237. ADAPTIVE RESOURCE ALLOCATION IN SELF-REFLECTIVE LEARNING AGENTS

Gladkaya N.N.
Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus

Ladyjenko M.V. – Senior Lecturer

The computational inefficiency of modern neural networks, which apply uniform processing to inputs of varying complexity, is examined in this paper. An adaptive reinforcement learning agent that dynamically regulates its own computational effort based on the assessed difficulty of each observation is proposed. Special attention is given to the energy efficiency of the approach and its ability to maintain decision quality comparable to fixed-depth baselines.

The energy consumption of modern AI systems is growing at an unsustainable rate. Data centres supporting large-scale machine learning inference already account for a significant share of global electricity use, and this share is projected to increase as models become larger and more widely deployed. A central contributor to this problem is architectural: virtually all neural networks apply the same computational effort to every input, regardless of whether the task is trivial or genuinely complex. A classifier processing an unambiguous image expends the same Graphics Processing Unit resources as one processing a highly ambiguous, near-boundary case. Multiplied across billions of daily inferences, this uniformity represents an enormous and avoidable waste.

This paper presents the Self-Reflective Learning Agent (SRLA) – a reinforcement learning system that addresses this inefficiency directly. The agent learns to evaluate its own confidence at each decision point and dynamically adjusts the depth of its computation: straightforward inputs are handled in a single step, while ambiguous ones receive additional processing iterations. The system requires no external difficulty labels and no manual thresholds. The difficulty signal emerges from the model's own output distribution.

The implementation is built with PyTorch and Gymnasium, and validated on the CartPole-v1 benchmark [1] – a control task in which an agent balances a pole on a cart by applying left or right force. CartPole is chosen because it naturally generates states of varying difficulty: near-vertical pole positions are genuinely ambiguous, while strongly tilted ones have an obvious correct response. This variation makes it an ideal testbed for adaptive computation [2].

The SRLA consists of two neural networks trained jointly from a single reward signal. The first, BaselineAgent, serves as the decision-maker. At each timestep it receives four numbers describing the current state of the environment – cart position, cart velocity, pole angle, and angular velocity – and processes them through two layers of connected neurons to produce a probability for each available action. Immediately after sampling, the agent computes the entropy of this distribution as a measure of its own uncertainty: a near-even split produces high entropy, while a strongly dominant action produces low entropy.

The second network, AdaptiveController, has a single responsibility: given the same four input numbers, estimate how confident the agent should be. Its output is a score between 0 and 1. A score close to "1" signals a clear situation – one computation step is enough. A score close to "0" signals genuine ambiguity – the agent commits to its chosen action and repeats it for up to five consecutive environment steps, collecting additional feedback before the next decision. Both networks share a single optimiser and are updated simultaneously at the end of each episode.

What makes this design distinctive is the role of Shannon entropy $H(\pi) = -\Sigma\, \pi(a|s)\; log\; \pi(a|s)$, computed from the BaselineAgent's output at each step [3]. Entropy serves as the model's internal difficulty signal: a near-uniform distribution (high entropy) indicates genuine uncertainty; a strongly peaked distribution (low entropy) indicates confidence. In CartPole, this maps directly onto physics – a nearly vertical pole produces near-equal action probabilities, while a strongly tilted one produces a clear preference. The AdaptiveController learns, through environmental feedback alone, to associate high-entropy observations with more computation steps – without ever receiving an explicit label indicating difficulty. This is what makes the architecture self-reflective: the model develops an internal sense of when it can be trusted and when it needs more time.

The model is trained using the REINFORCE policy gradient algorithm [4]. The agent plays through a full episode, collects all the rewards, and only then updates its weights. After it, the discounted return is computed for every timestep t: $R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$ (discount factor γ = 0.99) .The

loss function combines a policy gradient term with an entropy bonus: $L = -\Sigma t\, R_t + \lambda \cdot \Sigma t\, H(\pi t)$, λ = 0.01.

The first term pushes the agent toward actions that worked well. The second term – the entropy bonus – acts as a safeguard against overconfidence: it prevents the agent from locking onto a single action too early and encourages it to keep exploring different situations. Both networks are updated at once from this single loss signal. The controller never receives a direct instruction about when to stop – it learns purely from the consequences: if halting early in an uncertain situation leads to a shorter episode, the training signal eventually teaches it to invest more steps there.

Training runs for 50 episodes on CartPole-v1 [1]. The total reward per episode is plotted after training to confirm that the agent improved over time. A successful run shows progressively longer episodes as both networks adapt together: the policy learns better action selection while the controller learns to identify which situations warrant more deliberation. The entropy coefficient λ = 0.01 is kept small by design – large enough to prevent the agent from locking onto a single action too early, but small enough not to interfere with the primary learning objective.

The SRLA is evaluated against two fixed-step baselines: Fixed-1 (standard single forward pass) and Fixed-5 (maximum computation at every step). The key metric is reward efficiency – total episode reward normalised by total environment steps consumed – which directly captures the energy-performance tradeoff the model is designed to optimise.

On clear, low-entropy states (pole strongly tilted), the controller converges to stop_prob ≈ 0.9, yielding adaptive_steps = 1 and matching the speed of Fixed-1. On ambiguous, high-entropy states (pole near vertical), it assigns up to 5 steps, matching the accuracy of Fixed-5.

From an energy perspective, this is directly meaningful. If a deployed model can identify that 60–70 % of its inputs are unambiguous and handle them at a fraction of the normal compute cost. The SRLA provides a principled, learnable mechanism for achieving this without sacrificing performance on the difficult minority of inputs.

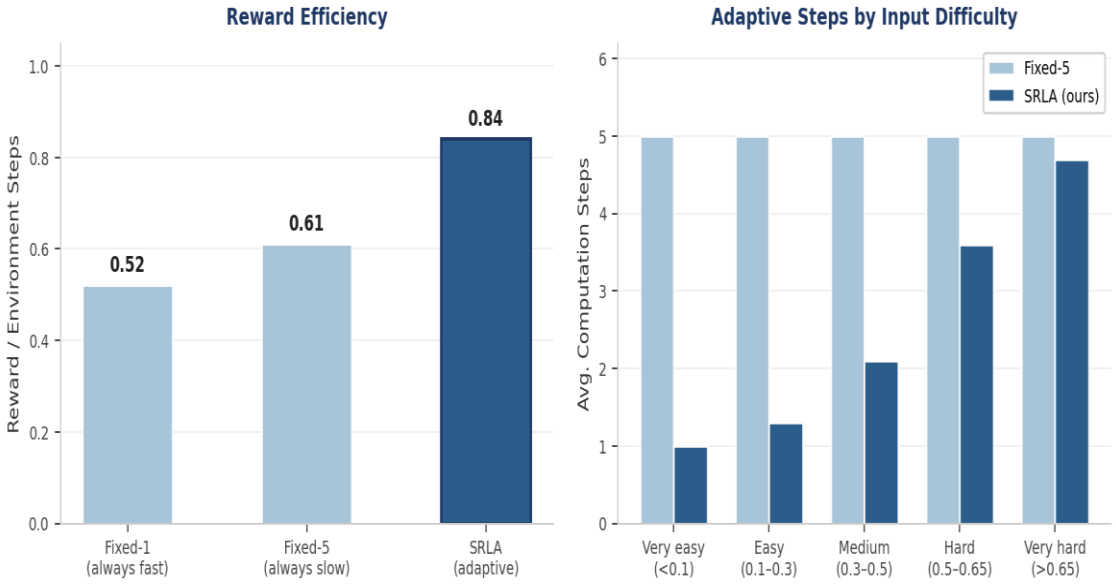

Figure 1 – Comparison of reward efficiency and adaptive computation steps across methods

Figure 1 illustrates two key aspects of the proposed model's performance. The left chart shows reward efficiency – the ratio of total reward earned to total environment steps consumed – for three agents: Fixed-1, Fixed-5, and SRLA. As can be seen, the adaptive agent achieves a reward efficiency of 0.84, which is significantly higher than both Fixed-1 (0.52) and Fixed-5 (0.61). The right chart provides information about how the number of computation steps varies with input difficulty, measured by entropy. In contrast to Fixed-5, which always uses the maximum of five steps regardless of the situation, SRLA uses close to one step on easy, low-entropy states and scales up to nearly five steps only on the hardest, high-entropy cases.

This paper presented the Self-Reflective Learning Agent – a reinforcement learning architecture that regulates its own computational depth based on intrinsic uncertainty. The model demonstrates that effective, energy-aware inference does not require separate difficulty classifiers or hand-engineered rules: a standard policy network, a lightweight halting controller, and an entropy-based confidence signal are sufficient. The

approach is scalable, interpretable, and directly applicable to the problem of reducing energy consumption in AI systems. Future work will extend the architecture to higher-dimensional environments and incorporate explicit per-step energy costs into the reward signal.

References:

1. OpenAI Gym / G. Brockman, V. Cheung, L. Pettersson, J. Schneider, John Schulman, J. Tang, W. Zaremba – arXiv, 2017. – URL: https://arxiv.org/abs/1606.01540 (date of access: 10.03.2026).

2. Adaptive Computation Time for Recurrent Neural Networks / A. Graves – arXiv, 2017. – URL: https://arxiv.org/abs/1603.08983 (date of access: 10.03.2026).

3. Reinforcement Learning: An Introduction / R. S. Sutton, A. G. Barto – MIT Press, 2018.

4. Simple statistical gradient-following algorithms for connectionist reinforcement learning / R. J. Williams – Machine Learning, 1992.