XMEN: X-Ray Multi-Input Engagement Navigators: A Recurrent Dueling DQN for Minecraft Zombie Combat

Bing-Hong Lin [1] Li-Heng Yang [1] Chih-Chuan Huang [1] Wei-Ning Huang [1] Che-Ning Chang [1] Chia-Chin Hsieh [1]

Abstract

Deep reinforcement learning (DRL) shows promise in complex video game environments, yet applying it to 3D first-person shooters like Minecraft's Zombies mini-game remains difficult due to partial observability, high-dimensional vision, and the need for precise combat control.

We propose XMEN, an Multi-modal Enhanced Network that fuses CNN visual features with structured "X-Ray" inputs (zombie position and distance). An LSTM-based DRQN models temporal dependencies, and training follows the Dueling Double DQN framework for stability. Built on Project Malmo, XMEN achieves substantially higher survivability and combat performance than single-modal or non-recurrent baselines, showing that combining memory with precise auxiliary sensing significantly improves first-person control.

1. Introduction

In the popular Minecraft server Hypixel, there is a mini-game called "Zombies." The gameplay consists of surviving against consecutive waves of zombies that spawn at fixed time intervals. As time progresses, the zombies become increasingly stronger and more numerous. Players must continuously fight them in order to prevent the city from being overrun, upgrading their weapons and mastering combat techniques to survive as long as possible.

For our final project, we aim to train a reinforcement learning agent to play a simplified version of this game. In our version, the player can no longer upgrade weapons; instead, survival relies purely on learning and mastering optimal combat movements.

1.1. Related Work

Deep Reinforcement Learning (DRL) for video game playing was significantly advanced by the Deep Q-Network (DQN), which demonstrated superhuman performance on Atari games by learning directly from pixels using experience replay and target networks (Mnih et al., 2013). To address the overestimation bias inherent in the original DQN, Double DQN was introduced to decouple action selection from evaluation (Hado van Hasselt, 2016). Furthermore, the Dueling Network architecture was proposed to improve learning efficiency by separating the representation of state values and action advantages (Wang et al., 2016). These advancements spurred broad research into DRL applications across various game genres, including Minecraft, as well as identifying key challenges such as sparse rewards and partial observability (Justesen et al., 2019). To address the limitations of DQN in partially observable environments (POMDPs), the Deep Recurrent Q-Network (DRQN) was introduced, which replaces frame-stacking with a recurrent (LSTM) layer to integrate information over time (Hausknecht & Stone, 2015). However, when tested on simple POMDP tasks within Minecraft, studies show that the standard DQN can still outperform the more complex DRQN, suggesting the recurrent architecture is not always the superior choice and its necessity depends on the task's complexity (Romac & Béraud, 2019).

2. Background

2.1. Project Malmo & Minecraft Environment

Minecraft presents significant challenges for RL agents due to its 3D, partially observable nature, unlike simple 2D environments. We utilize Project Malmo (Johnson et al., 2016), a platform enabling external Python scripts to interact with the game via a standard RL loop: the agent receives visual and auxiliary observations (e.g., health, position), executes commands, and receives reward signals. Crucially, Malmo allows pre-

cise environment control via XML definitions. In this project, we leverage this feature to recreate the "Zombies" mini-game mechanics for training combat agents.

2.2. Double Dueling DQN (D3QN)

We employ the D3QN algorithm to enhance stability and data efficiency by integrating Double Q-learning and Dueling architectures.

Double DQN (Hado van Hasselt, 2016) addresses the overestimation bias of standard DQN by decoupling action selection from evaluation. The Main Network (θ) selects the action, while the Target Network (θ^-) estimates its value:

$$Y_t^{Double} = R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \theta_t); \theta_t^-) \tag{1}$$

Furthermore, the Dueling architecture (Wang et al., 2016) improves efficiency by separating the network into State Value $V(s)$ and Advantage $A(s, a)$ streams. These are aggregated with a mean-subtraction normalization to ensure stability:

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a') \right) \tag{2}$$

Together, D3QN mitigates bias and accelerates the learning of state representations in complex 3D environments.

2.3. Long Short-Term Memory (LSTM)

Standard DQN assumes full observability (MDP), but Minecraft is a Partially Observable MDP (POMDP) where single frames lack temporal context (e.g., enemy velocity). To address this, we adopt the Deep Recurrent Q-Network (DRQN) architecture (Hausknecht & Stone, 2015) by integrating Long Short-Term Memory (LSTM) units (Hochreiter & Schmidhuber, 1997). Unlike standard RNNs, LSTMs employ gating mechanisms (input, output, forget) to mitigate vanishing gradients. This allows the agent to maintain an internal memory state, aggregating historical observations to infer motion and track out-of-view threats effectively.

2.4. Multi-Modal Fusion

In 3D environments, CNNs excel at extracting spatial topology from pixels but struggle with precise metric estimation (e.g., exact distance), whereas symbolic data (coordinates) offers precision but lacks context. To leverage both, we employ a multi-modal fusion architecture consisting of two streams: a Visual Stream using CNNs to process image data for obstacle avoidance, and a Vector Stream using MLPs to encode structured "X-Ray" data (e.g., entity positions). By concatenating these representations, the agent combines visual navigation with mathematical aiming precision, addressing partial observability more effectively than uni-modal approaches.

3. The XMEN Architecture

Building upon the theoretical foundations of DRQN and Dueling Networks described in Section 2, we propose the XMEN (X-ray Multi-modal Environment Network) architecture. While standard architectures typically process a single modality, XMEN is designed to master the partial observability of Minecraft by fusing high-dimensional visual data with structured sensor information.

As illustrated in Figure 1, our model extends the standard R2D2 architecture by introducing a multi-modal embedding layer before the recurrent memory, and applying the Dueling split strictly after temporal aggregation.

3.1. Multi-modal Input Representation

To account for the non-Markovian nature of the environment (e.g., estimating zombie velocity and attack cooldowns), the agent operates on a sequence of observations over time steps $t \in [1, T]$. The architecture processes two distinct input streams:

- Visual Stream (I_t): A sequence of 4 consecutive RGB frames ($T = 4$), preprocessed to 84×112. Unlike standard DQNs that stack frames into channels, we maintain the temporal dimension, resulting in an input shape of $(B, T, 84, 112, 3)$.
- X-Ray Sensor Stream (S_t): A sequence of 4 structured vectors containing agent-centric coordinates (x_{local}, z_{local}) and Euclidean distance to targets. This explicitly encodes spatial relationships that may be occluded in the visual stream. Input shape: $(B, T, 3)$.

3.2. Spatio-Temporal Feature Fusion

The feature extraction process utilizes a Time-Distributed strategy to preserve temporal boundaries before memory processing:

1. Visual Encoding: The visual stream is processed by a Time-Distributed CNN. This ensures that the same set of convolutional filters (32, 64, and 128 filters with ReLU and Max Pooling) is applied to each frame in the sequence independently.

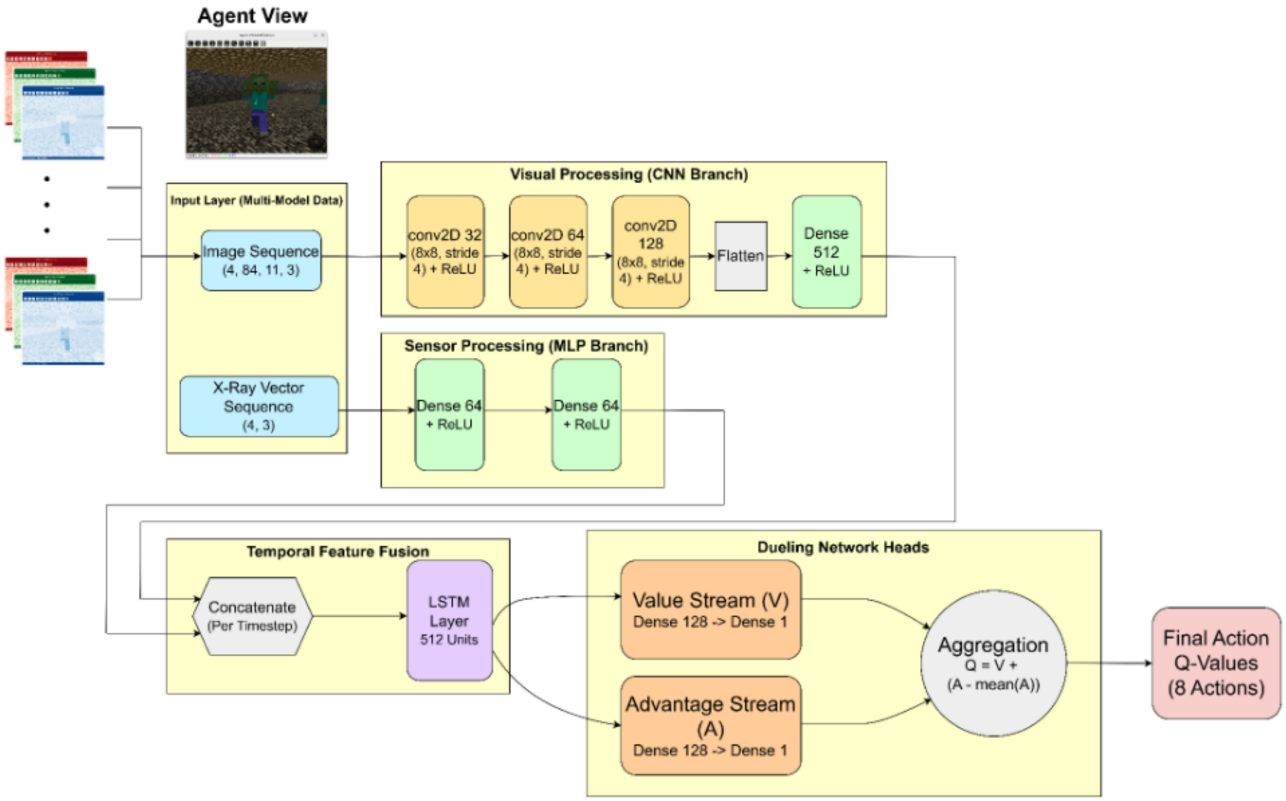

Figure 1. The XMEN Architecture. The model processes a sequence of 4 visual frames via a Time-Distributed CNN and X-ray sensor vectors via a generic MLP. These multi-modal features are fused and passed through an LSTM (512 units) to capture temporal dynamics. Finally, the network splits into Dueling Value and Advantage streams to compute the final Q-values for the 8 discrete actions.

2. Sensor Embedding: Simultaneously, the X-ray vectors are projected into a higher-dimensional space via a Time-Distributed MLP (two dense layers of 64 units).

3. Fusion & Memory: The flattened visual features ϕ_{vis} and sensor embeddings ϕ_{sensor} are concatenated at each time step. This fused sequence is then fed into a Long Short-Term Memory (LSTM) layer with 512 units.

Crucially, the LSTM serves as the central integration point, fusing distinct modalities across time into a single hidden state h_t that represents the agent's complete belief state.

3.3. Recurrent Dueling Head

Unlike standard Dueling Networks that split immediately after convolution, XMEN applies the Dueling architecture after the LSTM layer. This design ensures that both the Value and Advantage estimates are conditioned on the full history of the episode.

The network bifurcates the LSTM output h_t into two streams:

1. Value Stream $V(s)$: A dense layer (128 units) projecting to a scalar output.

2. Advantage Stream $A(s, a)$: A dense layer (128 units) projecting to $|\mathcal{A}| = 8$ action outputs.

These streams are aggregated using the mean-subtraction operator defined in Equation (2) (Section 2.2) to compute the final Q-values.

3.4. Implementation Details

The model is implemented in TensorFlow/Keras. We use the Double DQN learning objective with an Experience Replay buffer of capacity 5,000. To stabilize training for this deeper recurrent architecture, we use a lower learning rate of $lr = 0.00025$ (compared to standard DQN's 0.001) and the Adam optimizer. The

target network is updated via soft updates ($\tau = 0.005$).

4. Experiments

We evaluate the efficacy of the XMEN architecture within the Project Malmo environment. Our experiments are designed to test the hypothesis that fusing structured sensor data with visual memory, combined with tactical reward shaping, yields superior combat performance compared to standard visual-only reinforcement learning approaches.

4.1. Experimental Setup

Environment Details. The agent is instantiated in a closed arena (20×20 blocks) and subjected to consecutive waves of zombies. To simulate a hardcore survival scenario, natural health regeneration is disabled (‘/gamerule naturalRegeneration false‘).

Reward Shaping. A critical challenge in this task is the sparsity of the reward signal. The standard baseline implementation (Rishav, 2020) relies on a mix of terminal outcomes and a "Per Action" noise reward, which often leads to suboptimal looping behavior.

To accelerate convergence, we redesigned the reward function as detailed in Table 1. Key modifications include:

- Survival Priority: We quadrupled the penalty for death ($-100 \rightarrow -400$) to prioritize survival over suicidal aggression.
- Tactical Guidance: We introduced auxiliary rewards for Aiming (+5) and Kiting (+0.5), guiding the agent to keep the crosshair on the enemy while maintaining a safe distance (3–6 blocks).
- Noise Reduction: We removed the baseline's generic "Per Action Taken" reward (+0.05) to prevent the agent from accumulating rewards by simply vibrating or spinning without purpose.

Training Hyperparameters. All models are trained using the Adam optimizer with a learning rate of 0.00025. The exploration strategy follows a linearly decaying ϵ-greedy policy ($\epsilon : 1.0 \rightarrow 0.01$).

4.2. Baselines

We benchmark our proposed XMEN architecture against the implementation provided by rishavb123 (Rishav, 2020), which serves as the representative baseline for this task.

- Baseline (Visual DQN): A standard CNN-DQN that predicts Q-values directly from pixel inputs. It lacks explicit temporal memory (LSTM) and relies solely on visual cues.
- XMEN (Ours): Our Recurrent Dueling architecture which fuses visual data with X-ray sensor vectors and processes them through an LSTM.

Table 1. Reward Function Design: Comparison between the Baseline and Ours (XMEN). We introduce dense rewards to encourage specific combat tactics while removing noisy action rewards.

Event	Baseline	Ours
Core Events		
Death	-100	-400
Zombie Killed	+150	+100
Damage Dealt	+15	+30
Damage Taken	-4	-10
Survival (per tick)	-	+0.1
Per Action Taken	+0.05	-
Tactical Shaping		
Aiming	-	+5.0
Kiting Bonus	-	+0.5
Safe Attack	-	+10
Penalties		
Missed Opportunity	-	-2.0
Inactivity	-	-0.5
Wall Penalty	-	-5.0

4.3. Results and Analysis

Table 2 presents the quantitative results comparing the Baseline agent with our XMEN agent. We report both the Average performance (over evaluation episodes) and the Highest single-episode performance achieved.

Table 2. Performance comparison. XMEN demonstrates a massive improvement in combat efficiency, achieving nearly 68× more kills on average than the baseline.

	Average		Highest	
Model	Kills	Survival	Kills	Survival
Baseline	0.14	21.49s	2	36.41s
Ours	9.52	81.95s	22	166.77s

Quantitative Analysis. The results in Table 2 show a dramatic performance gap. The Baseline agent struggles significantly, averaging only 0.14 kills per episode with a survival time of roughly 21 seconds. This suggests the baseline often dies to the first zombie without mounting an effective defense.

In contrast, the XMEN agent achieves an average of 9.52 kills per episode—an improvement of over 6,700%. The average survival time also quadrupled to 81.95s. The peak performance record further highlights the stability of our model, with the agent capable of surviving for nearly 3 minutes and eliminating 22 zombies in a single run.

Behavioral Analysis. The "Per Action" reward in the baseline caused the agent to exhibit jittery, non-committal movements to farm scores. By removing this and adding the Aiming and Kiting rewards, XMEN learned to lock onto targets efficiently. The high kill count confirms that the X-ray sensor fusion successfully solved the problem of partial observability, allowing the agent to react to threats even when visual features were ambiguous.

5. Discussion

XMEN achieves a substantial performance leap over the baseline. We analyze the contributing factors, stability challenges, and future directions.

5.1. The Role of Multi-Modal Fusion

The baseline Visual DQN suffers from partial observability. XMEN fuses X-ray sensor data to bypass depth estimation issues. This fusion is synergistic: visual inputs provide environmental topology (preventing collisions), while sensor data offers target precision. The LSTM acts as a temporal bridge, integrating these sources into a coherent strategy.

5.2. Behavioral Impact of Reward Shaping

Our reward configuration evolved to address specific behavioral pathologies:

- "Coward" Problem: High death penalties caused purely evasive behavior. We added a range penalty to force engagement.
- "Kamikaze" Problem: High kill rewards caused "suicidal aggression." We balanced expected kill value with damage penalties ($E[\text{Kills}] \times R_{kill} \approx R_{death} + R_{dmg}$).
- "Aiming" Problem: To correct wasted attacks, we added Aiming (15°) and Wall penalties for fine-grained feedback.

This confirms that while architecture determines capacity, the reward landscape dictates the behavioral outcome.

5.3. Training Instability

Despite high peak performance, stability remains a challenge (Figure 2):

- Catastrophic Forgetting: Performance occasionally collapsed, likely due to replay buffer overfitting or aggressive updates.
- Initialization Sensitivity: Convergence depended on fortuitous early exploration, highlighting the "Cold Start" problem.

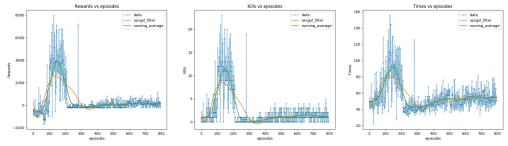

Figure 2. Training progression illustrating Catastrophic Forgetting. Metrics peak near ep. 150 before collapsing.

5.4. Limitations and Future Work

XMEN relies on privileged X-ray data. To address this and exploration challenges, we propose:

- Video PreTraining (VPT): Incorporating imitation learning (Baker et al., 2022) to mitigate "Cold Start".
- Gym-based Environments: Migrating to MineDojo (Fan et al., 2022) for standardized benchmarks.

Note: Reward clipping was tested but degraded tactical prioritization (see Appendix A).

6. Conclusion

We presented XMEN, a recurrent multi-modal agent that achieves a 68-fold increase in kills and 4-fold increase in survival in Minecraft combat.

Our analysis distinguishes the roles of architecture versus task design. We observed that the Architecture (LSTM + Fusion) governs the learning rate and temporal patterns, whereas the Reward Function determines the final outcome. Without iterative reward balancing—countering "cowardly" or "suicidal" tendencies—even sophisticated architectures fail. We conclude that solving complex 3D tasks requires a synergistic approach: robust architectures for efficiency, paired with domain-aware rewards for direction.

7. Contributions

B.-H. Lin (20%): code implementation, model training, and writing the repository documentation, including README files and architecture descriptions.

L.-H. Yang (16%): experimental results analysis, theoretical analysis, and providing revision suggestions.

C.-C. Huang (16%): experimental results analysis, theoretical analysis, and preparation of the final presentation.

W.-N. Huang (16%): experimental results analysis, theoretical analysis, and preparation of the final presentation.

C.-N. Chang (16%): code implementation, model training, and writing the final report.

C.-C. Hsieh (16%): code implementation, model training, and writing the final report.

References

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Maxwell, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In Advances in Neural Information Processing Systems, volume 35, pp. 24639–24654, 2022.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In Advances in Neural Information Processing Systems, volume 35, pp. 18343–18362, 2022.

Hado van Hasselt, Arthur Guez, D. S. Deep reinforcement learning with double q-learning. arXiv:1509.06461, 2016.

Hausknecht, M. and Stone, P. Deep recurrent q-learning for partially observable mdps. In AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15), 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. Neural computation, 1997.

Johnson, M., Hofmann, K., Hutton, T., and Bignell, D. The malmo platform for artificial intelligence experimentation. Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI), 2016.

Justesen, N., Bontrager, P., Togelius, J., and Risi, S. Deep learning for video game playing. arXiv preprint arXiv:1708.07902, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

Rishav, B. Minerl: Deep q learning for zombie combat in minecraft. https://github.com/rishavb123/MineRL, 2020. Accessed: 2025-01-01.

Romac, C. and Béraud, V. Deep recurrent q-learning vs deep q-learning on a simple partially observable markov decision process with minecraft. arXiv preprint arXiv:1903.04311, 2019.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. Dueling network architectures for deep reinforcement learning. Proceedings of the 33rd International Conference on Machine Learning (ICML), 2016.

A. Supplementary Experiments

In this appendix, we detail supplementary experiments investigating stabilization techniques, system dynamics, and the "Cold Start" problem.

A.1. Impact of Normalization and Reward Clipping

We tested if standard regularization (Reward Clipping $[-1, 1]$ and State Normalization) could mitigate training instability. However, as shown in Figure 3, these techniques negatively impacted performance by compressing the hierarchy of rewards.

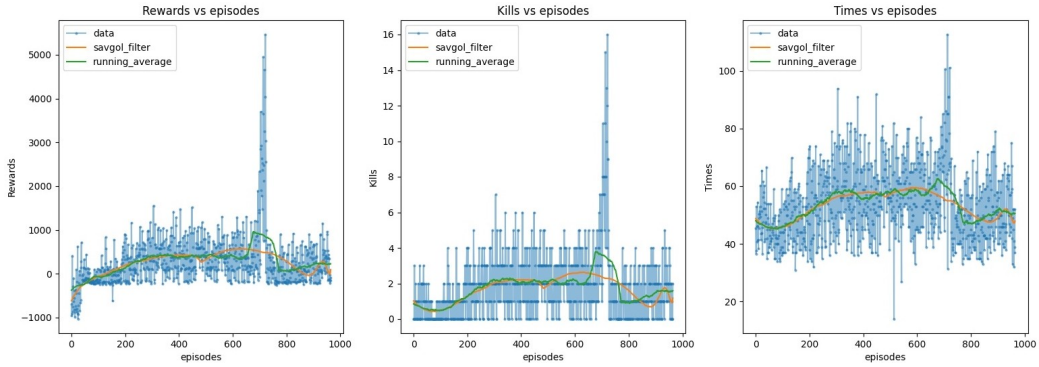

Figure 3. Performance comparison: Clipped models fail to capture the hierarchy of rewards (e.g., distinguishing Aiming from Killing).

A.2. Impact of Simulation Tick Rate

We investigated the trade-off between simulation speed and training stability by reducing the environment's Tick Rate (increasing 'msPerTick').

Hypothesis: We hypothesized that at high simulation speeds, the latency of the GPU forward pass (inference time) might exceed the tick window, causing actions to be executed on stale states (asynchronous execution). Slowing down the simulation ensures strict synchronization between observation, inference, and action execution.

Results: Figure 4 confirms that a slower tick rate yields significantly smoother convergence and higher stability compared to the baseline. While this increases the wall-clock training time, the improvement in data quality and policy stability makes it a worthwhile trade-off for complex 3D environments.

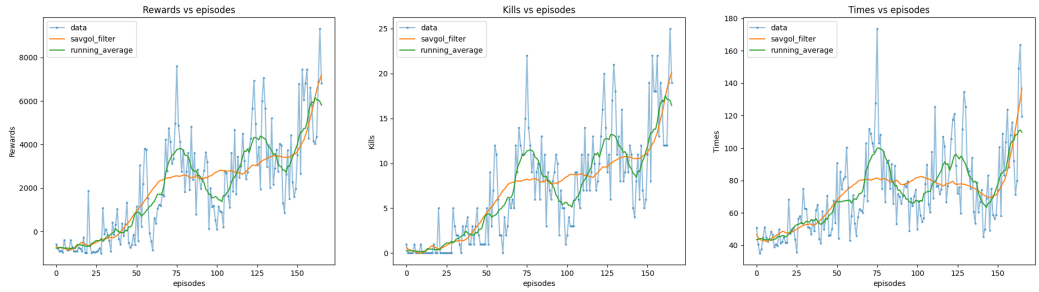

Figure 4. Effect of Slower Tick Rate. Reducing the simulation speed results in highly stable learning curves (Green line), likely due to better synchronization between GPU inference and game physics. Note: While the results are promising, we acknowledge that due to limited computational resources, we could not perform large-scale averaging. Thus, we cannot entirely rule out the possibility that this specific run also benefited from a fortuitous initialization.

A.3. Impaction of Initialization

This section illustrates the "Cold Start" problem discussed in Section 5, where convergence depends heavily on early exploration outcomes.

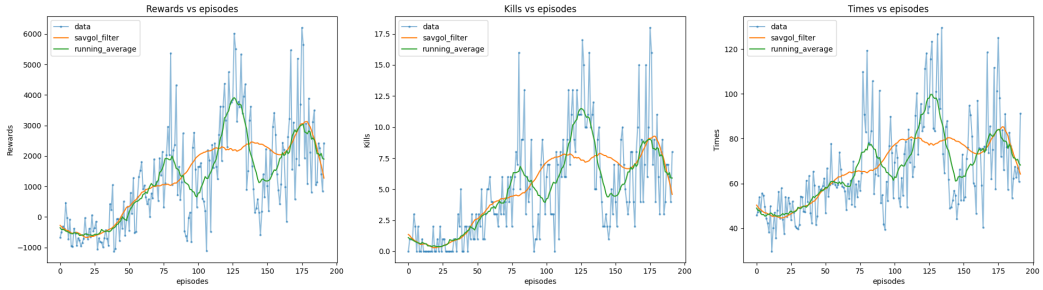

Figure 5. Case A: Unsuccessful Initialization. The agent's initial random actions failed to secure early kills, resulting in slow convergence.

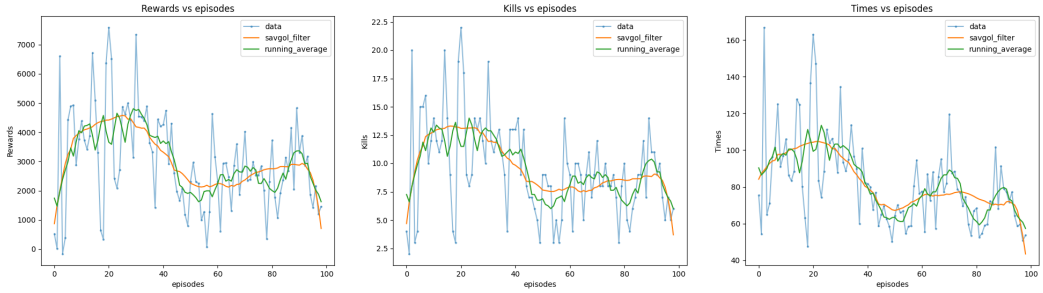

Figure 6. Case B: Fortuitous Initialization. Early successful kills allowed the agent to quickly lock onto the aggressive policy, resulting in rapid learning. However, it is important to note that a strong start does not guarantee permanent stability; the agent remains susceptible to catastrophic forgetting in later stages if stochastic exploration leads to the accumulation of poor experiences.