

## A Limitation

Although EDELINE maintains computational efficiency equivalent to that of DIAMOND and simultaneously addresses its memory limitations, the fundamental computational requirements of diffusion-based world models remain substantial relative to alternative approaches. The iterative characteristics of the diffusion process necessitate multiple forward passes throughout both training and inference phases, which consequently elevates computational overhead. The enhancement of computational efficiency within diffusion processes for world modeling applications constitutes a promising avenue for future research endeavors.

## B Boarder Impact

EDELINE addresses fundamental limitations inherent in diffusion-based world models through the enhancement of memory capabilities and imagination consistency. Our research contributes three pivotal advances to data-efficient reinforcement learning: (1) we overcome the fixed context window constraints of prior methodologies such as DIAMOND through the implementation of selective state space models, which substantially enhances memory capabilities, (2) we establish an end-to-end training methodology that effectively leverages hidden embeddings and incorporates HarmonyDream to augment world modeling performance, and (3) we demonstrate superior performance across visually rich 2D environments (i.e., Atari), challenging 3D first-person perspectives (i.e., ViZDoom), and memory-intensive procedurally-generated survival scenarios (i.e., Crafter), which illustrates the potential of our approach to address increasingly complex domains.

World models represent a promising direction for the improvement of sample efficiency and safety in reinforcement learning through the reduction of direct environment interaction requirements. While EDELINE advances this research domain, we acknowledge that imperfections in world models may continue to result in suboptimal agent behaviors. We believe our contributions toward more accurate and memory-capable world models will serve to mitigate such risks in future applications.

## C Additional Background Material Section

### C.1 Score-based Diffusion Generative Models

Diffusion probabilistic modeling [59, 19, 60] and score-based generative modeling [21, 61, 62] can be unified through a forward stochastic differential equation (SDE) formulation [22]. The forward diffusion process  $\{\mathbf{x}^\tau\}$  with continuous time variable  $\tau$  transforms the data distribution  $p^0 = p^{\text{data}}$  to prior distribution  $p^T = p^{\text{prior}}$ , expressed as:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, \tau)d\tau + g(\tau)d\mathbf{w}, \quad (7)$$

where  $\mathbf{f}(\mathbf{x}, \tau)$  represents the drift coefficient,  $g(\tau)$  denotes the diffusion coefficient, and  $\mathbf{w}$  is the Wiener process. The corresponding reverse-time SDE can then be formulated as:

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, \tau) - g(\tau)^2 \nabla_{\mathbf{x}} \log p^\tau(\mathbf{x})] d\tau + g(\tau)d\bar{\mathbf{w}}, \quad (8)$$

where  $\bar{\mathbf{w}}$  is the reverse-time Wiener process. Eq. (8) enables sampling from  $p^0$  when the (Stein) score function  $\nabla_{\mathbf{x}} \log p^\tau(\mathbf{x})$  is available. A common approach to estimate the score function is through the introduction of a denoiser  $D_\theta$ , which is trained to minimize the following objective:

$$\mathbb{E}_{\sigma \sim p^{\text{train}}} \mathbb{E}_{\mathbf{x}^0 \sim p^{\text{data}}} \mathbb{E}_{\mathbf{n} \sim \mathcal{N}(0, \sigma^2 I)} [\|D_\theta(\mathbf{x}^0 + \mathbf{n}; \sigma) - \mathbf{x}^0\|_2^2], \quad (9)$$

where  $\mathbf{n}$  is Gaussian noise with zero mean and variance determined by a variance scheduler  $\sigma(\tau)$  that follows a noise distribution  $p^{\text{train}}$ , and  $(\mathbf{x}^0 + \mathbf{n})$  corresponds to the perturbed data  $\mathbf{x}^\tau$ . The score function can then be estimated through:  $\nabla_{\mathbf{x}} \log p^\tau(\mathbf{x}) = \frac{1}{\sigma^2}(D_\theta(\mathbf{x}; \sigma) - \mathbf{x})$ . In practice, modeling the denoiser  $D_\theta$  directly can be challenging due to the wide range of noise scales. To address this, EDM [63] introduces a design space that isolates key design choices, including preconditioning functions  $\{c_{\text{skip}}, c_{\text{out}}, c_{\text{in}}, c_{\text{noise}}\}$  to modulate the unconditioned neural network  $F_\theta$  to represent  $D_\theta$ , which can be formulated as:

$$D_\theta(\mathbf{x}; \sigma) = c_{\text{skip}}(\sigma)\mathbf{x} + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)\mathbf{x}; c_{\text{noise}}(\sigma)). \quad (10)$$

The preconditioners serve distinct purposes:  $c_{\text{in}}(\sigma)$  and  $c_{\text{out}}(\sigma)$  maintain unit variance for network inputs and outputs across noise levels,  $c_{\text{noise}}(\sigma)$  provides transformed noise level conditioning, and  $c_{\text{skip}}(\sigma)$  adaptively balances signal mixing. This principled framework improves the robustness and efficiency of diffusion models, enabling state-of-the-art performance across various generative tasks.

### C.2 Multi-task Essence of World Model Learning

Modern world models [15, 7, 2, 8] typically address two fundamental prediction tasks: the modeling of environment dynamics through observations and the prediction of reward signals. The learning of these tasks requires distinct considerations based on the complexity of the environment. In simple low-dimensional settings, separate learning approaches suffice for each task. However, the introduction of high-dimensional visual inputs fundamentally alters this paradigm, as partial observability creates an inherent coupling between state estimation and reward prediction. This coupling necessitates joint learning through shared representations, an approach that aligns with established multi-task learning principles [64]. The implementation of such joint learning through shared representations introduces several technical challenges. The integration of multiple learning objectives requires careful consideration of their relative importance and interactions. A fundamental difficulty stems from the inherent scale disparity between high-dimensional visual observations and scalar reward signals. This disparity manifests in the world model learning objective, which combines observation modeling  $\mathcal{L}_o(\theta)$ , reward modeling  $\mathcal{L}_r(\theta)$ , and dynamics modeling  $\mathcal{L}_d(\theta)$  losses with weights  $w_o$ ,  $w_r$ ,  $w_d$  to control relative contributions:

$$\mathcal{L}(\theta) = w_o \mathcal{L}_o(\theta) + w_r \mathcal{L}_r(\theta) + w_d \mathcal{L}_d(\theta). \quad (11)$$

HarmonyDream [55] demonstrated that observation modeling tends to dominate this objective due to visual inputs' high dimensionality compared to scalar rewards. Their work introduced a variational formulation:

$$\mathcal{L}(\theta, w_o, w_r, w_d) = \sum_{i \in \{o, r, d\}} \mathcal{H}(\mathcal{L}_i(\theta), \frac{1}{w_i}) = \sum_{i \in \{o, r, d\}} w_i \mathcal{L}_i(\theta) + \log(\frac{1}{w_i}), \quad (12)$$

where  $\mathcal{H}(\mathcal{L}_i(\theta), w_i) = w_i \mathcal{L}_i(\theta) + \log(1/w_i)$  dynamically balances the losses by maintaining  $\mathbb{E}[w^* \cdot \mathcal{L}] = 1$ . This harmonization technique can substantially enhance sample efficiency and performance. Our work extends these insights through the integration of dynamic task balancing mechanisms into our EDELINE world model architecture.

## D Additional Experiments and Experiment Details

In this section, we provide extensive additional experiments and detailed experimental setups that complement the primary results presented in the main manuscript. These supplementary materials include in-depth analyses, ablation studies, and technical details that further validate and illuminate EDELINE’s advantages.

### D.1 Experimental Setup

We evaluate EDELINE on the Atari 100k benchmark [15], which serves as the standard evaluation protocol in recent model-based RL literature for fair comparison. In addition, our experimental validation extends to ViZDoom [18], MiniGrid [16], and Crafter [17] environments to demonstrate broader applicability. To ensure statistical significance, all reported results represent averages across three independent runs.

We evaluate EDELINE on the Atari 100k benchmark [15], which serves as the standard evaluation protocol in recent model-based RL literature for fair comparison. In addition, our experimental validation extends to ViZDoom [18] and MiniGrid [16] environments to demonstrate broader applicability. To ensure statistical significance, all reported results represent averages across three independent runs. The Atari 100k benchmark [15] encompasses 26 diverse Atari games that evaluate various aspects of agent capabilities. Each agent receives a strict limitation of 100k environment interactions for learning, in contrast to conventional Atari agents that typically require 50 million steps. EDELINE’s performance is evaluated against current state-of-the-art world model-based approaches, including DIAMOND [8], STORM [33], DreamerV3 [2], IRIS [35], TWM [32], and Drama [48]. For evaluating 3D scene understanding capabilities, we employ VizDoom scenarios that demand sophisticated 3D spatial reasoning in first-person environments. This provides a crucial testing ground beyond the third-person perspective of Atari environments. Furthermore, the Crafter and MiniGrid memory scenarios evaluate memorization capabilities through tasks that require information retention across extended time horizons.

### D.2 EDM Network Preconditioners and Training

Following DIAMOND [8], we use the EDM preconditioners from [63] for normalization and rescaling to improve network training:

$$c_{in}^{\tau} = \frac{1}{\sqrt{\sigma(\tau)^2 + \sigma_{data}^2}} \quad (13)$$

$$c_{out}^{\tau} = \frac{\sigma(\tau)\sigma_{data}}{\sqrt{\sigma(\tau)^2 + \sigma_{data}^2}} \quad (14)$$

$$c_{noise}^{\tau} = \frac{1}{4} \log(\sigma(\tau)) \quad (15)$$

$$c_{skip}^{\tau} = \frac{\sigma_{data}^2}{\sigma_{data}^2 + \sigma^2(\tau)}, \quad (16)$$

where  $\sigma_{data} = 0.5$ .

The noise parameter  $\sigma(\tau)$  is sampled to maximize the effectiveness of training:

$$\log(\sigma(\tau)) \sim \mathcal{N}(P_{mean}, P_{std}^2), \quad (17)$$

where  $P_{mean} = -0.4$ ,  $P_{std} = 1.2$ .

### 1033 D.3 Atari 100K Training Curves

1034 Fig. 6 presents the detailed training curves for all individual games in the Atari100k benchmark.

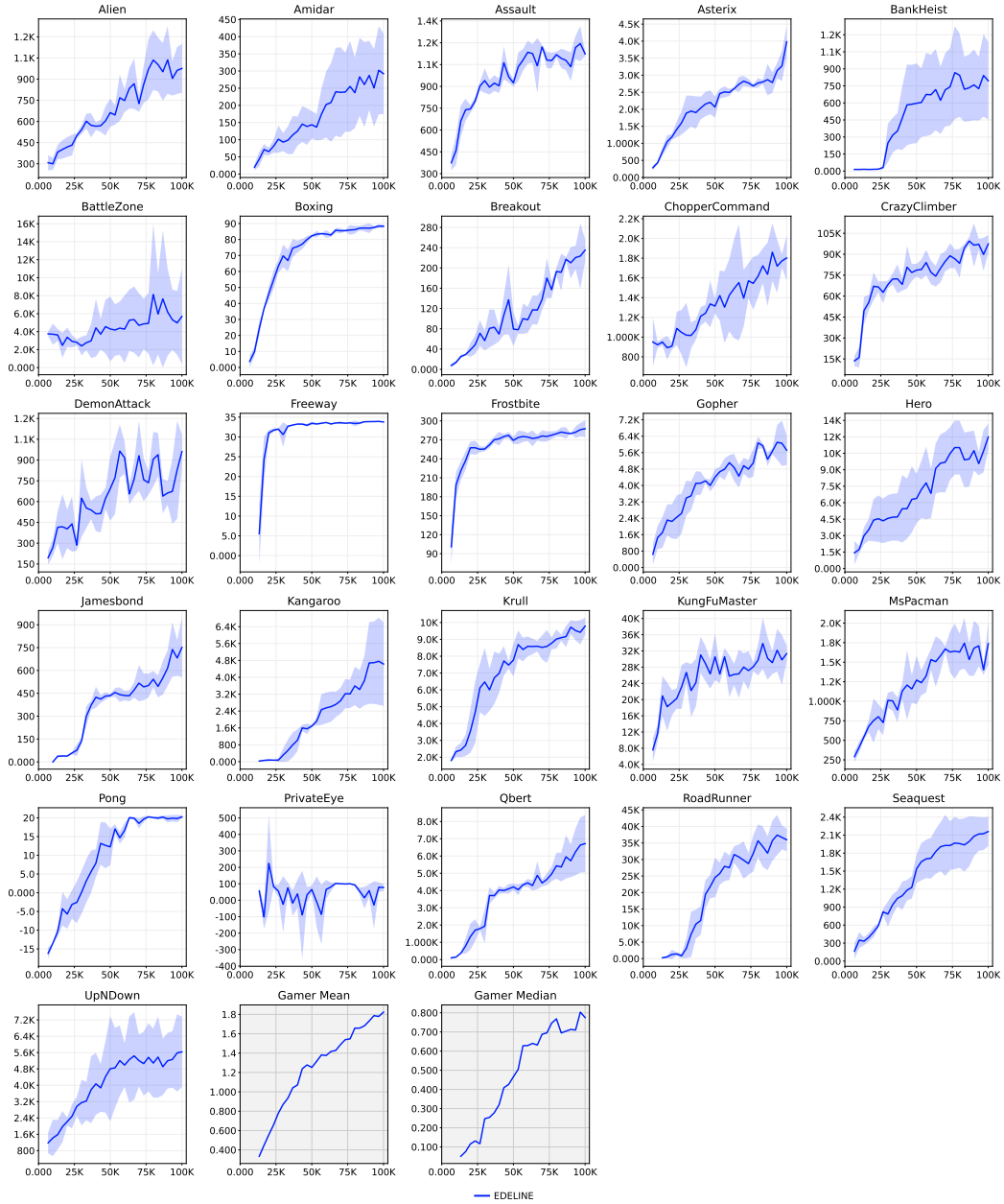


Figure 6: Training curves of EDELINE on the Atari100k benchmark for individual games (400K environment steps). The solid lines represent the average scores over 3 seeds, and the filled areas indicate the standard deviation across these 3 seeds.

#### 1035 D.4 Atari 100k Qualitative Analysis

1036 To provide deeper insights into EDELINE’s superior performance, we conduct qualitative analysis  
 1037 on three Atari games where our approach demonstrates the most significant improvements over  
 1038 DIAMOND: BankHeist, DemonAttack, and Hero. Fig. 7 presents temporal sequences comparing  
 1039 ground truth gameplay with predictions from both EDELINE and DIAMOND world models.

1040 In BankHeist, agents maximize scores through repeated map traversal to encounter new enemies.  
 1041 EDELINE’s SSM-enhanced world model maintains consistent tracking of the player character posi-  
 1042 tion throughout prediction sequences, while DIAMOND’s model shows progressive degradation with  
 1043 the character eventually disappearing from predictions. For DemonAttack, EDELINE successfully  
 1044 captures the relationship between enemy hits and score updates in its predictions. DIAMOND  
 1045 preserves basic visual structure but fails to reflect these crucial state transitions. The Hero envi-  
 1046 ronment showcases EDELINE’s long-range prediction capabilities, accurately capturing sequences  
 1047 of the player breaking obstacles and navigating new areas. These qualitative results highlight how  
 1048 EDELINE’s architectural innovations - SSM-based memory, unified training, and diffusion model-  
 1049 ing - enable robust state tracking, action-consequence modeling, and temporal consistency. These  
 capabilities directly contribute to EDELINE’s superior performance on the Atari 100k benchmark.

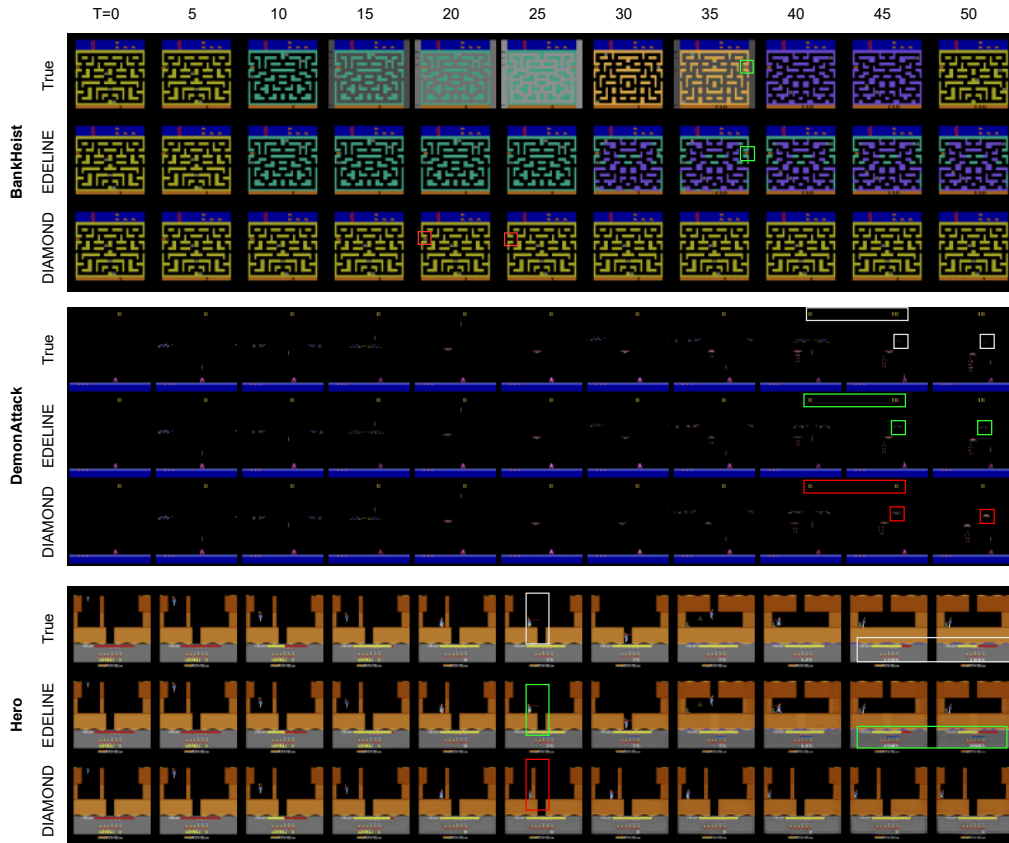


Figure 7: Qualitative comparison of world model predictions on three Atari games. Each panel shows temporal sequences comparing ground truth with EDELINE and DIAMOND predictions. In BankHeist (top), EDELINE successfully tracks the player character while DIAMOND loses this information. DemonAttack (middle) demonstrates EDELINE’s accurate modeling of score updates upon successful hits. Hero (bottom) showcases EDELINE’s ability to maintain consistent predictions of complex character-environment interactions across extended sequences. Colored boxes highlight successful (green) and failed (red) predictions of key game elements.

1050

## 1051 D.5 Extended Atari 100k Benchmark Performance Analysis

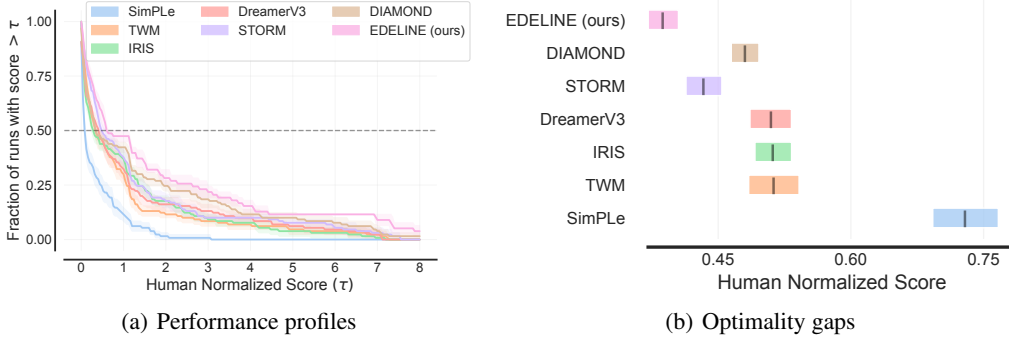


Figure 8: Additional performance analyses. (a) Performance profiles showing fraction of runs achieving scores above different human normalized score thresholds. (b) Optimality gaps demonstrating relative distance to theoretical optimal performance across different methods.

1052 To provide additional performance insights, we analyze EDELINE using performance profiles and  
 1053 optimality gaps [57]. Fig. 8(a) shows the empirical cumulative distribution of human-normalized  
 1054 scores (HNS) across all 26 Atari games. The y-axis indicates the fraction of games achieving scores  
 1055 above each HNS threshold  $\tau$ . EDELINE consistently maintains a higher fraction of games across  
 1056 different thresholds compared to baseline methods, particularly in the middle range ( $\tau$  between 1  
 1057 and 4). We further analyze model performance through optimality gaps shown in Fig. 8(b). The  
 1058 optimality gap measures the distance between model performance and theoretical optimal behavior.  
 1059 EDELINE achieves the smallest optimality gap among all compared methods, demonstrating its  
 1060 effectiveness in approaching optimal performance across the Atari 100k benchmark suite. These  
 1061 detailed analyses complement the aggregate metrics presented in Section 6.1 by providing a more  
 1062 granular view of performance distribution and theoretical efficiency.

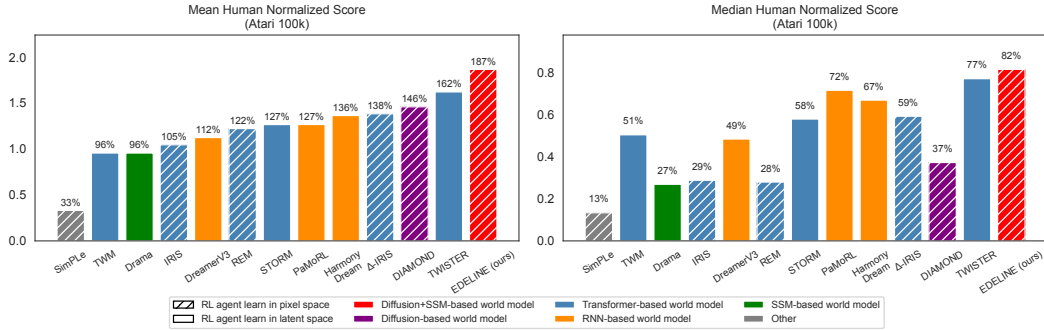


Figure 9: **Comparison of state-of-the-art model-based RL methods without using look-ahead search techniques on the Atari 100k benchmark.** EDELINE outperforms all existing model-based approaches on the Atari 100k benchmark. Previous methods can be categorized by their world model architectures: Transformer-based models (TWM [32], IRIS [35], REM [65], STORM [33],  $\Delta$ -IRIS [58], TWISTER [66]), RNN-based models (DreamerV3 [2], PaMoRL [67], Harmony-Dream [55]), SSM-based models (Drama [48]), and Diffusion-based models (DIAMOND [8]). Our proposed EDELINE advances the state-of-the-art by integrating diffusion modeling with state space models, combining their respective strengths in visual generation and temporal modeling.

## D.6 Atari 100k Linear Probing Analysis

To quantitatively evaluate the information content captured in EDELINE’s hidden representations, we conducted a comprehensive linear probing analysis, which assesses how effectively the model’s hidden states encode information critical for both reward prediction and observation prediction tasks.

Table 3: Linear probing loss of reward and observation prediction from EDELINE and DIAMOND hidden states.  $\mathcal{L}_{\text{rew}}$  (reward loss) was measured using cross-entropy loss on reward prediction, while  $\mathcal{L}_{\text{obs}}$  (observation loss) was measured using MSE loss on predicting encoded observation features. EDELINE outperforms DIAMOND in observation prediction with an average 57.3% reduction in loss, while both models achieve comparable performance on reward prediction (i.e., average  $\mathcal{L}_{\text{rew}}$  of 0.0246 vs. 0.0258). These results demonstrate that EDELINE’s unified hidden representation successfully encodes information required for both tasks.

Environment	EDELINE $\mathcal{L}_{\text{rew}}$	DIAMOND $\mathcal{L}_{\text{rew}}$	EDELINE $\mathcal{L}_{\text{obs}}$	DIAMOND $\mathcal{L}_{\text{obs}}$
Alien	<b>0.0434</b>	0.0663	<b>0.6161</b>	1.1798
Amidar	<b>0.0186</b>	0.0187	<b>0.6430</b>	0.7520
Assault	0.0779	<b>0.0270</b>	0.4894	<b>0.4199</b>
Asterix	0.0114	<b>0.0018</b>	<b>0.8247</b>	1.4411
BankHeist	<b>0.0023</b>	0.0039	<b>0.0312</b>	2.5398
BattleZone	<b>0.0129</b>	0.0220	0.2265	<b>0.1413</b>
Boxing	0.0482	<b>0.0011</b>	<b>0.3089</b>	0.5328
Breakout	0.0615	<b>0.0264</b>	0.6224	<b>0.4871</b>
ChopperCommand	0.0261	<b>0.0113</b>	<b>0.6815</b>	1.3819
CrazyClimber	<b>0.0029</b>	0.0034	<b>0.2844</b>	0.5513
DemonAttack	<b>0.0609</b>	0.0726	<b>0.8588</b>	1.0039
Freeway	<b>0.0000</b>	<b>0.0000</b>	<b>0.2163</b>	0.3911
Frostbite	<b>0.0001</b>	0.0002	<b>0.1992</b>	0.7435
Gopher	<b>0.0090</b>	0.0091	<b>0.3139</b>	0.9558
Hero	<b>0.0065</b>	0.0088	<b>0.0708</b>	0.1013
Jamesbond	<b>0.0054</b>	0.0058	<b>0.5931</b>	1.1080
Kangaroo	<b>0.0009</b>	0.0046	<b>0.0992</b>	0.5054
Krull	0.0728	<b>0.0455</b>	<b>0.3216</b>	0.8615
KungFuMaster	0.0048	<b>0.0043</b>	<b>0.2674</b>	0.8735
MsPacman	<b>0.0044</b>	0.0669	<b>0.2510</b>	0.8769
Pong	<b>0.0000</b>	<b>0.0000</b>	<b>0.1770</b>	1.3162
PrivateEye	<b>0.0014</b>	0.0144	<b>0.1188</b>	0.4931
Qbert	0.0240	<b>0.0025</b>	<b>0.1813</b>	0.2530
RoadRunner	0.0658	<b>0.0310</b>	<b>0.2032</b>	0.7896
Seaquest	<b>0.0008</b>	0.0299	<b>0.2159</b>	0.7852
UpNDown	<b>0.1012</b>	0.2110	<b>0.5967</b>	1.0783
ViZDoom-DeadlyCorridor	<b>0.0008</b>	0.0094	<b>0.3354</b>	1.2848
Average Loss	<b>0.0246</b>	0.0258	<b>0.3610</b>	0.8462

We trained linear probes on the hidden states extracted from both EDELINE and DIAMOND models. For our experimental setup, we collected 50 trajectories as training data and 10 trajectories as validation data for each environment, with five independent random seeds employed to ensure statistical robustness. For reward prediction, we trained a single linear layer to predict rewards for 50 epochs. For observation feature prediction, we trained a multi-layer perceptron (MLP) to predict encoded features of the subsequent observation (extracted from the pre-trained actor-critic network encoder) for 100 epochs. For EDELINE, we utilized the Mamba hidden states, while for DIAMOND, we employed the LSTM hidden states from its reward-termination model. We conducted these experiments across 26 Atari environments and the challenging ViZDoom-DeadlyCorridor environment to ensure comprehensive evaluation.

Table 3 presents the results of our linear probing analysis. EDELINE and DIAMOND demonstrate comparable performance on reward prediction, with average losses of 0.0246 and 0.0258 respectively. EDELINE outperforms DIAMOND in 15 out of 27 environments on this task. More significantly, EDELINE substantially outperforms DIAMOND in observation prediction, with an average 57.3% reduction in loss. EDELINE achieves lower observation prediction loss in 24 out of 27 environments. The substantial improvement in observation prediction while maintaining comparable reward prediction performance demonstrates that EDELINE’s unified hidden representation successfully captures information required for both tasks.

## D.7 Generation Quality Evaluation

To quantitatively assess the generation quality of our proposed world model, we conducted a comprehensive evaluation measuring pixel-wise Mean Squared Error (MSE) across 26 Atari environments and the challenging ViZDoom-DeadlyCorridor environment. This analysis provides direct evidence of EDELINE’s improved predictive accuracy compared to DIAMOND.

Table 4: Pixel-wise MSE comparison between EDELINE and DIAMOND across 27 environments. Lower values (in **bold**) indicate better performance. The rightmost column shows the imagination horizon length for each environment. The bottom row reports the average normalized score (EDELINE MSE / DIAMOND MSE), with values below 1.0 indicating EDELINE’s overall superior performance.

Environment	EDELINE	DIAMOND	Imagine Horizon Length
Alien	<b>0.0063</b>	0.0064	635
Amidar	<b>0.0177</b>	0.0235	1029
Assault	<b>0.1011</b>	0.2528	566
Asterix	0.0190	<b>0.0177</b>	1493
BankHeist	<b>0.1510</b>	0.1733	2019
BattleZone	0.0306	<b>0.0292</b>	1407
Boxing	<b>0.0068</b>	0.0174	1371
Breakout	0.0420	<b>0.0415</b>	1796
ChopperCommand	0.0084	<b>0.0082</b>	3006
CrazyClimber	<b>0.0666</b>	0.0686	3219
DemonAttack	<b>0.0312</b>	0.0318	1474
Freeway	<b>0.0015</b>	0.0030	1986
Frostbite	<b>0.0338</b>	0.0370	564
Gopher	<b>0.0152</b>	0.0172	2689
Hero	<b>0.1559</b>	0.2005	2103
Jamesbond	<b>0.2967</b>	0.3023	2212
Kangaroo	<b>0.0061</b>	0.0063	3241
Krull	0.1577	<b>0.1575</b>	1326
KungFuMaster	0.0210	0.0210	1864
MsPacman	<b>0.0179</b>	0.0197	892
Pong	0.0035	<b>0.0034</b>	1532
PrivateEye	0.1043	<b>0.0638</b>	2454
Qbert	<b>0.0435</b>	0.0460	1434
RoadRunner	<b>0.0526</b>	0.0532	960
Seaquest	<b>0.0107</b>	0.0136	1810
UpNDown	<b>0.1223</b>	0.1234	1279
ViZDoom-DeadlyCorridor	<b>0.0179</b>	0.0184	73
Mean Normalized Score	<b>0.918</b>	1.000	

Table 4 presents the comprehensive results of our pixel-wise MSE comparison. The analysis reveals that EDELINE achieves lower MSE than DIAMOND in 20 out of 27 environments. For meaningful comparison across environments with varying visual complexity, we calculated the average normalized score (EDELINE MSE / DIAMOND MSE) across all environments. Our analysis demonstrates that EDELINE achieves an average normalized score of 0.918, which represents an overall 8.2% reduction in pixel-level error compared to DIAMOND.

The superior generation quality of EDELINE can be attributed to its utilization of Mamba to incorporate longer observation history beyond the four frames employed by DIAMOND. This capability enables EDELINE to maintain more consistent predictions over extended horizons, which proves particularly advantageous in environments that necessitate memory (such as ViZDoom) or involve complex dynamics.



## D.8 Training Time Profile

We performed detailed training time profiling of EDELINE to evaluate its computational efficiency compared to DIAMOND. Table 5 provides a comprehensive breakdown of training time components across different scales, while Table 6 directly compares EDELINE with DIAMOND. Our profiling reveals that EDELINE achieves notable efficiency improvements in world model training. For world model updates, EDELINE is approximately 26.8% faster than DIAMOND. This efficiency gain stems from our unified architecture approach. While DIAMOND employs a two-stage training process (diffusion model for observations plus a separate CNN-LSTM network for reward/termination prediction), EDELINE’s unified architecture enables joint learning of observations, rewards, and terminations through shared representations. In addition, Mamba’s parallel scan algorithm contributes to this computational advantage during training. For actor-critic updates, EDELINE requires approximately 17.7% more computation time than DIAMOND. This difference occurs because while both methods use identical actor-critic architectures, EDELINE’s world model involves more complex inference during imagined rollouts due to SSM processing and cross-attention mechanisms. Specifically, as shown in Table 5, each imagination step requires 24.4ms, with 17.2ms dedicated to observation prediction and 6.5ms to reward/termination prediction. Overall, these timing differences balance out, resulting in comparable total training time between the two approaches. This demonstrates that EDELINE successfully maintains computational efficiency comparable to DIAMOND while delivering significantly enhanced memory capabilities and performance.

Table 5: Detailed breakdown of training time components across different scales. Profiling performed using a Nvidia RTX 4090 with default hyperparameters. Measurements are representative, as exact durations depend on specific hardware, environment, and training stage.

Single update	Time (ms)	Detail (ms)
Total	548.6	148.6 + 400
World model update	148.6	-
Actor-Critic model update	400	$15 \times 24.4 + 34$
Imagination step (x 15)	24.4	$17.2 + 6.5 + 0.7$
Next observation prediction	17.2	-
Denoising step (x 3)	5.7	-
Reward/Termination prediction	6.5	-
Action prediction	0.7	-
Loss computation and backward	34	-
Epoch	Time (s)	Detail (s)
Total	219.4	$59.4 + 160$
World model	59.4	$400 \times 148.6 \times 10^{-3}$
Actor-Critic model	160	$400 \times 400 \times 10^{-3}$
Run	Time (days)	Detail (days)
Total	2.9	$2.5 + 0.4$
Training time	2.5	$1000 \times 219.4 / (24 \times 3600)$
Other (collection, evaluation, checkpointing)	0.4	-

Table 6: Computational efficiency comparison between DIAMOND and EDELINE across different training stages, showing per-stage timing and relative differences.

Module	DIAMOND (ms/s/days)	EDELINE (ms/s/days)	Difference (%)
<b>Single Update</b>			
Total	543 ms	548.6 ms	+1.03%
World Model Update	203 ms	148.6 ms	-26.80%
Actor-Critic Model Update	340 ms	400 ms	+17.65%
<b>Epoch</b>			
Total	217 s	219.4 s	+1.11%
World Model	81 s	59.4 s	-26.67%
Actor-Critic Model	136 s	160 s	+17.65%

## 1120 D.9 Ablation Studies

1121 To systematically validate the effectiveness of EDELINE’s key architectural components, we perform  
 1122 comprehensive ablation studies across multiple environments. These studies isolate the contribution  
 1123 of each design decision and demonstrate their impact on model performance. The ablation studies in  
 1124 Appendix D.9.1 and D.9.2 focus on five representative environments for validating the proposed key  
 1125 components. These include four Atari games where EDELINE demonstrates significant improvements  
 1126 over DIAMOND (BankHeist, DemonAttack, Hero, Seaquest), and MiniGrid-MemoryS9 for memory  
 1127 capability evaluation. This selection provides comprehensive validation across visual prediction  
 1128 quality and memorization requirements. To further validate the advantages of MAMBA compared to  
 1129 Transformer architectures with quadratic time complexity, we conducted experiments on memory  
 1130 tasks such as MiniGrid MemoryS7/S9 and Crafter. These results are presented in Appendix D.9.4.

### 1131 D.9.1 Choice of REM architecture

1132 To validate the selection of Mamba for REM, we compare its performance against traditional linear-  
 1133 time sequence models GRU and LSTM across five environments. Fig. 10 illustrates that although  
 1134 all models achieve reasonable performance, Mamba demonstrates more stable learning curves and  
 1135 superior final performance, particularly in memory-intensive tasks such as BankHeist and MiniGrid-  
 1136 MemoryS9. GRU and LSTM models exhibit increased training variance with lower final scores,  
 1137 which validates Mamba’s effectiveness.

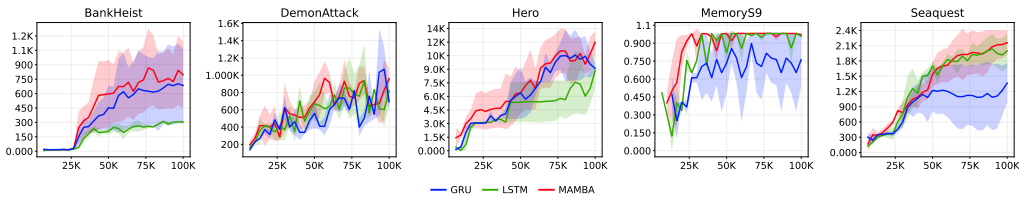


Figure 10: Performance comparison of different linear-time sequence models as REM architecture across five environments. Training curves show mean and standard deviation over three seeds. Mamba (red) shows more stable training progression and superior final performance compared to GRU (blue) and LSTM (green).

### 1138 D.9.2 Cross-Attention in Next-Frame Predictor

1139 To evaluate whether cross-attention improves the Next-Frame Predictor’s ability to process  
 1140 information-rich hidden embeddings, we examine EDELINE with and without this mechanism.  
 1141 As depicted in Fig. 11, the original EDELINE demonstrates superior performance in BankHeist,  
 1142 MemoryS9, and Seaquest where rich contextual information processing proves essential. The Memo-  
 1143 ryS9 environment validates this necessity, as models must integrate historical information for complete  
 1144 representation reconstruction. Cross-attention enables effective fusion of hidden embeddings with  
 1145 visual features for temporal context integration.

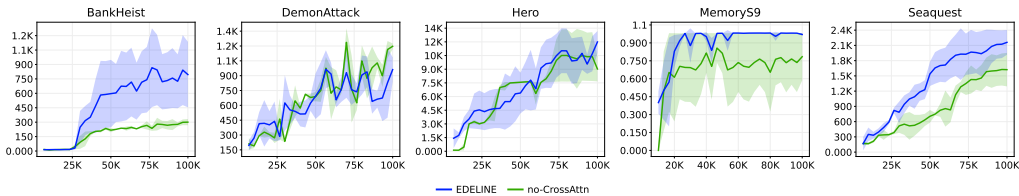
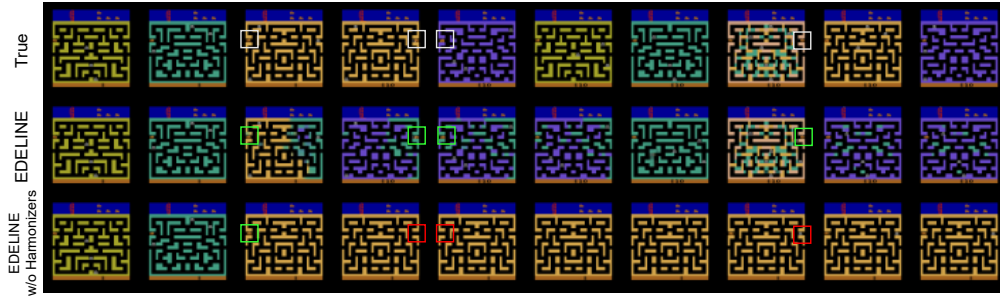


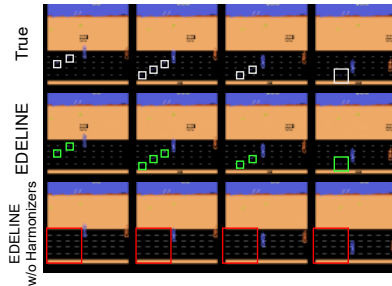
Figure 11: Ablation study comparing EDELINE with cross-attention blocks (blue) and without (green) across five test environments. Training curves depict mean and standard deviation over three seeds. EDELINE’s cross-attention mechanism provides advantages in environments requiring rich contextual information processing.

### 1146 D.9.3 Effect of Harmonizers

1147 To validate the effectiveness of harmonizers, we evaluate performance across the full Atari 100k  
 1148 benchmark.



(a) BankHeist



(b) RoadRunner

Figure 12: Effect of harmonizers on world model predictions. (a) Without harmonizers, EDELINE fails to maintain consistent character modeling in BankHeist. (b) In RoadRunner, removal of harmonizers leads to loss of reward-relevant visual details. Colored boxes highlight successful (green) and failed (red) predictions of key game elements.

1149 Table 7 presents quantitative performance comparison between EDELINE with and without har-  
 1150 monizers. The ablation results demonstrate significant performance degradation across multiple  
 1151 environments when harmonizers are removed, with particularly notable drops in games requiring  
 1152 precise reward related visual detail retention (BankHeist and RoadRunner). Qualitative analysis on  
 1153 these two environments, which showed the largest performance improvements with harmonizers,  
 1154 reveals how harmonizers contribute to world model performance. In BankHeist (Fig. 12(a)), removing  
 1155 harmonizers causes the world model to lose track of game characters, failing to maintain consistent  
 1156 agent representation across prediction sequences. Similarly, in RoadRunner (Fig. 12(b)), the model  
 1157 without harmonizers fails to capture reward-relevant visual details, degrading its ability to model  
 1158 critical game state information.

1159 These results demonstrate how harmonizers help achieve a crucial balance between observation  
 1160 modeling and reward prediction. Without harmonizers, the world model struggles with fine-grained  
 1161 task-relevant observations in both environments - tracking small character sprites in BankHeist  
 1162 and capturing critical game state details in RoadRunner. By maintaining this dynamic equilibrium  
 1163 between observation and reward modeling, harmonizers enable EDELINE to effectively learn compact  
 1164 task-centric dynamics while preserving essential visual details for sample-efficient learning.

Table 7: To evaluate the benefits of Harmonizers in detail, we compare DreamerV3 and its variant with Harmonizers (HarmonyDream), alongside EDELINE without Harmonizers and the complete EDELINE on the 26 games in the Atari 100k benchmark. Bold numbers indicate the highest scores.

Game	Random	Human	DreamerV3	HarmonyDream	EDELINE w/o Harmonizers	EDELINE (ours)
Alien	227.8	7127.7	959.4	889.7	<b>1086.2</b>	974.6
Amidar	5.8	1719.5	139.1	141.1	212.5	<b>299.5</b>
Assault	222.4	742.0	705.6	1002.7	1180.1	<b>1225.8</b>
Asterix	210.0	8503.3	932.5	1140.3	<b>4612.8</b>	4224.5
BankHeist	14.2	753.1	648.7	<b>1068.6</b>	139.8	854.0
BattleZone	2360.0	37187.5	12250.0	<b>16456.0</b>	2685.0	5683.3
Boxing	0.1	12.1	78.0	79.6	<b>88.3</b>	88.1
Breakout	1.7	30.5	31.1	52.6	<b>255.7</b>	250.5
ChopperCommand	811.0	7387.8	410.0	1509.6	<b>2616.5</b>	2047.3
CrazyClimber	10780.5	35829.4	97190.0	82739.0	96889.5	<b>101781.0</b>
DemonAttack	152.1	1971.0	303.3	202.6	924.2	<b>1016.1</b>
Freeway	0.0	29.6	0.0	0.0	33.8	<b>33.8</b>
Frostbite	65.2	4334.7	<b>909.4</b>	678.7	289.0	286.8
Gopher	257.6	2412.5	3730.0	<b>13042.8</b>	7982.7	6102.3
Hero	1027.0	30826.4	11160.5	<b>13378.0</b>	9366.2	12780.8
Jamesbond	29.0	302.8	444.6	317.1	527.5	<b>784.3</b>
Kangaroo	52.0	3035.0	4098.3	5117.6	3970.0	<b>5270.0</b>
Krull	1598.0	2665.5	7781.5	7753.6	8762.7	<b>9748.8</b>
KungFuMaster	258.5	22736.3	21420.0	22274.0	14088.5	<b>31448.0</b>
MsPacman	307.3	6951.6	1326.9	1680.7	1773.3	<b>1849.3</b>
Pong	-20.7	14.6	18.4	18.6	18.4	<b>20.5</b>
PrivateEye	24.9	69571.3	881.6	<b>2932.2</b>	73.0	99.5
Qbert	163.9	13455.0	3405.1	3932.5	5062.3	<b>6776.2</b>
RoadRunner	11.5	7845.0	15565.0	14646.4	23272.5	<b>32020.0</b>
Seaquest	68.4	42054.7	618.0	665.3	1277.2	<b>2140.1</b>
UpNDown	533.4	11693.2	7567.1	<b>10873.6</b>	2844.6	5650.3
#Superhuman ( $\uparrow$ )	0	N/A	9	11	11	<b>13</b>
Mean ( $\uparrow$ )	0.000	1.000	1.124	1.364	1.674	<b>1.866</b>

Table 8: Performance and computational efficiency comparison between Transformer and Mamba-based REM architectures across memory-demanding environments. Values for MiniGrid environments represent success rates, while Crafter values show average return over three independent seeds after 1M environment steps. Mamba demonstrates comparable or superior performance with significantly better computational efficiency.

REM Architecture	MiniGrid-MemoryS7	MiniGrid-MemoryS9	Crafter	Training Time (per epoch)
Transformer	0.980	0.978	$8.9 \pm 0.6$	293.0 sec
Mamba	0.981	0.982	<b><math>11.5 \pm 0.9</math></b>	<b>219.4 sec</b>

#### D.9.4 Transformer-based Recurrent Embedding Module

To investigate the efficacy of Mamba compared to Transformer architecture, we implemented a Transformer-based variant of the Recurrent Embedding Module (REM). This ablation study aims to assess both performance and computational efficiency across memory-demanding environments, which provides critical insights into architecture selection for world modeling. We evaluated both architectures on the MiniGrid-MemoryS7, MiniGrid-MemoryS9, and Crafter environments, which specifically challenge a model’s ability to retain and utilize historical information.

As demonstrated in Table 8, both architectures achieve comparable performance on the simpler MiniGrid-Memory environments, with success rates that approach optimal performance. Nevertheless, MAMBA demonstrates a substantial 29.2% improvement over Transformers in the more complex Crafter environment, which requires nuanced long-term memory and planning capabilities. Moreover, Mamba delivers approximately 25% faster training times due to its linear time complexity compared to the quadratic complexity of Transformers.

These results substantiate our selection of Mamba as the foundation for EDELINE. Although both architectures perform comparably in simpler memory tasks, Mamba demonstrates superior performance in complex environments with intricate memory dependencies while maintaining superior computational efficiency. The combination of enhanced modeling capabilities and linear scaling with sequence length makes MAMBA particularly well-suited for world modeling applications that must process lengthy observation histories efficiently.

## 1184 **D.10 ViZDoom Environment Specifications**

1185 This appendix provides detailed specifications for the ViZDoom scenarios used in our experiments.  
1186 We evaluate EDELINE on five key scenarios that test different agent capabilities in first-person 3D  
1187 environments.

### 1188 **D.10.1 DeadlyCorridor**

- 1189 • **Objective:** Navigate through enemy fire to acquire armor at corridor end
- 1190 • **Reward Mechanism:**
  - 1191 – +1 for each enemy killed
  - 1192 – +1 for armor acquisition
  - 1193 – -1 for each damage instance received
- 1194 • **Evaluation Metric:** Binary success if armor acquired before episode termination
- 1195 • **Episode Termination:** Agent death or 512 environment interaction steps.

### 1196 **D.10.2 HealthGathering**

- 1197 • **Objective:** Survive by collecting medkits in toxic environment
- 1198 • **Reward Mechanism:**
  - 1199 – +1 for each medkit collected
- 1200 • **Evaluation Metric:** Final health percentage (0-100%)
- 1201 • **Episode Termination:** Agent death or 512 environment interaction steps.

### 1202 **D.10.3 PredictPosition**

- 1203 • **Objective:** Anticipate enemy movement to land delayed rocket hit
- 1204 • **Reward Mechanism:**
  - 1205 – +1 for successful enemy kill
- 1206 • **Evaluation Metric:** Binary success if enemy eliminated
- 1207 • **Episode Termination:** Successful kill or 75 environment interaction steps.

### 1208 **D.10.4 Basic**

- 1209 • **Objective:** Eliminate stationary enemy with limited ammunition
- 1210 • **Reward Mechanism:**
  - 1211 – +1 for successful kill
- 1212 • **Evaluation Metric:** Binary success if enemy eliminated
- 1213 • **Episode Termination:** Successful kill or 75 environment interaction steps.

### 1214 **D.10.5 DefendCenter**

- 1215 • **Objective:** Survive against infinite enemy waves
- 1216 • **Reward Mechanism:**
  - 1217 – +1 per enemy killed
  - 1218 – -1 for agent death
- 1219 • **Evaluation Metric:** Total enemies killed per episode
- 1220 • **Episode Termination:** Agent death or 512 environment interaction steps.

1221 All scenarios use the following common configuration parameters unless otherwise specified:

- 1222 • Observation space: (64,64,3)
- 1223 • Action space: Discrete movement and attack actions

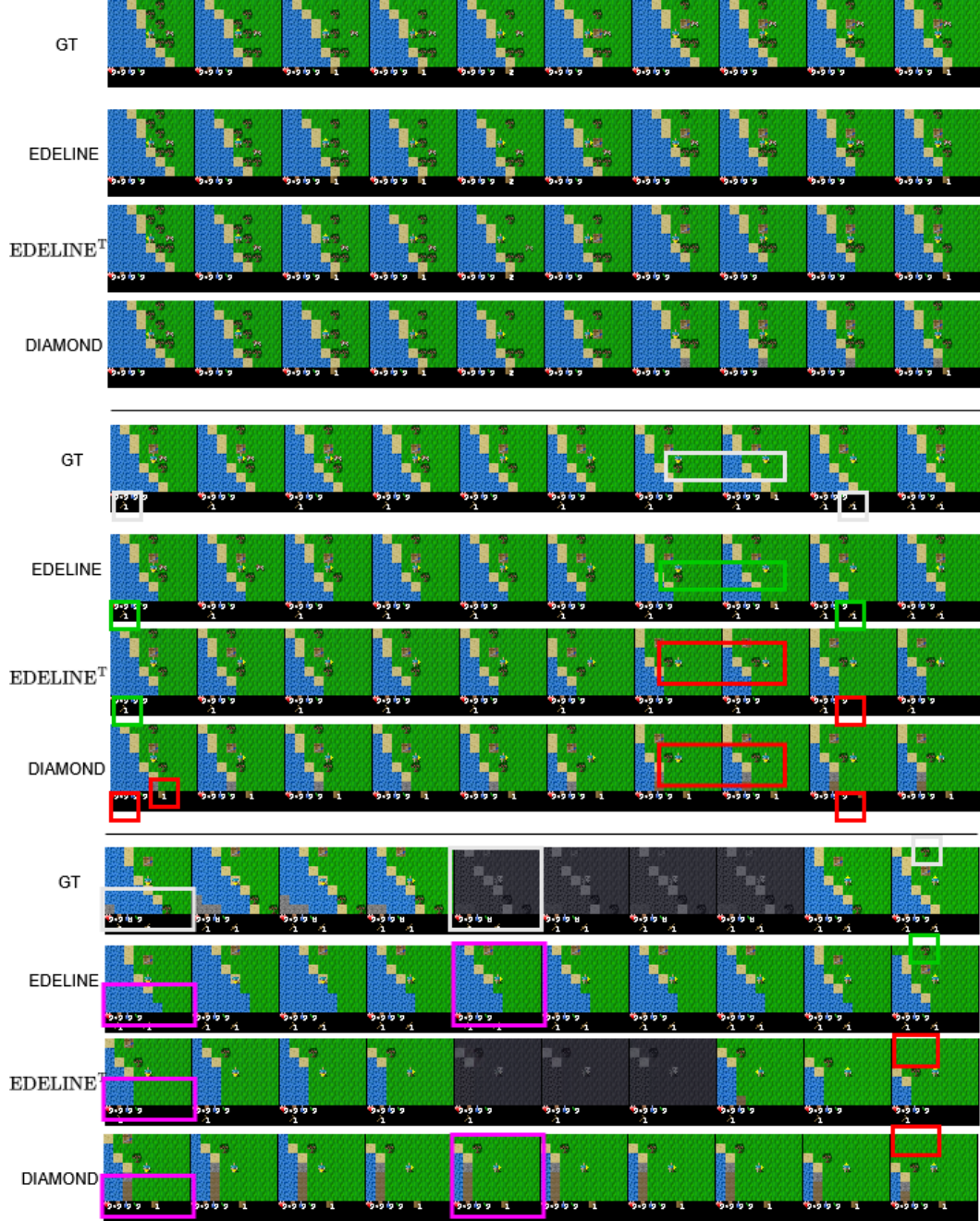


Figure 13: Qualitative comparison of world model predictions in Crafter. We compare GT (Ground Truth), EDELINE, EDELINE<sup>T</sup> (Transformer-based variant), and DIAMOND. The red boxes indicate prediction errors, the pink boxes show acceptable errors, and the green boxes highlight correct predictions of key elements.

In Fig. 13, we can observe comparative results between the different model architectures. In timesteps 1-10 (top row), all three methods demonstrate reasonable consistency in their predictions. Starting from timesteps 11-20 (middle row), notable differences emerge: DIAMOND fails to correctly represent crafted materials, and both DIAMOND and EDELINE<sup>T</sup> incorrectly predict outcomes when the player chops trees and crafts new materials. EDELINE maintains higher prediction accuracy throughout these interactions. In timesteps 21-30 (bottom row), all methods generate acceptable

1231 predictions for previously unseen regions. While these differ from Ground Truth, they remain  
1232 acceptable as they preserve gameplay-critical elements. During sleep actions, both EDELINE and  
1233 DIAMOND fail to predict the sleeping state, which we classify as acceptable error given that sleep  
1234 is a random event triggered independently of player actions. Most importantly, when the player  
1235 returns to previously visited areas, only EDELINE successfully remembers and reproduces critical  
1236 environmental features such as trees. EDELINE<sup>T</sup> and DIAMOND both fail to recall the crafting table  
1237 and trees outside the map border. This evidence demonstrates EDELINE’s superior memorization  
1238 capability for maintaining environmental consistency over long horizons.

Table 9: Hyperparameters for EDELINE.

Hyperparameter	Symbol	Value
<b>General</b>		
Number of epochs	—	1000
Training steps per epoch	—	400
Environment steps per epoch	—	100
Batch size	—	32
Sequence Length	$T$	19
Epsilon (greedy) for collection	$\epsilon$	0.01
Observation shape	(h,w,c)	(64,64,3)
<b>Actor Critic</b>		
Imagination horizon	$H$	15
Discount factor	$\gamma$	0.985
Entropy weight	$\eta$	0.001
$\lambda$ -returns coefficient	$\lambda$	0.95
LSTM dimension	—	512
Residual blocks layers	—	[1,1,1,1]
Residual blocks channels	—	[32,32,64,64]
<b>World Model</b>		
Number of conditioning observations	$L$	4
Burn-in length	$B$	4
Hidden embedding dimension	d_model	512
Reward / Termination Model Hidden Units	—	512
<b>Mamba</b>		
Mamba layers	n_layers	3
Mamba state dimension	d_state	16
Expand factor	—	2
1D Convolution Dimension	d_conv	4
Residual blocks layers	—	[1,1,1,1]
Residual blocks channels	—	[64,64,64,64]
Action embedding dimension	—	128
<b>Diffusion</b>		
Sampling method	—	Euler
Number of denoising steps	—	3
Condition embedding dimension	—	256
Residual blocks layers	—	[2,2,2,2]
Residual blocks channels	—	[64,64,64,64]
<b>Optimization</b>		
Optimizer	—	AdamW
Learning rate	$\alpha$	1e-4
Epsilon	—	1e-8
Weight decay (World Model)	—	1e-2
Weight decay (Actor-Critic)	—	0
Max grad norm (World Model)	—	1.0
Max grad norm (Actor-Critic)	—	100.0



## 1240 E EDELINE Algorithm

1241 We summarize the overall training procedure of EDELINE in Algorithm 1 below, which is modified  
 1242 from Algorithm 1 in [8]. We denote as  $\mathcal{D}$  the replay dataset where the agent stores data collected from the real environment, and other notations are introduced in previous sections or are self-explanatory.

---

### Algorithm 1: EDELINE

---

#### Procedure training\_loop():

```

for epochs do
  collect_experience(steps_collect)
  for steps_world_model do
    update_world_model()
  for steps_actor_critic do
    update_actor_critic()

```

#### Procedure collect\_experience(n):

```

 $o_0^0 \leftarrow \text{env.reset}()$ 
for  $t = 0$  to  $n - 1$  do
  Sample  $a_t \sim \pi_\theta(a_t \mid o_t^0)$ 
   $o_{t+1}^0, r_t, d_t \leftarrow \text{env.step}(a_t)$ 
   $\mathcal{D} \leftarrow \mathcal{D} \cup \{o_t^0, a_t, r_t, d_t\}$ 
  if  $d_t = 1$  then
     $o_{t+1}^0 \leftarrow \text{env.reset}()$ 

```

#### Procedure update\_world\_model():

```

Sample indexes  $\mathcal{I} := \{t, \dots, t + T - 1\}$  // sequence length  $T$ 
Sample sequence  $(o_i^0, a_i, r_i, d_i)_{i \in \mathcal{I}} \sim \mathcal{D}$ 
Initialize  $h_{t-1}$  // MAMBA hidden state
Parallel for  $i \in \mathcal{I}$  do
   $h_i \leftarrow f_\phi(o_i^0, a_i, h_{i-1})$  // processed in parallel via MAMBA parallel scan
   $\hat{r}_i \sim p_\phi(\hat{r}_i \mid h_i)$ 
   $\hat{d}_i \sim p_\phi(\hat{d}_i \mid h_i)$ 
Compute  $\mathcal{L}_{\text{rew}}(\phi) = \sum_{i \in \mathcal{I}} \text{CE}(\hat{r}_i, r_i)$  // CE: cross-entropy loss
Compute  $\mathcal{L}_{\text{end}}(\phi) = \sum_{i \in \mathcal{I}} \text{CE}(\hat{d}_i, d_i)$  // CE: cross-entropy loss
Sample index  $j \sim \text{Uniform}\{t + B, \dots, t + T - 1\}$  // burn-in  $B$  steps
Sample  $\log(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$  // log-normal sampling from EDM
Define  $\tau := \sigma$  // identity schedule from EDM
Sample  $o_j^\tau \sim \mathcal{N}(o_j^0, \sigma^2 \mathbf{I})$  // add Gaussian noise
Compute  $\hat{o}_j^0 = D_\phi(o_j^\tau, \tau, o_{j-L}^0, \dots, o_{j-1}^0, h_{j-1})$ 
Compute observation modeling loss  $\mathcal{L}_{\text{obs}}(\phi) = \|\hat{o}_j^0 - o_j^0\|^2$ 
Update  $\phi$  according to Eq. (5)

```

#### Procedure update\_actor\_critic():

```

Sample initial buffer  $(o_{t-B+1}^0, a_{t-B+1}, \dots, o_t^0) \sim \mathcal{D}$ 
// Burn in LSTM states  $\pi_\theta, V_\theta$  and MAMBA states  $f_\phi$  with the buffer
for  $i = t$  to  $t + H - 1$  do
  Sample  $a_i \sim \pi_\theta(a_i \mid o_i^0)$ 
  Compute  $h_i \leftarrow f_\phi(o_i, a_i, h_{i-1})$ 
  Sample reward  $r_i$ , next observation  $o_{i+1}^0$ , and termination  $d_i$  via  $p_\phi$ 
Compute  $V_\theta(o_i^0)$  for  $i = t, \dots, t + H$ 
Compute RL losses  $\mathcal{L}_V(\theta)$  and  $\mathcal{L}_\pi(\theta)$ 
Update  $\pi_\theta$  and  $V_\theta$ 

```

---

## 1244 F Actor-Critic Learning Objectives

1245 We follow DIAMOND [8] in the design of our agent behavior learning. Let  $o_t$ ,  $r_t$ , and  $d_t$  denote the  
 1246 observations, rewards, and boolean episode terminations predicted by our world model. We denote  $H$   
 1247 as the imagination horizon,  $V_\theta$  as the value network,  $\pi_\theta$  as the policy network, and  $a_t$  as the actions  
 1248 taken by the policy within the world model.

1249 For value network training, we use  $\lambda$ -returns to balance bias and variance in the regression target.  
 1250 Given an imagined trajectory of length  $H$ , we define the  $\lambda$ -return recursively:

$$\Lambda_t = \begin{cases} r_t + \gamma(1 - d_t)[(1 - \lambda)V_\theta(o_{t+1}) + \lambda\Lambda_{t+1}] & \text{if } t < H \\ V_\theta(o_H) & \text{if } t = H. \end{cases} \quad (18)$$

1251 The value network  $V_\theta$  is trained to minimize  $\mathcal{L}_V(\theta)$ , the expected squared difference with  $\lambda$ -returns  
 1252 over imagined trajectories:

$$\mathcal{L}_V(\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{H-1} (V_\theta(\mathbf{x}_t) - \text{sg}(\Lambda_t))^2 \right], \quad (19)$$

1253 where  $\text{sg}(\cdot)$  denotes the gradient stopping operation, following standard practice [2, 35].

1254 For policy training, we leverage the ability to generate large amounts of on-policy trajectories in  
 1255 imagination using a REINFORCE objective [68]. The policy is trained to minimize:

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{H-1} \log(\pi_\theta(a_t|o_{\leq t})) \text{sg}(\Lambda_t - V_\theta(o_t)) + \eta \mathcal{H}(\pi_\theta(a_t|o_{\leq t})) \right], \quad (20)$$

1256 where  $V_\theta(o_t)$  serves as a baseline to reduce gradient variance, and the entropy term  $\mathcal{H}$  with weight  $\eta$   
 1257 encourages sufficient exploration.