Permutation Equivariant Deep Reinforcement Learning for Multi-Armed Bandit

Zhuofan Xu, Benedikt Bollig, Matthias Függer, Thomas Nowak

Permutation Equivariant Deep Reinforcement Learning for Multi-Armed Bandit

Zhuofan Xu*, Benedikt Bollig*, Matthias Függer* and Thomas Nowak*†
*Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, Gif-sur-Yvette, France
†Institut Universitaire de France, Gif-sur-Yvette, France

***Abstract*—Permutation equivariance (PE) is a property widely present in mathematics and machine learning. Classic deep reinforcement learning (DRL) algorithms, such as Deep Q-Network (DQN), require thoroughly exploring the state space to achieve optimal performance. For a PE problem such as the Multi-Armed Bandit (MAB) problem, the PE property helps reduce the space that needs to be explored. This paper proposes PEDQN, a PE DRL framework based on DQN by applying a PE neural network structure. Our MAB experiments show that PEDQN has clear advantages compared to DQN with a fully connected network and achieves the same or better performance than UCB1 when tested in the same environment as the training.**

***Index Terms*—Reinforcement Learning, Deep Reinforcement Learning, Permutation Equivariance, Multi-Armed Bandit, Q-Learning, Deep Q-Network**

I. Introduction

Reinforcement learning (RL) is a machine learning paradigm that trains agents through their interactions with the environment. Many reinforcement learning problems can be modeled as a Markov Decision Process (MDP) of an agent within an environment: starting from an initial state, at each step, the agent chooses among possible actions, and depending on the played action and the current state, the environment transits, stochastically or not, the agent to a new state and rewards the agent.

Classic RL methods, such as Q-learning that estimates the reward for each state-action pair, quickly reach their limits in the presence of a large state or action space since most state-action pairs remain unobserved during learning. Introducing a method with inference capabilities like a neural network (NN) mitigates this problem: Deep reinforcement learning (DRL) algorithms have successfully used NNs to approximate policy or value functions, significantly compressing mapping representation. For example, Deep Q-Network (DQN) is a "deep" version of Q-learning, and it achieved remarkable results in complex tasks such as Atari games [1].

Despite the significant space reduction and generalizing capabilities of DQNs over classical Q-learning, DQN still requires a thorough exploration of the state and action space to achieve satisfying performance. The state and action space quickly grow exponentially in specific environments with the problem size. A simple example is the Multi-Armed Bandit (MAB) problem: over a fine number of rounds, a player can choose to play one out of n machines in each round. Each such machine has an apriori unkown, fixed probability to win, with the reward/loss being communicated to the player at the end of a round. When adding a new slot machine to a game of maximum T rounds with n machines, the possible play sequences pass from n^T to $(n+1)^T$. For example, adding a machine to $n = 9$ machines with a time horizon $T = 200$ increases the state space by $1.4\text{e}9$ times. This growth speed may lead to excessive training costs.

However, many multi-agent MDP problems studied in the literature are inherently symmetric in their state-action space. For example, in the MAB problem, when the order of the slot machines changes, the decision of which machine to play changes accordingly: it should not depend on the machine's identifier. Similar symmetries are present in distributed algorithms for multi-agent systems: often the searched for solution does or should not depend on the agent's identifier. In particular, MAB can be understood as a multi-agent systems with the machines being the agents and only one agent being allowed to play at a time. This symmetry has been studied as permutation equivariance (PE): when the input of a permutation equivariant function is permuted, then its output is permuted identically.

In the context of multi-agent RL frameworks, if a strategy, i.e., the function of choosing actions according to the agent's states, is PE, it can potentially be learned from a significantly reduced dataset: training on one permutation instead of all possible permutations, diminishes the state-action space by a factor of $n!$ where n is the number of individuals in the environment: e.g., for a game with 10 individuals, the state space becomes $1/3628800$ of the original size for a PE agent.

In this work, we propose a variant of a DQN for the MAB problem that benefits from the PE property: we use a so-called PE network as the Q-network. Unlike fully connected NNs that have to learn the PE property by training on permutations of data, which is costly and non-robust, the PE network is by design PE.

We identify simple network structures that are PE by design: (i) networks constructed from individual networks for each agent, (ii) pooling networks, and (iii) self-attention [2]. These networks can be combined systematically to form more

The work was supported by the French National Research Agency (ANR) project DREAMY (ANR-21-CE48-0003), as well as the SAIF project, funded by the "France 2030" government investment plan managed by ANR, under the reference ANR-23-PEIA-0006.

complex networks that again fulfill the PE property and are candidate solutions for the MAB problem. We show that the constructed PE DQNs are robust and accurate solutions, which we demonstrate on the binary MAB problem, where the trained PE DQNs outperform a classical DQN and match the performance of the classical UCB1 algorithm [3].

II. Related Work

In this section, we briefly review the PE neural network structures proposed in other works as well as current approaches to applying the DRL method to the MAB problem.

A. Permutation Equivariant Network

Permutation equivariance of a function is a property often observed in the context of sets and graphs. PE networks have numerous applications, notably in computer vision. For example, studies such as [4], [5] and [6] use their networks for symmetry group image classification, achieving higher accuracy with less data compared to CNNs, and [7] employs it for image selection and image deblurring. Due to their intrinsic properties, PE networks are also well-suited for tasks involving sets, including anomaly detection [8], point cloud classification, and segmentation [8], [9]. Similarly, the work in [10] focuses on permutation invariance (PI) and proposes a graph convolution method for tasks on the point cloud. Furthermore, PE networks are finding new applications, such as in the development of PE neural functionals [11]. Even if the model's equivariance does not precisely match the environment's symmetry, it has been shown that equivariant models can still outperform non-equivariant methods [12].

The work in [8] characterized the single-layer NNs that are PE as precisely those where all the diagonal elements of the weight matrix are identical, and all the off-diagonal elements are identical. For n scalar inputs, this reduces the parameter space from n^2 to 2. The work [13] generalizes the method to higher dimensional PE inputs via the Kronecker product of the weight matrix. At the same time, [7] generalizes to cases where elements are also symmetric. In [14], the authors use multihead attention to construct a "Set Transformer," which can also be PE.

Reducing the number of independent weights in an NN layer by identifying the same weights has also been previously used, e.g., in convolutional neural networks (CNNs), where a small kernel is applied to each input neighborhood. Some works extend the feature to other parts of CNNs and propose group equivariant CNNs [4], [5], [15], in most cases, rotation and translation equivariant. This idea of parameter sharing is then broadly adapted in the design of PE or PI structures: [16] propose a parameter-sharing scheme of an NN layer equivariant to given discrete group-actions. Network structures satisfying higher-order PE for graphs were proposed in [17], [18]. The work [19] uses a pairwise structure to achieve PE. Lastly, there is a family of PE networks among CNNs called steerable CNNs [20], which constrain the convolution kernels. A recent example is [21], which uses Clifford-steerable kernels to achieve equivariance of multivector fields in pseudo-Euclidean spaces.

B. Deep Reinforcement Learning

Deep reinforcement learning gained attention after successfully playing Atari games using DQN [1]. Various types of DRL methods have been proposed, such as asynchronous advantage actor-critic (A3C) [22] and Proximal Policy Optimization (PPO) [23]. Since then, DQNs have been improved in several aspects, such as Double DQNs [24], prioritized experience replay [25], and auxiliary networks [26].

Several works have exploited the PE property in DRL. The most common method of adding PE to DQN is to use group equivariant CNNs: e.g., [27] uses group-equivariant convolution to abstract equivariant feature in a Q-network, however the final layer is not equivaraint by construction. Similarly, [28] use equivariant convolution to replace convolutional layers in a Q-network to make it fully group equivariant. For multi-agent RL, [29] proposes to use weight selection or hypernetworks to match PE or PI weights. For applications in software-defined networks, [30] use a network combining PE and PI outputs. An approach to implement PE RL without PE networks, such as group equivariant state and action embedding, is followed in [31].

C. Multi-Armed Bandit

Multi-Armed Bandit (MAB) is a classic, widely-studied RL problem illustrating the exploration-exploitation dilemma. Numerous variant of the problem have been studied in literature. In this work we consider the classical variant with a single player, finite horizon, n machines, and a fixed winning probability per machine. Several non-deep learning algorithms have been applied to address the MAB problem, with prominent ones being Thompson Sampling [32] and UCB1 [3]. The work in [33] uses Q-learning to solve MAB, but instead of using Q-learning to evaluate actions directly, the Gittins index is predicted. A deep-learning approach is taken by [34], who propose a procedure for network learning and study the balance between exploration and exploitation.

III. Preliminaries

A. Permutation Equivariance (PE)

For an input $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n) \in \mathbb{R}^{n\times d}$ with n components of dimension d, a permutation of $\mathbf{X}$ is denoted as $\pi(\mathbf{X})$, and the set of possible permutations is denoted as $\mathbf{S}_n$.

Definition III.1 (Permutation equivariant function)**.** A function $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d'}$ is permutation equivariant if:

$$\forall \pi \in \mathbf{S}_n, \forall \mathbf{X} \in \mathbb{R}^{n\times d}\colon\ f(\pi(\mathbf{X})) = \pi(f(\mathbf{X}))\ .$$

Combining PE functions with corresponding input and output dimensions through composition retains the PE property.

Theorem III.1 (Composition of PE functions)**.** Whenever $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d'}$, $g : \mathbb{R}^{n\times d'} \to \mathbb{R}^{n\times d''}$, and $h : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d'}$ are PE functions, so are $g \circ f$ and $f + h$.

B. Deep Q-Network

Deep reinforcement learning applies deep neural networks to traditional RL techniques. One of the pioneering DRL frameworks is the class of Deep Q-Networks (DQNs) [1]. The approach builds upon Q-learning: The Q-function on the domain of state-action pairs estimates the expected cumulative future reward $Q(\mathbf{s}, a)$ of choosing action a in state $\mathbf{s}$. In DQNs, a neural network approximates the Q-function. During the learning phase, each transition from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$ and its reward r_t form an experience $(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}, r_t, \mathit{done})$ where $\mathit{done} \in \{0, 1\}$ is a Boolean flag denoting if the transition is terminal ($\mathit{done} = 1$) or not ($\mathit{done} = 0$). The target Q-values $Q(\mathbf{s}_t, a_t)$ are calculated with the Bellman Equation [1]:

$$Q(\mathbf{s}_t, a_t) = r_t + (1 - \mathit{done}) \cdot \gamma \cdot \max_{a'} Q(\mathbf{s}_{t+1}, a') \quad (1)$$

where γ is an appropriately chosen discount factor.

During the inference phase, in a given state $\mathbf{s}$, the action $\operatorname{argmax}_a Q(\mathbf{s}, a)$ with the maximal Q-value is chosen. The DQN framework gained widespread attention with its success in playing various Atari video games at a superhuman level [1].

IV. Permutation Equivariant Deep Q-Networks

In deep learning, certain classes of inputs have been shown to profit from corresponding NN structures. Recurrent Neural Networks (RNNs) are capable of capturing information in time series, Convolutional Neural Networks (CNNs) abstract local patterns, and the attention mechanism in Transformers [35] is capable of interpreting the relations between input pairs. By analogy, problems that are inherently PE may profit from respective networks. In this section, we formulate the MAB problem as an RL problem to show the potential benefit of PE RL agents. Then, we study methods to construct PE neural networks, in particular, obtaining a PE DQN for the MAB problem.

A. Multi-Armed Bandit as RL Problem

We assume a classical variant with a player facing n slot machines (one-armed bandits), each machine $i \in [\![1, n]\!]$ having a fixed unknown probability p_i of giving a winning reward r_{win}. Players aim to maximize the cumulative reward from the machine within a fixed (but potentially unknown) number of rounds. A strategy of a player has to gather information about the winning probabilities (exploration) and use the gathered information to generate reward (exploitation).

Stated as an MDP (Fig. 1), we choose some properties of machines as components of the global state: the number of rounds machine i was played and how often it returned a reward up to (and including) round t is recorded in $\mathbf{s}_{t,i} = (\text{played}, \text{rewarded})$. In particular, $\mathbf{s}_{0,i} = (0, 0)$. The global state at round t is given as $\mathbf{s}_t = (\mathbf{s}_{t,1}, \mathbf{s}_{t,2}, \ldots, \mathbf{s}_{t,i}, \ldots, \mathbf{s}_{t,n})$. At each round there are n possible actions: one for each machine i. If the player chooses to play machine i at round t (that is, $a_t = i$), they win with probability p_i the reward $r_t = r_{win}$, or lose the reward r_{loss}. In both cases, the updated state $\mathbf{s}_{t+1}$ is identical to $\mathbf{s}_t$, except for $\mathbf{s}_{t+1,i} = \mathbf{s}_{t,i} + (1, 1)$ in case it was a win and $\mathbf{s}_{t+1,i} = \mathbf{s}_{t,i} + (1, 0)$ in case it was a loss.

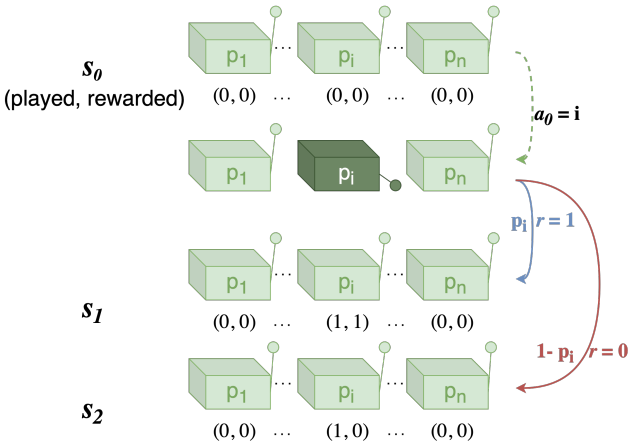


Fig. 1: Illustration of the MAB problem as an MDP. Global states $\mathbf{s}_0$, $\mathbf{s}_1$, ... are composed of individual machine states and changed upon actions and a probabilistic state transition.

A classical DQN approach uses a fully connected network that takes a flattened state as input and outputs a vector of Q-values, with components corresponding to the machines. However, for such a network, a permutation of the machines is interpreted as a different state. For an MAB instance with n machines, a fully connected DQN has to discover a state space $n!$ times more extensive than an agent that considers the permutations of a state to be identical, potentially shortening the required learning phase.

B. Permutation Equivariant Networks

Towards networks that are PE by design, we discuss basic networks with n input and output components before composing them into more complex solutions.

a) Shared Individual Networks: The most straightforward PE structure is using a common individual-machine network for all machines. Each machine-component of the input $\mathbf{X} \in \mathbb{R}^{n \times d}$ passes through a common NN $f : \mathbb{R}^d \to \mathbb{R}^{d'}$, and the individual outputs are composed into a global output, maintaining the initial input order. The network (Fig. 2a) is PE by design:

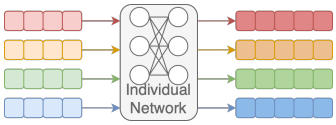
$$\text{PENN}_{\text{ind}}(\mathbf{X}) = (f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_i), \ldots, f(\mathbf{x}_n)) \quad (2)$$

While this structure can capture local information from each individual machine, it lacks a view of the global state.

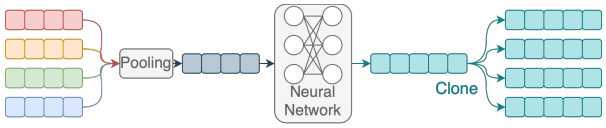
b) Pooling Network: A pooling network (Fig. 2b) comprises two main components: a pooling function $\text{Pooling} : \mathbb{R}^{n \times d} \to \mathbb{R}^d$, and a subsequent neural network $g : \mathbb{R}^d \to \mathbb{R}^{d'}$. The choice of pooling function is flexible, with options including max-pooling, average-pooling, or sum-pooling. Initially, the input undergoes pooling before being fed through the neural network. The resulting network output is then replicated n times to match the input dimension:

$$\text{PENN}_{\text{pooling}}(\mathbf{X}) = g(\text{Pooling}(\mathbf{X})) \otimes \mathbb{1}_n \quad (3)$$

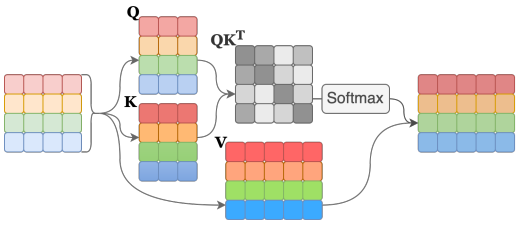
The pooling network is permutation invariant, and thus PE, since all output components are identical. Pooling networks



(a) Shared Individual Network



(b) Pooling Network



(c) Self-Attention

Fig. 2: Basic PE network structures. Colors indicate the dependence of outputs on inputs.

are, to some extent, opposite to individual networks: They provide a global view of the entire input, but the local information of input individuals is no longer distinguished.

c) Self-Attention: Proposed in [2] to align the input and output of encoder-decoder models dedicated to machine translation, the attention mechanism is an essential component of the state-of-the-art Transformer architecture [35]. The work [14] uses multi-head attention modules to build a PE "Set Transformer." Self-Attention (Fig. 2c shows a simplified structure) is PE by design as can be demonstrated by multiplying the input with a permutation matrix and observing that Q, K, and V are permuted accordingly.

d) Activation Function: Most commonly-used activation functions, such as $\tanh$, ReLU, and ELU, are element-wise, and are thus PE similar to the individual networks mentioned earlier. Other non-element-wise functions, like softmax and maxout, also meet the definition of a PE function. Therefore, we can use common activation functions with the basic PE networks discussed before, obtaining more complex PE networks (Theorem III.1).

C. Composed Permutation Equivariant Structures

Basic PE structures like the shared individual network lead to non-accurate Q-value predictions. This is because, according to (1), the value $\mathrm{Q}(\mathbf{s}_t, a_i)$ depends on $\max_a(\mathrm{Q}(\mathbf{s}_{t+1}, a))$. When calculating $\mathrm{Q}(\mathbf{s}_t, a_i)$, however, the individual network does not take into account the states of the machines other than i. It is thus necessary to combine basic PE structures into more expressive ones.

According to Theorem III.1 more elaborate PE networks can be obtained by composing or summing fundamental PE structures. For example, to integrate the local scope of an individual network with the global scope of a pooling network, we can first merge their outputs using addition and then apply an activation function (σ) to the combined output (Fig. 3):

$$f(\mathbf{X}) = \sigma(\mathrm{PENN}_{\mathrm{pooling}}(\mathbf{X}) + \mathrm{PENN}_{\mathrm{ind}}(\mathbf{X})) \qquad (4)$$

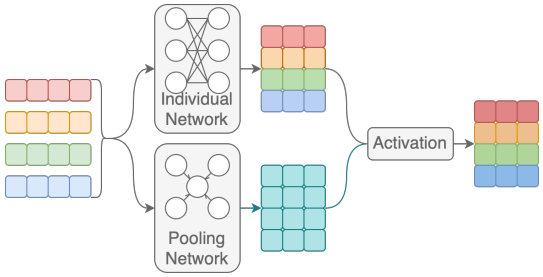


Fig. 3: Example of a composed PE structure as given in (4).

This "stacking-up" of structures and reduction to a single output, e.g., by summation can be used to obtain the single-layer PE network studied in [8]: Indeed, choosing a one-layer perceptron without bias as a "shared individual network" structure and another one as a "pooling network", and then combining them by summation and a subsequent PE activation function as in (4), one obtains the network presented in [8] as a special case.

V. Experiments

With the previously discussed method to systematically construct PE DQNs, we next examine the performance of PE DQNs as Q-networks compared to vanilla DQNs for solving the MAB problem.

A. Training and validation setting

We consider the binary MAB problem with $n = 10$ machines. Rewards are $r_{win} = 1$ and $r_{loss} = -1$. The state of a machine at time t is composed of how many times this machine was played, $cp_{t,i}$, how many times it was winning, $cr_{t,i}$, and how many rounds are left for the game (horizon), $h_t = T - t$. We ran two types of experiments: (i) In so-called single-task experiments, 9 machines have a winning probability of $p = 0.1$ and one machine of $p = 0.9$. The order of the machines is shuffled before each episode. (ii) For the so-called generalization-task experiment, the winning probability $p_i \sim \mathrm{Uniform}(0, 1)$ of each machine is randomly sampled from a uniform distribution before each episode of the game.

To measure the impact of the network structure on performance, we checked four DQN agents with different Q-networks: three are PE, and one is a fully connected network.

- **PE(Indiv+Pooling) DQN.** The PE network of this agent consists of three layers with similar PE compositions of different sizes, connected with ELU activation functions between layers. Each layer is the sum of an individual network and a pooling network of the same dimension, both without activation functions or biases. This structure

is akin to the DeepSet layer. The networks in each layer have dimensions 3×15, 15×15, and 15×1. Mean-pooling is used as the pooling function.

- **PE(Indiv) DQN.** This agent uses a network with the same dimensions as the Ind+Pooling structure but without the pooling module. It can also be viewed as a fully connected network where inputs are individual machine-states and outputs are the corresponding individual Q-values.
- **PE(Att) DQN.** Here, the network consists of one layer of self-attention with dimension $d = 3$ and $d_k = d_v = 9$, followed by an ELU activation function and one Individual+Pooling layer similar to those in the PE(Indiv+Pooling) DQN, with dimensions 9×1.
- **Vanilla DQN.** As a benchmark, the vanilla DQN uses a fully connected network with dimensions 30×150, 150×150, and 150×10. The input dimension is the size of a machine's state times n, as opposed to a machine's state size used for the PE DQNs, since the FCNN cannot take a matrix as a single input. The sizes of the subsequent layers are adjusted accordingly.

All these DQNs use the double DQN architecture proposed in [1], with a separate network to predict the $Q(\mathbf{s}_{t+1}, a_i)$. The target network is synchronized with the online network with a fixed frequency (optimized hyperparameter).

The following classic algorithms have been included in the benchmarks for comparison:

- **UCB1.** First proposed in [3] for MAB, the upper confidence bound (UCB) algorithm is well studied and has many variants. The main idea of the UCB algorithms is choosing the arm with the highest upper bound of the confidence interval for the arm's reward. In UCB1, the bound is calculated as $\mathrm{UCB1}(\mathbf{s}_{t,i}) = \frac{cr_{t,i}}{cp_{t,i}} + c\sqrt{\frac{\ln t}{cp_{t,i}}}$, where c is the exploration constant. We set $c = 0.1\sqrt{2}$ based on hyperparameter optimization (data not shown).
- **Thompson Sampling.** Thompson Sampling [32] (TS) is a probabilistic algorithm that balances exploration and exploitation by sampling from a random distribution of beliefs of the reward distribution. In the case of binary MAB, the rewards follow Bernoulli distributions; hence, the distribution of p_i, following Bayes' rule, is a Beta distribution with $\alpha_{t,i} = cr_{t,i}$ and $\beta_{t,i} = cp_{t,i} - cr_{t,i}$.

We also include one random choice algorithm and one oracle algorithm in the comparison to estimate the lower and upper bounds of performance in the stochastic environment.

- **Random Action.** This algorithm randomly selects machines with equal probabilities for each action.
- **MinRegret.** This algorithm has access to the probabilities p of all machines and systematically chooses to play the machine with the highest probability.

In evaluating the algorithms, the most critical metric is the total rewards collected during the game. In single-task experiments, we assess algorithm performance precisely through the fraction of time the best action is chosen. However, in generalization tasks where probabilities p_i are randomly generated, the advantage of the best arm may not always be evident. Therefore, we use cumulative reward as the primary metric. The machines' winning probabilities influence the possible cumulative rewards collected in one episode. In order to ensure fairness across all algorithms, the winning probabilities are fixed for all algorithms after random initialization in each game during testing.

Another metric we consider is the differences between Q-values of different actions at the same state. For a perfect Q-agent, knowing all the probabilities p_i without the need for exploration, according to the Bellman equation (1), the Q-value $Q(\mathbf{s}_t, a_{t,i})$ is:

$$Q(\mathbf{s}_t, a_{t,i}) = \mathbb{E}(r_i) + \max_j \mathbb{E}(r_j)\frac{1-\gamma^{h_t}}{1-\gamma} \tag{5}$$

where r_i is the immediate reward of playing machine i and $\mathbb{E}(r_i)$ is the expected reward in the total game when playing machine i. The formula indicates that the Q-function grows exponentially with respect to the horizon h_t (and consequently with t) given perfect knowledge of p_i. Specifically, the gap between Q-values of two actions at the same state are approximately constant and bounded above by $\max_i \mathbb{E}(r_i) - \min_j \mathbb{E}(r_j)$. Despite trained Q-values varying due to exploration and uncertainty in practice, the theoretical predictions still serve as a valuable reference for expected Q-values.

Training and hyper-parameters: Optimized hyper parameters are the learning rate $\eta = 0.000005$ for PE(Att) DQN and $\eta = 0.00003$ for the other networks, batch size $|B| = 1$, discount factor $\gamma = 0.99$, and $\epsilon = 0.1$. For each round number T, the agents are trained for 10^4 episodes. The online Q-network and the target Q-network are synchronized every 10 episodes. For the generalization experiments, 10 DQN models were trained for each method, and the best one was selected for validation.

B. Single-task experiment

In the initial phase of the single-task experiment, we evaluate the performance and learning efficiency of DQNs with different network architectures on a single task. We set $T = 200$ for both the training and validation phases. Each DQN undergoes training for 10^4 games, and after every 10 games, the agent is tested on 100 episodes of games to record the average fraction of games choosing the best action. To mitigate the effects of randomness during training, this process is repeated 10 times.

The results are smoothed using a sliding window of 5 epochs and presented in Fig. 4. All PE DQNs outperform the vanilla DQN, with PE(Indiv+Pooling) demonstrating significant advantages. It achieves a precision over 0.8 in choosing the best action in less than 10^3 episodes and maintains this performance consistently throughout training, with the smallest variance among all agents. PE(Indiv) DQN learns rapidly initially but plateaus early. Due to its structure, PE(Att) DQN shows considerable variance and requires more training time but ultimately also outperforms the vanilla DQN.

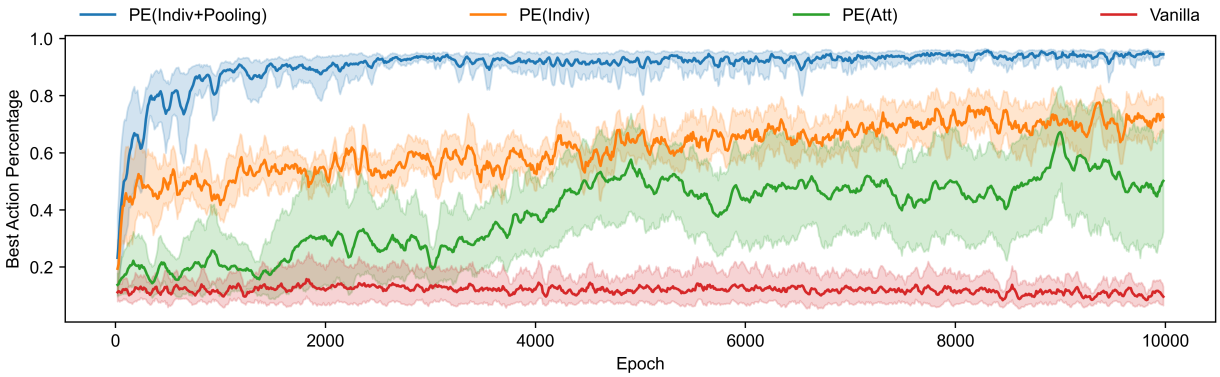


Fig. 4: The percentage of choosing the best action among $n = 10$ machines during training smoothed with a sliding window of 5 episodes, of DQNs. The line and the shaded region represent the mean and 95% confidence interval, respectively.

C. Generalization-task experiment

We tested two aspects of generalization capabilities: (i) Generalization to different settings (different T), testing if a DQN can perform well on games with varying duration T. (ii) Generalization to unseen states, testing if a DQN can play effectively on games with states not encountered during training, assessing adaptability to novel environments.

For that purpose, the DQNs are trained on games with horizons $T = 10a$, where $a \in \{10, 11, \ldots, 20\}$, with each configuration trained for 10^4 episodes per T. After training, they are validated on games with round numbers $T = 10a$, where $a \in \{5, 6, \ldots, 50\}$. For each DQN, 10 agents are trained, and the best-performing agent is selected for validation. The cumulative rewards collected by different algorithms are presented in Fig. 5. Results are averaged over 100 test episodes for each round number and normalized between MinRegret and Random Action.

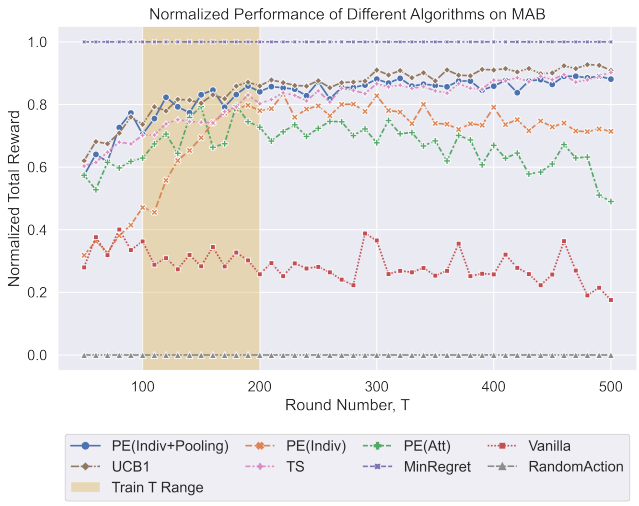


Fig. 5: Performance on the MAB problem (normalized to MinRegret and Random Action) with game duration T from 50 to 500. DQNs are trained for T between 100 and 200.

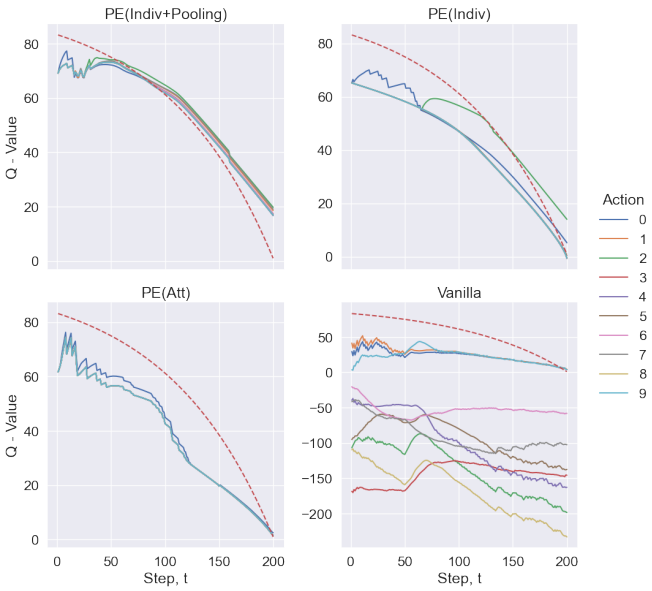
For tasks encountered during training ($100 \leq T \leq 200$), all PE DQN variants noticeably outperform the Vanilla DQN. PE(Indiv+Pooling) DQN demonstrates an advantage compared to Thompson Sampling and matches the performance of UCB1 across most games. Both PE(Indiv) DQN and PE(Att) DQN show improved performance (within $100 \leq T \leq 200$) as the game duration increases, with PE(Att) DQN notably achieving results comparable to Thompson Sampling, while PE(Indiv) DQN catches up after $T > 160$.

For tasks beyond the training range ($T < 100 \mid T > 200$), PE(Indiv+Pooling) DQN maintains its superiority over all other DQNs. It outperforms Thompson Sampling and matches the performance of UCB1 in most cases, even when T reaches 500. On the other hand, PE(Indiv) and PE(Att) demonstrate less strong generalization outside the training T range. PE(Indiv) DQN experiences a rapid performance decline as T decreases to 50, but the decline is less pronounced as T increases to 500. PE(Att) DQN performs best around $T = 200$, with a noticeable decline as T moves away from this value. Its performance reaches a minimum around $T = 400$ but remains superior to Vanilla DQN.

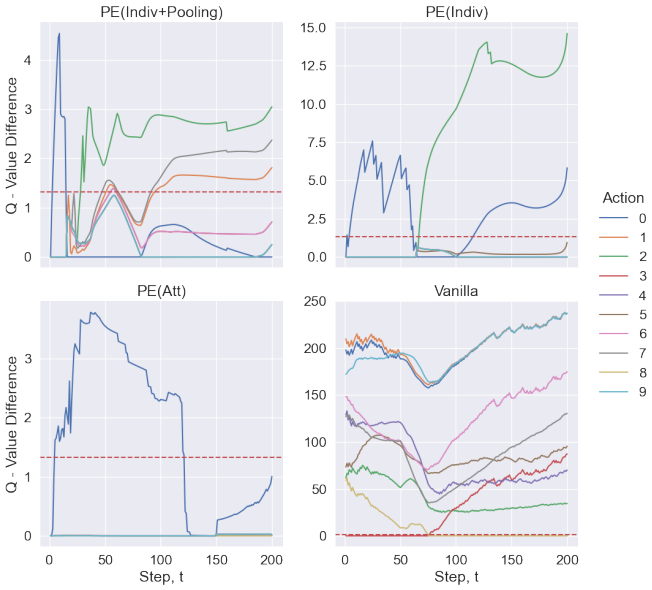
The generalization-task demonstrates that PE DQNs were able to develop universal strategies across similar task settings and extend these strategies to unseen scenarios. The degree of generalizability is specific to the agent's PE network structure, with PE(Indiv+Pooling) showing the highest generalizability and PE(Att) the lowest.

We next take a closer look at the Q-values predicted by the DQNs. Fig. 6 shows the Q-value predictions and the differences between the Q-values of different actions and the minimum predicted Q ($\mathrm{Q}(\mathbf{s}_t, i) - \min_j \mathrm{Q}(\mathbf{s}_t, j)$) in a game with $T = 200$. The red dashed lines in 6a and 6b respectively indicate the Q-values for the best action and the differences of Q-values between the best and worst machines in an optimal network, according to (5).

In Fig. 6a, we observed that the PE(Indiv+Pooling) DQN predicts the Q-values close to optimal, and PE(Indiv) matches the form globally. The PE(Att) DQN predicts Q-values that decrease with t, even though the form of the curve is not perfect. In contrast, the Vanilla DQN's Q-values deviate sig-



(a)



(b)

Fig. 6: Q-values predicted by different DQNs (6a) and their differences between options (6b) in a game with $T = 200$. Training was performed with $T = 10a$, where a varied from 10 to 20. The red dotted line indicates the Q-values of best machine or the difference between the best and the worst machine for an optimal Q-function as in (5).

nificantly from the reference, with the Q-value curve exhibiting varied trends. The biases toward specific machines are also significant, leading to uneven exploration: the machine with the highest initial Q-value is explored more initially. We observe the same pattern of Q-values for some actions towards the end of the game, suggesting that the FC network approximates the PE features via learning, although sub-optimal.

In terms of differences (Fig. 6b), all PE agents show relatively small differences in Q-values of actions, with PE(Indiv+Pooling) having the differences the closest to the reference. The difference predicted by PE(Indiv) is the largest among the PE DQNs, which corresponds to its lack of a global view of the game: when computing $\max_{a'} Q(\mathbf{s}_{t+1}, a')$ during the training phase to obtain target $\mathrm{Q}(\mathbf{s}_t, a_i)$, PE(Indiv) can only consider the case where a' refers to playing itself, i.e., $\max_{a'} Q(\mathbf{s}_{t+1}, a') = Q(\mathbf{s}_{t+1}, a_i)$. Consequently, the differences between Q-values are larger than the differences of the immediate rewards. Unlike the PE DQNs, the differences predicted by Vanilla DQN are more significant, failing to capture the homogeneity of the inputs.

VI. Conclusion

We explored the benefits of directly using permutation equivariance (PE) in reinforcement learning, which allows us to significantly reduce the state space, leading to faster learning and improved performance. We introduced a method to systematically construct PE DQN architectures and validated three such architectures for performance. Our experiments demonstrate that PE DQNs with various network structures consistently outperform vanilla DQNs in MAB games. They exhibit superior generalization capabilities and show a strong correlation in Q-value predictions. Among the PE DQNs tested, the combination of individual and pooling networks emerges as the top performer among DQNs and competes favorably with classical algorithms like UCB1. In future work we aim to apply these methods to more complex PE games.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, ISSN: 1476-4687. DOI: 10.1038/nature14236.

[2] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: 1409.0473 [cs.CL].

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, 2002, ISSN: 1573-0565. DOI: 10.1023/A:1013689704352.

[4] R. Gens and P. M. Domingos, "Deep symmetry networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014.

[5] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 2990–2999.

[6] E. Jenner and M. Weiler, "Steerable Partial Differential Operators for Equivariant Neural Networks," in *International Conference on Learning Representations (ICLR)*, 2022.

[7] H. Maron, O. Litany, G. Chechik, and E. Fetaya, "On learning sets of symmetric elements," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 6734–6744.

[8] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[9] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] Z. Li, Y. Zhang, and Y. Bai, "Geometric invariant representation learning for 3d point cloud," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 2021, pp. 1480–1485. DOI: 10.1109/ICTAI52525.2021.00235.

[11] A. Zhou, K. Yang, K. Burns, *et al.*, "Permutation equivariant neural functionals," in *Advances in Neural Information Processing Systems*, 2023.

[12] D. Wang, J. Y. Park, N. Sortur, L. L. Wong, R. Walters, and R. Platt, "The surprising effectiveness of equivariant models in domains with latent symmetry," in *The Eleventh International Conference on Learning Representations*, 2023.

[13] J. Hartford, D. Graham, K. Leyton-Brown, and S. Ravanbakhsh, "Deep models of interactions across sets," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1909–1918.

[14] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 3744–3753.

[15] R. Kondor and S. Trivedi, "On the generalization of equivariance and convolution in neural networks to the action of compact groups," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 2747–2755.

[16] S. Ravanbakhsh, J. Schneider, and B. Póczos, "Equivariance through parameter-sharing," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 2892–2901.

[17] E. H. Thiede, T. S. Hy, and R. Kondor, *The general theory of permutation equivarant neural networks and higher order graph variational encoders*, 2020. arXiv: 2004.03990 [cs.LG].

[18] H. Pan and R. Kondor, "Permutation equivariant layers for higher order interactions," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, 2022, pp. 5987–6001.

[19] N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai, *Permutation-equivariant neural networks applied to dynamics prediction*, 2016. arXiv: 1612.04530 [cs.CV].

[20] T. S. Cohen and M. Welling, "Steerable CNNs," in *International Conference on Learning Representations*, 2017.

[21] M. Zhdanov, D. Ruhe, M. Weiler, A. Lucic, J. Brandstetter, and P. Forré, "Clifford-steerable convolutional neural networks," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 61 203–61 228.

[22] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 1928–1937.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347 [cs.LG].

[24] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16, Phoenix, Arizona: AAAI Press, 2016, pp. 2094–2100.

[25] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *CoRR*, vol. abs/1511.05952, 2015.

[26] W. Yu, R. Wang, R. Li, J. Gao, and X. Hu, "Historical best Q-networks for deep reinforcement learning," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, pp. 6–11. DOI: 10.1109/ICTAI.2018.00012.

[27] A. K. Mondal, P. Nair, and K. Siddiqi, *Group equivariant deep reinforcement learning*, 2020. arXiv: 2007.03437 [cs.LG].

[28] D. Wang, R. Walters, X. Zhu, and R. Platt, "Equivariant Q learning in spatial action spaces," in *Proceedings of the 5th Conference on Robot Learning*, 2022, pp. 1713–1723.

[29] J. Hao, X. Hao, H. Mao, *et al.*, "Boosting multiagent reinforcement learning via permutation invariant and permutation equivariant networks," in *The Eleventh International Conference on Learning Representations*, 2023.

[30] H. Tsang, I. Akbari, M. A. Salahuddin, N. Limam, and R. Boutaba, "ATMoS+: Generalizable threat mitigation in SDN using permutation equivariant and invariant deep reinforcement learning," *IEEE Communications Magazine*, 2021.

[31] A. K. Mondal, V. Jain, K. Siddiqi, and S. Ravanbakhsh, "EqR: Equivariant representations for data-efficient reinforcement learning," in *Proceedings of the 39th International Conference on Machine Learning*, 2022, pp. 15 908–15 926.

[32] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933, ISSN: 00063444.

[33] M. O. Duff, "Q-learning for bandit problems," in *Machine Learning Proceedings 1995*, A. Prieditis and S. Russell, Eds., San Francisco (CA): Morgan Kaufmann, 1995, pp. 209–217, ISBN: 978-1-55860-377-6.

[34] M. R. W. Dawson, B. Dupuis, M. L. Spetch, and D. M. Kelly, "Simple artificial neural networks that match probability and exploit and explore when confronting a multiarmed bandit," *IEEE Transactions on Neural Networks*, 2009.

[35] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.