PDiT: Interleaving Perception and Decision-making Transformers for Deep Reinforcement Learning

Hangyu Mao
SenseTime Research
Beijing, China
maohangyu@sensetime.com

Rui Zhao
SenseTime Research
Shenzhen, China
zhaorui@sensetime.com

Ziyue Li
University of Cologne
Cologne, Germany
zlibn@wiso.uni-koeln.de

Zhiwei Xu
University of Chinese Academy of Sciences
Beijing, China
xuzhiwei2019@ia.ac.cn

Hao Chen
University of Chinese Academy of Sciences
Beijing, China
chenhao915@mails.ucas.ac.cn

Yiqun Chen
Gaoling School of AI, Renmin University of China
Beijing, China
chenyiqun990321@ruc.edu.cn

Bin Zhang
University of Chinese Academy of Sciences
Beijing, China
zhangbin2020@ia.ac.cn

Zhen Xiao
Peking University
Beijing, China
xiaozhen@pku.edu.cn

Junge Zhang
Institute of Automation, Chinese Academy of Sciences
Beijing, China
jgzhang@nlpr.ia.ac.cn

Jiangjin Yin
Huazhong Agricultural University
Wuhan, China
jiangjinyin@mail.hzau.edu.cn

ABSTRACT

Designing better deep networks and better reinforcement learning (RL) algorithms are both important for deep RL. This work studies the former. Specifically, the Perception and Decision-making Interleaving Transformer (PDiT) network is proposed, which cascades two Transformers in a very natural way: the perceiving one focuses on *the environmental perception* by processing the observation at the patch level, whereas the deciding one pays attention to *the decision-making* by conditioning on the history of the desired returns, the perceiver's outputs, and the actions. Such a network design is generally applicable to a lot of deep RL settings, e.g., both the online and offline RL algorithms under environments with either image observations, proprioception observations, or hybrid image-language observations. Extensive experiments show that PDiT can not only achieve superior performance than strong baselines in different settings but also extract explainable feature representations. Our code is available at https://github.com/maohangyu/PDiT.

KEYWORDS

Deep Reinforcement Learning; Neural Architecture Design for Reinforcement Learning; Transformer for Reinforcement Learning

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

ACM Reference Format:
Hangyu Mao, Rui Zhao, Ziyue Li, Zhiwei Xu, Hao Chen, Yiqun Chen, Bin Zhang, Zhen Xiao, Junge Zhang, and Jiangjin Yin. 2024. PDiT: Interleaving Perception and Decision-making Transformers for Deep Reinforcement Learning. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Deep reinforcement learning (RL) has made great progress in automating decisions in complex tasks, from virtual environments such as mastering video games [31] to real-world applications such as object grasp, autonomous driving, and 6-DoF manipulation [12]. These tasks usually involve multi-modal data.

Generally, an RL agent receives the environmental observations (**perception**) and takes actions accordingly (**decision-making**) to maximize the accumulated rewards. Dealing with environments with multi-modal data can be rather challenging, and various deep learning modules have been employed in perception and decision-making to enhance performance.

For most of the deep RL models to perceive inputs in a specific modality from the environment, the perception module needs to be chosen accordingly. Commonly, Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) with its variances have been adopted to encode low-dimensional or proprioception observation [15], vision [5, 31], and language [17], respectively. It will be more challenging when the environment contains more than one modality. The intuitive solution was combining different perception modules. For instance, DreamerV3 [15] uses a combination of MLPs and CNN for a mixture

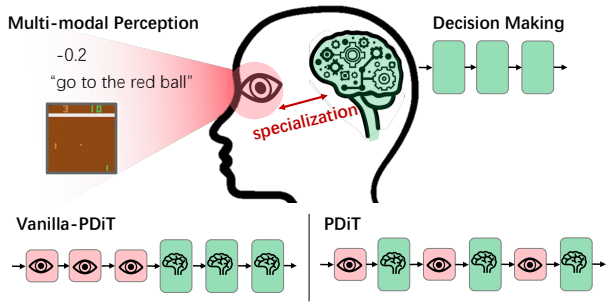

Figure 1: Perception and Decision Making.

of low-dimensional and image inputs. However, such a combination inevitably complicates the models.

From the perspective of decision-making, deep modules such as MLP are commonly used to parameterize the policy, e.g., in classic RL models such as DQN [31] and Actor-Critic [36]. Given the nature of RL as a sequential decision process, Transformer [43] has also recently been adopted, proving that it is a powerful decision-maker.

However, when considering perception and decision-making together in multi-modal environments, challenges emerge when designing the model: an intuitive solution could still be stacking various modules together. We denote Perception and Decision-making as P and D for short: For example, CoBERL [3] use residual network (P) and Transformer (D); Catformer [8] uses CNN (P) and Transformer (D); when dealing with vision-language tasks, RT-1 [4] uses residual network for vision (P), Universal-Sentence-Encoder for language (D), and Transformer for action (D). The models tend to be gradually over-designed and complicated.

In parallel, owing to the Transformer's growing ability in vision-language domains, such as Vision Transformer [2], Uni-Perceiver-v2 [27], PaLM-E [10], it has been proved that Transformer has powerful vision-language multi-modal perception. This development largely encourages researchers to simplify the model design for a multi-modal generalist agent. Ground-breaking offline RL works such as Decision Transformer (DT) [6], Trajectory Transformer (TT) [21], Gato [35], and MGDT [26] only use a single Transformer sequence model: it encodes inputs from various tasks in different modalities with a causal Transformer, and predict the next step's action with the same Transformer in an autoregressive manner. Undoubtedly, this simple design eases the model complexity; however, **fusing perception and decision-making in the same Transformer model could only be sub-optimal**. Instead, delegating perception and decision to two separate modules is intuitively more efficient, and Stooke et al. [41] also proves that a specialized perceiver and a specialized decision-maker could largely accelerate learning. At the same time, a compact model design is still preferred over the module-stacking models.

To this end, we propose the Perception and Decision-making Interleaving Transformer (PDiT) network, which specializes in perception and decision with two respective Transformers as shown in Figure 1. Specifically, the PDiT network is made up of a Perceiving Transformer (perceiver) and a Deciding Transformer (decision-maker): the perceiver processes the observation at the patch level to learn a *good environmental understanding*, while the decision-maker focuses on the *good policy learning* by conditioning on the history of the desired returns, the perceiver's outputs and the actions. We first propose a *Vanilla-PDiT*, which naively stacks the Deciding Transformer on top of the Perceiving Transformer.

However, our experiments find that Vanilla-PDiT cannot always achieve the expected performance. Possible reasons are that the vanilla version has no information exchange between the perception and decision; besides, stacking several Transformer blocks may cause over-global attention and loss of focus [1, 9]. Since the deciding layer is to make decisions, once the hidden deciding layer receives the signal from the perceiving layer, it should directly take actions instead of further embedding the input's representations abstractly by stacking multiple deciding layers.

Thus, we propose the final model, which builds a PDiT block by interleaving the perceiving and deciding Transformer blocks and then stacks L PDiT blocks to form the network. In this way, both perception and decision are considered, and they are still specialized from the perspective of each PDiT block, and the performance is better in practice. Our contributions are summarized:

- The core contribution is the proposed PDiT network (Vanilla-PDiT and full PDiT), which is generally applicable to various deep RL settings: (1) **online and offline**: the online RL like PPO [36], the offline RL like CQL [24], and the offline supervised learning like DT; (2) **multi-modal inputs**: the environments with *image* observations like Atari, the environments with *proprioception* observations like MuJoCo, and the environments with *hybrid image-language* observations like BabyAI; (3) **multi-task reinforcement learning** without changing the network architecture.
- Furthermore, extensive experiments show that the full PDiT can achieve superior performance than strong baselines in different settings. Ablation study and feature visualization give a better understanding of our methods.

2 RELATED WORK

A Landscape. Recently, there is a trend of applying Transformers to deep RL [18, 28, 46]. As shown in Figure 2, we categorize the Transformer-based methods in two dimensions: a) Function specialization: whether the two essential functions of environmental perception and decision-making are fused in one model or specialized with two models; b) Model architecture: whether it is *Transformer-only* or *X+Transformer*, which combines a Transformer with other deep X modules such as CNN to perceive the environment. Specifically, we have the following four combinations: (1) *Specialized: X+Transformer* methods such as GTrXL [33], Catformer [8], CoBERL [3], DT [6] in Atari environment. For instance, Catformer [8] combines CNN with Transformer, where the CNN is specialized for perception, and the Transformer is specialized for decision. In practice, they need to change their perception module when handling different observation types, which may hinder their scalability in complex environments. (2) In contrast, the *Fused: Transformer-only* methods like MGDT [26], TT [21], DT in MuJoCo environment, Gato [35], are purely based on the Transformer, which fuses the role of environmental perception and decision-making simultaneously in one Transformer. These models significantly simplified the model structure and improved the scalability,

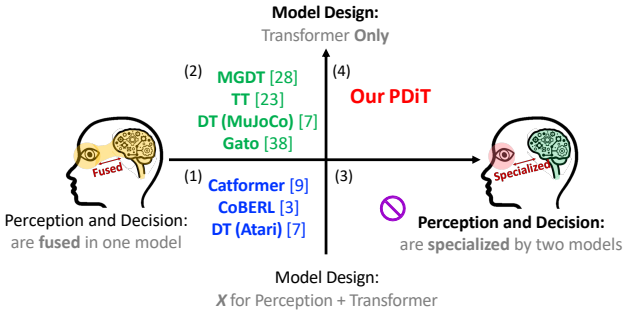

Figure 2: The categories of Transformer-based RL methods, in terms of function specialization (i.e., x-axis) and model design (i.e., y-axis). Our PDiT combines the advantages of recent methods. This illustration does not include all related works for simplicity.

but such a perception-decision fusion in the same model could be sub-optimal and slow in training [41]. For example, DT [6] mixes environmental observations with returns and actions in one Transformer when testing on the MuJoCo tasks, but the performance can be further improved by processing observations separately with another Transformer, as shown by our experiments. (3) There are no *Fused: Transformer+X*, which could be reasonable due to them being potentially over-complicated. (4) The proposed PDiT is, to the best of our knowledge, **the first model that delegates perception and decision-making into two Transformers**, leveraging Transformer's power of dealing with multi-modal data and making sequential decisions.

Concurrent Work. StARformer [38], HDT [7], GDT [19] and our preliminary work TIT [30] also apply two Transformers for better scalability and performance. But we have fundamentally different motivations and implementations: StARformer focuses on balancing short-term and long-term dependencies with mixed Transformer-CNN; HDT aims at hierarchical control with subgoal prediction; GDT applies a causal graph to capture potential dependencies between different concepts and facilitate temporal and causal relationship learning; in contrast, our PDiT combines the advantages of the existing methods by dividing environmental perception and decision-making into two Transformers, and it is applicable for many deep RL settings.

Applying Two Transformers in Other Fields. In multi-agent reinforcement learning, two Transformers are used for asynchronous action coordination [47]. In computer vision, some methods apply two Transformers to handle tasks like image classification and person search, e.g., TNT [16]., ViViT [2], DualFormer [29] and COAT [45]. However, our PDiT and these works are totally different. Although both divide the image input into patches, it is worth noting that this is a standard process for all Vision Transformers. Besides, both have two Transformers but with different designs. For example, TNT's outer Transformer takes the coarse-grain patches, and the inner one further takes the finer patches from the coarse patch. For our PDiT, the perceiver's input is the multi-modal observation (image, proprioception, image-language), and the decision-maker's input is return-observation-action embedding. The perceiving Transformer perceives the environment, and the deciding one takes action. As a result, TNT can only be used for vision tasks, but our PDiT is designed for RL tasks.

3 APPROACH

We consider problems that can be formulated as a Markov Decision Process (MDP), which is formally defined by a tuple $\langle S, A, T, R, \gamma \rangle$, where S is the set of possible states $s \in S$; A represents the set of possible actions $a \in A$; $T(s'|s,a) : S \times A \times S \to [0,1]$ denotes the state transition function; $R(s,a) : S \times A \to \mathbb{R}$ is the reward function; $\gamma \in [0,1]$ is the discount factor. We use s^t, a^t and $r^t = R(s^t, a^t)$ to denote the state, action and reward at timestep t, respectively. Our goal is to learn a policy $\pi(a^t|s^t)$ that can maximize $\mathbb{E}[\hat{R}^t]$ where $\hat{R}^t = \Sigma_{k=t}^{H} \gamma^{k-t} r^k$ is the discounted return, and H is the time horizon. Reinforcement learning [42] is a popular approach to solve the MDP problems. In practice, the environment can be noisy, so we can only get an observation o^t, which contains partial information of the state s^t. We have to learn the policy based on the observation history $[o^{t-K}; ...; o^{t-1}; o^t]$, which will be represented by $[o^k]_{k=t-K}^{t}$ for simplicity and K is the history horizon. This setting is called partially-observable MDP (POMDP) [40]. Given these settings and notations, we introduce our PDiT network as follows.

3.1 Vanilla-PDiT

Vanilla-PDiT is the minimal implementation of PDiT. As shown in Figure 3, it stacks multiple perceiving Transformers to encode the multi-modal input and multiple deciding Transformers to learn the policy via the MDP decision sequences.

Specifically, the perceiving Transformer (which consists of L perceiving blocks) focuses on *the environmental perception* by processing the observation at the patch level, while the deciding Transformer (which is made up of L deciding blocks) pays attention to *the decision-making* (i.e., generating the action a^t) by conditioning on the history of the desired returns $[\hat{R}^k]_{k=t-K}^{t}$, the perceiver's outputs $[z^k]_{k=t-K}^{t}$ and the actions $[a^k]_{k=t-K}^{t-1}$.

However, naively stacking the deciding Transformer on top of the perceiving one cannot always achieve the expected performance, as shown in our experiments. We suppose this is because there is no information interaction between the perceiving and deciding blocks, which may hinder the representation learning for reinforcement learning, as demonstrated by our feature visualization. So we propose the full PDiT to improve the performance.

3.2 PDiT

As shown in Figure 4, the full PDiT first builds a PDiT block by one perceiving block and one deciding block on the top, then stacks L PDiT blocks to form the network. In this way, we **interleave** the perceiving and deciding Transformer blocks, such that the information can be fully fused in every PDiT block, while the perception and decision are still specialized by the perceiving and deciding blocks, respectively. In the following, we introduce the details.

3.2.1 ***Dealing with the Multi-modal Observations.*** Different environments have different types of observations. As shown in Figure 5, we consider three common types: image observation in

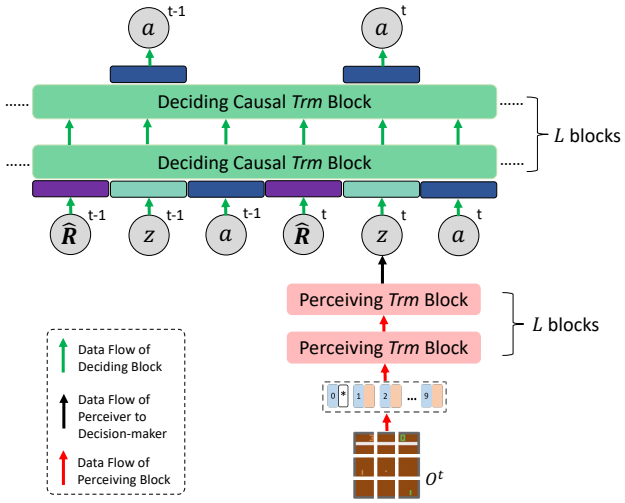

Figure 3: The architecture of the proposed Vanilla-PDiT.

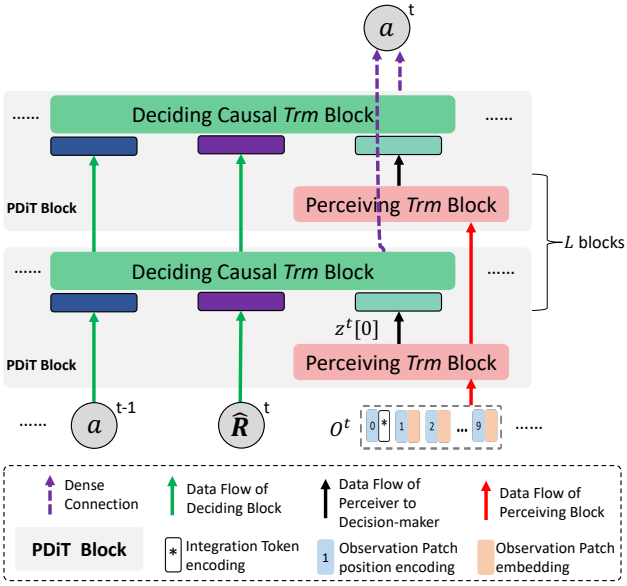

Figure 4: The architecture of the proposed full PDiT, which stacks L PDiT blocks (i.e., the grey rectangle). In each PDiT block, there is a perceiving block and a deciding block that are exactly the same as those of Vanilla-PDiT. Note that the perceiving blocks in the same layer share model parameters across different timesteps.

Atari, proprioception observation in MuJoCo, and hybrid image-language observation in BabyAI.

(1) Transform into Sequential Input. To feed multi-modal input into a Transformer, we will first "sequentize" the data.

Given an **image observation** $o \in \mathbb{R}^{H\times W\times C}$, we split it into a sequence of *observation patches* $o^p \in \mathbb{R}^{N\times(P^2\times C)}$, where (H, W) is the resolution of the original image observation; C is the number of

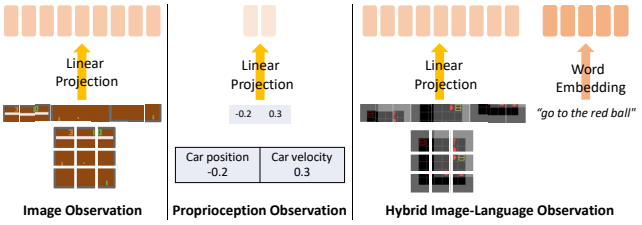

Figure 5: The illustration of generating the observation patch embedding for different types of observations.

channels; (P, P) is the resolution of each patch; and $N = HW/P^2$ is the resulting number of patches, which is known as the context length for the perceiving Transformer. After getting the observation patches, we map each patch o_i^p into the *observation patch embedding* with a trainable linear projection E^p:

$$\hat{z}_0 = [o_1^p E^p; o_2^p E^p; ...; o_N^p E^p] \in \mathbb{R}^{N\times D^p} \tag{1}$$

where $E^p \in \mathbb{R}^{(P^2C)\times D^p}$, and D^p is the dimension of the observation patch embedding.

Given a **proprioception observation** with D entries, i.e., $o \in \mathbb{R}^D$, we take each entry as an observation patch independently: the sequence of observation patches will be $o^p \in \mathbb{R}^{N\times 1}$ where N is equal to D. This is convenient and reasonable since each entry usually has atomic semantics, e.g., in the classic Mountain Car environment, the 2 entries of observation represent the atomic meaning of "Car position" and "Car velocity", respectively. If there is prior knowledge, we can generate the observation patches in other ways (e.g., several entries as a patch). Then, we again use a trainable linear projection $E^p \in \mathbb{R}^{1\times D^p}$ to map the observation patches into the observation patch embeddings.

Given a **hybrid image-language observation**, the image will be processed as above, and the language will be processed using Word Embedding [25]. We concatenate them and get a total number of $N = HW/P^2 + N_w$ observation patch embeddings, where N_w is the number of words in the language. Our method is motivated by Vision-Language Pre-training [22].

(2) Integration Token Encoding. After getting the observation patch embeddings, a trainable *integration token* $z_0^{integration} \in \mathbb{R}^{1\times D^p}$ is added:

$$\check{z}_0 = [z_0^{integration}; \hat{z}_0] \in \mathbb{R}^{(1+N)\times D^p} \tag{2}$$

(3) Observation Patch Position Encoding. We then add the *observation patch position encoding*, which is a trainable parameter E^p_{pos}, to retain positional information:

$$z_0 = \check{z}_0 + E^p_{pos} \qquad E^p_{pos} \in \mathbb{R}^{(1+N)\times D^p} \tag{3}$$

The resulting z_0 serves as the input of the perceiving Transformer.

3.2.2 ***Interleaving Perceiver and Decision Maker.*** Now we introduce the PDiT block.

(1) Perceiving Transformer Block. The concrete operations in the l-th perceiving block $z_l = P\text{-}Trm_l(z_{l-1})$ are as follows:

$$\tilde{z}_l = z_{l-1} + \text{MSA}(\text{LN}(z_{l-1})) \qquad l = 1, ..., L \tag{4}$$

$$z_l = z_{l-1} + \text{FFN}(\text{LN}(\tilde{z}_{l-1})) \qquad l = 1, ..., L \tag{5}$$

where MSA(·), LN(·), and FFN(·) stand for the multi-headed self-attention, layer normalization, and feed-forward network used in the original Transformer [43]. There is no mask in the MSA so that all patches in an observation can be used for good environmental perception. Moreover, the perceiving block can build the *spatial relationships* among observation patches within an observation by computing interactions between two observation patches.

For any layer l, there are $1+N$ tokens in z_l (i.e., $z_l \in \mathbb{R}^{(1+N)\times D^p}$), and the integration token $z_l[0] \in \mathbb{R}^{1\times D^p}$ will serve as the integrated representation of all tokens.

(2) Deciding Transformer Block. As shown in Figure 4, the input of the deciding block is the trajectory across K timesteps. We use the superscript t and the subscript l to represent the timestep and the layer, respectively. The input of the l-th deciding block is:

$$y_{l-1} = [\hat{R}_{l-1}^{t-K}; z_l^{t-K}[0]; a_{l-1}^{t-K}; ...; a_{l-1}^{t-1}; \hat{R}_{l-1}^{t}; z_l^{t}[0]] \quad (6)$$

The detailed operations in the l-th deciding block $y_l = D\text{-}Trm_l(y_{l-1})$ can be shown as follows:

$$\tilde{y}_l = y_{l-1} + \text{MSA}(\text{LN}(y_{l-1})) \qquad l = 1, ..., L \quad (7)$$

$$y_l = y_{l-1} + \text{FFN}(\text{LN}(\tilde{y}_{l-1})) \qquad l = 1, ..., L \quad (8)$$

There are causal masks in the MSA so that the deciding block can build the *temporal relationships* among different trajectory tokens across K timesteps, which is helpful for decision-making in the POMDP setting.

(3) PDiT Block. As shown in Figure 4, the PDiT block combines one perceiving block and one deciding block. Formally, the operations in the l-th PDiT block are:

$$z_l^k = P\text{-}Trm_l(z_{l-1}^k) \qquad k = t-K, ..., t \quad (9)$$

$$y_{l-1} = [\hat{R}_{l-1}^{t-K}; z_l^{t-K}[0]; a_{l-1}^{t-K}; ...; a_{l-1}^{t-1}; \hat{R}_{l-1}^{t}; z_l^{t}[0]] \quad (10)$$

$$y_l = D\text{-}Trm_l(y_{l-1}) \quad (11)$$

In Eq. (9), the perceiving blocks in the same layer share model parameters across different timesteps, so the number of model parameters in PDiT is comparable to existing methods.

Note that for any layer l, there are $2+3K$ tokens in y_l. This is because there are 3 tokens (i.e., $\hat{R}, z[0], a$) for each timestep before t and $\hat{R}^t, z^t[0]$ for timestep t. We use the last token $y_l[-1]$ as the representation of all tokens, which is a conventional operation for causal Transformers.

(4) PDiT Backbone Network. We stack L PDiT blocks to form the full PDiT network, ensuring that the spatial-temporal information is fully fused for good representation and decision. Then, we apply a dense connection design to concatenate the last token $y_l[-1]$ from all PDiT blocks and apply an FFN upon the concatenation to generate the final action:

$$a^t = \text{FFN}([y_1[-1]; y_2[-1]; ...; y_L[-1]]) \quad (12)$$

We found that the dense connection is empirically helpful for performance in the experiments.

3.2.3 ***Why interleaving the perceiving and deciding blocks is better (than stacking all deciding blocks to the end)?*** Possible reasons are as follows. First, the outer is to make decisions. Once it receives the output from the perceiving, it should directly take actions instead of further abstractly embedding the representations by stacking multiple deciding blocks. Second, this interleaving design encourages the information interaction between the perceiving and deciding blocks (but not mixed or fused up). Third, stacking too deep transformer layers may also cause losing focus.

Table 1: PPO, CQL, and DT cover a lot of RL settings.

Setting	PPO	CQL	DT
Online / Offline	√ / ×	× / √	-
On-Policy / Off-Policy	√ / ×	× / √	-
Policy Gradient / Q-Learning	√ / ×	× / √	-
RvS [11]	-	-	√

3.3 Training Methods

This paper focuses on designing networks that are generally applicable to different deep RL settings, so we restrain ourselves from modifying the RL training algorithms and use the existing algorithms to train PDiT. Specifically, we train PDiT with PPO [36] based on Stable-baseline3 [1], CQL [24] based on D3rlpy [2], and Reinforcement Learning via Supervised Learning (RvS) [11] based on the official Decision Transformer (DT) [3]. As shown by Table 1, these algorithms cover a lot of RL settings. We refer the readers to their original papers for more details.

4 EXPERIMENT

4.1 Setting

4.1.1 Evaluation Environments. We use Atari [4], MuJoCo[5], and BabyAI [6] as the evaluation environments. Specifically, Atari has image observations and discrete action spaces; MuJoCo has proprioception observations and continuous action spaces; while BabyAI has hybrid image-language observations and discrete action spaces. We consider these environments because they are credible benchmarks and their settings are diverse.

4.1.2 Baseline Deep Networks. Since PDiT is based on Transformers, we will mainly compare it with other Transformer-based networks. (1) Environment with image observations: we compare with NatureCNN [31], ResNet [37], ResNet + Transformer (i.e., Catformer [8]), ResNet + Transformer + Gating + LSTM (i.e., CoBERL [3]). (2) Environments with proprioception observations and hybrid observations: we compare with MLP, DT [6], Transformer-based Behavior Cloning (i.e., GATO [35]).

4.1.3 Hyperparameter Tuning. To ensure fair comparison, we follow recent works [6, 19] and report results of most baselines from their original papers. We believe that these results are already tuned to the best by the original authors. For baselines run by us (only ResNet, Catformer, CoBERL and GATO), the common hyperparameters (e.g., training steps, learning rate, discount factor) are set as

[1] https://stable-baselines3.readthedocs.io/
[2] https://d3rlpy.readthedocs.io/
[3] https://github.com/kzl/decision-transformer/
[4] https://www.gymlibrary.dev/environments/atari/
[5] https://www.gymlibrary.dev/environments/mujoco/
[6] https://minigrid.farama.org/environments/babyai/

Table 2: The episode returns for online Atari environments (trained by PPO). For NatureCNN, we report numbers directly from the official Stable-baseline3; other numbers are run by us. Compared to other CNN-based and Transformer-CNN-mixed networks, PDiT achieves the highest average performance with five independent runs (improving ResNet by 15.0%). Best result is in boldface, and the second best is underlined.

	Breakout	MsPacman	Pong	SpaceInvaders	Average
NatureCNN	398±33	1754±172	20.989±0.105	960±425	783.25
ResNet	397±57	1807±405	**21.000±0.000**	1700±511	981.25
ResNet + Transformer (i.e., Catformer)	242±41	1579±461	19.980±0.139	1427±597	817.00
ResNet + Transformer + LSTM + Gating (i.e., CoBERL)	358±34	2190±327	19.460±1.557	821±314	847.12
PDiT (ours)	**411±68**	**2246±326**	20.750±1.577	**1837±168**	**1128.69**

Table 3: The episode returns for offline MuJoCo (trained by RvS). We also include several concurrent methods (i.e., StARformer and GDT). DT and GDT results are reported from original papers; StARformer are taken from GDT; GATO is run by us. PDiT achieves the highest average performance over five runs (beat DT, StARformer, and GDT by 6.4%, 40.7%, and 2.8%).

Dataset	Environment Name	DT	GATO	PDiT (ours)	StARformer	GDT
Medium	Halfcheetah	42.6±0.1	42.9±1.7	42.8±2.3	**42.9±0.1**	**42.9±0.1**
	Hopper	67.6±1.0	58.7±2.5	**68.2±2.4**	59.5±4.2	65.8±5.8
	Walker2d	74.0±1.4	**77.8±1.2**	77.6±0.6	73.8±3.5	**77.8±0.4**
Medium-Replay	Halfcheetah	36.6±0.8	36.9±3.8	**40.8±2.3**	36.8±3.3	39.9±0.1
	Hopper	82.7±7.0	33.8±4.3	**89.6±2.7**	29.2±4.3	81.6±0.6
	Walker2d	66.6±3.0	64.7±0.8	74.1±0.6	39.8±5.1	**74.8±1.9**
Average-normalized		61.68	52.47	**65.52**	47.00	63.80
Average-original		4065.94	3443.94	**4325.27**	3074.53	4209.11

Table 4: The episode returns for offline MuJoCo (trained by CQL). We also compare other offline RL algorithms, i.e., BEAR [23], BRAC-v [44] and AWR [34]. Here, CQL-MLP numbers are reported from the original paper; BEAR, BRAC-v and AWR are reported from the D4RL paper [13]. Compared to baselines, PDiT achieves the highest performance (improving MLP by 42.5%).

Dataset	Task Name	CQL-MLP	PDiT (ours)	BEAR	BRAC-v	AWR
Medium	Halfcheetah	44.4	42.6±0.1	41.7	**46.3**	37.4
	Hopper	58.0	**100.8±0.2**	52.1	31.1	35.9
	Walker2d	79.2	**84.1±0.9**	59.1	81.1	17.4
Medium-Replay	Halfcheetah	46.2	**47.8±0.1**	38.6	47.7	40.3
	Hopper	48.6	**99.2±1.7**	33.7	0.6	28.4
	Walker2d	26.7	**53.6±3.7**	19.2	0.9	15.5
Average-normalized		50.52	**71.35**	40.73	34.62	29.15
Average-original		3312.25	**4719.00**	2651.08	2238.44	1869.03

the open-source code repositories, while some specific hyperparameters (e.g., the number of network layers, embedding size) are tuned using the random grid-search as PDiT.

4.2 Result

Training with PPO under Atari Environments. The results are shown in Table 2. As can be seen, the conventional NatureCNN and ResNet can achieve satisfactory scores in most Atari environments. Compared to these conventional networks, ResNet + Transformer (i.e., Catformer) and ResNet + Transformer + LSTM + Gating (i.e., CoBERL) can achieve better average performance than NatureCNN but are worse than ResNet. It is because these mixed CNN-Transformer methods are unstable across different tasks, e.g., CoBERL gets a very high score in MsPacman, but the lowest score in SpaceInvaders. Compared to these baselines, PDiT achieves good performance in most testing cases.

Training with RvS under MuJoCo Environments. As observed in Table 3, PDiT is better than DT and GATO, especially in practical datasets with complex distributions. Specifically, PDiT matches or exceeds DT and GATO by a small margin in "Medium"

Table 5: The episode returns for offline BabyAI tasks.

	DT	GATO	PDiT (ours)
GoToRedBall	0.969±0.01	0.985±0.00	**0.994±0.00**
GoToLocal	0.884±0.02	0.923±0.03	**0.991±0.00**
PickupLoc	0.719±0.02	0.755±0.01	**0.954±0.01**
PutNextLocal	0.342±0.01	0.435±0.02	**0.881±0.01**
Average	0.729	0.775	**0.955**

datasets generated from a single policy. In contrast, PDiT outperforms DT and GATO by a large margin in "Medium-Replay" datasets (also known as the "Mixed" datasets) that combine multiple policies. Since the mixed datasets with complex distributions are more likely to be common in practice, as mentioned in [24], we expect that PDiT will work better than DT and GATO in practical applications.

Training with CQL under MuJoCo Environments. In the previous experiments, we find that training with RvS can sometimes result in small improvements. One possible reason may be that the ability of RvS to stitch offline data is not as strong as TD-learning. Thus, we test whether training with CQL, the state-of-the-art in offline TD-learning, can further improve PDiT. The results in Table 4 show that PDiT can improve CQL-MLP by 42.5% on average.

Training with RvS under BabyAI Environments. The results shown in Table 5 demonstrate that PDiT is obviously better than DT and GATO under environments with hybrid image-language observations and sparse rewards.

Summary: All of the above results together prove that the proposed PDiT is generally applicable to a lot of RL settings, i.e., different RL training algorithms and evaluation environments with diverse characteristics.

More Results: We provide more results on other classical scenarios and multi-task reinforcement learning in the Appendix 4. To highlight, we perform multi-task reinforcement learning with one compact network, our PDiT, without changing the network architecture. We use multiple tasks (e.g., GoToRedBall, GoToObj, and GoToSeq) to train PDiT, then evaluate PDiT on each task separately. PDiT with multi-task learning achieves a better average return than baselines with single-task learning.

4.3 Ablation Study

We consider the following ablation networks:

(1) **Existing networks like DT**: There is one Transformer, which is worse than PDiT as shown by Table 2 and 3.
(2) **w/o Perceiver**: This removes the perceiving Transformer of PDiT, so only the deciding Transformer is used. In this network, the input of the deciding blocks is the *observation embedding*, which is generated by linearly mapping the whole observation with a trainable parameter.
(3) **w/o Decision-maker**: This removes the deciding Transformer of PDiT, so only the perceiving Transformer is used. In this network, the K observations are stacked in the same way as DQN and then processed by the perceiving blocks to generate the action.

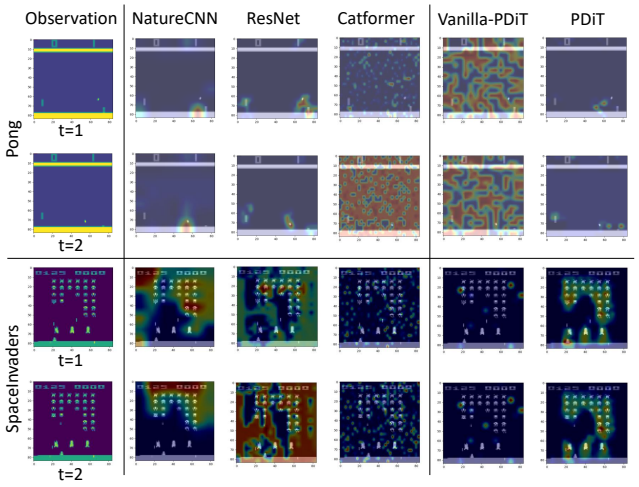

Figure 6: The feature gradient visualization of 5 different methods. The first column shows the original observations randomly generated from Pong and SpaceInvaders: in the observations, objects need to be perceived.

(4) **Vanilla-PDiT**: There are two Transformers, but they are stacked naively as mentioned in Section 3.1.
(5) **w/o Dense**: There are two Transformers, but the dense connection design is removed from the PDiT. In this network, only the output of the last PDiT block is used to generate the action (i.e., $a^t = \text{FFN}(y_L[-1])$).

The ablation results are shown in Table 6. (1) Compared to PDiT, Vanilla-PDiT has the maximum performance degradation. It implies that although two Transformers are necessary for PDiT, naively combining them cannot get the expected performance, and the design of interleaving the perception and decision makes the most contribution. (2) Besides, "w/o Dense" has the second worst performance: this result verifies that the dense connection design is important. This is intuitive and reasonable since the dense network feeds all the outputs of each deciding Transformer to the final output, with more abundant information. As far as we know, previous studies have shown that the dense connection is also important for networks that are based on the CNN [20] and MLP [32, 39], but PDiT is the first work to demonstrate this for purely Transformer-based networks. (3) Furthermore, both "w/o Perceiver" and "w/o Decision-maker" perform worse than PDiT, which indicates that both perceiving and deciding blocks are necessary for good performance.

4.4 Visualization

4.4.1 ***Feature Gradient Visualization***. We visualize the feature gradient of different methods by Grad-CAM [14], which is a typical method for visualization and explainability of deep networks. The results are shown in Figure 6.

From the spatial perspective, PDiT generates more explainable gradient maps than other methods. As can be seen in Figure 6, the gradient of NatureCNN, ResNet, and PDiT is highly correlated with the objects in the observations in most cases. In contrast,

Table 6: The episode returns of ablation networks for online Atari environments (trained by PPO).

		Breakout	MsPacman	Pong	SpaceInvaders	Average	Improve
One Transformer	w/o Perceiver	276±59	1588±516	19.350±2.441	1363±91	811.59	-28.09%
	w/o Decision-maker	229±83	1591±396	20.180±2.034	1295±233	783.80	-30.56%
Two Transformers	Vanilla-PDiT	169±91	748±205	9.600±6.445	752±77	419.65	**-62.82%**
	w/o Dense	121±34	1372±192	18.620±3.267	938±256	612.41	-45.74%
	Full PDiT	411±68	2246±326	20.750±1.577	1837±168	1128.69	-

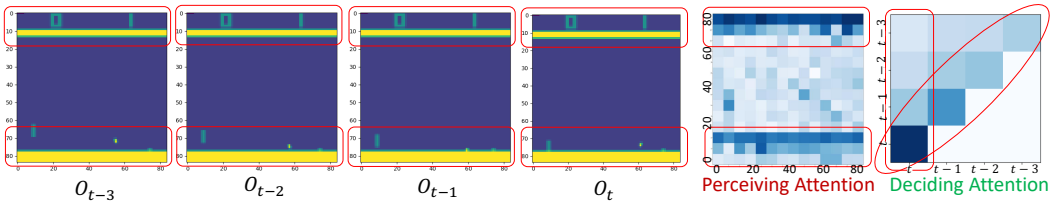

Figure 7: The attention weights visualization of PDiT. Observations (i.e., the first four images) are randomly generated from Pong. The fifth and last images are the attention weights of the perceiving block and the deciding block, respectively.

Catformer and Vanilla-PDiT have disorganized gradient maps, and they sometimes generate unexplainable gradients, as shown by the Pong case. Furthermore, in the SpaceInvaders task, the gradient values are often positively-correlated with the density of the invaders (namely, the attention color is darker where there are many invaders that are close to the agent).

From the temporal perspective, PDiT generates more stable gradient maps than other methods. For example, when the observation changes slightly over a small time step, its gradient map does not change significantly. Thus, PDiT may learn consistent temporal representations for good decision-making, which may explain its stable performance across different tasks. In contrast, gradient maps of NatureCNN and Catformer have changed a lot.

4.4.2 ***Attention Weight Visualization***. In the above section, we have visualized the *feature gradient* of different methods by *Grad-CAM*. In this section, we visualize the *attention weights* of the *MSA operation* from the perceiving and deciding Transformers of our PDiT, as shown in Figure 7.

The analyses from the spatial perspective. For the *PDiT's perceiving block*: Here, we use the integration token as the key in the MSA operation because it serves as the integrated representation of all observation patches. As we can see, the observation image's top and bottom regions get higher attention weights since the top region is the game score, and the bottom region contains the paddles and balls, which are important for the game. In contrast, the central region of the observation image is mainly the background, which may have little influence on the game, so the attention weight values of the middle part are small. Based on this visualization, we conclude that the perceiving Transformer of PDiT can indeed capture important spatial information for good decision-making.

The analyses from the temporal perspective. From *the deciding block*'s perspective: Here, we use four observations to form the observation history to match the setting of the original DQN [31]. As we can see, the current observation o_t usually gets the highest attention weights (i.e., the color on the diagonal is darker); moreover, the observation (e.g., o_{t-1}) close to the current timestep gets higher attention weight than the far away ones (e.g., o_{t-3} and o_{t-2}); therefore the most updated information can be attended and used for decision-making. Based on this visualization, we conclude that PDiT (and the deciding Transformer) can indeed capture important temporal information for good decision-making.

5 CONCLUSION AND FUTURE WORK

This paper proposed the Perception and Decision-making Interleaving Transformer (PDiT) network for deep RL. The key insight is cascading two specialized Transformers in a very natural way: the perceiving one processes the observations for good environmental perception, while the deciding one focuses on good decision-making. As far as we know, specializing perception and decision by a pure Transformer-based network is achieved for the first time. The experiments demonstrated that PDiT can achieve good results in many RL settings. Furthermore, PDiT can extract explainable representations from both spatial and temporal perspectives.

With the goal to conceptually demonstrate the potential advantages of interleaving perception and decision-making for deep RL, we simply implement PDiT using the basic Transformer [43]. We expect that PDiT could further improve the performance with more advanced Transformers and dedicated optimization skills (although this is not the focus of this paper). We give more discussions about the limitations and future directions in Appendix 5.

ACKNOWLEDGMENTS

The authors would like to thank Dong Li, Jianye Hao and the anonymous reviewers for their insightful comments. This work is supported by Basic Cultivation Fund project, CAS (JCPYJJ-22017), Strategic Priority Research Program of Chinese Academy of Sciences (XDA27010300), and the National Natural Science Foundation of China (Grants No. 62302189).

REFERENCES

[1] Gosthipaty Aritra Roy and Paul Sayak. 2023. *Investigating Vision Transformer representations.* https://keras.io/examples/vision/probing_vits/

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. 2021. ViViT: A Video Vision Transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 6816–6826. https://doi.org/10.1109/ICCV48922.2021.00676

[3] Andrea Banino, Adria Puigdomenech Badia, Jacob C Walker, Tim Scholtes, Jovana Mitrovic, and Charles Blundell. 2022. CoBERL: Contrastive BERT for Reinforcement Learning. In *International Conference on Learning Representations*.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817* (2022).

[5] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. 2023. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13734–13744.

[6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems* 34 (2021), 15084–15097.

[7] André Correia and Luís A Alexandre. 2022. Hierarchical Decision Transformer. *arXiv preprint arXiv:2209.10447* (2022).

[8] Jared Q Davis, Albert Gu, Krzysztof Choromanski, Tri Dao, Christopher Re, Chelsea Finn, and Percy Liang. 2021. Designing stable transformers via sensitivity analysis. In *International Conference on Machine Learning*. PMLR, 2489–2499.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[10] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378* (2023).

[11] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. 2021. RvS: What is Essential for Offline RL via Supervised Learning?. In *International Conference on Learning Representations*.

[12] Richard Evans and Jim Gao. 2016. Deepmind AI reduces Google data centre cooling bill by 40%. *DeepMind blog* 20 (2016), 158.

[13] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219 [cs.LG]

[14] Jacob Gildenblat and contributors. 2021. PyTorch library for CAM methods. https://github.com/jacobgil/pytorch-grad-cam.

[15] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering diverse domains through world models. *preprint arXiv:2301.04104* (2023).

[16] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. 2021. Transformer in transformer. *Advances in Neural Information Processing Systems* 34 (2021), 15908–15919.

[17] Matthew Hausknecht and Peter Stone. 2015. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*.

[18] Shengchao Hu, Li Shen, Ya Zhang, Yixin Chen, and Dacheng Tao. 2022. On Transforming Reinforcement Learning by Transformer: The Development Trajectory. *arXiv preprint arXiv:2212.14164* (2022).

[19] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. 2023. Graph Decision Transformer. *arXiv preprint arXiv:2303.03747* (2023).

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2261–2269. https://doi.org/10.1109/CVPR.2017.243

[21] Michael Janner, Qiyang Li, and Sergey Levine. 2021. Reinforcement learning as one big sequence modeling problem. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*.

[22] Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*. PMLR, 5583–5594.

[23] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems* 32 (2019).

[24] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[25] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*. PMLR, 957–966.

[26] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. 2022. Multi-game decision transformers. *Advances in Neural Information Processing Systems* 35 (2022), 27921–27936.

[27] Hao Li, Jinguo Zhu, Xiaohu Jiang, Xizhou Zhu, Hongsheng Li, Chun Yuan, Xiaohua Wang, Yu Qiao, Xiaogang Wang, Wenhai Wang, et al. 2023. Uni-perceiver v2: A generalist model for large-scale vision and vision-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2691–2700.

[28] Wenzhe Li, Hao Luo, Zichuan Lin, Chongjie Zhang, Zongqing Lu, and Deheng Ye. 2023. A Survey on Transformers in Reinforcement Learning. *arXiv preprint arXiv:2301.03044* (2023).

[29] Yuxuan Liang, Pan Zhou, Roger Zimmermann, and Shuicheng Yan. 2022. DualFormer: Local-Global Stratified Transformer for Efficient Video Recognition. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 577–595.

[30] Hangyu Mao, Rui Zhao, Hao Chen, Jianye Hao, Yiqun Chen, Dong Li, Junge Zhang, and Zhen Xiao. 2022. Transformer in Transformer as Backbone for Deep Reinforcement Learning. *arXiv preprint arXiv:2212.14538* (2022).

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

[32] Kei Ota, Tomoaki Oiki, Devesh Jha, Toshisada Mariyama, and Daniel Nikovski. 2020. Can Increasing Input Dimensionality Improve Deep Reinforcement Learning?. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 7424–7433. https://proceedings.mlr.press/v119/ota20a.html

[33] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. 2020. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*. PMLR, 7487–7498.

[34] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177* (2019).

[35] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. 2022. A generalist agent. *arXiv preprint arXiv:2205.06175* (2022).

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[37] Rutav M Shah and Vikash Kumar. 2021. RRL: Resnet as representation for Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 9465–9476. https://proceedings.mlr.press/v139/shah21a.html

[38] Jinghuan Shang, Kumara Kahatapitiya, Xiang Li, and Michael S Ryoo. 2022. StARformer: Transformer with State-Action-Reward Representations for Visual Reinforcement Learning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*. Springer, 462–479.

[39] Samarth Sinha, Homanga Bharadhwaj, Aravind Srinivas, and Animesh Garg. 2020. D2rl: Deep dense architectures in reinforcement learning. *arXiv preprint arXiv:2010.09163* (2020).

[40] Matthijs TJ Spaan. 2012. Partially observable Markov decision processes. In *Reinforcement Learning*. Springer, 387–414.

[41] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. 2021. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*. PMLR, 9870–9879.

[42] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

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

[44] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).

[45] Rui Yu, Dawei Du, Rodney LaLonde, Daniel Davila, Christopher Funk, Anthony Hoogs, and Brian Clipp. 2022. Cascade Transformers for End-to-End Person Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7267–7276.

[46] William Yuan, Jiaxing Chen, Shaofei Chen, Lina Lu, Zhenzhen Hu, Peng Li, Dawei Feng, Furong Liu, and Jing Chen. 2023. Transformer in Reinforcement Learning for Decision-Making: A Survey. (2023).

[47] Bin Zhang, Hangyu Mao, Lijuan Li, Zhiwei Xu, Dapeng Li, Rui Zhao, and Guoliang Fan. 2023. Stackelberg Decision Transformer for Asynchronous Action Coordination in Multi-Agent Systems. *arXiv preprint arXiv:2305.07856* (2023).