A Appendix: EfficientZero Objectives

To warrant the latent dynamics that can mirror the true states of the environment, MuZero is trained to predict three necessary quantities directly relevant for planning: (1) the policy target π obtained from visit count distribution of the MCTS (2) immediate reward u from environment (3) bootstrapped value target z where $z = \sum_{i=0}^{k-1} \gamma^i u^i + \gamma^k v_{t+k}$. On top of MuZero, EfficientZero adds a self-supervised consistency loss term, and predicts sum of rewards instead of single-step reward. We refer reader to the original manuscript for more implementation details (Ye et al., 2021). Here, we present the learning objective for EfficientZero at time step t with k unroll steps:

$$\mathcal{L}_t^{\text{ez}}(\theta) = \sum_{k=0}^{K} \underbrace{\|\mathcal{L}^r(u_{t+k}, \hat{r}_t^k)\|_2^2}_{\text{reward}} + \lambda_1 \underbrace{\|\mathcal{L}^p(\pi_{t+k}, \hat{p}_t^k)\|_2^2}_{\text{policy}} + \lambda_2 \underbrace{\|\mathcal{L}^v(z_{t+k}, \hat{v}_t^k)\|_2^2}_{\text{value}} + \lambda_3 \underbrace{\|\mathcal{L}^s(s_{t+1}, \hat{s}_{t+1})\|_2^2}_{\text{consistency}} + c||\theta|| \tag{3}$$

B Appendix: Task Weights Computation

Within the N-steps update, we maintain a task-specific counter s^i and update the counter by Δs_n^i at each step n as follows:

$$\Delta s_n^i = \begin{cases} 1, & \text{if } \ \text{Sim}(\hat{\mathcal{M}}^i, \mathcal{M}) > 0.1 \\ 0, & \text{otherwise.} \end{cases}$$
$$s^i = s^i + \Delta s_n^i \tag{4}$$

After N steps, the new task weights η^i can be reset by $\eta^i = \frac{s^i}{N}$, and be adopted in our online adaptation objective in Equation 1. In practice, we start task weights update at 10k steps to ensure enough data from the online target task is collected. All task weights are initialized as 1 for the first 10k steps, and reset at the beginning of each every T-cycle.

Figure 9 shows how task weights are adaptively adjusted by the model during 100k environment steps during online finetuning stage for 10 games reported in Figure 6.

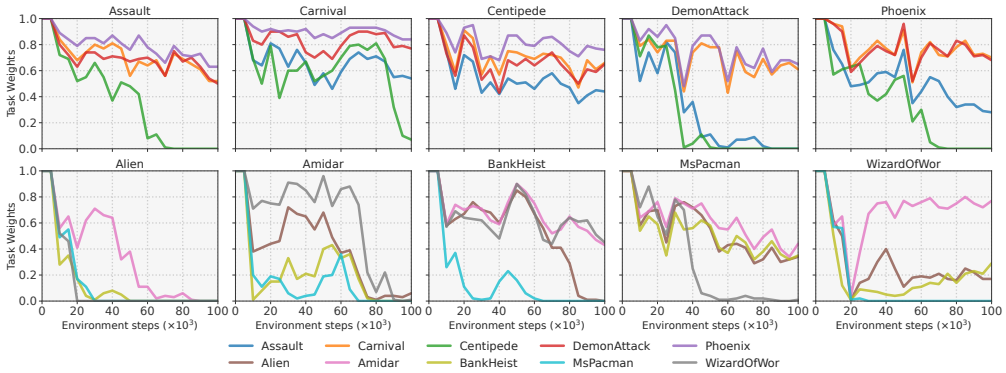

Figure 9: Task weights adaptively adjusted by the model as it progresses in online finetuning to the target task.

C Appendix: Comparison between Multi-Game Offline Training and Distillation

In Figure 10, we present results from multi-game offline training and multi-game distillation for 4 games. The multi-game results are evaluated from a single set of model parameters trained with all 4 games. While the model can easily learn each game individually (offline single game), it fails to learn 4 games simultaneously in the offline setting. On the opposite, our multi-game distillation successfully learns 4 games from teacher targets in multi-task fashion.

D Appendix: XTRA Game Scores for Table 1.

We report raw game scores from our XTRA stated in Table 1 for all 5 seeds we have run. We attach the mean, median, and standard deviation of game scores from 5 seeds of each individual game. We

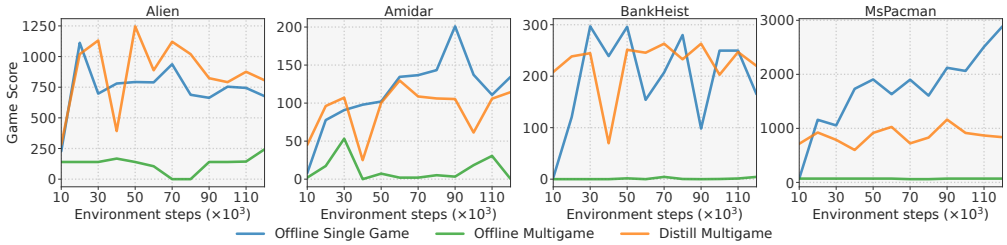

Figure 10: Comparison between Multi-Game Offline Training and Distillation

also list the random and human score from Badia et al. (2020), and calculate the Human Normalized Score based on the formula: $(\text{score}_{\text{agent}} - \text{score}_{\text{random}})/(\text{score}_{\text{human}} - \text{score}_{\text{random}})$. There is no random or human score available for Carnival, and therefore the aggregated Human Normalized Mean and Median Scores are calculated from the other 9 games. Details are in Table 3.

Table 3: Game score per seed and its aggregate results. Random and human scores are adopted from Badia et al. (2020).

Game	Game Score per Seed					Aggregated Metrics			References		Human Normed
	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean	Median	Std	Random	Human	
Assault	1450.19	1356.53	1236.19	1215.12	1214.72	1294.55	1236.19	105.06	222.40	742.00	2.06
Carnival	3865.31	4867.50	4155.62	2601.88	3814.38	3860.94	3865.31	819.67	-	-	
Centipede	7596.38	6179.25	5380.41	5300.03	3950.88	5681.39	5380.41	1336.58	2090.90	12017.00	0.36
DemonAttack	10470.78	8051.25	27574.06	8117.81	16490.47	14140.88	10470.78	8258.35	152.10	1971.00	7.69
Phoenix	20875.94	10988.44	10521.88	15803.44	14709.06	14579.75	14709.06	4198.81	761.40	7242.60	2.13
Alien	569.69	807.50	814.06	1388.12	1194.38	954.75	814.06	329.77	227.80	7127.70	0.11
Amidar	93.00	104.34	76.47	97.38	79.59	90.16	93.00	11.84	5.80	1719.50	0.05
BankHeist	303.12	316.56	270.62	270.00	364.06	304.88	303.12	38.83	14.20	753.10	0.39
MsPacman	1109.69	1960.00	1865.94	1228.44	1134.38	1459.69	1228.44	417.48	307.30	6951.60	0.17
WizardOfWor	1275.00	687.50	1056.25	990.62	915.62	985.00	990.62	213.62	563.50	4756.50	0.10
Human Normed Mean											1.45
Human Normed Median											0.36

E Appendix: Additional Evaluation Curves of XTRA on Atari 100k Benchmark

We show additional evaluation curves of XTRA on 5 games shown in Table 2. The averaged game score is evaluated by 5 seeds each with 32 evaluation episodes.

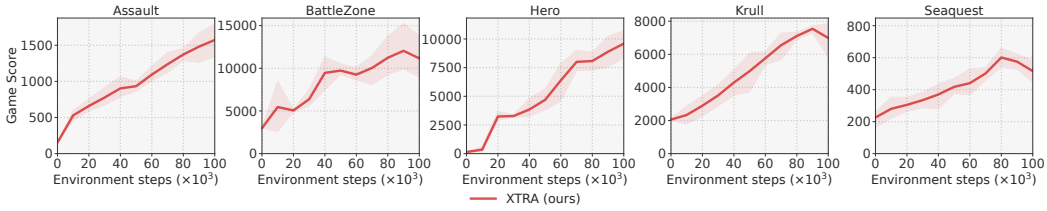

Figure 11: Evaluation curves of XTRA on Atari 100k benchmark

F Appendix: Hyper-parameters for XTRA and EfficientZero Model Architecture

We follow the EfficientZero's official architecture implementation to make straight comparison. For EfficientZero-L and XTRA, we increase number of convolutions layers from 1 to 4 to linearly scale up the size of representation networks.

The architecture of the **representation networks** is as follows:

- 1 convolution with stride 2 and 32 output planes, output resolution 48x48. (BN + ReLU)

- 1 residual block with 32 planes.
- 1 residual downsample block with stride 2 and 64 output planes, output resolution 24x24.
- 1 residual block with 64 planes.
- Average pooling with stride 2, output resolution 12x12. (BN + ReLU)
- 1 residual block with 64 planes.
- Average pooling with stride 2, output resolution 6x6. (BN + ReLU)
- 1 residual block with 64 planes.

, where the kernel size is 3×3 for all operations.

The architecture of the **dynamics networks** is as follows:

- Concatenate the input states and input actions into 65 planes.
- 1 convolution with stride 2 and 64 output planes. (BN)
- A residual link: add up the output and the input states. (ReLU)
- 1 residual block with 64 planes.

The architecture of the **reward prediction network** is as follows:

- 1 1x1convolution and 16 output planes. (BN + ReLU)
- Flatten.
- LSTM with 512 hidden size. (BN + ReLU)
- 1 fully connected layers and 32 output dimensions. (BN + ReLU)
- 1 fully connected layers and 601 output dimensions.

The architecture of the **value and policy prediction networks** is as follows:

- 1 residual block with 64 planes.
- 1 1x1convolution and 16 output planes. (BN + ReLU)
- Flatten.
- 1 fully connected layers and 32 output dimensions. (BN + ReLU)
- 1 fully connected layers and D output dimensions.

, where $D = 601$ in the value prediction network and $D = action\ space$ in the policy prediction network.

G APPENDIX: HYPER-PARAMETERS FOR ONLINE FINETUNING XTRA

In this paper, we adopted all hyper-parameters for EfficientZero to make straight comparison except training steps and reanalyzed policy ratio. In the original EfficientZero paper, EfficientZero trains the model for additional 20k steps after collecting 100k environment steps data and decay learning rate by 0.1. We set training steps to 120k as original EfficientZero setting for Table 2 to make direct and fair comparison to the reported number from EfficientZero paper. We set training steps to 100k to match exact 100k environment steps with no additional training for all other results reported in this paper for running both EfficientZero baselines (EfficientZero and EfficientZero-L) and our XTRA models.

Table 4: Hyper-parameters for Online Finetuning XTRA on Atari games

Parameter	Setting
Observation down-sampling	96×96
Frames stacked	4
Frames skip	4
Reward clipping	True
Terminal on loss of life	True
Max frames per episode	108K
Discount factor	0.997^4
Minibatch size	256
Optimizer	SGD
Optimizer: learning rate	0.2
Optimizer: momentum	0.9
Optimizer: weight decay (c)	0.0001
Learning rate schedule	$0.2 \rightarrow 0.02$
Max gradient norm	5
Priority exponent (α)	0.6
Priority correction (β)	$0.4 \rightarrow 1$
Training steps	100K/120K
Evaluation episodes	32
Min replay size for sampling	2000
Self-play network updating inerval	100
Target network updating interval	200
Unroll steps (l_{unroll})	5
TD steps (k)	5
Policy loss coefficient (λ_1)	1
Value loss coefficient (λ_2)	0.25
Self-supervised consistency loss coefficient (λ_3)	2
LSTM horizontal length (ζ)	5
Dirichlet noise ratio (ξ)	0.3
Number of simulations in MCTS (N_{sim})	50
Reanalyzed policy ratio	1.0

H APPENDIX: GAME INFORMATION

We provide information on games appeared in pretraining and finetuning for XTRA. The **Similar Task** column marks all the games used in Table 1 , and the two **Diverse Task** column mark all the games used in Table 2 for pretraining and finetuning the model. We define all the games in the following categories: Maze, Shooter, Tank, Adventure, and Ball Tracking.

In the Maze category, the actor is required to move horizontally or vertically to find a path to (1) pick up objectives to gain scores or special abilities, and (2) avoid being caught by moving enemies. In the Shooter category, the actor is required to shoot horizontally or vertically toward objectives, and at the same time avoid being shot by enemies. In the Tank category, the actor is required to move a fighting vehicle into any directions on a plane, shoot to the enemies appearing on the map and avoid being hit; it is worth mentioning that in this category, the visual observation appears in the First-Person Perspective (FPP), which is different from the visual observation from the shooter category, where the visual observation appears in Third-Person Perspective (TPP) with absolute positioning (e.g. the background is fixed). In the adventure category, the actor is required to perform a diverse set of skills which can differ from game to game; these skills often include shooting, hitting the enemies, solving mazes, and protecting targets, some of which can also be seen from other game categories. In the

Ball Tracking category, the actor is required to keep track of a moving ball, and hit it to prevent the ball from moving off the gaming area.

We also include attributes in terms of **scene continuity** and **action space**. A continuous scene represents that within the game, there will not be sudden changes of a visual observation possibly due to actor moving out of the current gaming zone, leveling up, etc. Action space refers to the number of actions that the actor is able to use for a game. The maximum action space for Atari games is 18.

Table 5: **Information on games relavant to Table 1 and Table 2.** In Table 1, we finetune the model to each game after pretraining it on the other games within the same category. In Table 2, we finetune the model to each game after pretraining it on all eight games.

Games	Similar Task (Pretrain & Fine Tune)	Diverse Task (Pretrain)	Diverse Task (Fine Tune)	Category	Scene Continuity	Action Space
Alien	✓			Maze		18
Amidar	✓			Maze	✓	10
Assault	✓		✓	Shooter	✓	7
Bank Heist	✓			Maze		18
Carnival	✓	✓		Shooter	✓	6
Centipede	✓	✓		Shooter	✓	18
DemonAttack	✓			Shooter	✓	6
MsPacman	✓			Maze		9
Phoenix	✓	✓		Shooter		8
WizardOfWor	✓	✓		Maze		10
BattleZone			✓	Tank	✓	18
Hero			✓	Adventure		18
Krull			✓	Adventure		18
Seaquest			✓	Shooter	✓	18
Pooyan		✓		Shooter		6
Riverraid		✓		Shooter	✓	18
VideoPinball		✓		Ball Tracking		9
YarsRevenge		✓		Shooter		18

I APPENDIX: GAME VISUALIZATIONS

To better understand the feasibility of cross-task transfer, we show sample trajectory from our experiment games. Figure 12 and 13 include all games involved in Table 1 (tasks sharing similar game mechanics). Figure 14 include pretraining/finetuning games involved in Table 2 (tasks sharing diverse game mechanics) excluding ones that are shown in Figure 12 and Figure 13.

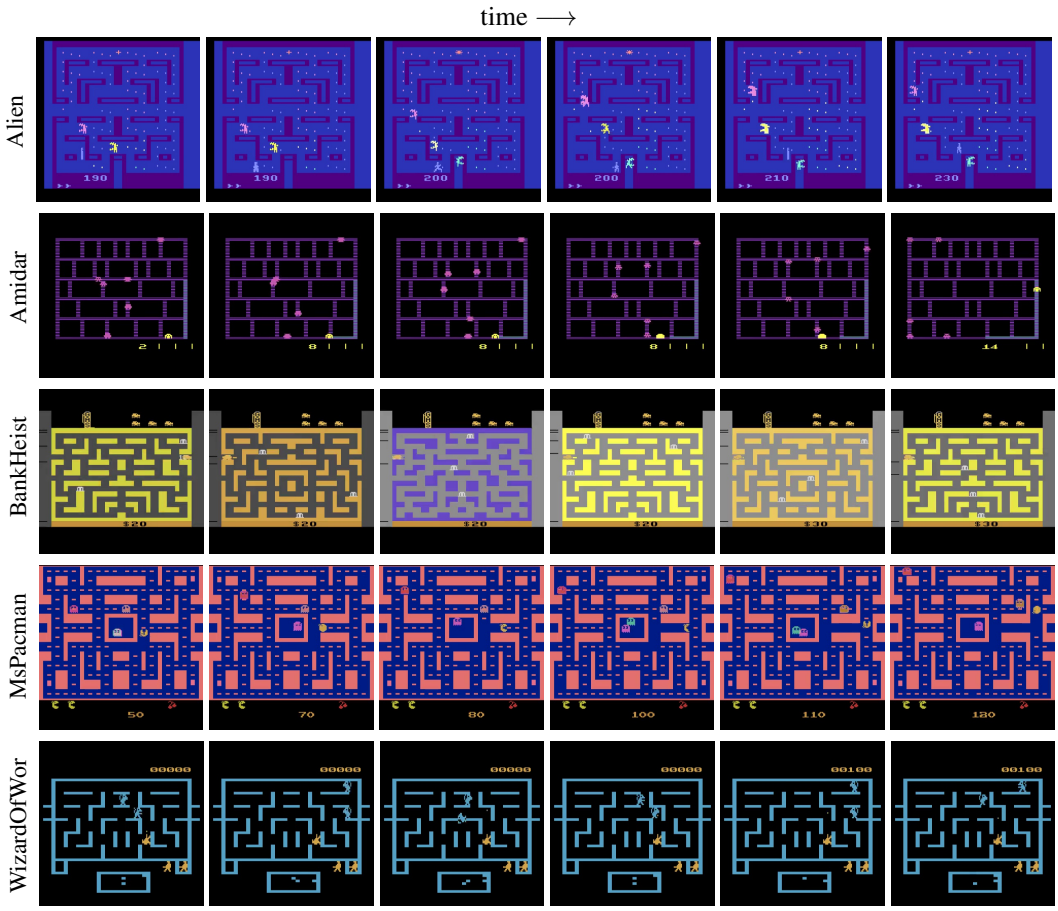

Figure 12: **Visualizations of maze games.**

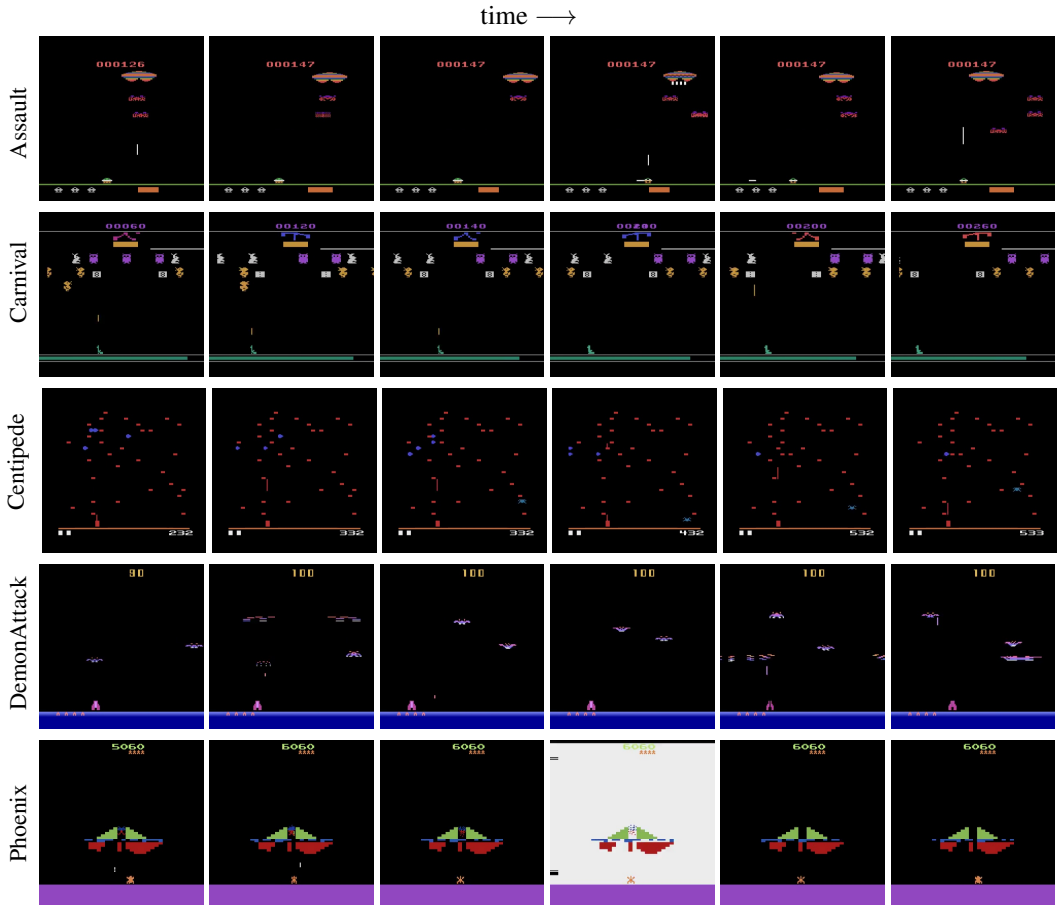

Figure 13: **Visualizations of shooter games.**

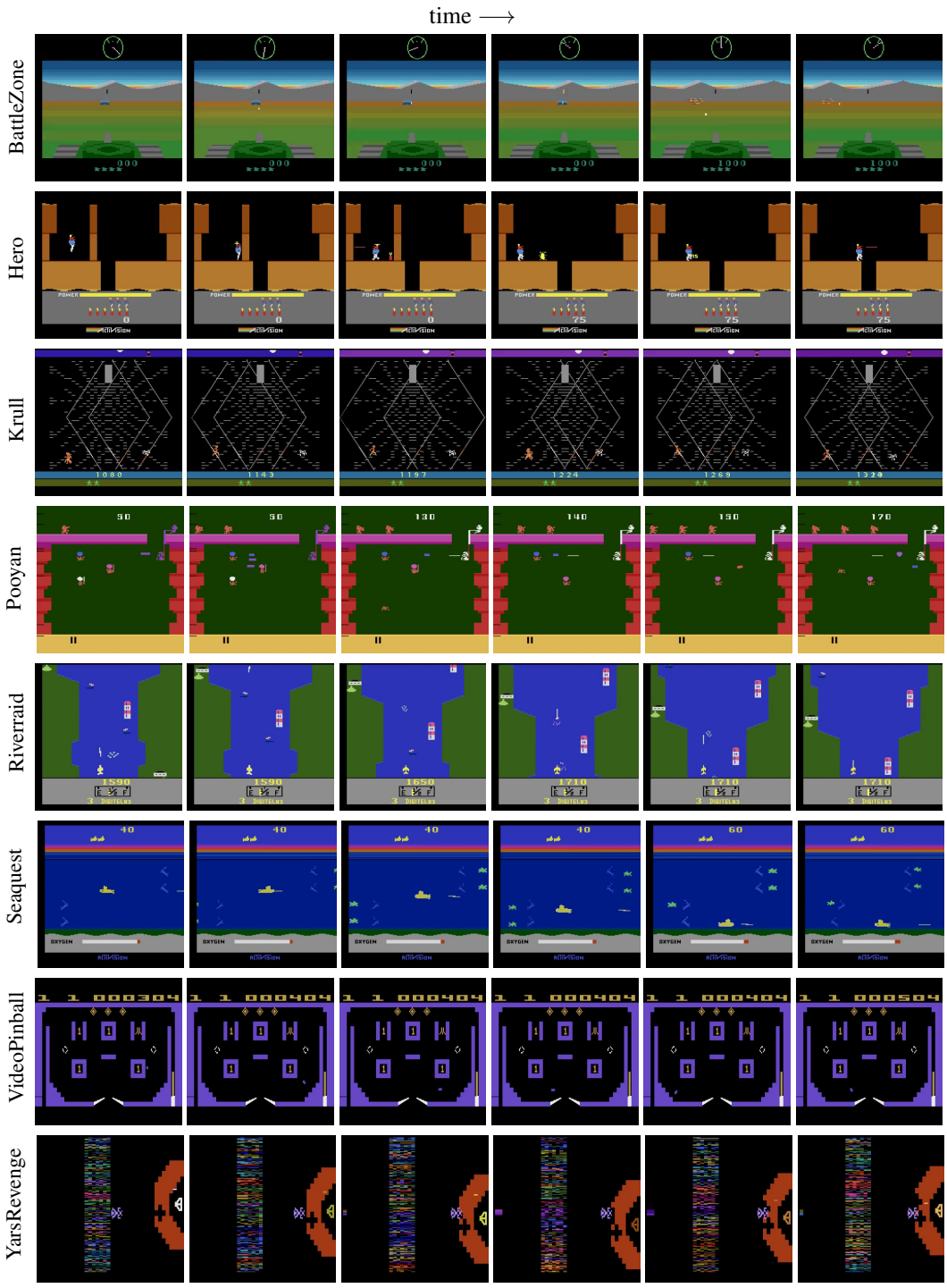

Figure 14: **Visualizations of games from diverse categories.**