with measurement vectors drawn from the state of a physical system. This brings us nearer to being able to treat the machine learning problem as a state-space control problem, where the learned model takes the place of the control plant.

Consider an RL environment with a polyvalent continuous state space and (without loss of generality) a component-wise normalized action vector space:

$$\mathcal{O} = (\mathbb{R}^{d_o})^{n_o}$$

$$\mathcal{A} = [0,1]^{d_a}$$

$$\pi_\theta : \mathcal{O} \to \mathcal{A}$$

$$\phi : \mathcal{S} \times \mathcal{A} \to \mathcal{S}, \mathbb{R}$$

$$\omega : \mathcal{S} \to \mathcal{O}$$

where:

- The observation space $\mathcal{O}$ is an ordered set of n_o zero-padded vectors in $\mathbb{R}^{d_o}$, with $n_o$the number of observed states and d_o the greatest state dimensionality;
- The action space $\mathcal{A}$ is a vector assigning a normalized control value to each of d_a control axes;
- π_θ, ϕ, ω are respectively the parametric agent control policy, environment state transition function, and observation mask function as defined previously.

Rather than a single vector, the observation space $\mathcal{O}$ is notated as an arbitrary-size set of vector observations of dynamical elements, or equivalently a matrix of n_o row vectors; the importance of this convention will become apparent. The semantics of the row vectors need not be homogeneous so long as they are internally separable, e.g. by a "one-hot" encoding of the data type; and a maximal n_o may be specified so that the dimension of the observation matrix is well-defined. Excess elements (or entire rows) may be set to zero with no effect on the output.

Note that the *observation* space is not necessarily equivalent to the *state* space (i.e. total observability is not guaranteed), although it may be so if the observation mapping ω is the identity.

Since the output of this mapping is always simply a vector, it is trivial to compose it with other (learned) transformation, i.e. fully-connected/feed-forward neural network layers, as part of a larger policy learning pipeline. Where $O_t\ [n_o \times d_o] = \omega(s_t)$, we can then define $Q_t = O_t W^Q$ for a matrix of weights $W^Q\ [d_o \times d_k]$, and likewise define K_t and V_t with $W^K\ [d_o \times d_k]$, $W^V\ [d_o \times d_v]$. The stochastic control policy π_θ with parameters $\theta \supset$

$\{W^Q, W^K, W^V\}$, having a single attention head layer and fully connected neural layers implementing a function $FC : \mathbb{R}^{d_v} \to \mathcal{A}$, is then the composition:

$$a_t = \pi_\theta(O_t) = Clip_0^1 \circ FC \circ Attention(O_t W^Q, O_t W^K, O_t W^V)$$

where

$$Clip_0^1(\boldsymbol{x})_i = \begin{cases} 0, x_i \leq 0 \\ 1, x_i \geq 1 \\ x_i \text{ otherwise} \end{cases}$$

is used to constrain the normalized output to a physically reasonable range.

Conveniently, since the attention output function is linear except for the addition of softmax (which is trivially differentiable), it is itself easy to differentiate. There is thus no additional difficulty in computing the backwards gradient step of a learning algorithm, except for the addition of dimension.

CHAPTER 4. METHODOLOGY

To study the proposed attentional MARL methodology, a series of simulated physical environments were implemented in the Unity engine, based on OpenAI's MPE benchmark environments for MARL [22]. First, in a proof of concept, a small homogeneous group of roving agents must coordinate to cover a set of landmarks as quickly and closely as possible while minimizing collisions amongst themselves, sharing only partial state information. Second, the MPE "Tag" environment implements a considerably more complex game: a team of evaders must work together to dodge and distract a team of pursuers, in the presence of multiple static obstacles.

This work takes the approach of time-series analysis of the reward trajectories of the agents in each task, reasoning that the reward functions are defined so as to directly measure some property relevant to the performance of the multi-agent system. By choosing this measure, we obtain a simultaneous view of the convergence trajectory, computational cost and final performance for each policy neural network tested. This also enables study of the effect of time-domain perturbations on training, as in Section 6.3.3.

The policy neural networks were implemented in PyTorch, using the Unity ML Agents framework [23] as an interface between the simulation state and the input/output layers of the network.

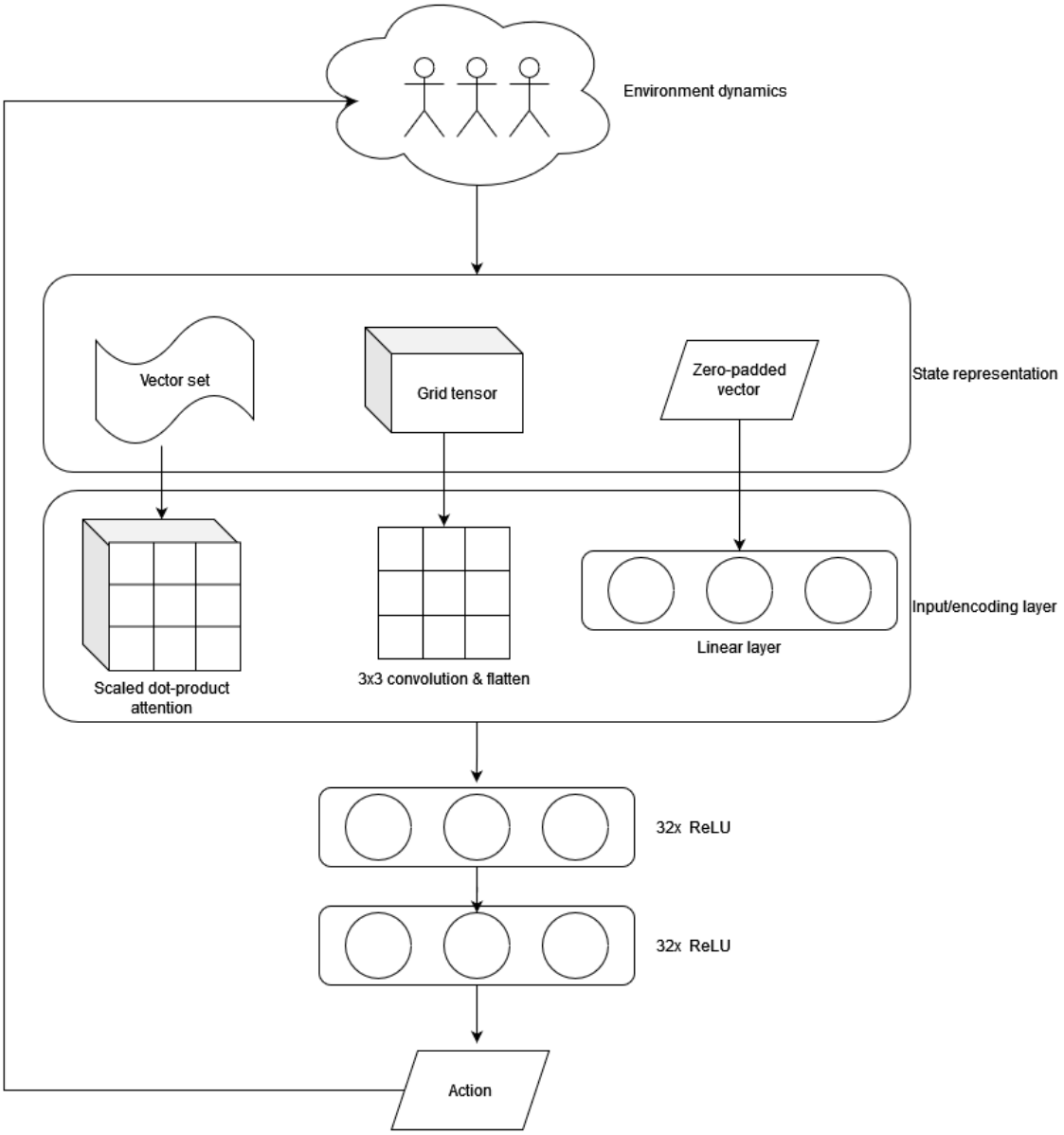

Figure 4.1: Comparison of architectures studied

Figure 4.1 presents the three neural network architectures compared in this work. The leftmost is the attention-based architecture discussed hitherto. The center architecture implements a convolutional neural network (CNN), summarized in [24], [25]. The local environment state surrounding each agent is represented by a discretization into volumetric

cells, where the value in each channel corresponds to the type of entity occupying its cell (i.e. the representation is permutation-invariant). Such architectures have produced expert-level performance on benchmark RL tasks in games [26] as well as model-based simulations of grid control problems subject to locality and partial observability [27].

Finally, a permutation-variant "feed-forward" policy neural network (right) is implemented in each task as a baseline for comparison.

CHAPTER 5. LEARNING POLICIES FOR COOPERATIVE NAVIGATION IN POLYVALENT ENVIRONMENTS

5.1. Summary

Group navigation strategies for arbitrary-sized multi-agent systems require permutation-invariant control policies which can accommodate a range in the number of sensed states without detriment to performance. This chapter presents a neural network architecture for a decentralized control policy on a 3D dynamical implementation the OpenAI MPE "Spread" cooperative navigation benchmark, using the permutation-invariant attention head structure as an input layer. In simulation, this policy architecture vastly outperforms a permutation-variant feed-forward architecture and demonstrates modest performance gains with substantially smaller compute costs compared to a more conventional CNN architecture.

5.2. Methodology

The "Spread" environment is a test case demonstrating the suitability of the attention-based architecture for cooperative decentralized control using a single policy for varying numbers of agents and objectives, in the absence of any communication other than shared state information.

Two models with otherwise identical hyperparameters (2 layers, 32 ReLU neurons [28]) are tested for comparison. The first is a simple fully connected neural network controller, taking as input a single concatenated, zero-padded observation vector $\in \mathbb{R}^{d_o \times n_o}$. This representation is not permutation-invariant and is considered as a baseline.

The second uses a convolutional input layer, taking as input a grid tensor representation of the agent's surroundings ($20 \times 20 \times 2$, with the channels corresponding to landmarks and agents, containing a 0-1 encoding of whether the given entity type is present in that cell).

This implementation of the environment differs from the standard implementation (described in [22] and utilized in [10]) in a few meaningful ways:

- The contact dynamics are simulated in three dimensions rather than two, and kinematics consider material properties (drag, elasticity of collisions) to provide a closer approximation to practical environments.
- The maximal number of agents and the sizes of the policy neural networks are considerably smaller due to compute constraints (1x NVIDIA RTX 3060 12GB). More conventional architectures are chosen for comparison, to provide a sense of the difference made by the attention layer even in minimal implementations.
- Learning is online rather than offline. That is, rather than learning to associate a value to each state-action pair sampled randomly from a historical dataset, agents in this implementation learn as they interact with the nonstationary environment, providing a sense of potential for "lifelong" learning *in situ*.

The overall environment specification is given by:

$$\mathcal{O} = \mathbb{R}^4 \times (\mathbb{R}^2)^{2n}$$

$$\mathcal{A} = [0,1]^2$$

$$r_g = -\sum_{i=1}^{n} \min_{1 \le j \le n} \left\| \boldsymbol{x}_i^{target} - \boldsymbol{x}_j^{agent} \right\|$$

$$r_i^k = \begin{cases} -1 \text{ if colliding with a teammate} \\ 0 \text{ otherwise} \end{cases} \quad \forall k \in 1, \dots, n$$

$$r_t^k = \hat{r}_g^k + r_i^k \; \forall k \in 1, \dots, n$$

with

- $\mathcal{O}, \mathcal{A}$ the observation and action spaces as defined previously;
- r_g the group reward component for the MA-POCA algorithm [6] as a function of target landmark positions $\boldsymbol{x}_i^{target}$ and agent positions $\boldsymbol{x}_j^{agent}$ in the plane;
- r_i^k the individual reward component;
- r_t^k the total reward assigned to agent *k*, which is the sum of the MA-POCA imputation of group reward $\hat{r}_g^k$ and the individual reward r_i^k.

The observation represents a concatenation of each agent's own position and velocity with the instantaneous *relative* position of each other agent and target at sample time (i.e. observations are in the local frame). The action vector is interpreted as a normalized command force to be applied to the agent along the ground plane axes.

5.3. Numerical results

Figure 5.1 summarizes the total group reward for the navigation task over a complete episode (2500 steps) at each time step of training. This presents an aggregate measure of the speed and accuracy with which the agents converge upon the targets and is therefore a useful proxy for total final performance. Figure 5.2 is a graph of the average individual collision penalty in each episode; thus, it roughly inversely proportional to the "per-capita" time rate of collisions in the system.

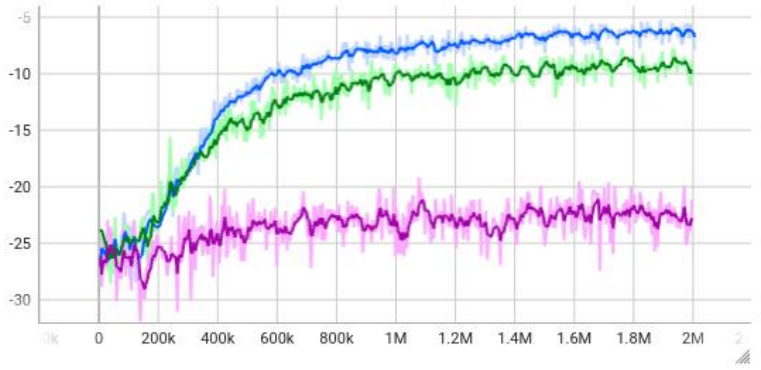

Figure 5.1: Group performance reward on cooperative navigation task vs. training step for each tested architecture

Table 5.1: Legend and summary of Figure 5.1

Run	Terminal group reward	Training time
• Attention	-6.765	37.06 min
• CNN	-9.642	1.166 hr
• Feed-forward	-22.763	35.11 min

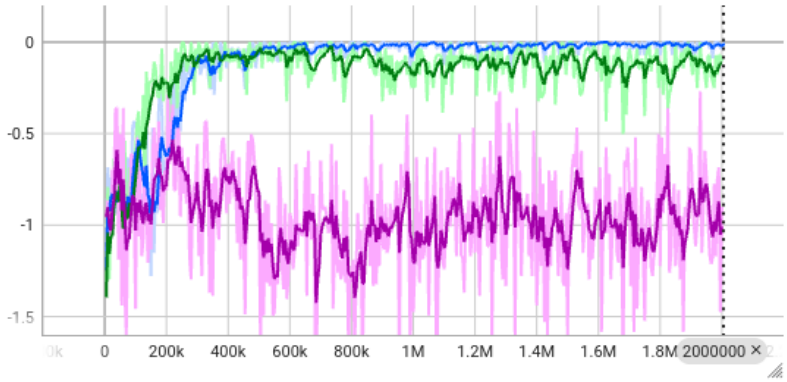

Figure 5.2: Mean individual collision penalty on cooperative navigation task vs. training step for each tested architecture

Table 5.2: Legend and summary of Figure 5.2

Run	Terminal individual reward	Training time
• Attention	-0.023	37.06 min
• CNN	-0.114	1.166 hr
• Feed-forward	-1.007	35.11 min

5.4. Discussion

It is apparent that, as expected, the permutation-variant feed-forward policy function fails to improve meaningfully by convergence. This follows because parameters are not shared between successive input neurons even when representing entities of the same type; thus, gradient updates are distributed unevenly over the input layer and it is substantially harder to learn any useful behavior.

The variadic, permutation-invariant grid tensor CNN architecture performs much better, and approaches convergence by step 2×10^6 . However, it takes the longest to train of the tested architectures, by nearly a factor of two for the same number of simulation steps.

The attention-based policy architecture provides a modestly better final group performance on the navigation task than the CNN architecture, with a substantially smaller compute requirement (37 minutes vs. 70 minutes).

CHAPTER 6. PERFORMANCE OF ATTENTION-BASED POLICY ARCHITECTURES IN MIXED COOPERATIVE-COMPETITIVE ENVIRONMENTS WITH TIME-VARYING DYNAMICS

6.1. Summary

An important consideration for multi-agent autonomous system control "in the wild" is whether the system will act safely and reliably in environmental conditions not part of the distribution considered at the time of design. This may be due to the presence of other autonomous actors in the environment, or simply due to a shift in the distribution of environmental conditions. Here we investigate the potential of learned attention-based evasion policies to adapt to such conditions, speculating that the relational structure of learned attention policies should provide certain benefits over alternative policy neural networks. In two experiments, we find that (1) small attention networks trained for group-level achieve both faster convergence of behavior and better final performance against expert teams of pursuers, as compared to other compact learned policies; and (2) this capability is robust to persistent distribution shift of the physical characteristics of the game area. To the best of the author's knowledge, this is the first such comparative study of the effects of time-domain distribution shift on *in situ* continued multi-agent reinforcement learning.

6.2. Methodology

In the "Tag" environment [22], agents are divided into two multi-agent teams: the "good" or "evader" agents $(1 \leq n \leq 3)$ receive a group penalty $r_{ge} = -1$ for colliding with pursuers, and an individual penalty at each timestep for exiting a unit bounding box. The

"bad" or "pursuer" team ($3 \leq m \leq 5$) receives a group reward $r_{gp} = 1$ for each collision with an evader, incentivizing group maneuvers. There are also $2 \leq l \leq 4$ randomly placed static obstacles with which agents may collide.

This section proceeds in three stages: first, since this is a mixed cooperative-competitive environment, "self-play" [4] is used to concurrently train attention-based pursuit and evasion policies, providing a baseline pursuer policy to train against in future steps as well as validating that the policies do at least roughly converge in the presence of changing environment dynamics.

Second, evasion control policies based on the same feed-forward, CNN and attention-based architectures used for comparison in Chapter 5 are successively trained against static teams of the pursuit agent trained in self-play. This is to provide a direct comparative measure of the performance on the evasion task.

Finally, the same is repeated, but with obstacles deterministically toggled on and off every 5×10^5 training steps. This time-domain reversal design is intended to investigate whether attention neural networks are as susceptible to the problem of "catastrophic forgetting" as reinforcement learners are historically known to be [29]; in short, will overfitting/maladaptation to a particular prior distribution of conditions require learning a subsequent distribution "from scratch?"

The complete environment specification is given by:

$$\mathcal{O} = \mathbb{R}^4 \times (\mathbb{R}^4)^{n-1} \times (\mathbb{R}^4)^{\mathrm{m}} \times (\mathbb{R}^2)^{\mathrm{l}}$$

$$\mathcal{A} = [0, 1]^2$$

$$r_{gp} = -r_{ge} = \text{number of pursuer} - \text{evader collisions}$$

$$r_i^k = \begin{cases} 0, \rho_k < 0.9 \\ 10(\rho_k - 0.9), 0.9 \leq \rho_k < 1 \\ \min(e^{2\rho_k - 2}, 10), \rho_k \geq 1 \end{cases}$$

$$r_{te}^k = \hat{r}_{ge}^k + r_i^k \; \forall k \in 1, \ldots, n$$

Variables are defined as in Section 5.2, with the addition that a subscript *e* indicates a member of the evader team, and subscript p a member of the pursuer team. Reward is imputed to these groups separately; that is, there are two concurrent instances of the MA-POCA algorithm optimizing the parameters of two separate policies.

6.3. Numerical results

6.3.1. Self-play

To begin with, two separate attention neural networks are randomly initialized and respectively assigned to the individual members of the pursuer and evader teams. Following the example of [4], the teams are pitted against each other successively, and checkpoints of their internal weights are taken periodically, in order to condition competitive behavior against opponents of a range of skill levels. Since there is no explicit stopping condition to indicate a satisfactory baseline, this process continues until the reward graph and policy gradient for each model is relatively smooth and stable, illustrated in Figure 6.1.

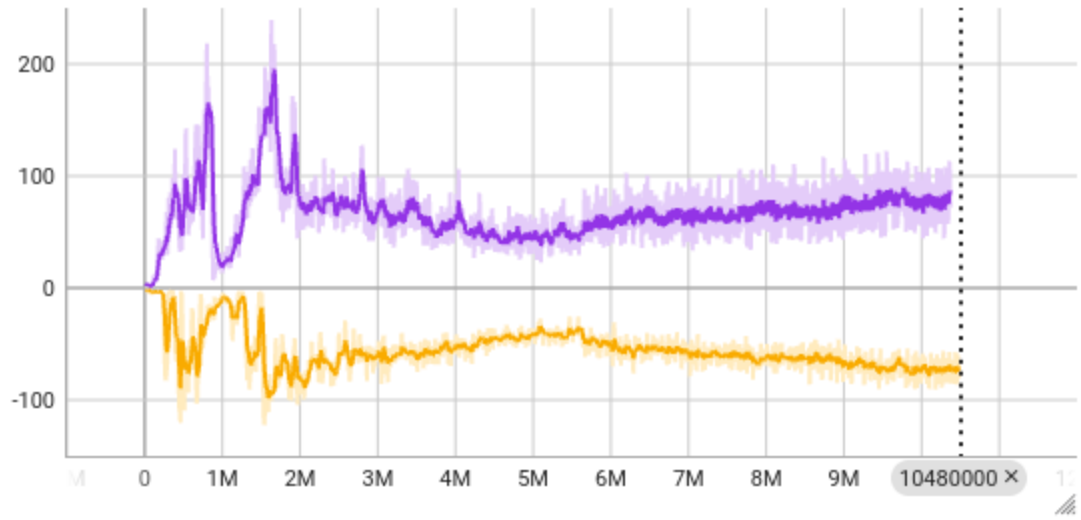

Figure 6.1: Self-play training reward trajectory in pursuit-evasion task

Table 6.1: Legend and summary of Figure 6.1

Team	Terminal total reward	Training time
• Pursuit	82.394	6.851 hr
• Evasion	-73.602	6.979 hr

The evasion policy learned in this process exists to provide a putative "self-play" opponent for the pursuers and is discarded; the pursuit policy is retained as the "antagonist" for future experiments. It is notable that the final reward distribution indicates that the environment is favorable to the pursuers.

6.3.2. Base task

The pursuit-evasion game is repeated with the "expert" pursuer team, and the evader team given, successively, a randomly initialized attention-based policy neural network, a CNN-based network, and a permutation-variant feed-forward neural network (as a baseline).

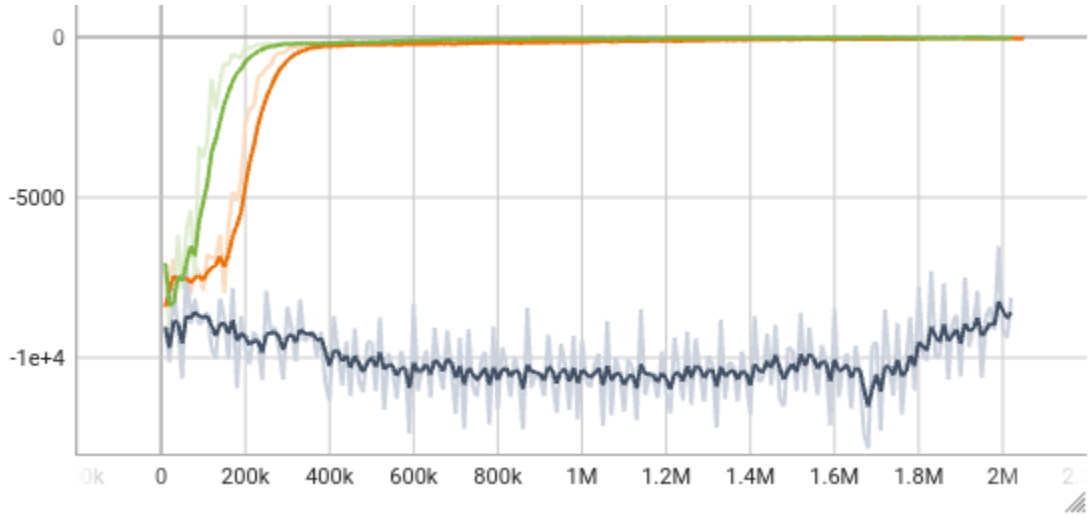

Figure 6.2: Total reward trajectory for tested architectures in pursuit-evasion task

Table 6.2: Legend and summary of Figure 6.2

Run	Terminal total reward	Training time
• Attention	-26.564	33.62 min
• CNN	-39.524	1.249 hr
• Feed-forward	-8,588.946	(Training interrupted)

Figure 6.2 presents the training trajectory of the net performance outcomes, largely to indicate (1) the attention neural network converges in substantially fewer iterations than

the CNN equivalent, and (2) the feed-forward network fails to learn the boundary constraint within the allotted 2×10^6 iterations, leading to drastically poorer performance than either of the permutation-invariant policies (although some improvement is indicated at the very last).

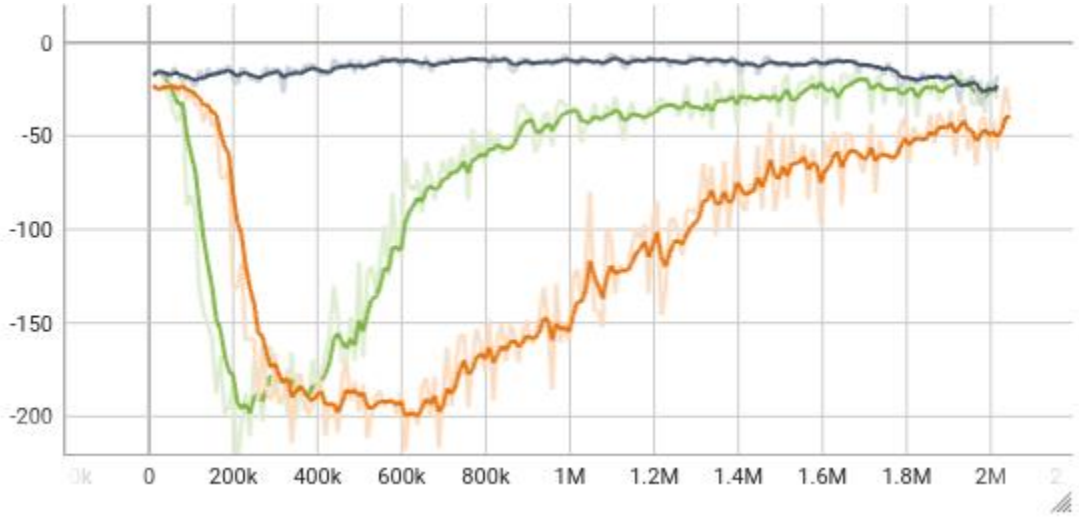

Figure 6.3: Group evasion performance trajectory for tested architectures

Table 6.3: Legend and summary of Figure 6.3

Run	Terminal group reward	Training time
• Attention	-26.459	33.62 min
• CNN	-39.331	1.231 hr
• Feed-forward	-23.217	(Training interrupted)

Figure 6.3 shows a slightly more nuanced picture of the evasion performance by presenting only the group reward, i.e. neglecting individual penalties for exiting the soft "bounding box." Here the feed-forward policy (in grey) achieves the best nominal evasion

performance, since it largely neglects the distance constraint and maladaptively flees as far as possible. Meanwhile, the attention-based policy provides the best representation of the intended behavior, quickly converging to a consistent average of 30.25 interceptions per episode (averaged over all cohorts).

6.3.3. Changing environment

Finally, we examine the evolution of the policies under shifting environment dynamics in a time-series reversal design. In this section, training begins in the absence of static obstacles for the agents to maneuver around; thus the states of such obstacles are likewise unobserved. At timestep $t = 5 \times 10^5$, the obstacles are abruptly enabled, taking the values defined in 6.2. At $t = 1 \times 10^6$, they are once again disabled, before finally being re-enabled at $t = 1.5 \times 10^6$.

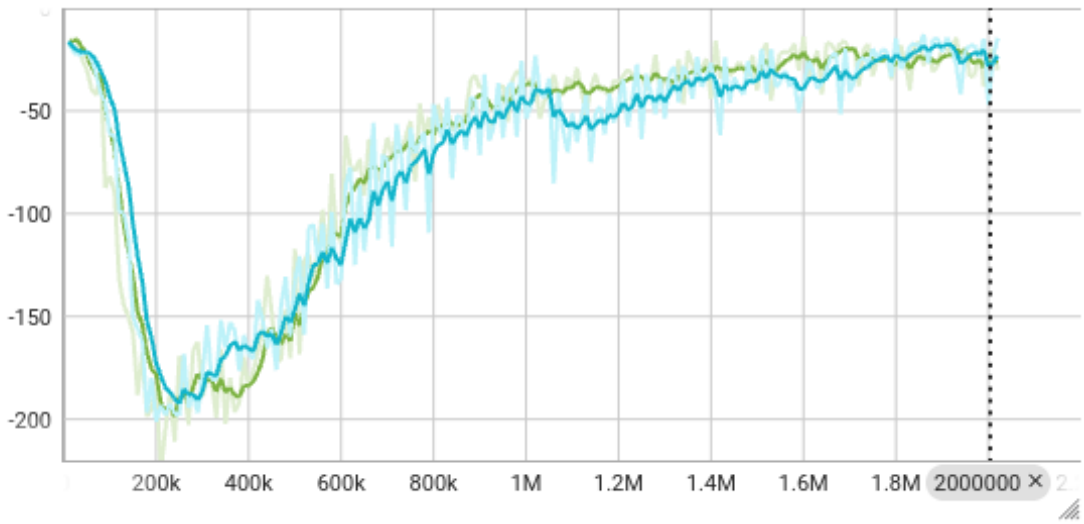

Figure 6.4: Attention architecture evasion performance trajectory, with and without periodic switching of environment configuration

Table 6.4: Legend and summary of Figure 6.4

Run	Terminal reward	Training time
• Attention	-27.0775	33.62 min
• Attention, switching dynamics	-28.2093	33.62 min

The attention-based policy neural network trained from scratch under these conditions (blue) shows some immediate loss in evasion performance upon these distribution shifts, then quickly recovers to approximately its prior efficiency and resuming learning. There is very little difference in the final reward trajectory as compared to the original policy trained under static conditions (green), demonstrating robustness to catastrophic forgetting and perhaps some utility for "lifelong learning" in environments outside the training distribution.

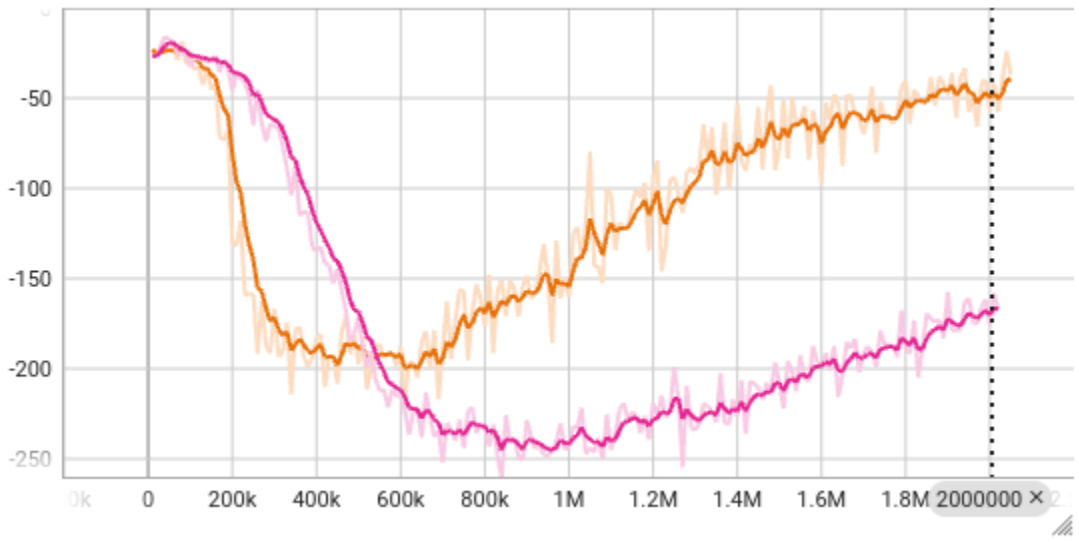

Figure 6.5: CNN architecture evasion performance trajectory, with and without periodic switching of environment configuration

Table 6.5: Legend and summary of Figure 6.5

Run	Terminal reward	Training time
• CNN	-49.890	1.231 hr
• CNN, switching dynamics	-170.209	1.233 hr

Figure 6.5 shows the results of the same experiment with the CNN architecture. Here it is evident that the policy trained under distribution shift (pink) fails to acquire the prior level of evasion performance under static conditions (orange).

6.4. Discussion

This chapter exhibits several critical advantages of the attention architecture, including unexpected performance gains that greatly aid the case for adoption in practical settings. 6.3.1. shows that performant competitive equilibria may be reached by attention-based policies in "self-play" competition, which evolve smoothly in response to the changing environmental conditions created by the opponent. This compounds work such as [4] and [30] as evidence for the potential of RL to achieve expert-level performance on broad classes of tasks with extremely large state spaces. 6.3.2 shows that even in extremely conservative cases, attention overtakes permutation-variant and convolutional architectures in final performance, computational complexity and time to convergence on a general pursuit-evasion task. Finally, 6.3.3. provides evidence that, among small architectures, the relational approach of the attention architecture is uniquely equipped to generalize previous learning under abrupt and persistent perturbations of the environment

configuration, even when this perturbation introduces novel elements and utilizes network parameters which have not hitherto been subject to counterfactual updates.

CHAPTER 7. LIMITATIONS

Several simplifying assumptions have been made in the foregoing experiments, and further study will be needed to translate the promise of these results to more practical scenarios.

For one, communication between agents has been modeled as consistently single-hop in these scenarios. That is, if an agent is observable to the putative "network," its state in the local frame may be known without the introduction of error by repeated relative measurements and signal transmissions. This is not generally true of multi-agent systems, and work is ongoing in the matter of how a consensus state measurement can be reached between agents in decentralized multi-agent systems with realistic directed communication topologies, even when individual measurements are subject to heterogeneous disturbances and interference [31] [32].

A more general concern is that, as is often the case for learning approaches to sensitive problems, the benefit of reaching unexpected solutions by stochastic optimization is a double-edged sword. At the beginning of this work, the example of [4] was used to illustrate that iterated competition between learning agents can result in surprisingly authentic, novel, animal- or human-like approaches to problems yielding solutions better than would have been expected. The authors of that work note, however, that these outcomes do not necessarily *stop* where intended; their learning agents, by the end of training, learn to exploit unintended behavior of the physical simulation to execute maneuvers which are not replicable in the real world, and which exceed the intended scope of permissible behavior. Thus, as in any discussion of autonomous systems, it is incumbent upon us to consider the safety of system operators and surroundings. This should motivate directed study of neural

network explainability, with particular focus on “decoding” attention weights, as the technology progresses and moves toward practical adoption.

CHAPTER 8. DISCUSSION AND FUTURE WORK

8.1. Summary

The goal of this work was to present a systematic treatment of the application of the attention head data structure to multi-agent systems, with an emphasis on how decentralized attention-based neural network control policy architectures may enable multi-agent reinforcement learning on complex cooperative and mixed cooperative-competitive multi-agent tasks, even with non-stationary dynamics. In Chapter 1, a structural argument is presented for the need for such study. Then, in Chapter 2, a review of the literature demonstrates that the problem of multi-agent reinforcement learning for general environments with variable numbers of states is a current challenge. Recent investigations of attention/transformer architectures for MARL tasks are summarized, showing that the translation from discrete, centralized and offline environments to continuous, decentralized and online practical applications with nonstationary dynamics is an open problem.

Chapter 4 summarizes the prospective approach of the work: the OpenAI MPE benchmarks are adopted and generalized slightly to account for three-dimensional contact physics and polyvalence of the sensed dynamical elements.

Chapter 5 presents a paper demonstrating multi-agent reinforcement learning on a cooperative navigation task with continuous control outputs. The number of agents and navigation targets are both unspecified at execution time, so the objective is to learn a performant control policy for any possible environment within the configuration space. It is shown that even with desktop-grade compute capabilities, an attention-based control

architecture provides substantial final performance gains over a convolutional neural network architecture, and both are vastly superior to a non-permutation-invariant fully connected neural network architecture. Additionally, the attention architecture requires substantially less computation time to train to convergence.

Chapter 6 presents a mixed cooperative-competitive pursuit-evasion environment with randomized pursuer, evader and obstacle sets. It is shown that the attention-based evader policy is consistently the most successful at delaying interception despite a noted pursuer advantage in the environment as specified. Additionally, to study the suitability of the architecture for in-situ "lifelong learning" modification of the control policy in response to sudden changes in environment dynamics, we show using a time-series reversal experiment design that attention enables convergence of the policy under sudden removal of obstacles to navigation, with no loss of prior performance when the obstacles are restored, suggesting robustness to the "forgetting problem" and generalization of the learned relational features.

8.2. Future work

As indicated in Chapter 7, future work is likely to be directed largely by considerations of practical reliability. The experiments in Chapters 5 and 6 show robust performance assuming single-hop communication without uncertainty due to iterated measurements; future experiments should focus on generalizing this performance to more realistic communication topologies with corresponding signal degradation [18], [33]. Such work might be complemented by the use of a "digital twin" to incorporate the non-idealities of a network of physical systems [34], [35].

Some work has been done on the use of attention head weights to interpret the causal relationships of observations to decisions, with mixed results [36] [37]. Despite the compelling tabular representations that they provide, attention architectures are still subject to the difficulties created by compositions of nonlinearities, and the resulting weight-based predictions are noisy at best [36] or, more conservatively, not at all usefully related to model outputs [38]. Further, much of this work has been driven by the extraordinary efficacy of transformers on NLP and image processing tasks; very little study has been done in the reinforcement learning setting, wherein poorly-understood policy outputs may have consequences for physical systems. For want of a compelling analytic approach at the present time, extensive inductive study is needed to show with sufficient confidence that neural network control policies can be made stable, or at least to prove under what conditions they may be known to be unstable.

8.3. Conclusion

The sum of these results demonstrates that, among comparable architectures of similar size, decentralized attention-based control policies are uniquely positioned to enable efficient group behavior for multi-agent systems in environments with unknown or non-constant numbers of measured states. Further, such policy models learn robustly even when the distributions of the environment configurations abruptly and persistently shift, a property that is unique among approaches known to the author. These represent reason to consider the (carefully considered) adoption of reinforcement learning of attention- or transformer-based policies as a tool in the implementation of multi-agent systems for complex or underdefined tasks, when it is not feasible to design such systems explicitly.

REFERENCES

[1] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-Agent Systems: A Survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018, doi: 10.1109/ACCESS.2018.2831228.

[2] L. Busoniu, R. Babuska, and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008, doi: 10.1109/TSMCC.2007.913919.

[3] L. Canese *et al.*, "Multi-Agent Reinforcement Learning: A Review of Challenges and Applications," *Appl. Sci.*, vol. 11, no. 11, Art. no. 11, Jan. 2021, doi: 10.3390/app11114948.

[4] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent Complexity via Multi-Agent Competition," arXiv.org. Accessed: Apr. 11, 2024. [Online]. Available: https://arxiv.org/abs/1710.03748v3

[5] C. Yu *et al.*, "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., Curran Associates, Inc., 2022, pp. 24611–24624. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf

[6] A. Cohen *et al.*, "On the Use and Misuse of Absorbing States in Multi-agent Reinforcement Learning." arXiv, Jun. 06, 2022. doi: 10.48550/arXiv.2111.05992.

[7] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, May 1992, doi: 10.1007/BF00992698.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms." arXiv, Aug. 28, 2017. doi: 10.48550/arXiv.1707.06347.

[9] S. Parsons and M. Wooldridge, "Game Theory and Decision Theory in Multi-Agent Systems," *Auton. Agents Multi-Agent Syst.*, vol. 5, no. 3, pp. 243–254, Sep. 2002, doi: 10.1023/A:1015575522401.

[10] F. Zhang, B. Liu, K. Wang, V. Tan, Z. Yang, and Z. Wang, "Relational Reasoning via Set Transformers: Provable Efficiency and Applications to MARL," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 35825–35838, Dec. 2022.

[11] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual Multi-Agent Policy Gradients," *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, Art. no. 1, Apr. 2018, doi: 10.1609/aaai.v32i1.11794.

[12] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate.” arXiv, May 19, 2016. doi: 10.48550/arXiv.1409.0473.

[13] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Mar. 29, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[14] J. Xie, A. Ajagekar, and F. You, “Multi-Agent attention-based deep reinforcement learning for demand response in grid-responsive buildings,” *Appl. Energy*, vol. 342, p. 121162, Jul. 2023, doi: 10.1016/j.apenergy.2023.121162.

[15] Z. Wang, J. Xuan, and T. Shi, “Multi-source information fusion deep self-attention reinforcement learning framework for multi-label compound fault recognition,” *Mech. Mach. Theory*, vol. 179, p. 105090, Jan. 2023, doi: 10.1016/j.mechmachtheory.2022.105090.

[16] J. Chen *et al.*, “Global-and-Local Attention-Based Reinforcement Learning for Cooperative Behaviour Control of Multiple UAVs,” *IEEE Trans. Veh. Technol.*, vol. 73, no. 3, pp. 4194–4206, Mar. 2024, doi: 10.1109/TVT.2023.3327571.

[17] “Reinforcement Learning in Multidimensional Environments Relies on Attention Mechanisms | Journal of Neuroscience.” Accessed: Apr. 17, 2024. [Online]. Available: https://www.jneurosci.org/content/35/21/8145.short

[18] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, “Multi-Agent Game Abstraction via Graph Attention Neural Network,” *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 05, Art. no. 05, Apr. 2020, doi: 10.1609/aaai.v34i05.6211.

[19] S. Hu, F. Zhu, X. Chang, and X. Liang, “UPDeT: Universal Multi-agent Reinforcement Learning via Policy Decoupling with Transformers.” arXiv, Feb. 07, 2021. doi: 10.48550/arXiv.2101.08001.

[20] Numiri, *English: calculation flow through a single attention head*. 2023. Accessed: Apr. 17, 2024. [Online]. Available: https://commons.wikimedia.org/wiki/File:Attention-qkv.png. Numiri, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

[21] T. Brown *et al.*, “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 1877–1901. Accessed: Mar. 29, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[22] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments.” arXiv, Mar. 14, 2020. doi: 10.48550/arXiv.1706.02275.

[23] A. Juliani *et al.*, “Unity: A General Platform for Intelligent Agents,” arXiv.org. Accessed: Apr. 11, 2024. [Online]. Available: https://arxiv.org/abs/1809.02627v2

[24] J. Gu *et al.*, “Recent advances in convolutional neural networks,” *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018, doi: 10.1016/j.patcog.2017.10.013.

[25] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[26] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning.” arXiv, Dec. 19, 2013. doi: 10.48550/arXiv.1312.5602.

[27] B. J. Claessens, P. Vrancx, and F. Ruelens, “Convolutional Neural Networks for Automatic State-Time Feature Extraction in Reinforcement Learning Applied to Residential Load Control,” *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3259–3269, Jul. 2018, doi: 10.1109/TSG.2016.2629450.

[28] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” arXiv.org. Accessed: Apr. 11, 2024. [Online]. Available: https://arxiv.org/abs/1803.08375v2

[29] A. Cahill, “Catastrophic Forgetting in Reinforcement-Learning Environments,” Thesis, University of Otago, 2011. Accessed: Apr. 12, 2024. [Online]. Available: https://ourarchive.otago.ac.nz/handle/10523/1765

[30] D. Silver *et al.*, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, doi: 10.1038/nature24270.

[31] G. Wang, C. Wang, Z. Ding, and Y. Ji, “Distributed Consensus of Nonlinear Multi-Agent Systems With Mismatched Uncertainties and Unknown High-Frequency Gains,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 68, no. 3, pp. 938–942, Mar. 2021, doi: 10.1109/TCSII.2020.3016977.

[32] G. Guo and R. Zhang, “Lyapunov Redesign-Based Optimal Consensus Control for Multi-Agent Systems With Uncertain Dynamics,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 69, no. 6, pp. 2902–2906, Jun. 2022, doi: 10.1109/TCSII.2022.3149911.

[33] S. C. Lokhande, H. Xu, and H. S. Toor, “Event triggered distributed adaptive consensus control for high-order nonlinear multi-agent systems in presence of system uncertainties,” in *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Jul. 2017, pp. 606–611. doi: 10.1109/RCAR.2017.8311929.

[34] M. Matulis and C. Harvey, “A robot arm digital twin utilising reinforcement learning,” *Comput. Graph.*, vol. 95, pp. 106–114, Apr. 2021, doi: 10.1016/j.cag.2021.01.011.

[35] K. Xia *et al.*, “A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence,” *J. Manuf. Syst.*, vol. 58, pp. 210–230, Jan. 2021, doi: 10.1016/j.jmsy.2020.06.012.

[36] S. Serrano and N. A. Smith, “Is Attention Interpretable?” arXiv, Jun. 09, 2019. doi: 10.48550/arXiv.1906.03731.

[37] H. Fukui, T. Hirakawa, T. Yamashita, and H. Fujiyoshi, “Attention Branch Network: Learning of Attention Mechanism for Visual Explanation,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10705–10714. Accessed: Apr. 15, 2024. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Fukui_Attention_Branch_Network_Learning_of_Attention_Mechanism_for_Visual_Explanation_CVPR_2019_paper.html

[38] S. Jain and B. C. Wallace, “Attention is not Explanation.” arXiv, May 08, 2019. doi: 10.48550/arXiv.1902.10186.