Supplementary Material: Action-Conditional Video Prediction using Deep Networks in Atari Games

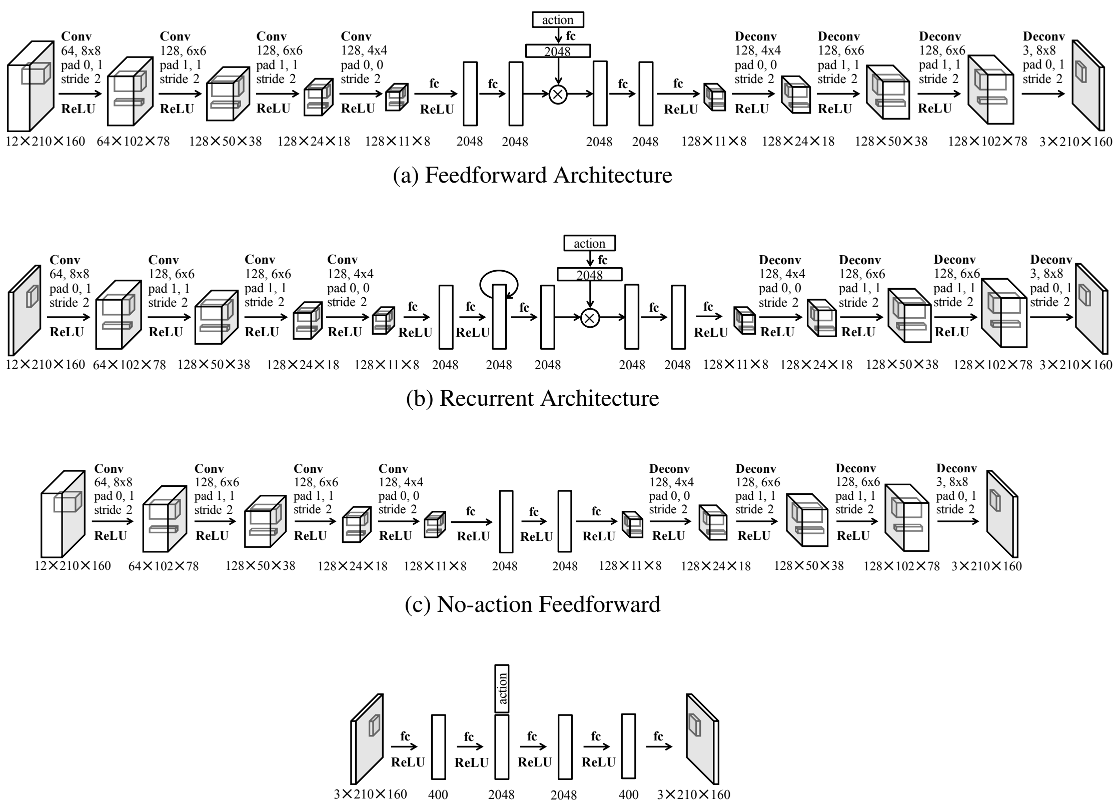
Junhyuk Oh **Xiaoxiao Guo** **Honglak Lee** **Richard Lewis** **Satinder Singh**
University of Michigan, Ann Arbor, MI 48109, USA
{junhyuk,guoxiao,honglak,rickl,baveja}@umich.edu

A Network Architectures and Training Details

The network architectures of the proposed models and the baselines are illustrated in Figure 1.

The weight of LSTM is initialized from a uniform distribution of $[-0.08, 0.08]$. The weight of the fully-connected layer from the encoded feature to the factored layer and from the action to the factored layer are initialized from a uniform distribution of $[-1, 1]$ and $[-0.1, 0.1]$ respectively.

The total number of iterations is 1.5×10^6, 10^6, and 10^6 for each training phase (1-step, 3-step, and 5-step). The learning rate is multiplied by 0.9 after every 10^5 iterations.



(a) Feedforward Architecture

(b) Recurrent Architecture

(c) No-action Feedforward

(d) MLP

Figure 1: Network architectures. '×' indicates element-wise multiplication. The text in each (de-)convolution layer describes the number of filters, the size of the kernel, padding (height and width), and stride.

B Informed Exploration

The entire DQN algorithm with informed exploration is described in Algorithm 1.

Algorithm 1 Deep Q-learning with informed exploration

Allocate capacity of replay memory R
Allocate capacity of trajectory memory D
Initialize parameters θ of DQN
while $steps < M$ **do**
  Reset game and observe image x_1
  Store image x_1 **in** D
  for t=1 to T **do**
    Sample c from Bernoulli distribution with parameter ϵ
    Set $a_t = \begin{cases} \textbf{argmin}_a \, n_D\left(x_t^{(a)}\right) & \text{if } c = 1 \\ \text{argmax}_a \, Q\left(\phi\left(s_t\right), a; \theta\right)\right) & \text{otherwise} \end{cases}$
    Choose action a_t, observe reward r_t and image x_{t+1}
    Set $s_{t+1} = x_{t-2:t+1}$ and preprocess images $\phi_{t+1} = \phi\left(s_{t+1}\right)$
    Store image x_{t+1} **in** D
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in R
    Sample a mini-batch of transitions $\{\phi_j, a_j, r_j, \phi_{j+1}\}$ from R
    Update θ based on the mini-batch and Bellman equation
    $steps = steps + 1$
  end for
end while

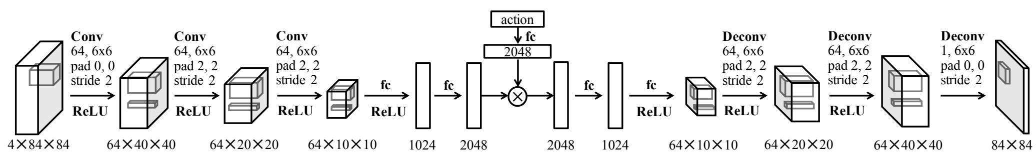
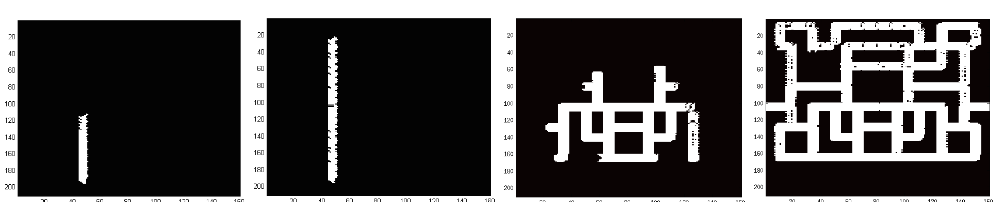


Figure 2: Feedforward encoding network for gray-scaled and down-sampled images.

Predictive Model for Informed Exploration. A feedforward encoding network (illustrated in Figure 2) trained on down-sampled and gray-scaled images is used for computational efficiency. We trained the model on 1-step prediction objective with learning rate of 10^{-4} and batch size of 32. The pixel values are subtracted by mean pixel values and divided by 128. RMSProp is used with momentum of 0.9, (squared) gradient momentum of 0.95, and min squared gradient of 0.01.

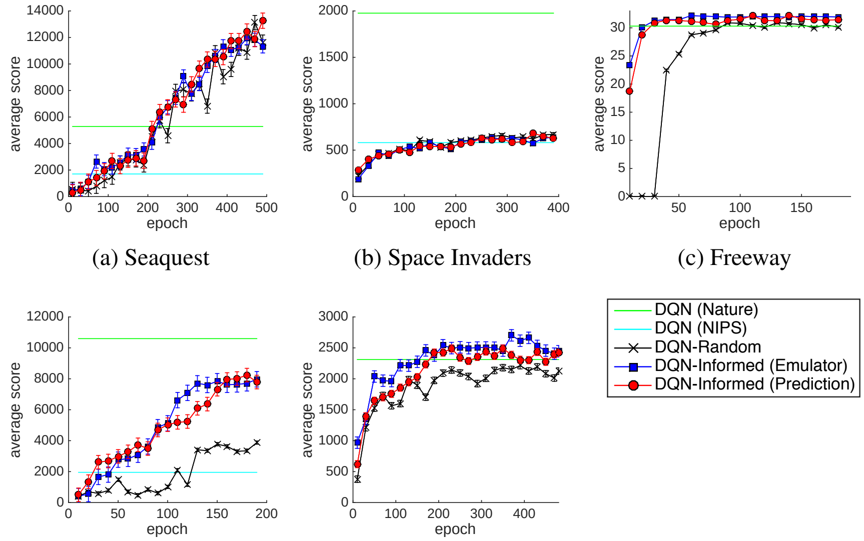
Comparison to Random Exploration. Figure 3 visualizes the difference between random exploration and informed exploration in two games. In Freeway, where the agent gets rewards by reaching the top lane, the agent moves only around the bottom area in the random exploration, which results in 4.6×10^5 steps to get the first reward. On the other hand, the agent moves around all locations in the informed exploration and receives the first reward in 86 steps. The similar result is found in Ms Pacman.

Application to Deep Q-learning. The results of the informed exploration using the game emulator and our predictive model are reported in Figure 4 and Table 1. Our DQN replication follows [1], which uses a smaller CNN than [2].



(a) Random (b) Informed (c) Random (d) Informed

Figure 3: Comparison between two exploration methods on Freeway (Left) and Ms Pacman (Right). Each heat map shows the trajectories of the agent measured from 2500 steps from each exploration strategy.



(a) Seaquest (b) Space Invaders (c) Freeway

(d) QBert (e) Ms Pacman

Figure 4: Learning curves of DQNs with standard errors. The red and blue curves are informed exploration using our predictive model and the emulator respectively. The black curves are DQNs with random exploration. The average game score is measured from 100 game plays with ϵ-greedy policy with $\epsilon = 0.05$.

Model	Seaquest	S. Invaders	Freeway	QBert	Ms Pacman
DQN (Nature) [2]	5286	1976	30.3	10596	2311
DQN (NIPS) [1]	1705	581	-	1952	-
Our replication of [1]	13119 (538)	698 (20)	30.9 (0.2)	3876 (106)	2281 (53)
I.E (Prediction)	13265 (577)	681 (23)	32.2 (0.2)	8238 (498)	2522 (57)
I.E (Emulator)	13002 (498)	708 (17)	32.2 (0.2)	7969 (496)	2702 (92)

Table 1: Average game score with standard error. 'I.E' indicates DQN combined with the informed exploration method. 'Emulator' and 'Prediction' correspond to the emulator and our predictive model for computing $\mathbf{x}_t^{(a)}$.

C Correlation between Actions

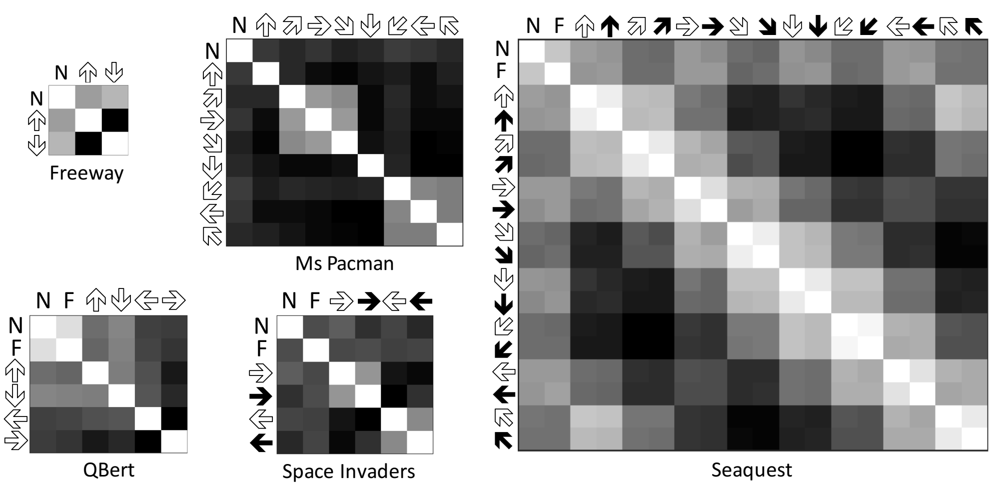
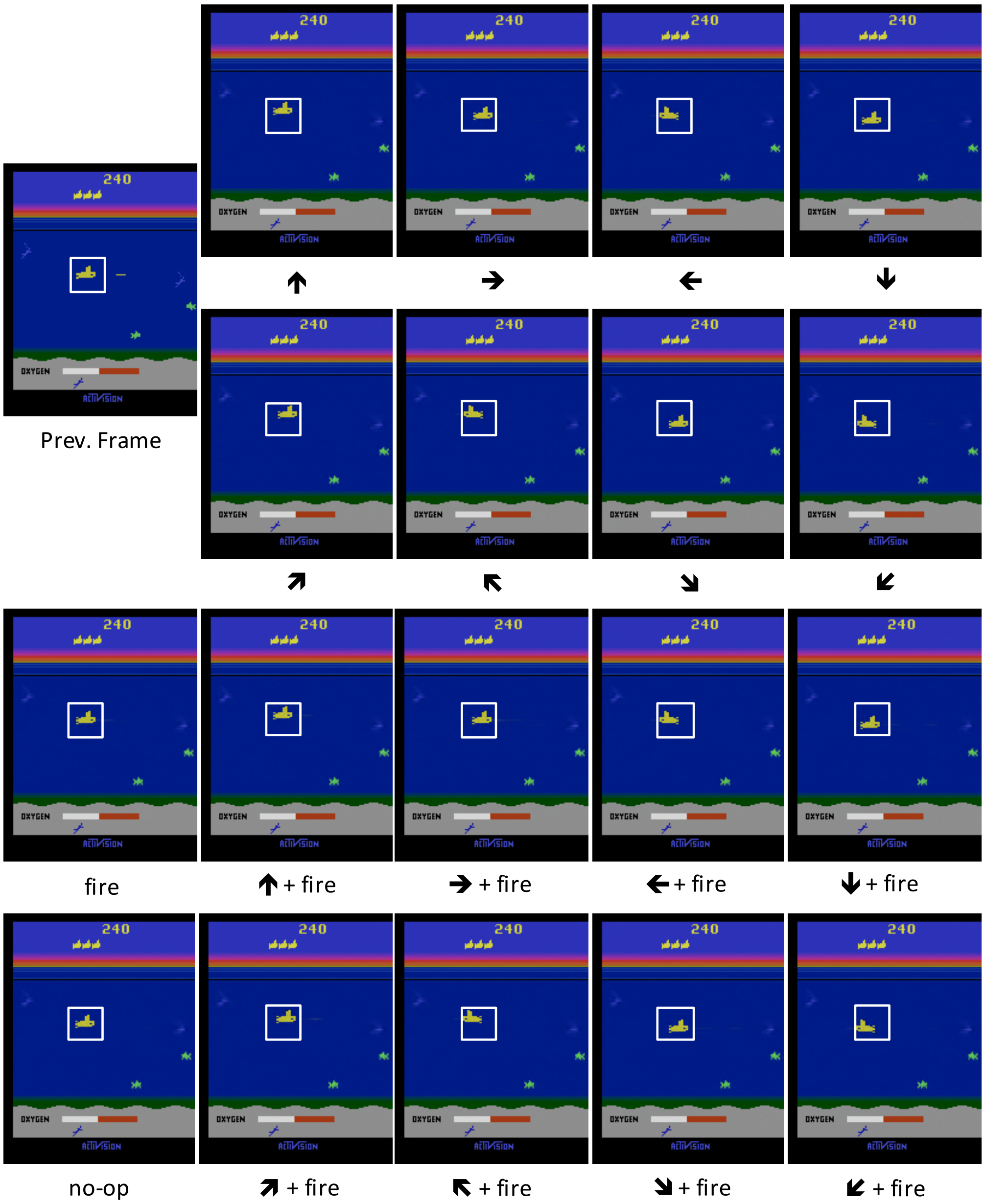
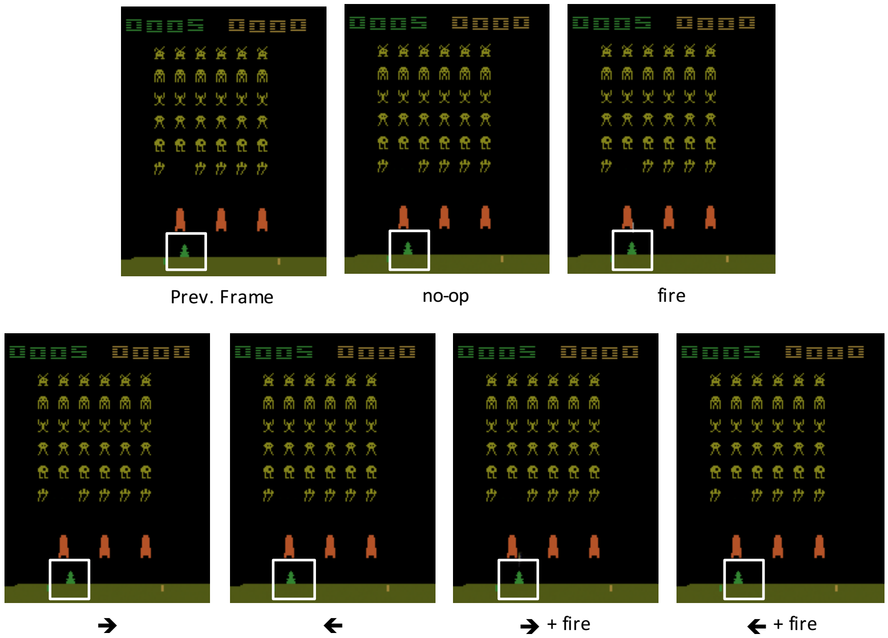


Figure 5: Correlations between actions. The brightness represents consine similarity between pairs of factors.

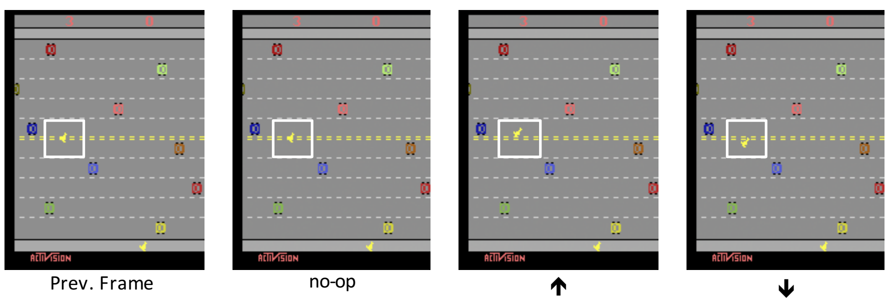
D Handling Different Actions



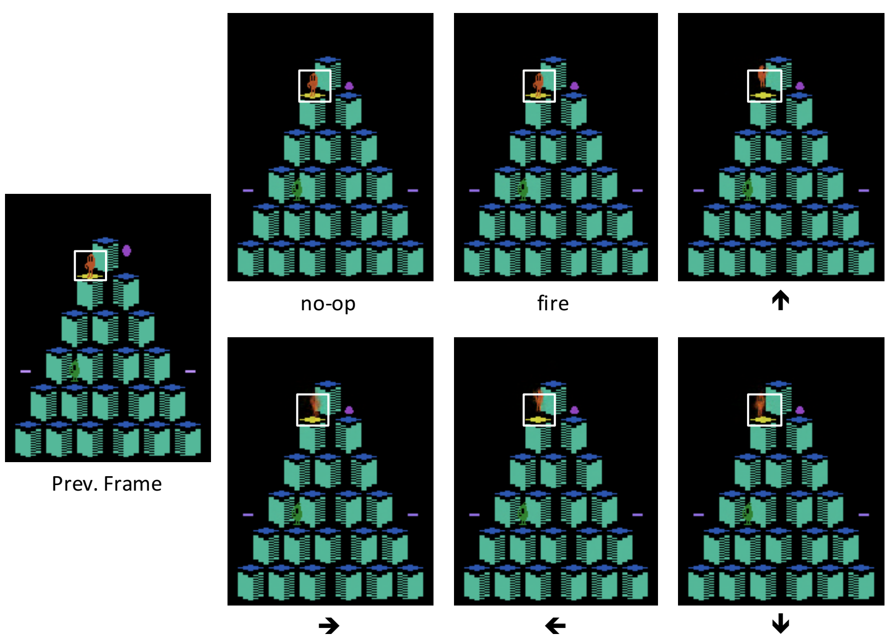
(a) Seaquest



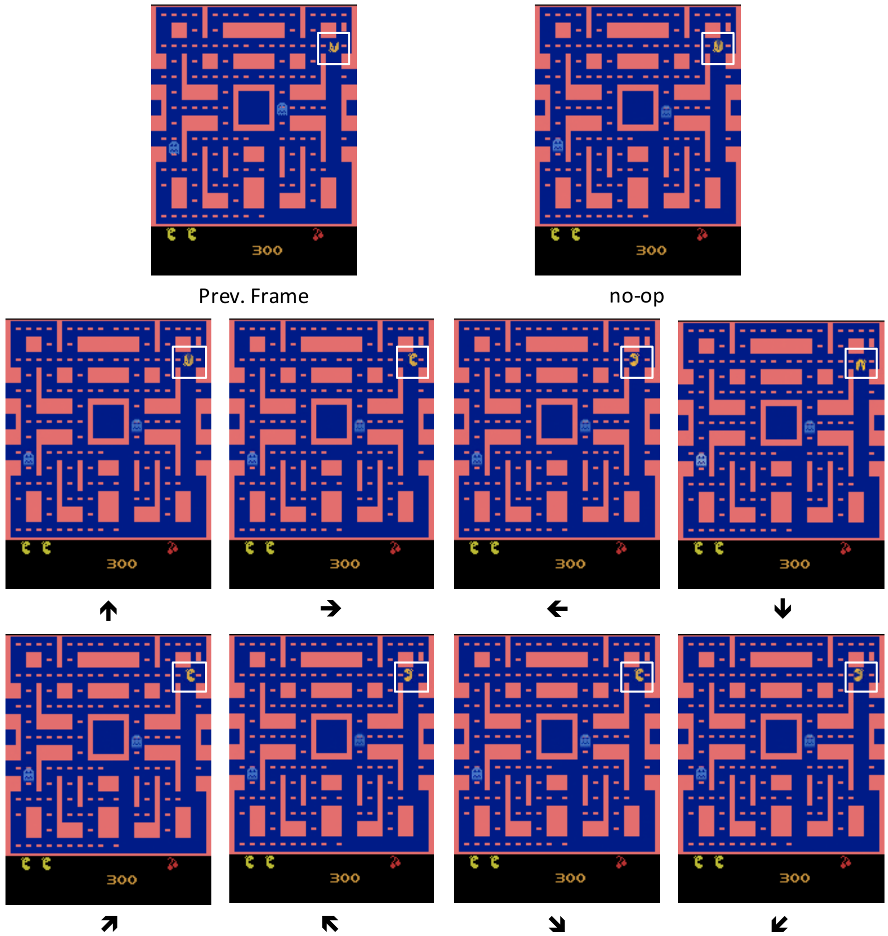
(b) Space Invaders



(c) Freeway



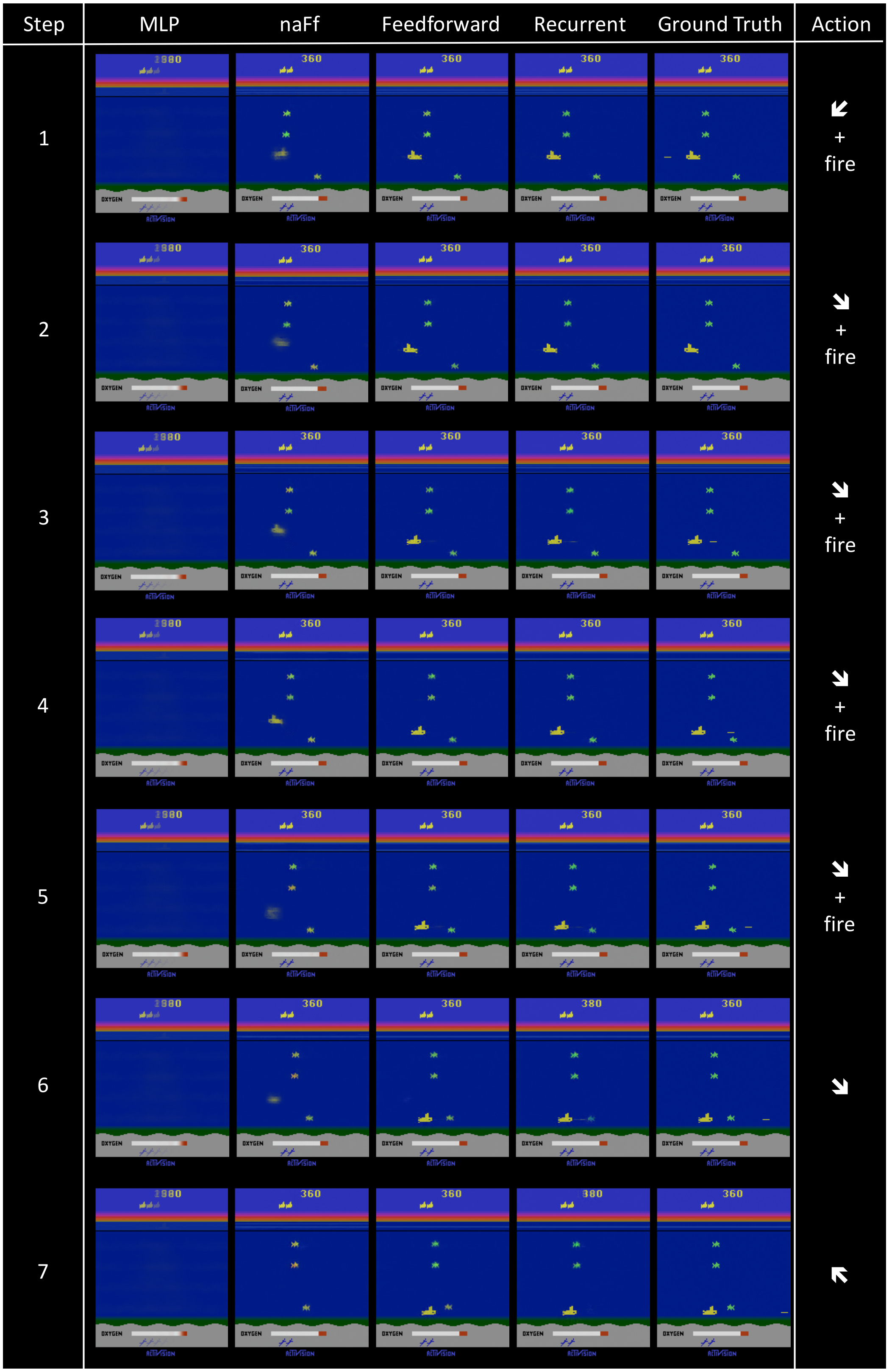
(d) QBert



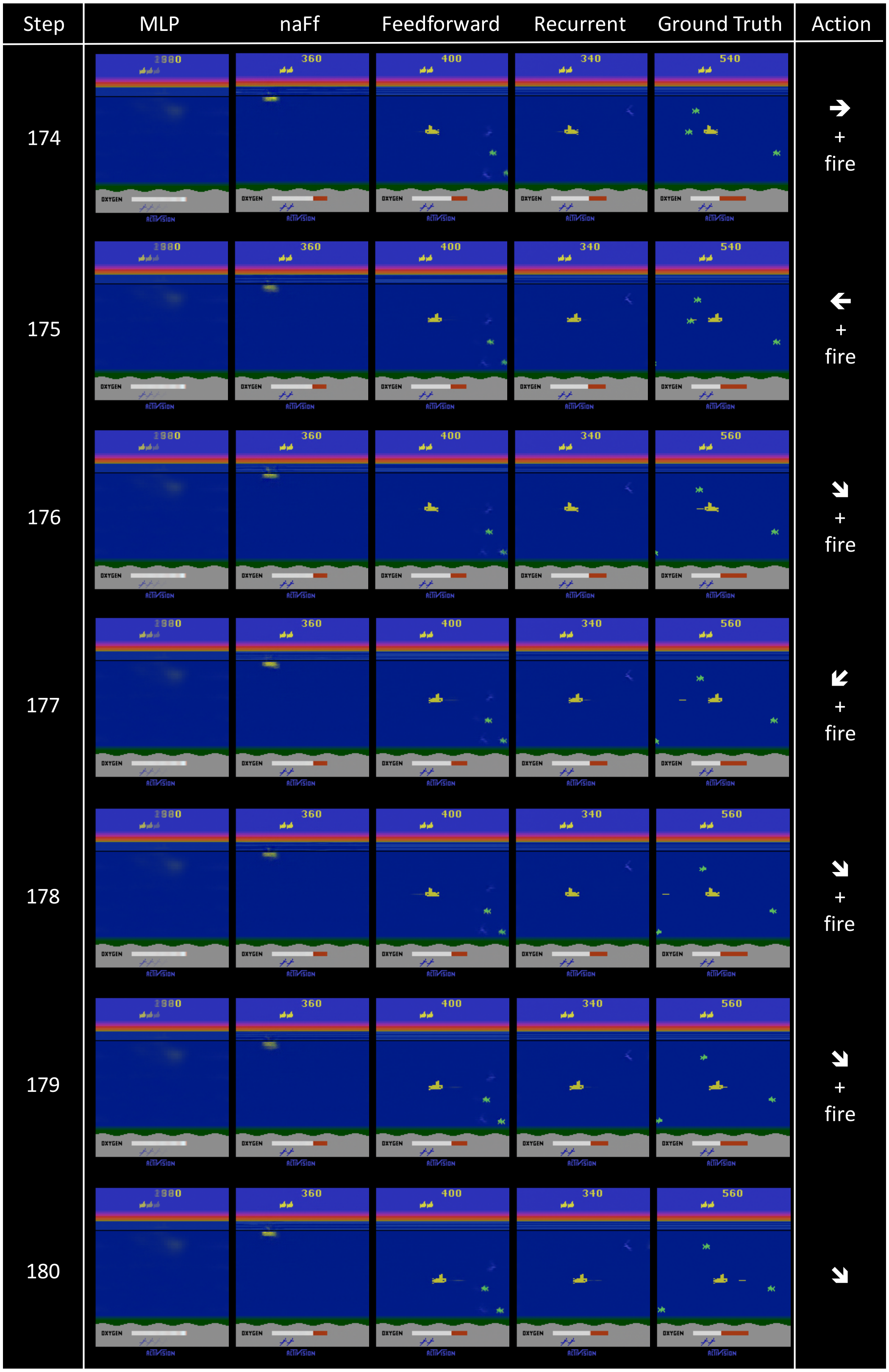
(e) Ms Pacman

Figure 6: Predictions given different actions

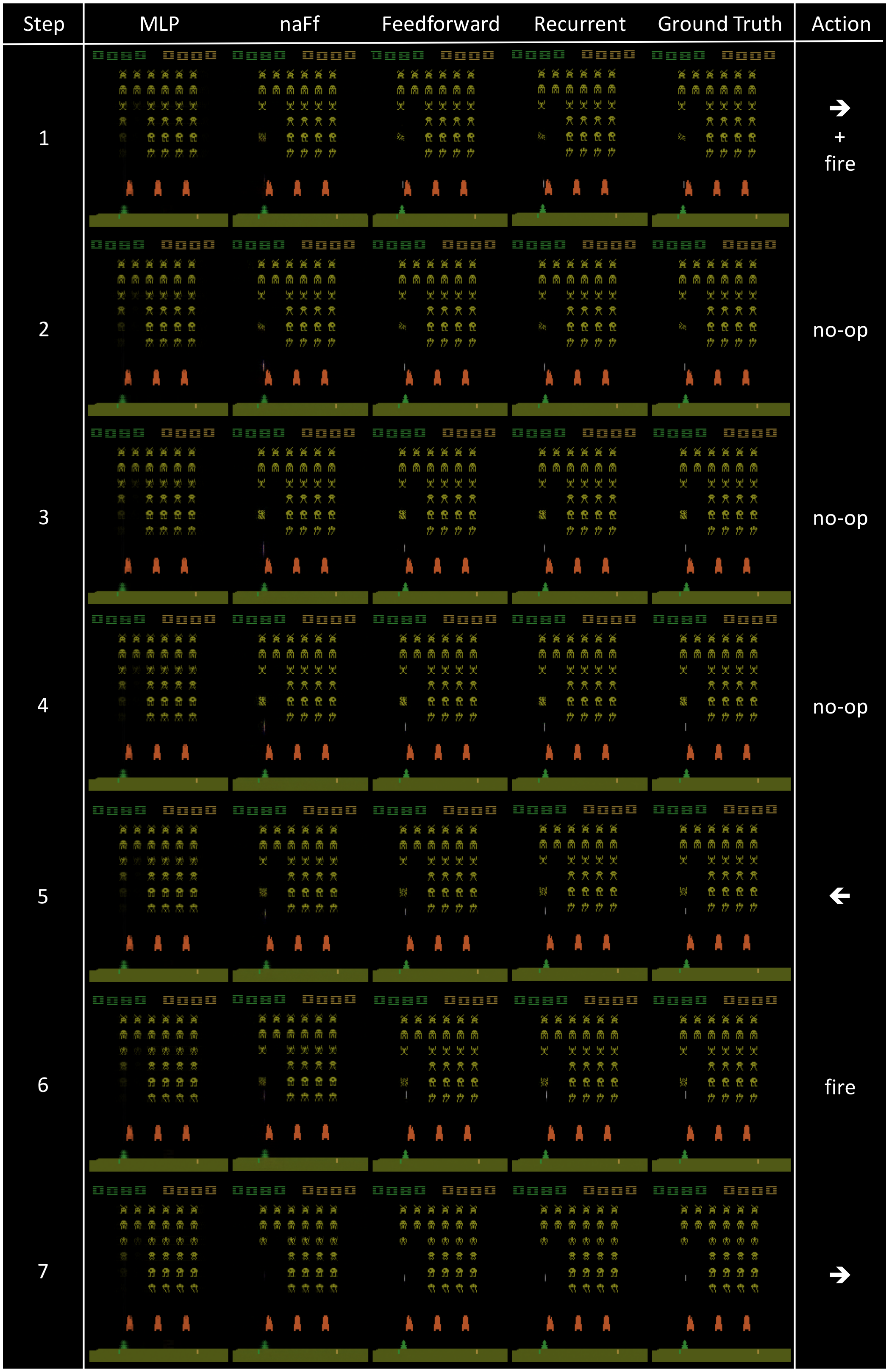
E Prediction Video



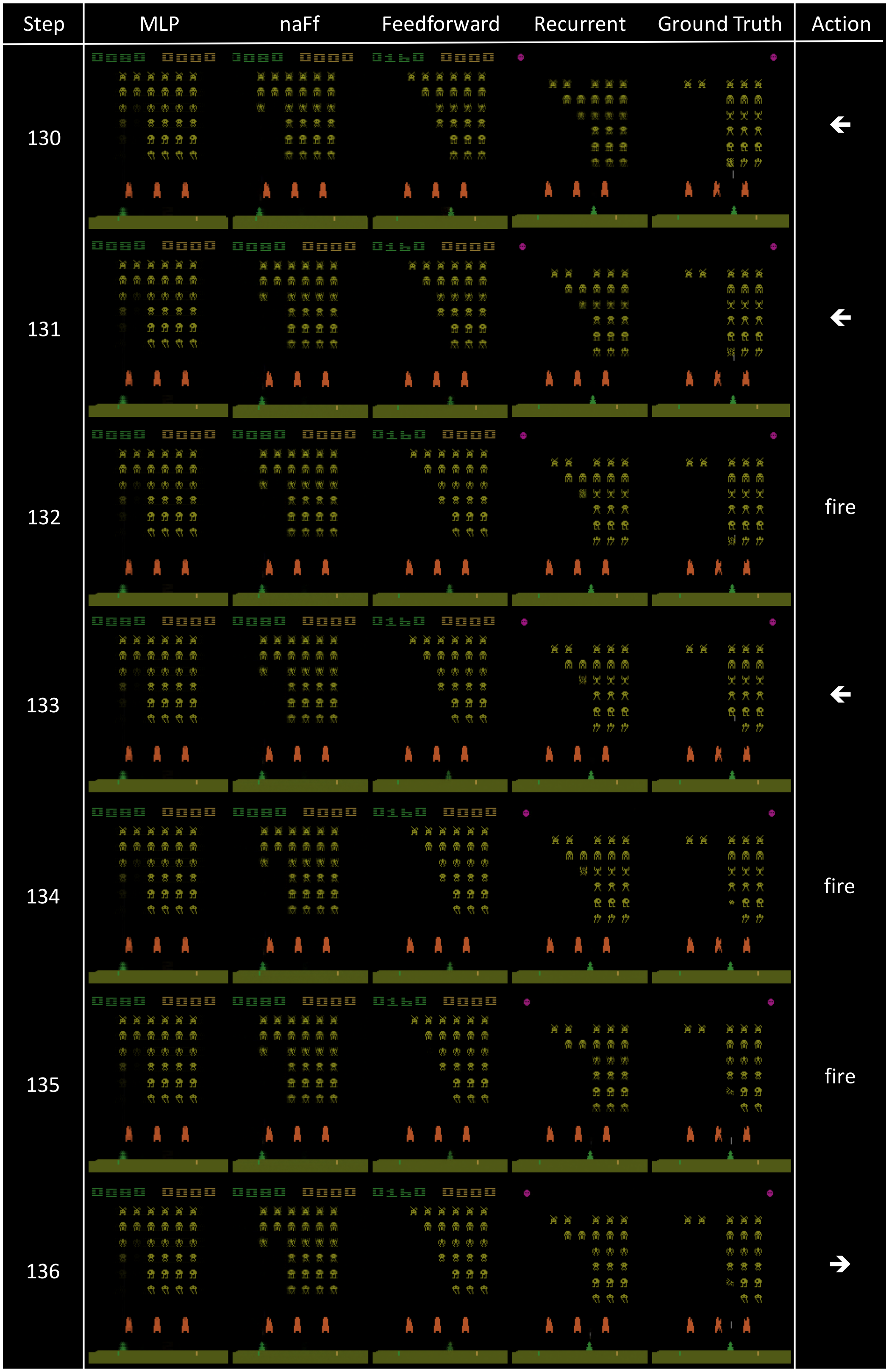
(a) Seaquest (1 ∼ 7 steps). Our models predict the movement of the enemies and the yellow submarine which is controlled by actions. 'naFf' predicts only the movement of other objects correctly, and the submarine disappears after a few steps. 'MLP' does not predict any objects but generate only the mean pixel image.



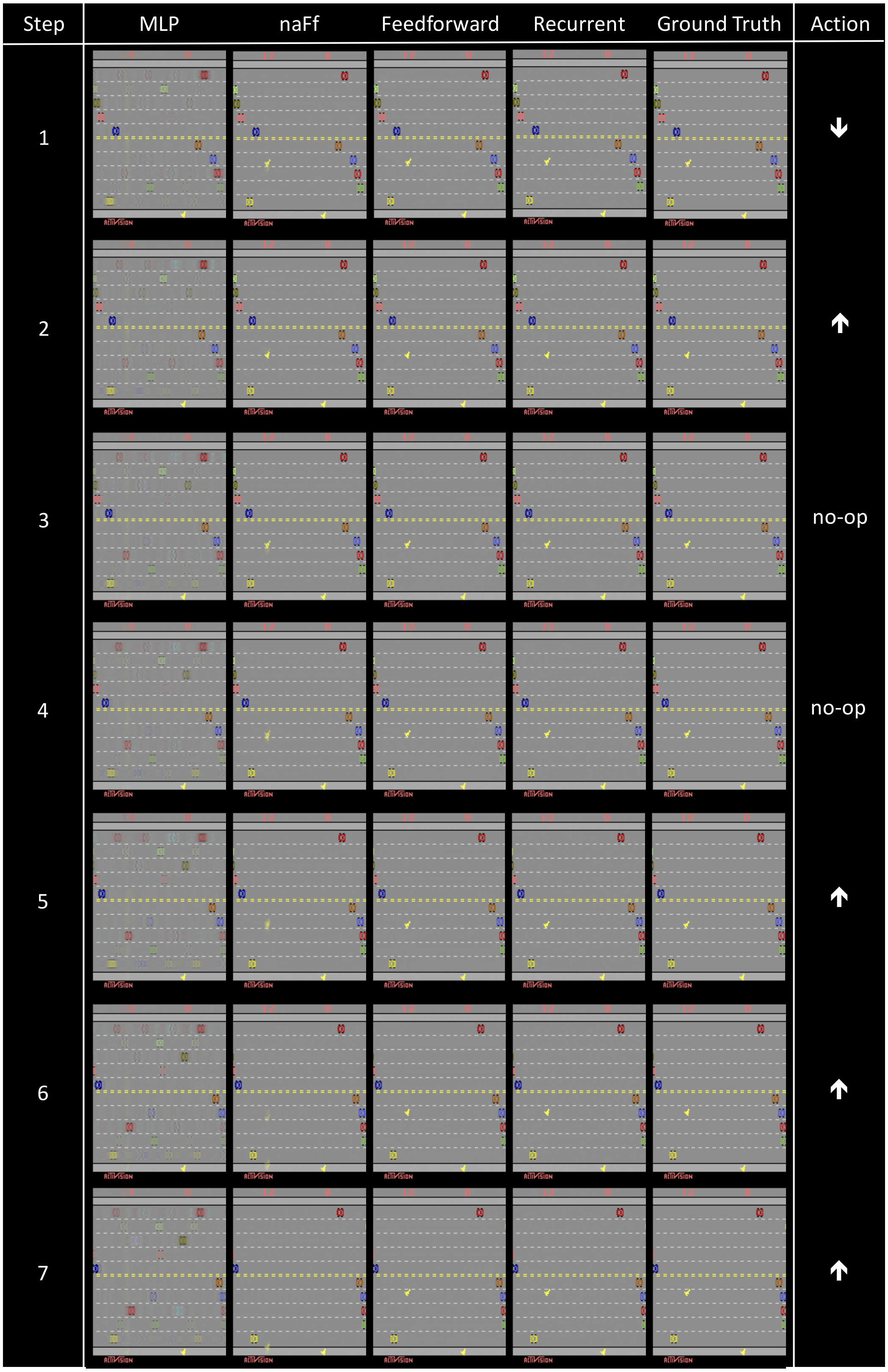
(b) Seaquest (174 $\sim$ 180 steps). The proposed models predict the location of the controlled object accurately over 180-step predictions. They generate new objects such as fishes and human divers. Although the generated objects do not match the ground-truth images, their shapes and colors are realistic.



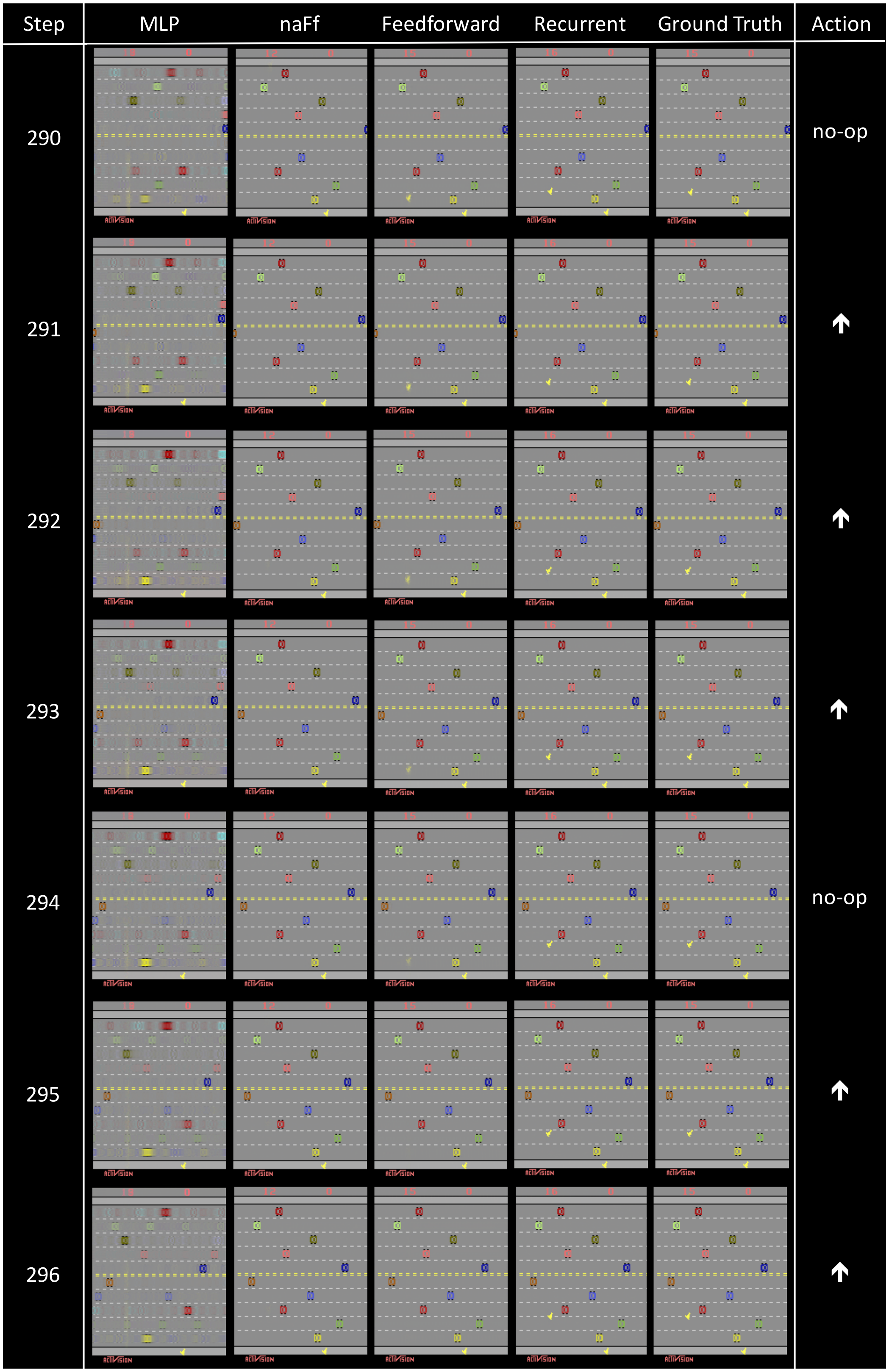
(c) Space Invaders (1 $\sim$ 7 steps). The enemies in the center move and change their shapes from step 6 to step 7. This movement is predicted by the proposed models and 'naFf', while the predictions from 'MLP' are almost same as the last input frame.



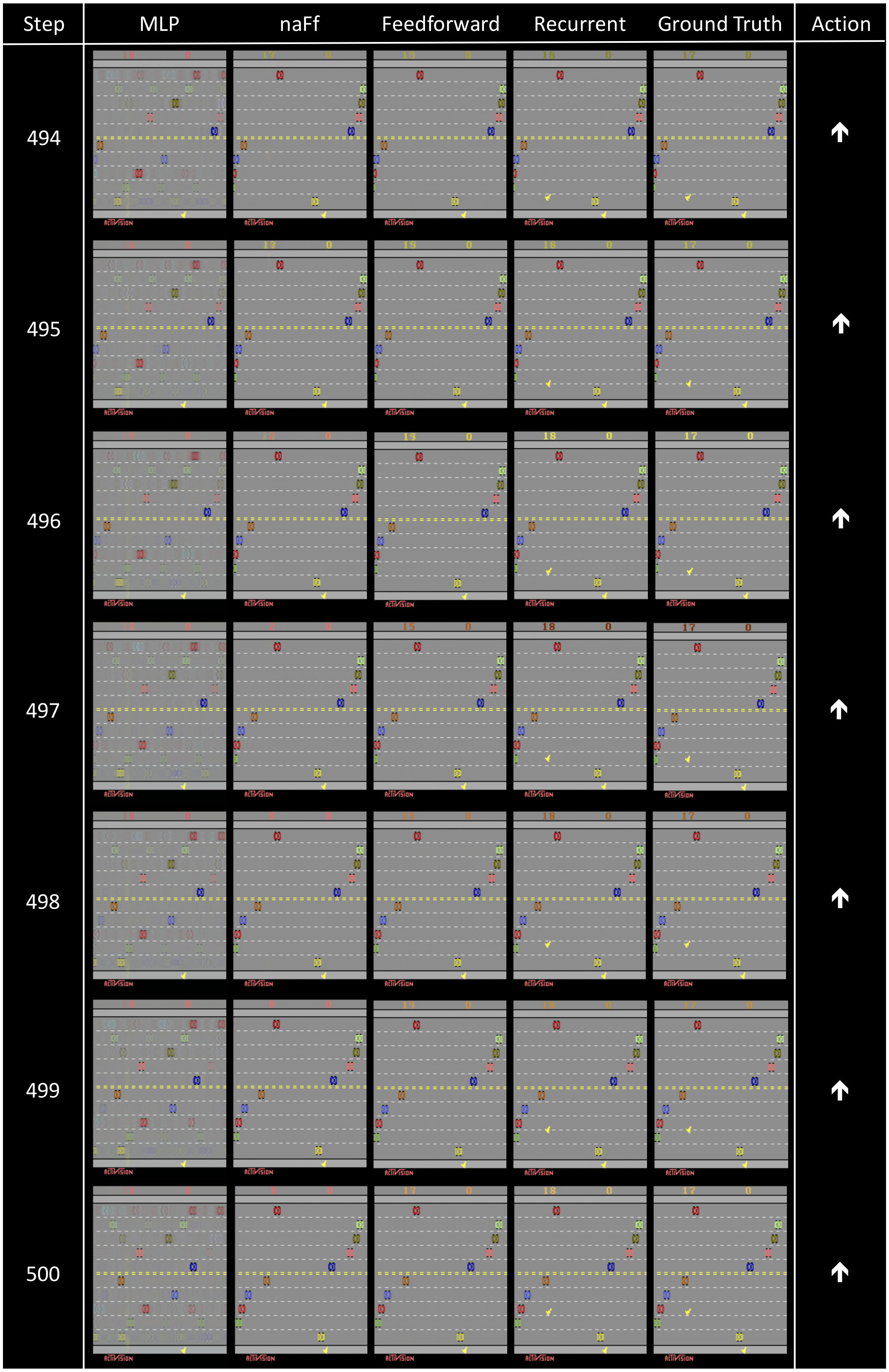
(d) Space Invaders (130 $\sim$ 136 steps). Although our models make errors in the long run, the generated images are still realistic in that the objects are reasonably arranged and moving in the right directions. On the other hand, the frames predicted by 'MLP' and 'naFf' are almost same as the last input frame.



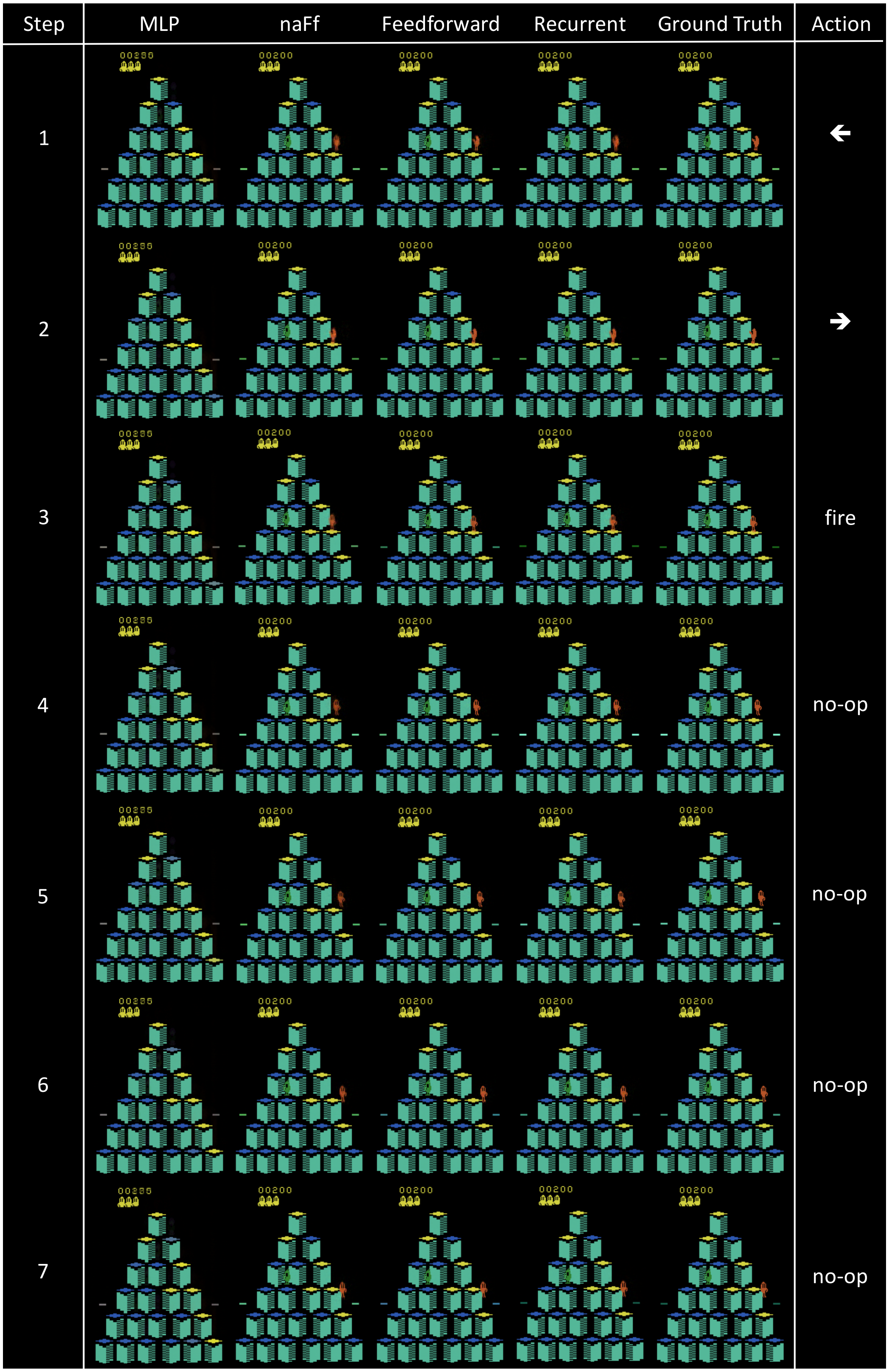
(e) Freeway (1 $\sim$ 7 steps). The proposed models predict the movement of the controlled object correctly depending on different actions, while 'naFf' fails to handle different actions. 'MLP' generates blurry objects that are not realistic.



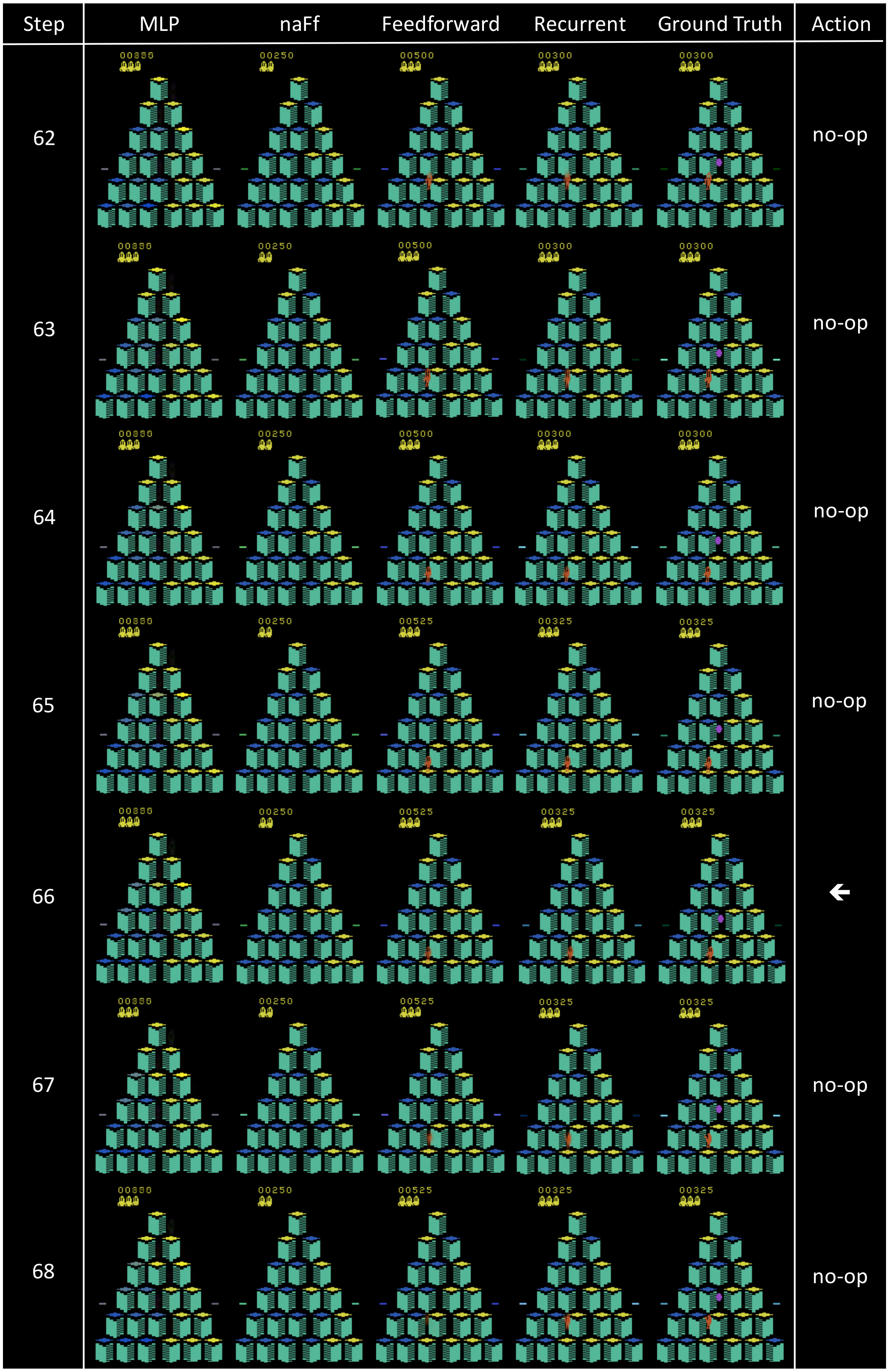
(f) Freeway (290 $\sim$ 296 steps). The feedforward network diverges at 294-step as the agent starts a new stage from the bottom lane. This is due to the fact that actions are ignored for 9-steps when a new stage begins, which is not successfully handled by the feedforward network.



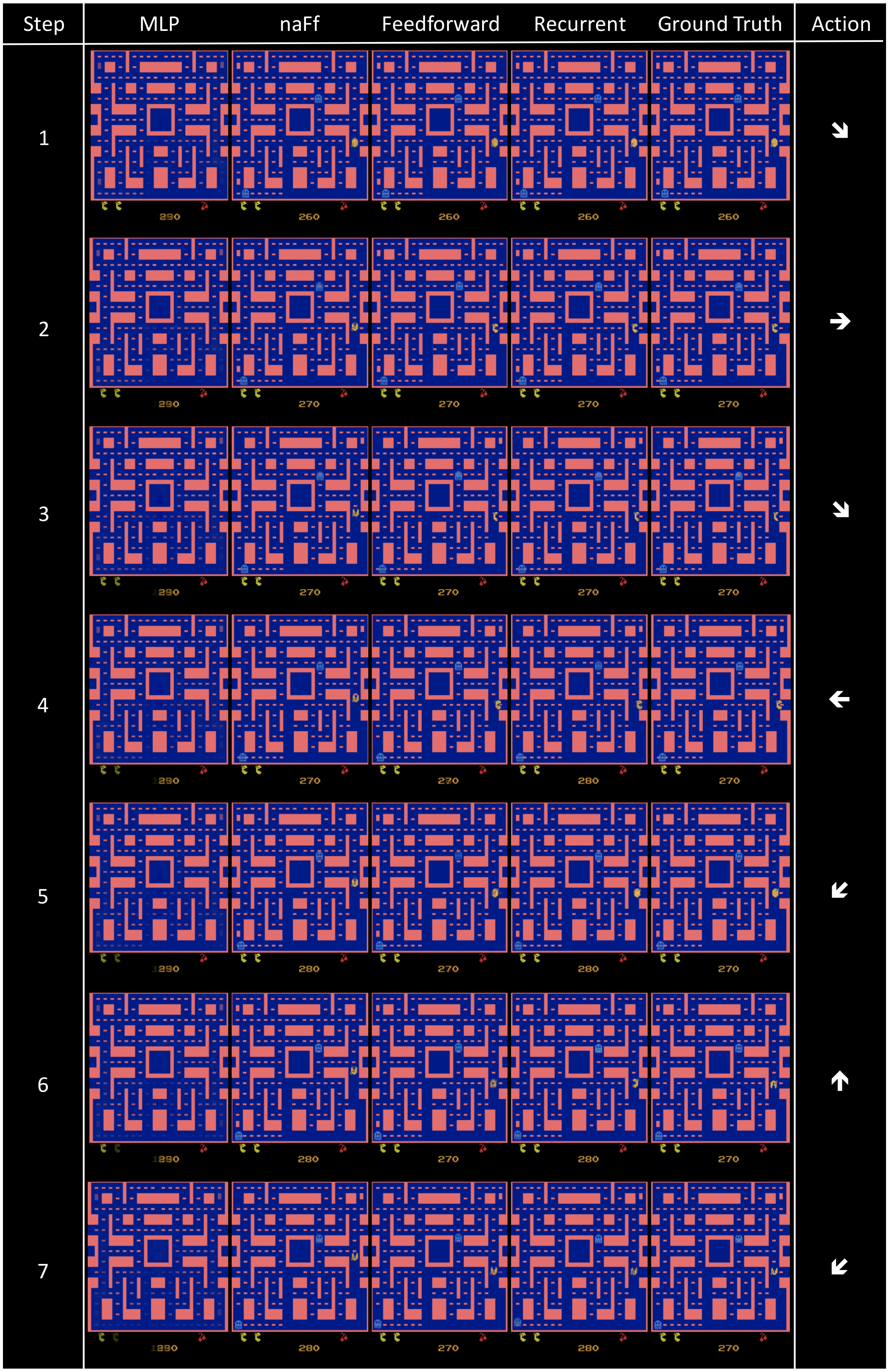
(g) Freeway (494 $\sim$ 500 steps). The recurrent encoding model keeps track of every object over 500 steps.



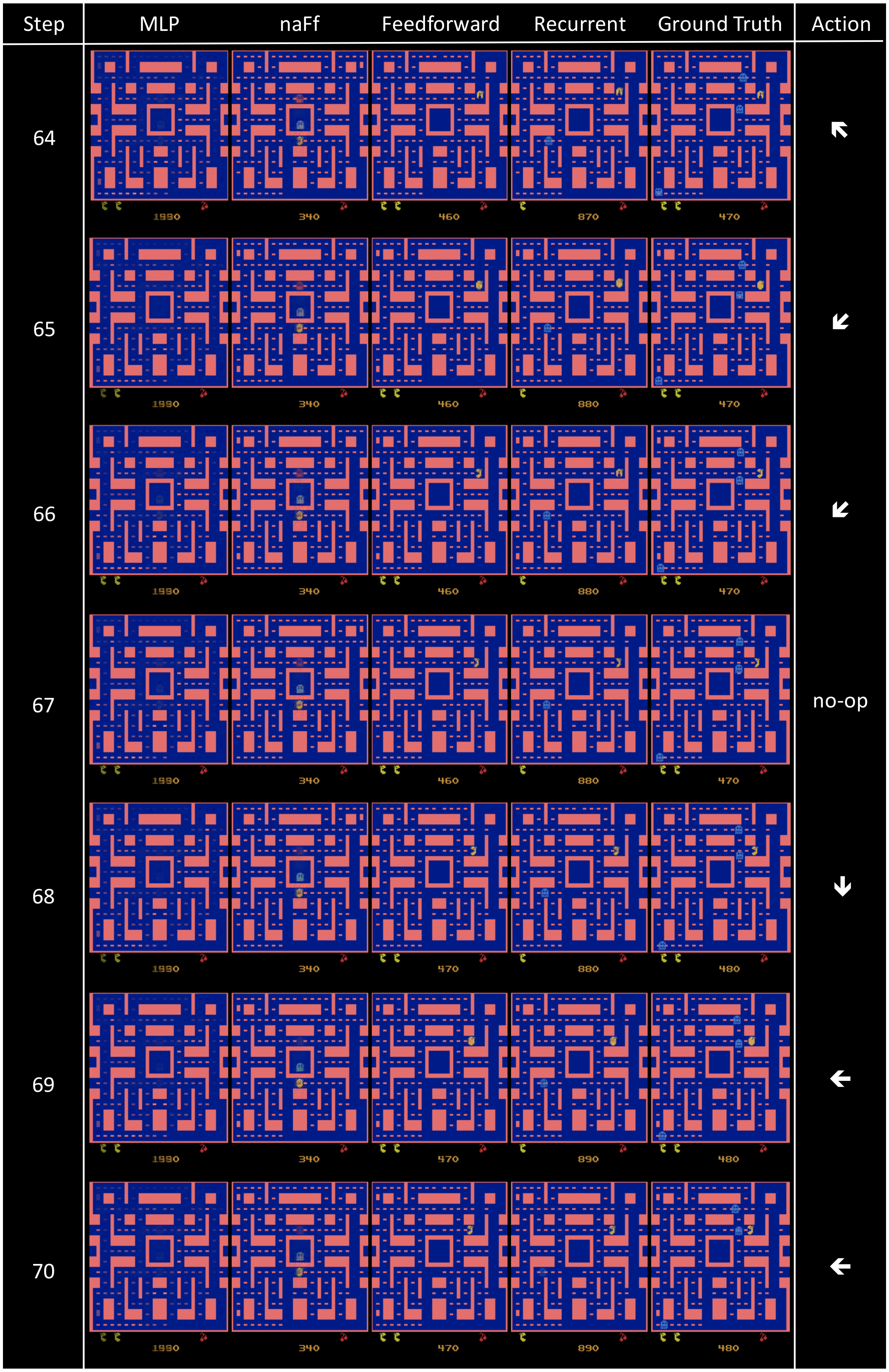
(h) QBert (1 $\sim$ 7 steps). The controlled object jumps from the third row to the fourth row. In the meantime (jumping), the actions chosen by the agent do not have any effects. Our models and 'naFf' predicts this movement, whereas 'MLP' does not predict any objects.



(i) QBert (62 $\sim$ 68 steps). The recurrent model predicts the controlled object and the color of the cubes correctly, while the feedforward model diverges at 68-step as it predicts a blurry controlled object at 66-step. The baselines diverged before 62-step.



(j) Ms Pacman (1 $\sim$ 7 steps). The proposed models predict different movements of Pacman depending on different actions, whereas 'naFf' ignores the actions. 'MLP' predicts the mean pixel image.



(k) Ms Pacman (64 $\sim$ 70 steps). Our models keep track of Pacman, while they fail to predict the other objects that move almost randomly.

References

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

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.