視覚言語モデルによる リアルタイム制御のための深層強化学習手法

森田 知颯 [1]　鶴岡 慶雅 [1]
[1] 東京大学
{morita-chihaya, tsuruoka}@logos.t.u-tokyo.ac.jp

概要

視覚言語モデル (Vision Language Model, VLM) は意思決定エージェントとして期待される一方，推論時間の長さによる「遅延」がリアルタイムな環境への応用の壁となっていた．

本研究では，遅延を考慮したマルコフ決定過程に基づき，一度の推論で未来の行動系列（アクションチャンク）を予測させることで遅延を補償する新たな強化学習フレームワークを提案する．これにより推論中も動作を継続可能にし，リアルタイム環境での安定した VLM 制御の実現を目指す．

1　はじめに

近年，大規模言語モデル (Large Language Model, LLM) の能力を視覚情報へと拡張した, 視覚言語モデル (Vision Language Model, VLM) が急速に発展している．VLM は，画像の内容を理解し，それに関する質問に答え，人間との対話を行うといった受動的なタスクにおいて目覚ましい性能を示してきた [1, 2]．その応用範囲は静的な画像理解に留まらず，VLM を意思決定の主体，すなわち「エージェント」として捉え，ゲーム [3, 4] やロボット制御 [5] といった動的な環境と直接インタラクションさせる研究が活発化している．

しかし，一瞬の判断を要するアクションゲーム等，リアルタイム環境への応用には課題がある．VLM は巨大モデルと自己回帰生成ゆえに，観測から行動決定までに無視できない推論遅延が生じる．例えば 60fps のゲームでは数百ミリ秒の遅延は数十フレームの遅れに相当し，常に過去の状況に対し行動することになる．この問題は，リアルタイム制御における最大のボトルネックとなっている．

本研究の目的は，推論遅延を克服し，VLM を説明可能なリアルタイム環境の制御エージェントとする新たな強化学習手法の提案・検証である．この目的を達成するため, 本稿では遅延を補償する形でアクションチャンキングを導入した手法を提案する．本手法は, VLM の推論遅延を観測遅延のあるマルコフ決定過程 [6] として明示的にモデル化する．その上で, VLM に一度の推論で未来の複数ステップ分の行動系列（アクションチャンク）を予測させ, VLM が次のチャンクを計算している間も, エージェントが行動を継続できる仕組みを構築する．実験の結果，シミュレーション環境において，推論遅延の存在下でも学習による報酬の向上が確認された．本研究は，適切な遅延モデル化とチャンキングにより，VLM がリアルタイム制御における実用的な意思決定主体として学習可能であることを示すものである．

2　VLM エージェントの強化学習

2.1　MDP による定式化と PPO

本研究では，VLM による環境制御をマルコフ決定過程 (Markov Decision Process, MDP) として定式化する．MDP は，状態集合 $\mathcal{S}$，行動集合 $\mathcal{A}$，遷移確率 P，報酬関数 $\mathcal{R}$，割引率 γ の組 $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$ で定義される．エージェントは時刻 t に状態 s_t を観測し方策 $\pi_\theta(a_t|s_t)$ に従い行動 a_t を選択，環境は次状態 s_{t+1} へ遷移し報酬 r_t を返す．目的は割引累積報酬 $J(\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$ を最大化するパラメータ θ の学習である．

本稿では，代表的な方策勾配法である Proximal Policy Optimization (PPO) [7] を採用する．PPO は，更新前後の方策の乖離を一定範囲に抑えることで学習の安定化を図る手法であり，以下のクリップされた目的関数を最大化する．

$$J^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(\rho_t(\theta) A_t,\ \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon) A_t \right) \right]$$

ここで，$\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$ は確率比，A_t はアドバン

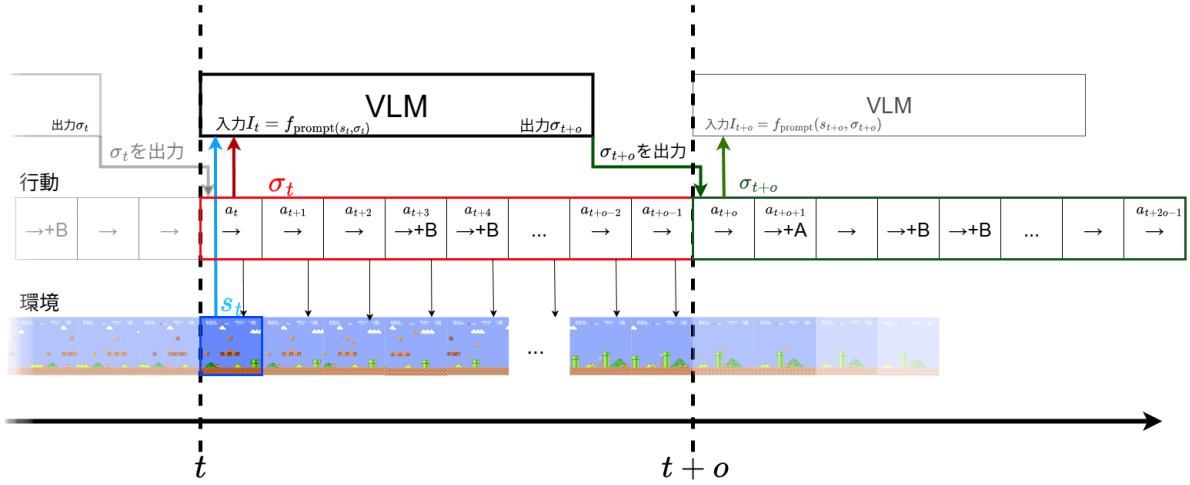

図 1　提案手法の推論のタイムチャート．VLM の推論時間よりも長い行動系列を決定することで，リアルタイムな制御を可能としている．

テージ関数，ϵ はハイパーパラメータである．アドバンテージ関数の推定には，Generalized Advantage Estimation (GAE) [8] が広く用いられ，本研究でも採用する．

2.2　VLM エージェントの学習手法

VLM を意思決定エージェントとして利用する研究は，近年では環境との相互作用を通じて学習する強化学習アプローチが提案されている．

Zhai ら [9] の RL4VLM では，VLM に対して Chain-of-Thought (CoT) [10] を用い，推論過程（思考）と，実行すべき行動の両方をテキストとして出力させ，その結果得られる報酬を用いてモデルを更新する．思考を出力させることで，タスク遂行能力が向上することが示されている．しかし，自己回帰生成では思考トークン数が行動トークン数より多いため，全体の対数尤度が思考部分に支配され，行動選択確率の適切な評価・更新が困難となる問題がある．RL4VLM では思考と行動の尤度を重み付け係数 λ で混合し，行動の選択確率として PPO を用いているが，この係数の調整は環境依存性が高く，学習が不安定になりやすい．また，PPO のための価値関数の計算に VLM の最後の隠れ状態を入力とする小規模な多層パーセプトロン（Multi Layer Perceptron, MLP）を用いている．価値ネットワークの更新に伴って VLM 本体にも勾配を伝播させている．

この課題に対処するため，Bredis ら [11] は Vision-Language Decoupled Actor-Critic (VL-DAC) を提案した．VL-DAC の最大の特徴は，方策と価値の学習を異なる時間スケールおよび勾配フローで分離する点にある．具体的には，方策の更新は思考の部分を除いて，行動部分のみで行い，価値関数の学習による勾配が VLM 本体に逆伝播しないよう，勾配停止 (stop-gradient) を適用する．これにより，価値学習の不安定性が言語モデルとしての表現力を損なうことを防ぎ，安定した学習を実現している．本研究では，この VL-DAC の安定した学習構造を踏襲する．

3　提案手法

3.1　アクションチャンキング

VLM 方策 π_θ は，単一の行動ではなく，長さ o の行動系列 $\sigma_t = (a_t, a_{t+1}, \ldots, a_{t+o-1})$ を出力する．これにより，VLM が次の推論を行っている間（$k \leq o$ なる k ステップ）も，エージェントはバッファにある行動を実行し続けることができる．

3.2　観測遅延と状態拡張

エージェントが時刻 $t+o$ からの行動系列 σ_{t+o} を生成するためには，推論時間を考慮して時刻 t に推論を開始する必要がある．このとき利用可能な最新の観測は s_t である．しかし，単に s_t を入力するだけでは，時刻 $t+o$ 時点の状況と乖離が生じる．そこで，観測遅延のある MDP への対処法として知られる状態拡張 [6] を適用する．拡張状態を，

$$(s_t, a_t, \ldots, a_{t+o-1}) = (s_t, \sigma_t)$$

として定義する．VLM はこれをプロンプト化した I_t 受け取り，行動履歴の結果として生じる未来の状態を予測しながら，次の系列 σ_{t+o} を生成する（図 1）．

3.3 推論プロセスとアルゴリズム

提案手法の推論プロセスは以下の並列処理で行われる（図 1）．

1. **行動の実行**: 時刻 t から $t+o-1$ の間，バッファに格納された行動系列 σ_t を環境で順次実行する．
2. **次期行動の生成**: 並行して，VLM は入力 I_t に基づき，次の o ステップ分の行動系列 σ_{t+o} の生成を開始する．この推論には k ステップを要する．
3. **バッファ更新**: 推論が完了すると，生成された σ_{t+o} がバッファに追加され，次のサイクルの実行に備える．

この一連の学習プロセスをアルゴリズム 1 に示す．本アルゴリズムでは，環境とのインタラクションを行うループと，収集した経験を用いてネットワークを更新するステップが含まれる．データバッファ $\mathcal{D}$ には，拡張状態 I_{t-o}，生成されたチャンク σ_t，および実際に環境で得られたチャンク報酬 R_{chunk} が格納される．

3.4 ネットワークの更新

学習には PPO を用いるが，チャンク単位での処理に適応させる．

3.4.1 方策の更新

チャンク内の各原始行動 a_i レベルで以下の目的関数を最大化する．

$$J^{\text{POLICY}}(\theta) = \mathbb{E}\left[\min\left(\rho_t(\theta)A_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$$

確率比 $\rho_t(\theta)$ は以下のように定義される．

$$\rho_t(\theta) = \sum_{i=t}^{t+o-1} \frac{\pi_\theta(a_i|I_{t-o}, u_{<a_i})}{\pi_{\theta_{\text{old}}}(a_i|I_{t-o}, u_{<a_i})}$$

ここで $u_{<a_i}$ は，VLM がそれまでに出力した全トークン（思考テキストおよび先行する行動トークン）の系列を表す．

3.4.2 価値関数の学習

価値関数 $V_\phi(I_t)$ は拡張状態に対する価値を推定する．アドバンテージの計算には，チャンク全体を 1 ステップとみなした GAE を用いる．

$$\delta_{t-o}^{(o)} = \sum_{j=0}^{o-1} \gamma^j r_{t-o+j} + \gamma^o V_\phi(I_t) - V_\phi(I_{t-o})$$

アルゴリズム 1

入力: Chunk size o, VLM Policy π_θ, Value V_ϕ
入力: Initial State s_0, Initial buffer σ_0

Initialize experience replay buffer $\mathcal{D} \leftarrow \emptyset$
$t \leftarrow 0$
while Training not converged **do**
  // 1. Inference Phase (at time t)
  Construct Augmented State:
    $I_t \leftarrow (s_t, \sigma_t)$
  Generate next chunk (taking k steps):
    $\sigma_{t+o} \sim \pi_\theta(\cdot \mid I_t)$
  // 2. Execution Phase (time $t \to t+o$)
  $R_{\text{chunk}} \leftarrow 0$
  for $j = 0$ **to** $o-1$ **do**
    $r_{t+j}, s'_{t+j} \leftarrow \text{Env.step}(\sigma_t[j])$
    $R_{\text{chunk}} \leftarrow R_{\text{chunk}} + \gamma^j r_{t+j}$
  end for
  $s_{t+o} \leftarrow s'_{t+o-1}$
  // 3. Store Data
  Store transition $(I_{t-o}, \sigma_t, R_{\text{chunk}}, I_t)$ in $\mathcal{D}$
  // 4. Update Variables
  $\sigma_t \leftarrow \sigma_{t+o}$ ▷ Update buffer
  $t \leftarrow t+o$
  if $\mathcal{D}$ is ready for update **then**
    Calculate Advantages $\hat{A}$ using Chunk-GAE
    Update θ via PPO Objective
    Update ϕ via Value Loss
    $\mathcal{D} \leftarrow \emptyset$
  end if
end while

$$\hat{A}_{t-o} = \delta_{t-o}^{(o)} + (\gamma\lambda)^o \hat{A}_t$$

この $\hat{A}_{t-o}$ を，チャンク内の全ての行動に対する共通のアドバンテージとして用いる．

4 実験

4.1 実験設定

提案手法の有効性を検証するため，Gymnasium [12] および gym-super-mario-bros [13] を用いた．

4.1.1 環境の詳細

gym-super-mario-bros ステージ 1-1 を使用．報酬は「右への移動距離 + 到達ボーナス - 時間経過ペナルティ」である．VLM が出力可能な行動は，何

もしない, 右, 右+A, 右+B, 右+A+B の 5 種類に制限した．

Mountain Car　カートを左右に振って勢いをつけて山を登るタスク．報酬は，通常ゴール以外-1 だが，ここでは学習を促進するため「カートの位置エネルギーの増分」に基づくシェイピング報酬を導入した．行動は操作はカートを右あるいは左に力を加えるか，何もしないの 3 種類である．

Break Out　ブロック崩し．ボールを落とさずに打ち返す．報酬はブロック破壊時のスコア．行動は，バーを右，左に移動させるか何もしないかの 3 種類である．

4.1.2　モデル設定

ベースモデルには Qwen2.5-VL-7B-Instruct [2] を使用し，全線形層に LoRA [14] を適用した．チャンクサイズ o は 16 または 32 とした．詳細なハイパーパラメータは A を参照されたい．

4.2　実験結果と考察

4.2.1　学習曲線と報酬の推移

各環境の学習結果を図 2-4 に示す．gym-super-mario-bros では，$o = 16, 32$ の双方で学習に伴う報酬の向上が確認された．特に $o = 16$（図 2：青）では，更新 32 回付近でピークに達した．$o = 32$（図 2：橙）でも学習は進んだが，$o = 16$ と比較して学習による報酬の最大値は劣っている．Mountain Car（図 3）においても，学習初期は底に留まっていたが，更新が進むにつれて勢いをつける動作を学習し，ゴールへの到達に成功した．

一方，Break Out（図 4）では，60 回の更新を行ってもベースラインからの有意な向上は見られなかった．

4.2.2　考察

マリオ等の成功は，提案手法が VLM の推論遅延を効果的に補償していることを示唆する．VLM はバッファ内の「これから実行される行動」を考慮し，その結果マリオがどの位置にいるか，敵がどこにいるかを予測した上で，次の行動（ジャンプなど）を生成できていた．

対して Break Out で学習が停滞した要因は，タスクの「疎な報酬」と「精密動作の要求」にあると考えられる．ボールを打ち返すには数ピクセル単位の位置合わせと正確なタイミングが必要だが，チャンク単位で同一の方策更新を行う本手法では，ミスの瞬間の特定の行動に対する負のフィードバック（クレジットアサインメント）が希薄になりやすかったと推察される．

5　おわりに

本研究では，VLM をリアルタイムアクションゲームのエージェントとして適用するための「遅延補償型アクションチャンキング」手法を提案した．実験の結果，リアルタイム性のある環境において，VLM が自身の遅延を考慮した行動計画を学習可能であることを示した．今後の課題として，チャンク内の個々の行動に対するより詳細な価値評価手法の確立や，精密動作を要するタスクへの適応能力の向上が挙げられる．

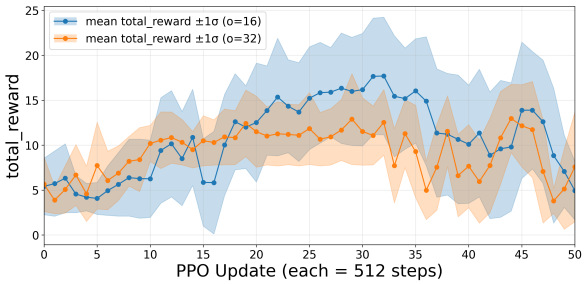

図 2　gym-super-mario-bros ($o = 16$：青，$o = 32$：橙) の学習推移．

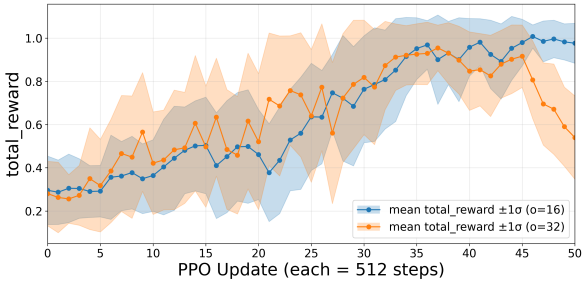

図 3　Mountain Car ($o = 16$：青，$o = 32$：橙) の学習推移．

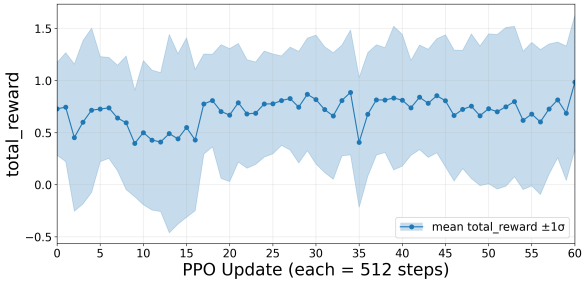

図 4　Break Out ($o = 16$) の学習推移．

参考文献

[1] Zongxia Li, Xiyang Wu, Hongyang Du, Fuxiao Liu, Huy Nghiem, and Guangyao Shi. A survey of state of the art large vision language models: Alignment, benchmark, evaluations and challenges. **arXiv preprint arXiv:2501.02189**, 2025.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. **arXiv preprint arXiv:2502.13923**, 2025.

[3] Xinyu Wang, Bohan Zhuang, and Qi Wu. Are large vision language models good game players? In **The Thirteenth International Conference on Learning Representations**, 2025.

[4] Peng Chen, Pi Bu, Jun Song, Yuan Gao, and Bo Zheng. Can vlms play action role-playing games? take black myth wukong as a study case. In **NeurIPS 2024 Workshop on Open-World Agents**, 2024.

[5] Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. Physically grounded vision-language models for robotic manipulation. In **2024 IEEE International Conference on Robotics and Automation (ICRA)**, pp. 12462–12469. IEEE, 2024.

[6] Sascha E. Engelbrecht Konstantinos V. Katsikopoulos. Markov decision processes with delays and asynchronous cost collection. **IEEE transactions on automatic control**, Vol. 48, No. 4, pp. 568–574, 2003.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. **arXiv preprint arXiv:1707.06347**, 2017.

[8] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. **arXiv preprint arXiv:1506.02438**, 2015.

[9] Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. In **Proceedings of the 38th International Conference on Neural Information Processing Systems**, pp. 110935–110971, 2024.

[10] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. **Advances in neural information processing systems**, Vol. 35, pp. 24824–24837, 2022.

[11] George Bredis, Stanislav Dereka, Viacheslav Sinii, Ruslan Rakhimov, and Daniil Gavrilov. Enhancing vision-language model training with reinforcement learning in synthetic worlds for real-world success. **arXiv preprint arXiv:2508.04280**, 2025.

[12] Mark Towers, Ariel Kwiatkowski, Jordan K Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. **CoRR**, 2024.

[13] Christian Kauten. gym-super-mario-bros: An openai gym environment for super mario bros. & the lost levels. https://github.com/Kautenja/gym-super-mario-bros, 2018.

[14] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In **International Conference on Learning Representations**, 2022.

A 実験の設定

A.1 実験のハイパーパラメータ

- **LoRA Target Modules**: qkv, proj, q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj, lm_head

表 1 実験パラメータ詳細

カテゴリ	パラメータ	設定値
Train	Scheduler	Cosine
	Decay Steps	100
	Initial Learning Rate	1.0×10^{-5}
	Final Learning Rate	1.0×10^{-9}
PPO	Rollout Steps	512
	Discount Factor (γ)	0.999
	GAE Lambda (λ)	0.999
	Clip Coefficient (ϵ)	0.2
	Entropy Coefficient	0.05
	Value Function Coefficient	0.15
	Max Grad Norm	0.5
	PPO Epochs	2
	Value Epochs	1
	Mini Batch Size	1
	Gradient Accumulation	128
Generation	Max New Tokens	256
	Temperature	0.2
	Top-p	0.85
	Top-k	40
	Chunk Size (o)	16 or 32
LoRA	Rank (r)	128
	Alpha (α)	256
	Dropout	0.05

A.2 環境設定の詳細

各環境の設定を表 2 に示す．「スキップ」はフレームスキップ数を指し，例えばスキップ 4 の場合，VLM が 1 つの行動を選択すると，環境内ではその行動が 4 フレーム連続して入力される．「報酬スケール」は，環境から得られる生の報酬に乗じる係数であり，学習の安定化のために調整されている．

表 2 環境ごとの設定

環境名	スキップ	報酬スケール
gym-super-mario-bros	4	1/16
Mountain Car	1	1.0
Break Out	4	1.0

B 入出力例

gym-super-mario-bros における System Prompt

```
You are an expert Super Mario Bros.(Family Computer game)
      speedrunner AI. Your goal is to output a sequence
     of {n} optimal actions based on the provided game
     screen and current action buffer.
```

gym-super-mario-bros における User Prompt

```
Analyze the current game screen and determine the next 16
      actions for Mario.

CURRENT SITUATION:
- Current game screen is shown in the image
- Action buffer currently contains: [3, 2, 2, 3, 0, 1, 2,
      3, 0, 1, 2, 3, 0, 1, 2, 3]
- These buffered actions will be executed first before
     your new actions


AVAILABLE ACTIONS:
0: "NOOP" (no operation)
1: "right" (move right)
2: "right+jump" (move right and jump)
3: "right+dash" (move right and dash)
4: "right+jump+dash" (move right, jump and dash)

OUTPUT FORMAT:
Respond with JSON containing your thoughts and a sequence
      of 16 or
     32actions:
{{{{"thoughts":"Your reasoning about the game situation
     and action sequence strategy","actions":(list of 16
     action IDs from 0 to 4)}}}}

CRITICAL RULES:
- Your entire output must be ONLY a single, valid JSON
     object - nothing else.
- Do NOT include any text before or after the JSON object
     .
- Do NOT use markdown code blocks (```json or ```).
- The "thoughts" field is mandatory and must explain your
      reasoning for the action.
- The "actions" field must be a JSON array of EXACTLY 16
     integers
- Each integer in the array must be 0, 1, 2, 3, or 4
- Include your strategic thinking in the "thoughts" field
- Consider the action buffer that will be executed before
      your actions which take about two seconds to
     execute
    - Prefer sequences that use at least two different
     actions. avoid long runs of identical actions.
    - If repetition is required, briefly explain why in
     the "thoughts" field.

STRATEGY TIPS:
- Plan ahead: Your actions will be executed after the
     current buffer
- Enemies/obstacles ahead: Use action 2 (right+jump) or 4
      (right+jump+dash)
- Mario can jump more higher if you select action 2 or 4
     when Mario is in the air
- Jump when facing obstacles (green pipes or bricks),
     enemies, pits
- Clear path: Use action 3 (right+dash) for speed
- Consider Mario's momentum and trajectory when planning
     the sequence
```

gym-super-mario-bros における出力例

```
{\"thoughts\":\"Mario needs to jump over the pipe. The
     action buffer has 2 right+jump actions, so I need to
      use them to make him jump over the pipe.\",\"
     actions\":[2, 3, 2, 2, 2, 2, 2, 2, 0, 1, 2, 3, 0, 1,
      2, 3]}
```