

1 A Prediction & reconstruction - benchmark results

2 In this Appendix we report the evaluations for each environment we tested. We report the results
 3 following the same template as Figure ??, ?? and ?. Additionally, we discuss each result individually,
 4 to better highlight environment-related strengths and weaknesses of each approach.

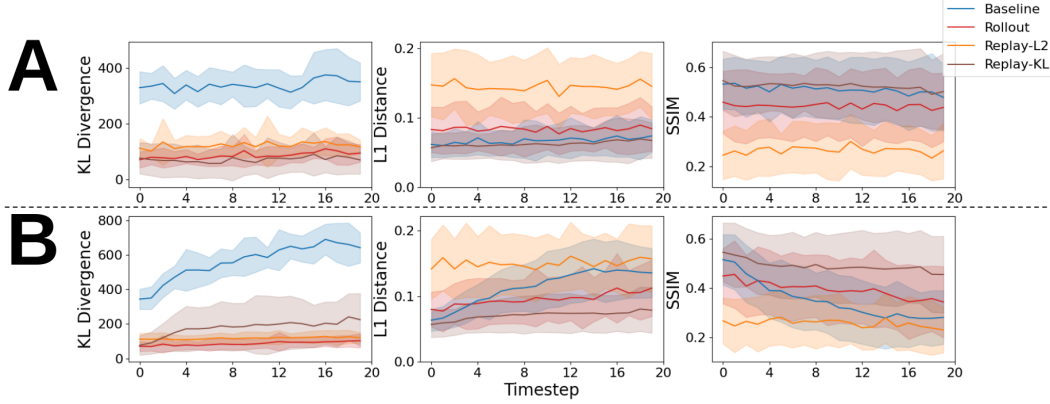


Figure 1: Numerical evaluation in SuperTuxKart "fortmagma" track. Results are reported in terms of KL divergence for latent dynamics prediction, and L1 distance & SSIM index for (A) one-step and (B) long-horizon predictions.

5 Results for the "fortmagma" (Figure 1) reflect the general behavior we have discussed in Section ??:
 6 all our models achieve significantly lower error in KL divergence, meaning that they are closer to the
 7 true dynamics in latent space. As for the reconstruction, which we remark could be affected also by
 8 an error in decoding the latent, Rollout and Replay-KL show comparable reconstructions with the
 9 baseline, while Replay-L2 is generally affected by hallucinations or inconsistencies.

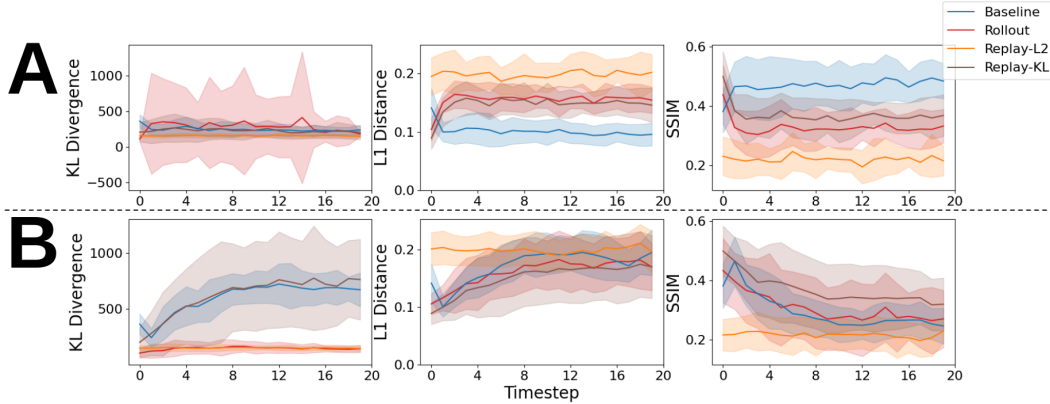


Figure 2: Numerical evaluation in SuperTuxKart "snes_rainbowroad" track. Results are reported in terms of KL divergence for latent dynamics prediction, and L1 distance & SSIM index for (A) one-step and (B) long-horizon predictions.

10 The results of our benchmark on "snes_rainbowroad", reported in Figure 2, show the importance
 11 of interpreting the quantitative and qualitative results jointly. The general behavior of the models
 12 is quite different from Figure 1: our models Rollout and Replay-KL perform either similarly (KL
 13 divergence) or slightly worse (L1, SSIM) than the baseline for one-step predictions, while match or
 14 improve in long-term dynamics prediction.

15 We attribute this difference to the visual complexity of the images from this specific environment, as
 16 shown in Appendix B: the sequence of colors in the track, along with the "falls" that the player might

17 experience heavily contribute to the divergence from the real sequence. However, we point out how
 18 our models are generally more consistent in reconstructing visually coherent sequences, even though
 19 they might be slightly more distant in terms of visual accuracy.

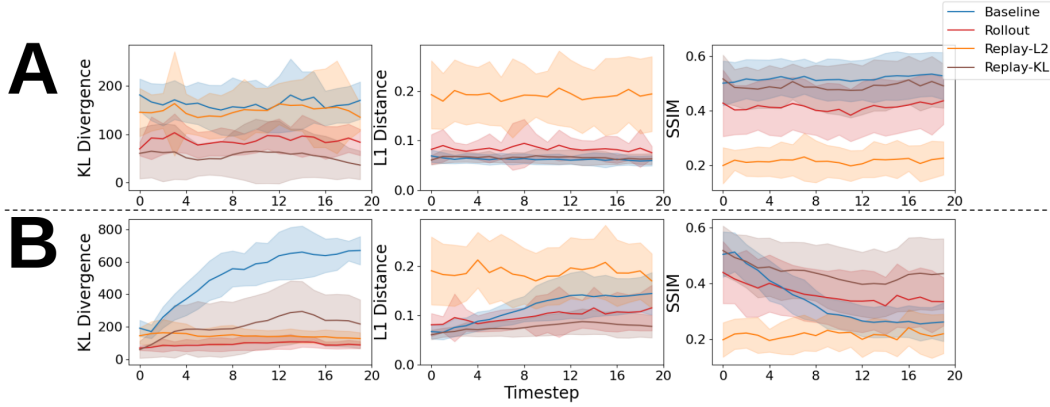


Figure 3: Numerical evaluation in SuperTuxKart "volcano_island" track. Results are reported in terms of KL divergence for latent dynamics prediction, and L1 distance & SSIM index for (A) one-step and (B) long-horizon predictions.

20 The results from track "volcano_island" (Figure 3) support the general results we have shown in the
 21 main text. Our agents Rollout and Replay-KL are generally closer to the real sequence in terms of
 22 latent dynamics prediction, while either match or slightly improve the visual reconstructions. Notably,
 23 the gap in difference is more pronounced in long-horizon predictions, where Rollout and Replay-KL
 24 diverge from the real sequence much more slowly than the baseline.

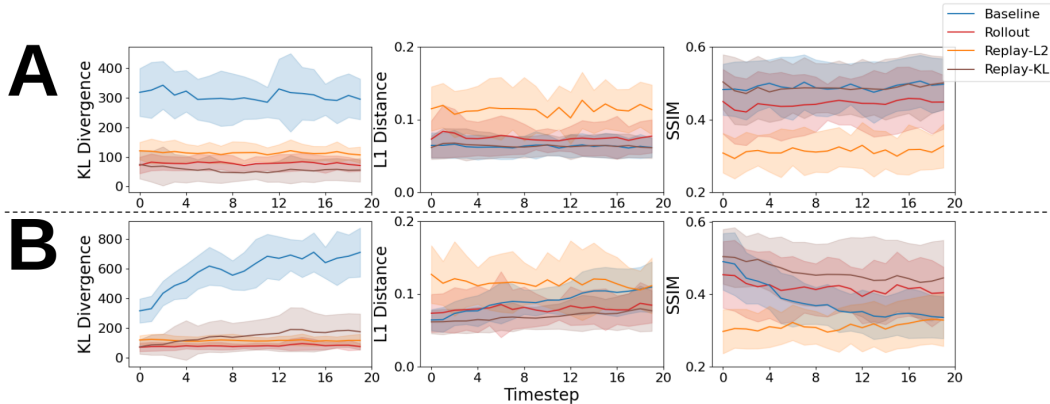


Figure 4: Numerical evaluation in SuperTuxKart "snowmountain" track. Results are reported in terms of KL divergence for latent dynamics prediction, and L1 distance & SSIM index for (A) one-step and (B) long-horizon predictions.

25 In the "snowmountain" track, the overall trend is also confirmed: all our models improve latent
 26 reconstructions, while either match or improve the baseline in decoded reconstructions. The only
 27 exception is represented by Replay-L2 that, despite achieving a better latent prediction, systematically
 28 fails to decode it. As reported in main text, we track this effect to the latent space learned by the
 29 VAE: while estimating the generative distributions from a batch of similar samples correctly points
 30 the model in the right direction, sampling from that distribution for the purpose of decoding usually
 31 results in visually unpleasant images.

32 As can be seen from the examples in B, "lighthouse" features a visually dark theme. As such,
 33 differences in this track are generally smaller and harder to assess from quantitative measures.

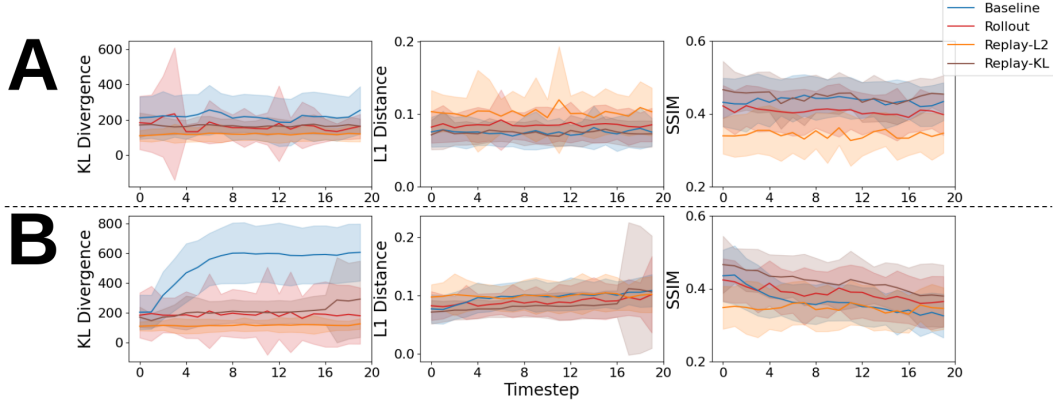


Figure 5: Numerical evaluation in SuperTuxKart "lighthouse" track. Results are reported in terms of KL divergence for latent dynamics prediction, and L1 distance & SSIM index for (A) one-step and (B) long-horizon predictions.

34 However, we see from Figure 5 how the general trend is confirmed, with Replay-KL and Rollout
 35 either matching or improving over the baseline, while Replay-L2 behaving better than the baseline in
 36 latent reconstructions (KL divergence), while failing to properly decode its predictions.

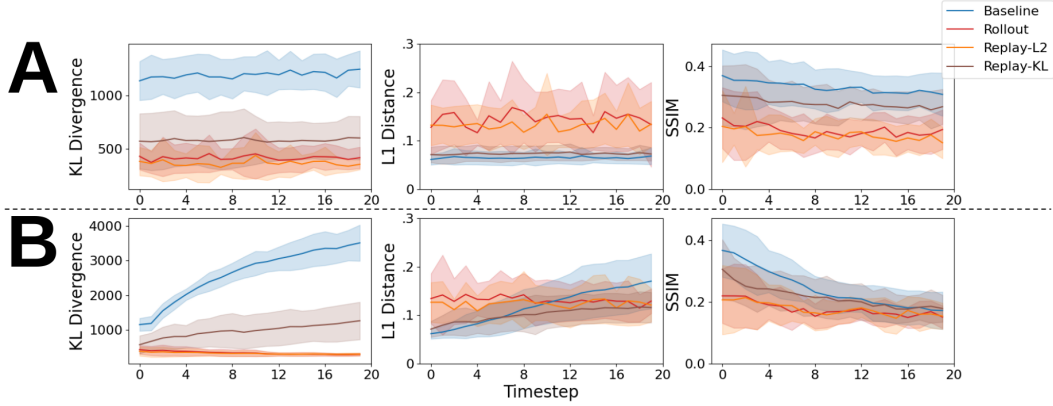


Figure 6: Numerical comparison for Minecraft "Treechop-v0" task. Results are divided in (A) one-step and (B) long-horizon predictions.

37 The results from "Treechop-v0" (Figure 6 from the Minecraft benchmark further confirm our findings.
 38 Intuitively, the state space of the task is limitless, with a few "bottlenecks" occurring in the proximity
 39 of a tree. Nonetheless, our agents are generally better at reconstructing latent dynamics, and on
 40 average match the baseline on visual reconstructions. We found this result surprising, as our agents
 41 only store a limited number of trajectories, while an SSM should theoretically be able to generalize
 42 better.

43 Similarly, in "Navigate-v0" a human expert is asked to explore the open world of the game, with no
 44 particular aim. Hence, it is nearly impossible to fully represent the full space without having access
 45 to an infinite number of trajectories. The results in Figure 7 reflect this intrinsic complexity: this
 46 is the only instance in our experiments where the latent reconstruction favors the baseline over our
 47 strongest model, Replay-KL. However, we notice how, despite similar or higher KL divergence, our
 48 models' decoded latents are generally closer to the real sequence.

49 Our experiments on "Space Invaders", reported in Figure 8 reflect the average trend shown in the
 50 main text. Notably, in this case the KL divergence computed on our models is very low, meaning
 51 that the predicted dynamics is generally very close to the original sequence. Also, due to the images

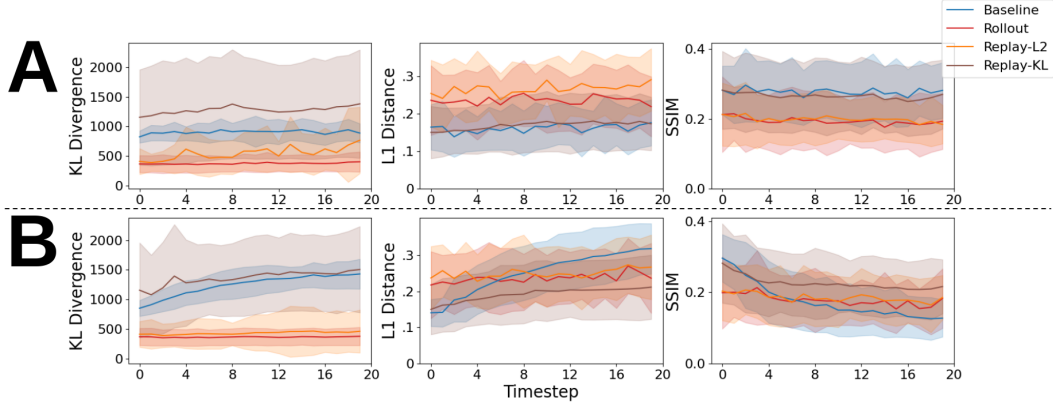


Figure 7: Numerical comparison for Minecraft "Navigate-v0" task. Results are divided in (A) one-step and (B) long-horizon predictions.

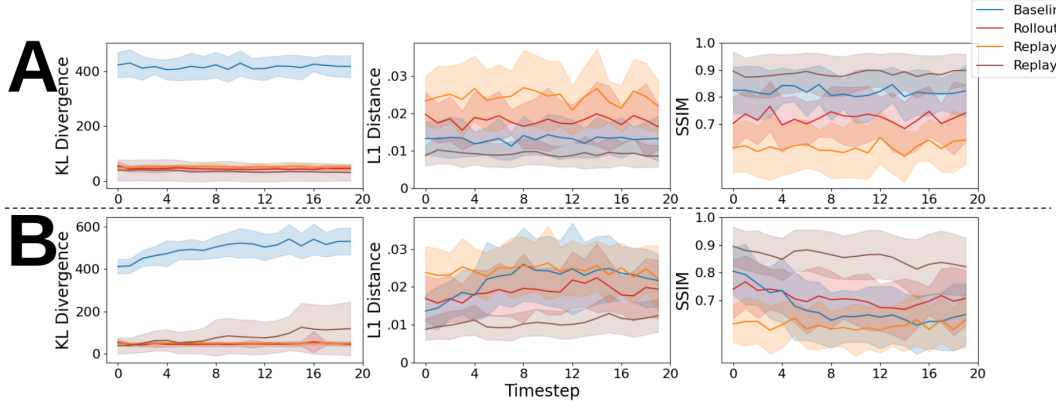


Figure 8: Numerical evaluation of "Space Invaders" from the Atari benchmark. Results report (A) one-step and (B) long-horizon predictions.

52 differing only on small details, the error on L1 distance is an order of magnitude lower w.r.t. the other
53 benchmarks, and SSIM is close to perfect.

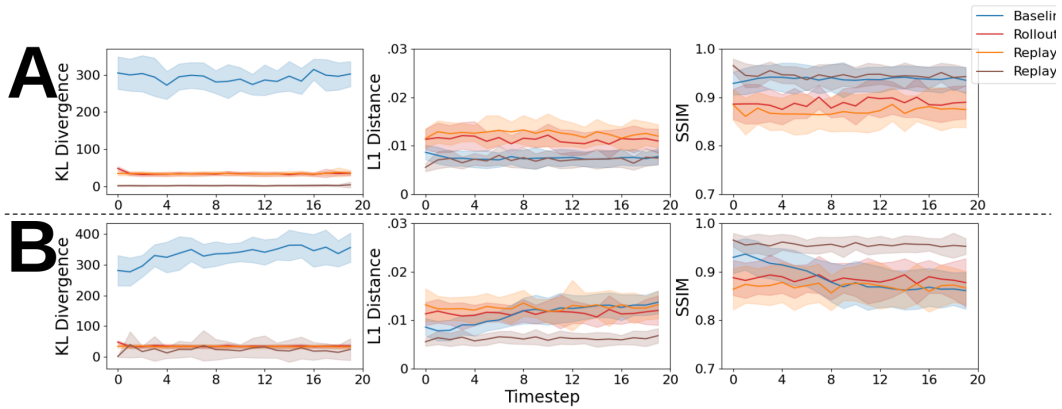


Figure 9: Numerical evaluation of "Seaquest" from the Atari benchmark. Results report (A) one-step and (B) long-horizon predictions.

54 Similarly, Figure 9 further confirms the validity of our approach, with Replay-KL performing either
 55 on par or better than the baseline. Moreover, like in "Space Invaders", the gap in KL divergence
 56 is abyssal, while L1 distance and SSIM are similar, but always favoring Replay-KL, especially in
 57 long-horizon reconstructions.

58 **B Prediction & reconstruction - qualitative evaluation**

59 This Appendix reports a number of visual reconstructions of predicted dynamics. For each environ-
 60 ment, we report two one-step and two long-horizon predictions, namely a positive example and a
 61 failure case for our models. Given the number of models we test, finding a totally positive/negative
 62 example that include all models is hard. As such, in order to maintain the length of our Appendix
 63 reasonable, we report the *best overall* positive/negative examples. Each subsection refers to an
 64 environment.

65 In "fortmagma" (Figure 10) we could not find a failure case for Replay-KL. However, Rollout and
 66 especiallt Replay-L2 are subject to hallucinations both in one-step and long-term reconstruction.

67 As stated in the main text, "snes_rainbowroad" represents a particularly hard case for this benchmark
 68 due to the visual complexity of the track and to the "falls" that a player may experience. As such, in
 69 Figure 11 we see how all models can incur into hallucinations.

70 The track "volcano_island" features a lot of visual variety, as it includes track sections on asphalt,
 71 rock, sand, and even water. Such visual diversity generally makes prediction harder. As expected,
 72 in Figure 12 we see how, to some extent, all models incur into hallucinations. An exception is
 73 represented by Replay-KL, which exhibits strong performance throughout the whole track.

74 As visible in Figure 13, in "snowmountain" all models except Replay-L2 are mostly capable of
 75 predicting both long-term and one-step situations. However, we see in Figure 13A that the baseline
 76 SSM model might diverge from the reference sequence.

77 As previously stated, "lighthouse" is a peculiar case of this benchmark due to its intrinsic darkness.
 78 As such, spotting hallucinations is a hard task. However, the tracks also features lightnings occurring
 79 at random times, which we use to determine failure cases. In particular, Figure 14B shows Replay-KL
 80 predicting a lightning when not existing. Also, in section D of the same Figure we see how all models
 81 fail to reconstruct a coherent background.

82 Tasks from the MineRL benchmark feature an almost limitless state space. As such, expecting a
 83 perfect prediction from quite simple models such as ours is quite unrealistic. However, in Figure 15 it
 84 is surprising to see Replay-KL achieving quite good reconstructions in A and C. Notably, in this task
 85 our baseline model (SSM) produces mostly blurry reconstructions.

86 An even more extreme case of limitless state space is represented by "Navigate-v0", for which we
 87 report some examples in Figure 16. Interestingly, quite some trajectories are centered on navigating
 88 the sea, hence easily leading all models to hallucinations if water is shown in the picture. For this
 89 task, finding a "good prediction" was particularly hard.

90 The most notable difference between the models in "Space Invaders" (Figure 17 is the consistency
 91 with which they predict the alien ships. In general, Replay-KL was the most consistent one, even
 92 though it was not perfect.

93 Similarly, as shown in Figure 18 in "Seaquest" the most notable example of hallucination is repre-
 94 sented by the enemy boats that, in some models, flicker throughout the sequence. Like in "fortmagma",
 95 Replay-KL was mostly correct and coherent in its predictions, while both SSM, Rollout and Replay-
 96 L2 struggle to maintain consistency in their predictions.

97 **C Ablation study - additional results**

98 In this Appendix we report some visual reconstructions coming from the ablation study presented in
 99 Section 5.2. Moreover, we report the results of an additional ablation study performed on Minecraft
 100 "Treechop-v0", using [10, 80] trajectories and a step of 10. This additional ablation study is also used
 101 to compute the wallclock time needed for each method to predict the latent dynamics. To fit all the
 102 models in one run, the test is run on CPU, using an Intel i7 12650HX.

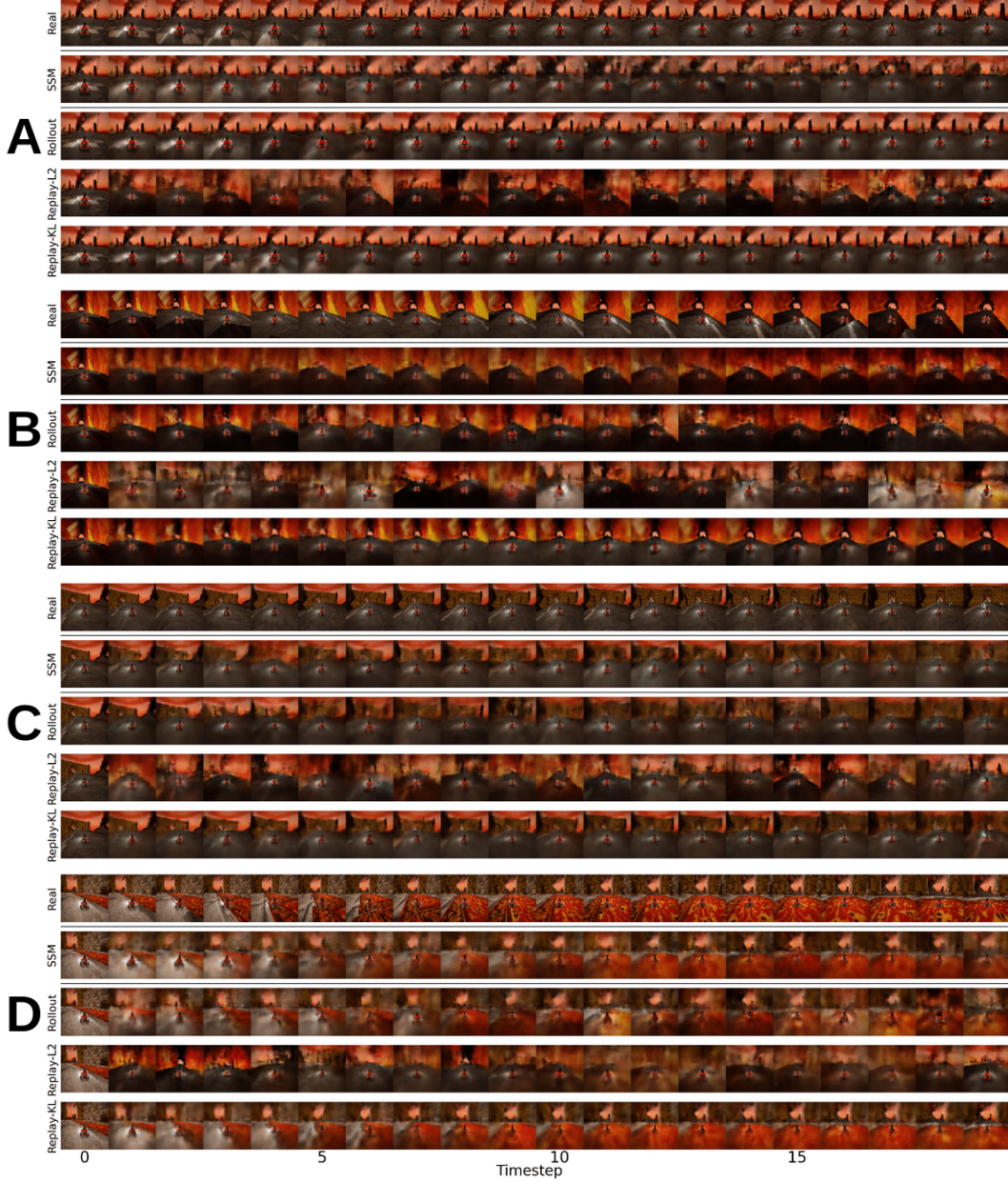


Figure 10: Examples from SuperTuxKart "fortmagma" track. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

103 C.1 Wallclock time

104 Figure 19 shows the average time required to recover a transition for each method in the one-step
 105 (top) and long-horizon (bottom) cases. As expected, the time required to perform the search scales
 106 linearly in the number of stored transitions. During the test, we occupied a maximum of 12GB of
 107 RAM, including VAE, SSMs, a rollout and a replay buffer.

108 Clearly, computing the KL divergence requires significantly more time than computing L2 distance.
 109 However, we highlight that for long-term predictions we only require to compute the distances
 110 once every N steps, where N is the horizon. Additionally, we inform that in our implementation
 111 we used the `torch.distributions` package to ensure correctness, regardless of efficiency. The

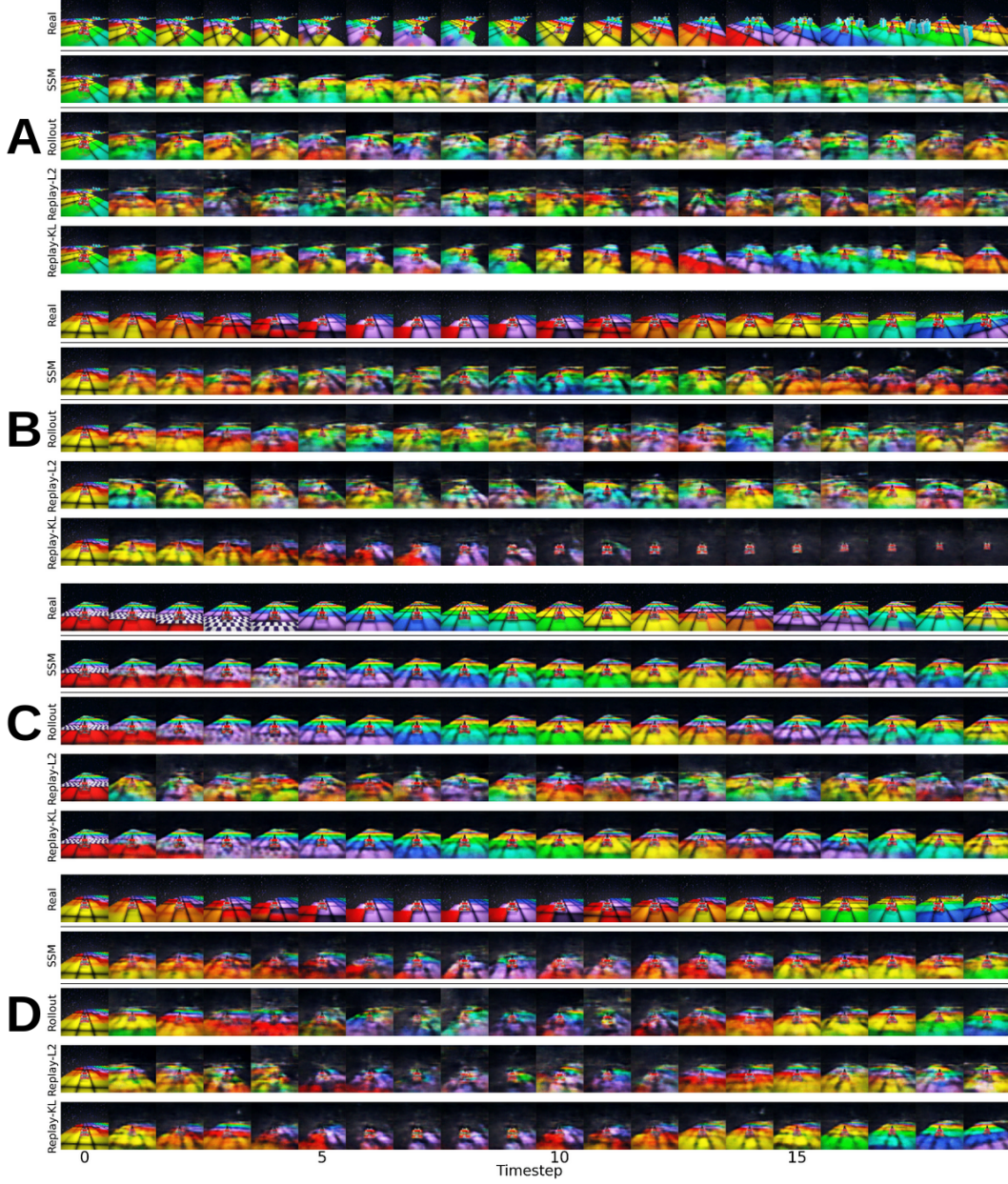


Figure 11: Examples from SuperTuxKart "snes_rainbowroad" track. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

112 package requires two `Distribution` objects to compute the KL distance between them *exactly*,
 113 hence making the computation quite expensive. However, if the type of distribution is set, efficiency
 114 can be improved by computing a closed-form solution and leveraging PyTorch parallel computation.
 115 Additionally, we remark that the test has been done in CPU, since all models could not fit in the
 116 VRAM of our GPU. However, when using only a single method, the computation can be parallelized
 117 in GPU to vastly improve speed.

118 Finally, we point out how encoding almost half a million transitions represents quite an extreme case
 119 needed only in open-ended tasks: in SuperTuxKart, for example, each dataset is composed of roughly
 120 10k transitions; similarly, in Atari we store around 20k transitions per task. Therefore, we expect the
 121 time gap between methods to be quite limited in practice.

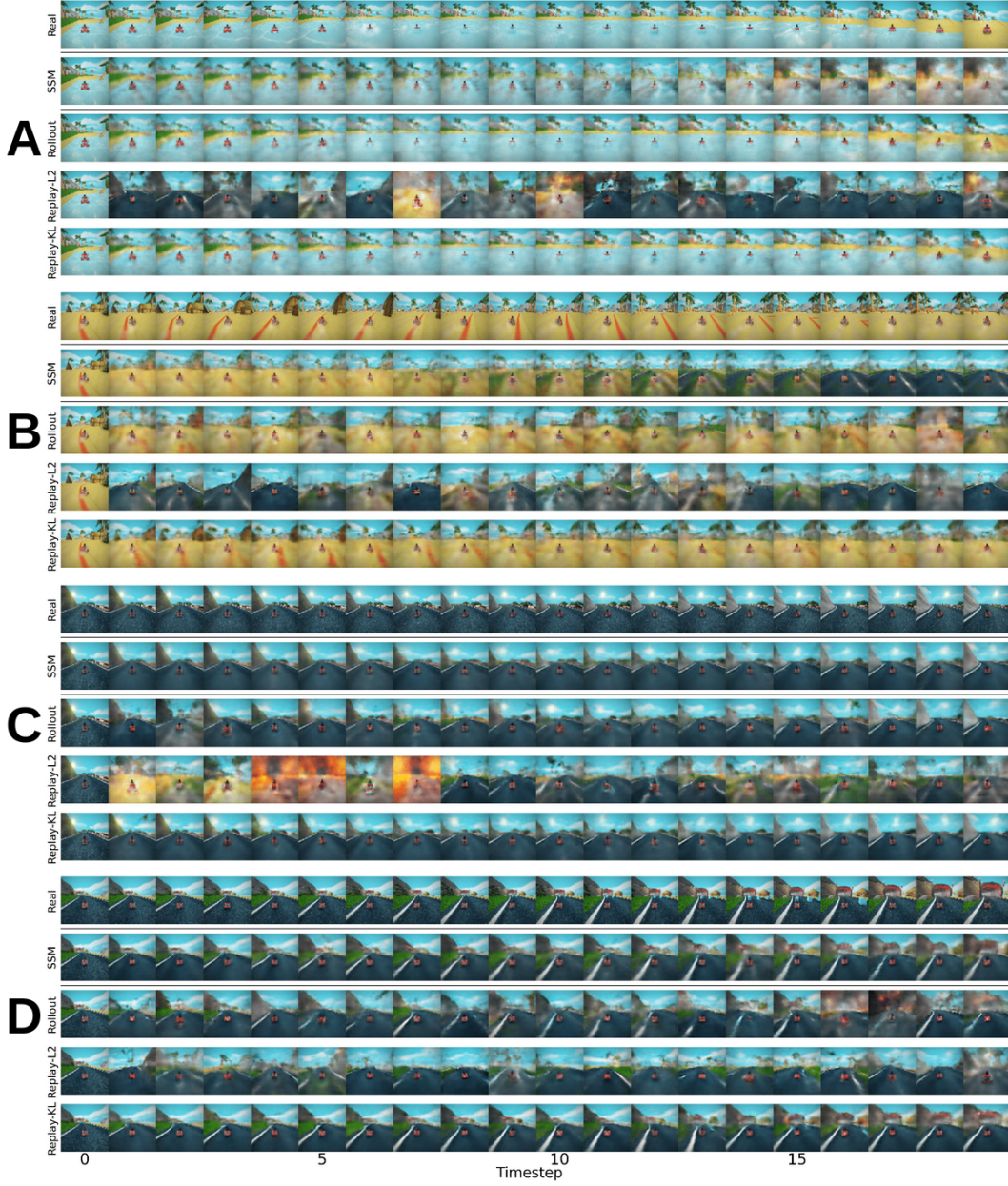


Figure 12: Examples from SuperTuxKart "volcano_island" track. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

122 C.2 Visual results

123 In Figure 20 we report an example of reconstruction for 10, 20, 40 and 80 encoded trajectories. The
 124 image suggests that encoding more images slightly improves visual resemblance in our method,
 125 mostly thanks to the more complete coverage of the state space. However, reconstruction is not
 126 always perfect. Interestingly, despite improving its latent prediction skills as reported in Section 5,
 127 the baseline cannot produce convincing decoded sequences from its predictions.

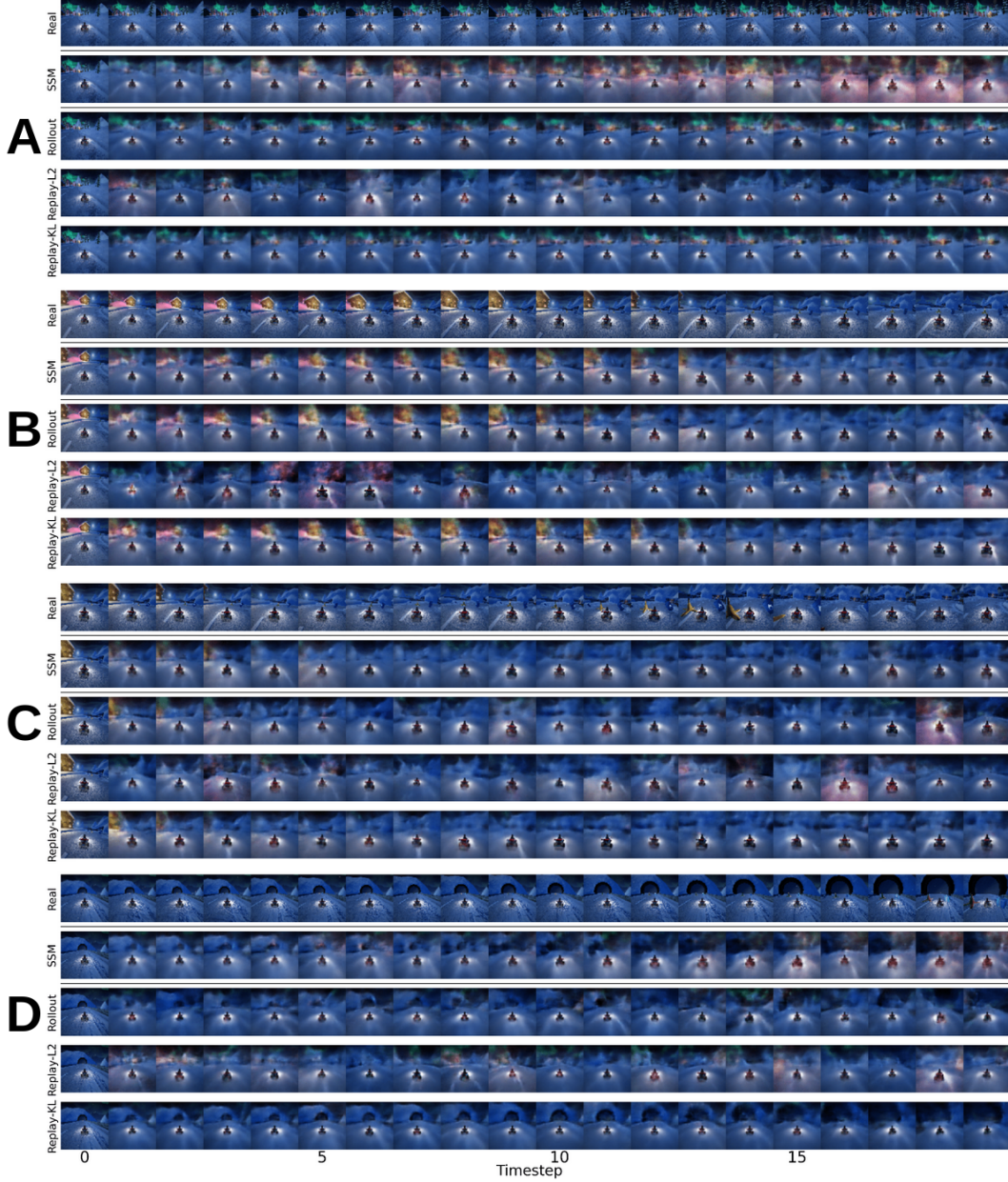


Figure 13: Examples from SuperTuxKart "snowmountain" track. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

128 D Action conditioning - visual examples

129 In Figures 21, 22, 23, 24 and 25 we report some visual examples of the effects of action conditioning
 130 for each track of SuperTuxKart. In each Figure, the first row shows the real sequence, while the three
 131 pairs of sequences show (top to bottom) Rollout, Replay-L2 and Replay-KL with (first row of a pair)
 132 and without (second row of a pair) action conditioning.

133 As reported in main text, conditioning on the action generally makes the prediction task harder,
 134 especially for Rollout and Replay-L2, as both models need to sample a batch on transitions to
 135 estimate the distribution. On the contrary, Replay-KL seems unaffected. We hypothesize that this is
 136 the result of matching the distribution rather than estimating it.

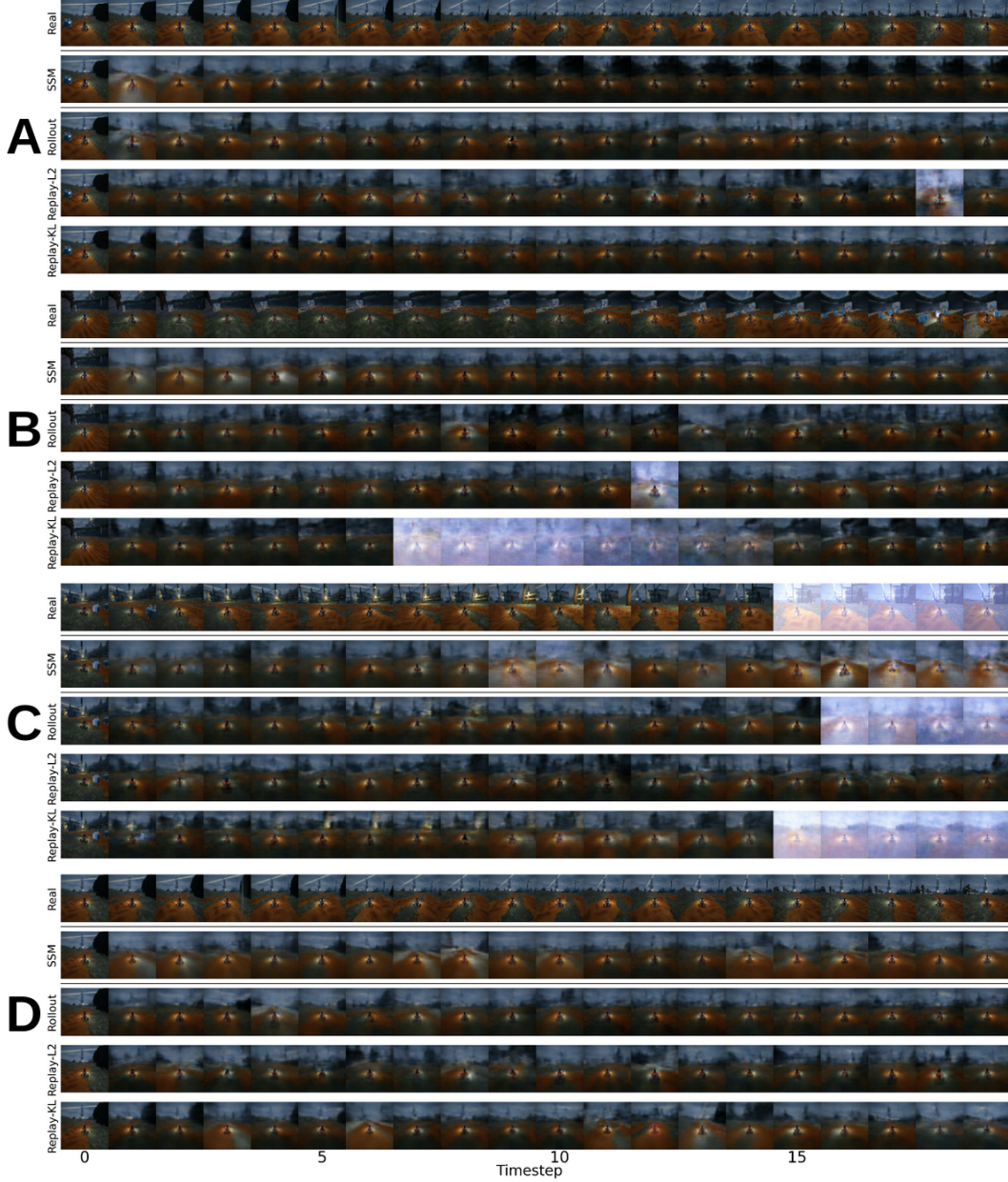


Figure 14: Examples from SuperTuxKart "lighthouse" track. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

E Algorithm & implementation details

In this Appendix, we report the architectures we have used for our study, along with some details and hyperparameters, to facilitate reproducibility of our results.

E.1 VAE

Table 1 shows the hyperparameters used to train the VAEs. We train the models using the β -VAE variant, which includes a warm-up procedure for the KL divergence term. Specifically, the KL divergence part of the loss is masked in the first 10% of the epochs, then it is gradually increased during the next 80% of training time. The architecture of the encoder and decoder follows [8].

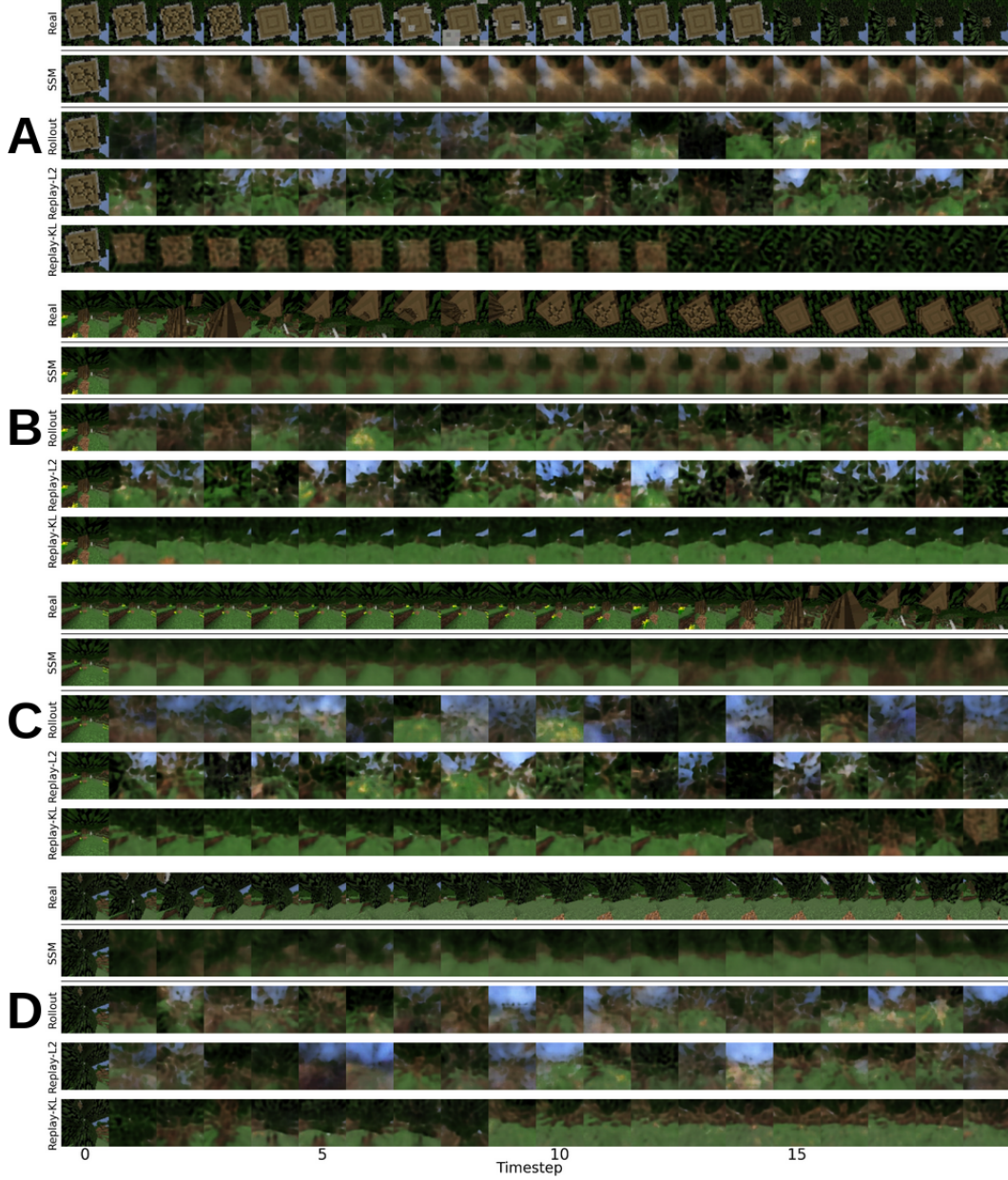


Figure 15: Examples from MineRL "Treechop-v0" task. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

For MineRL, we have increased the latent size due to the higher amount of relevant information of an image. Additionally, we have decreased the number of epochs to 50, as empirically the model showed signs of convergence around that time. Finally, we have increased the learning rate from 5×10^{-5} to 3×10^{-4} , as it empirically led to the best results.

E.2 SSM

We report the hyperparameters used to train the SSM model in Table 2. The training procedure follows [8]. From the original paper, we changed the values of the hyperparameters to obtain the best results, according to our empirical tests. Our PyTorch implementation uses <https://github.com/abhayraw1/planet-torch> as reference, even though the source code

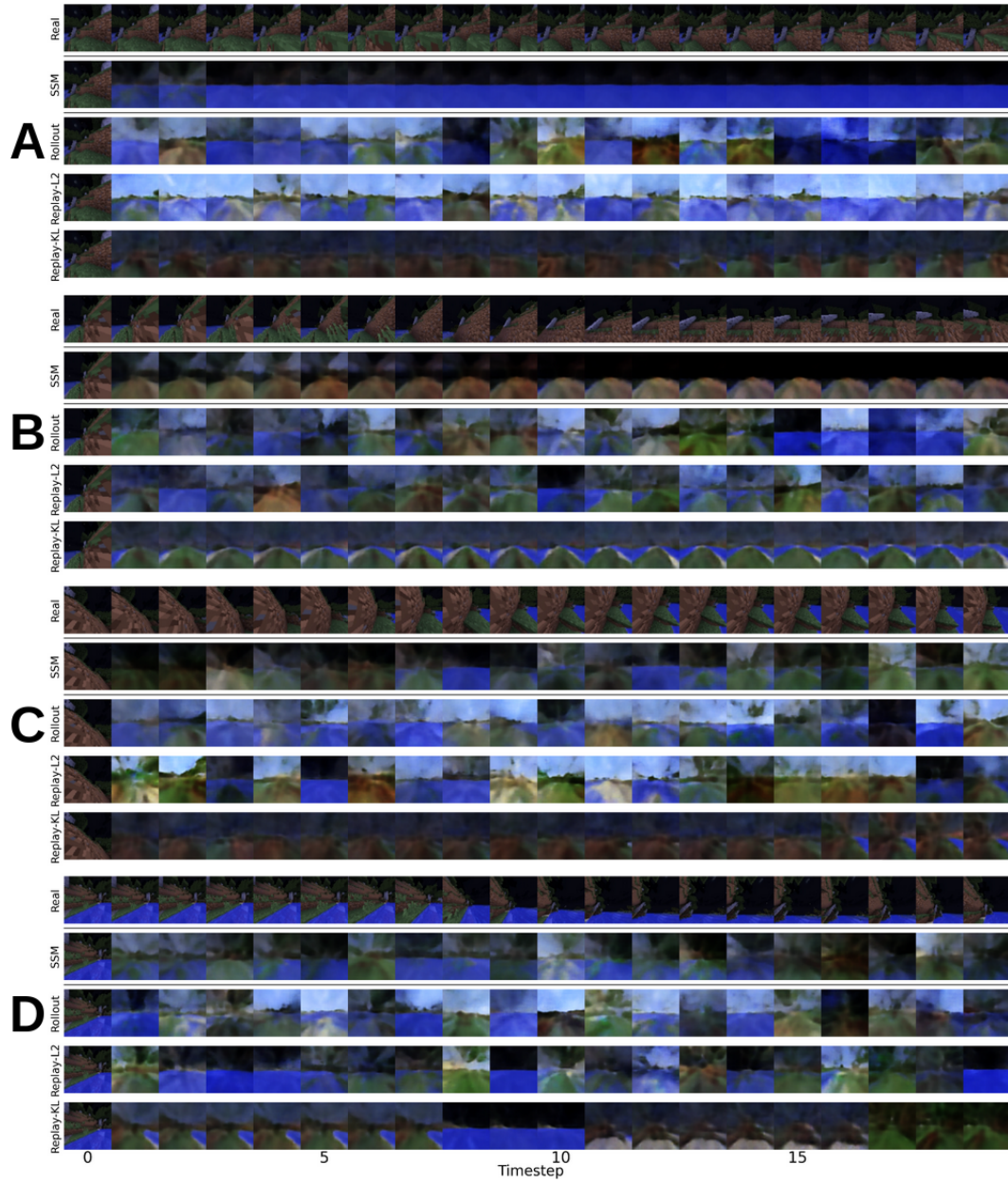


Figure 16: Examples from MineRL "Navigate-v0" task. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

154 of the repo has been used only as a guide. We re-implemented the model as shown in the source code
 155 linked to this study.

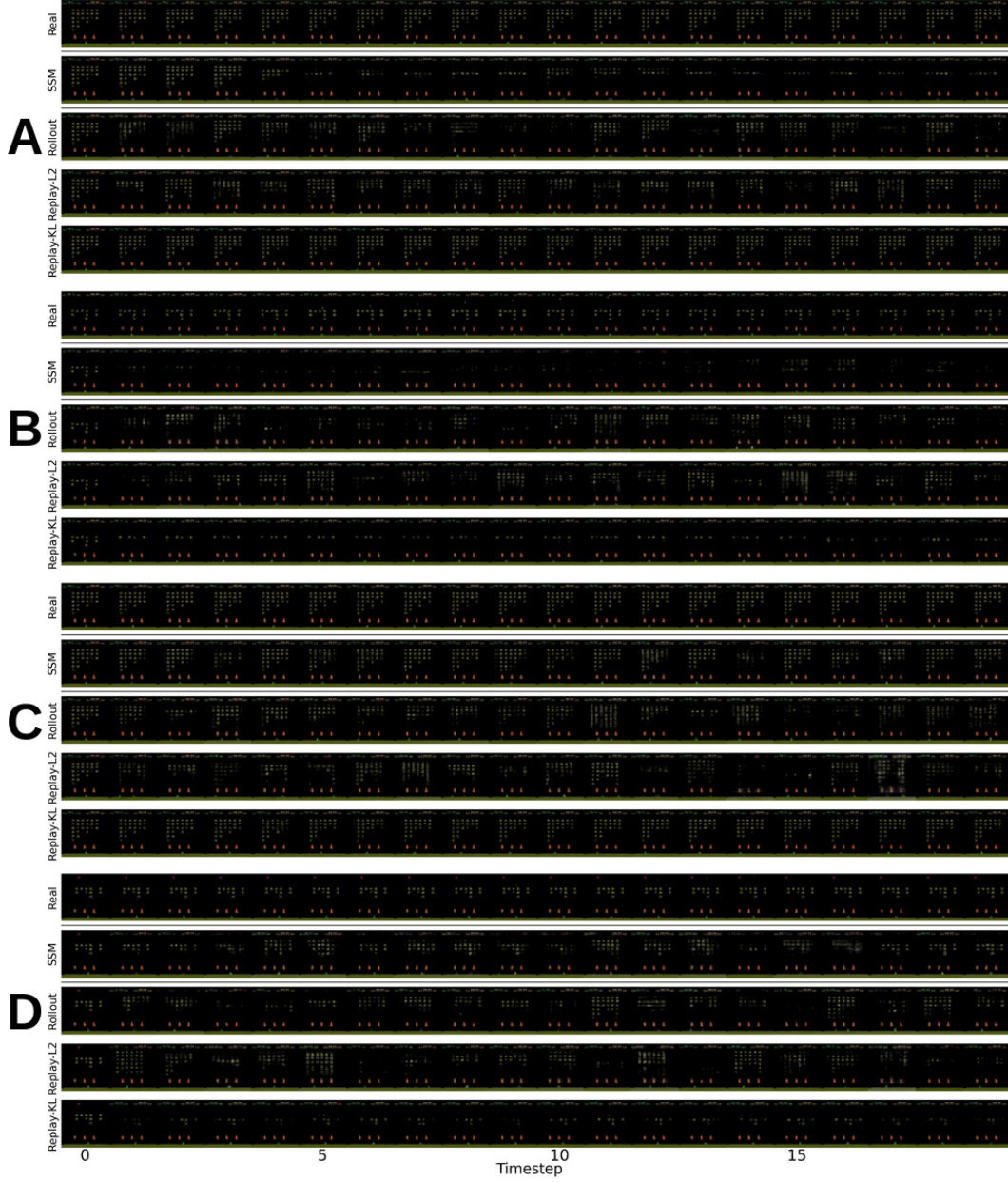


Figure 17: Examples from Atari "Space Invaders" games. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

Name	SuperTuxKart	MineRL	Atari
image size	64x64x3	64x64x3	64x64x3
latent size	128	512	128
learning rate	5×10^{-5}	3×10^{-4}	5×10^{-5}
epochs	250	50	250
batch size	128	128	128
beta	$0.0 \rightarrow 5 \times 10^{-8}$	$0.0 \rightarrow 5 \times 10^{-8}$	$0.0 \rightarrow 5 \times 10^{-8}$
beta interval (epochs)	$25 \rightarrow 225$	$5 \rightarrow 45$	$25 \rightarrow 225$

Table 1: Hyperparameters for VAE models used in this study.

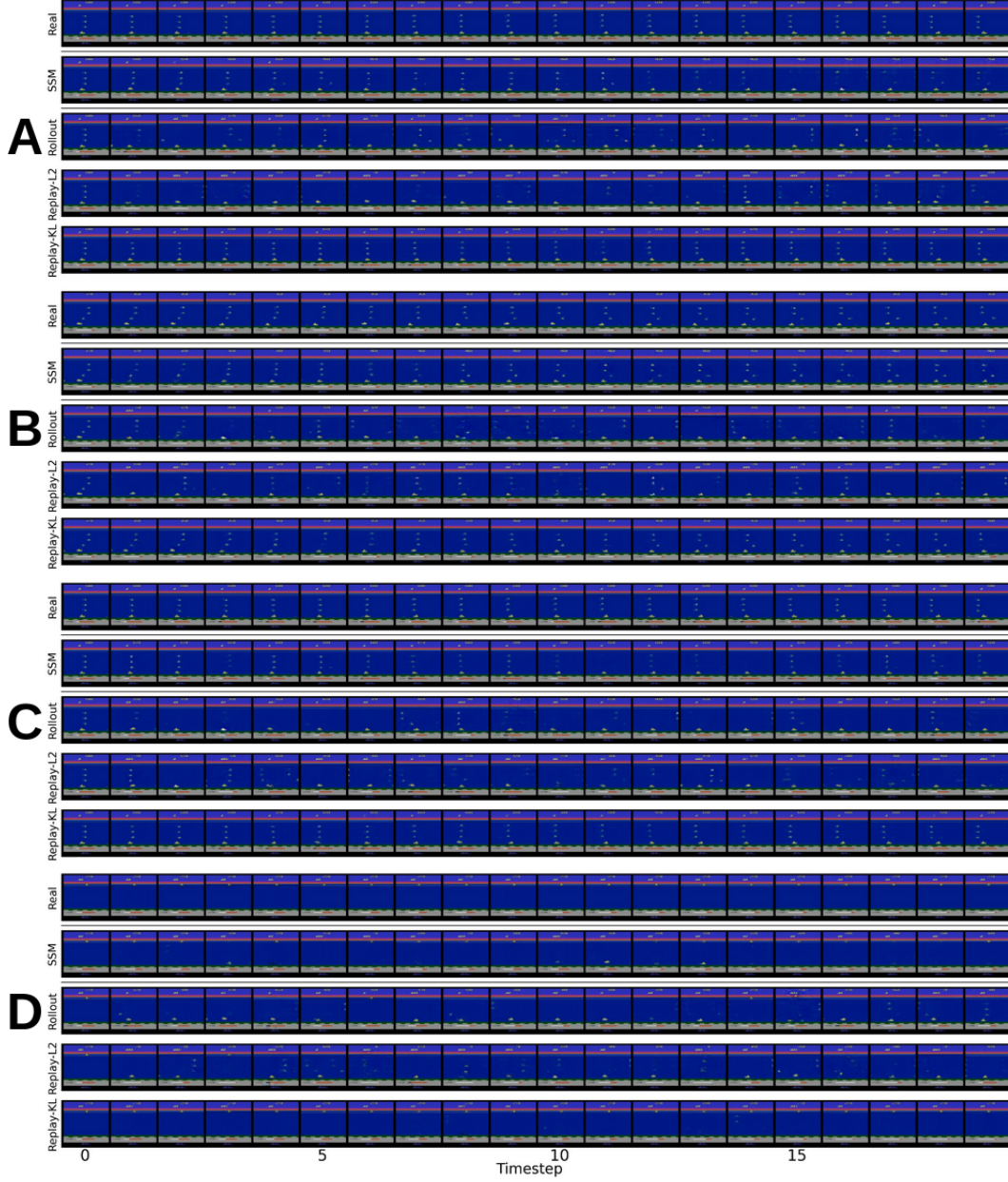


Figure 18: Examples from Atari "Seaquest" game. (A) Long-horizon success; (B) long-horizon failure; (C) One-step success; (D) One-step failure.

Name	SuperTuxKart	MineRL	Atari
latent size	128	512	128
hidden size	256	256	256
learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}
epochs	250	250	250
batch size	64	64	64
beta	0.1	0.1	0.1

Table 2: Hyperparameters used to train the PlaNet baseline.

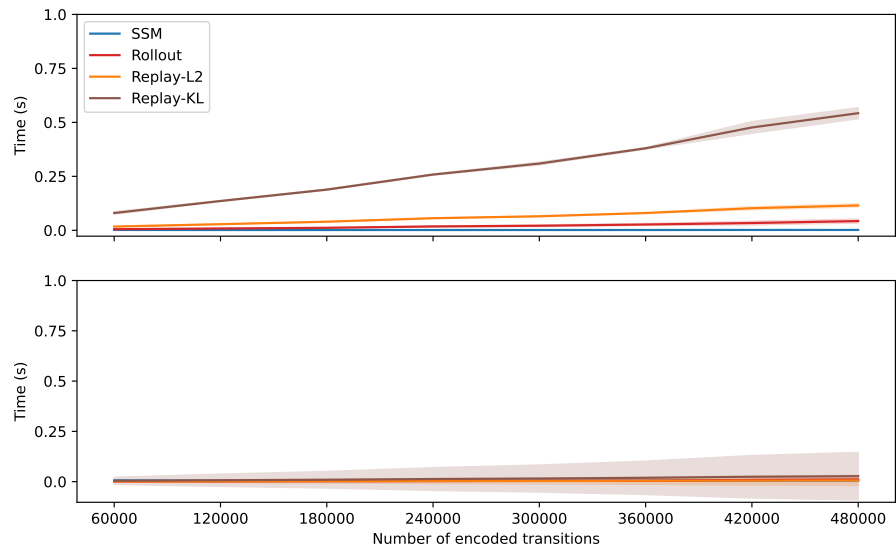


Figure 19: Average time needed for one-step (top) and long-horizon (bottom) latent dynamics prediction.

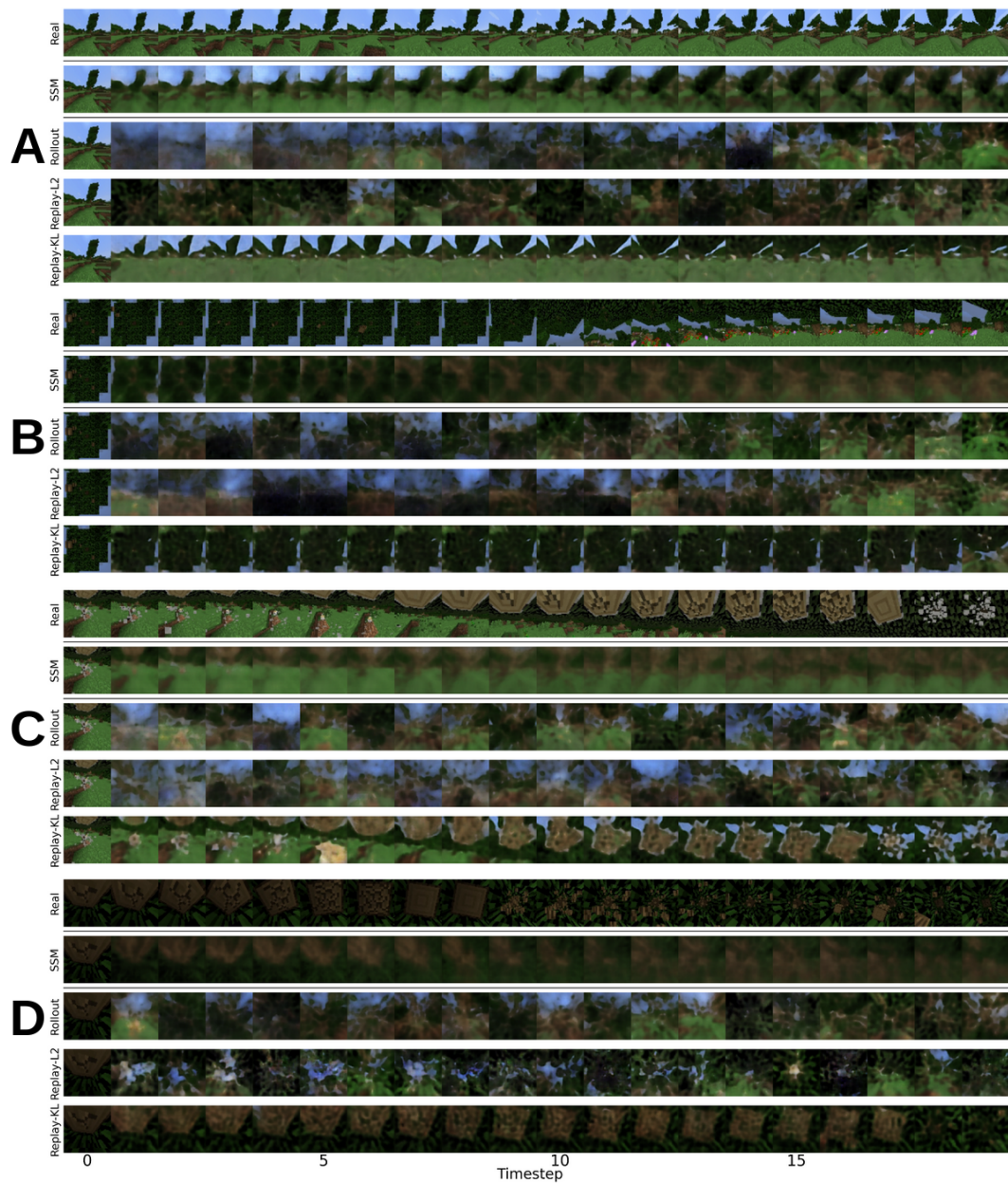


Figure 20: Visual examples of reconstructions from MineRL "Treechop-v0" using (A) ten, (B) twenty, (C) forty, and (D) eighty trajectories.

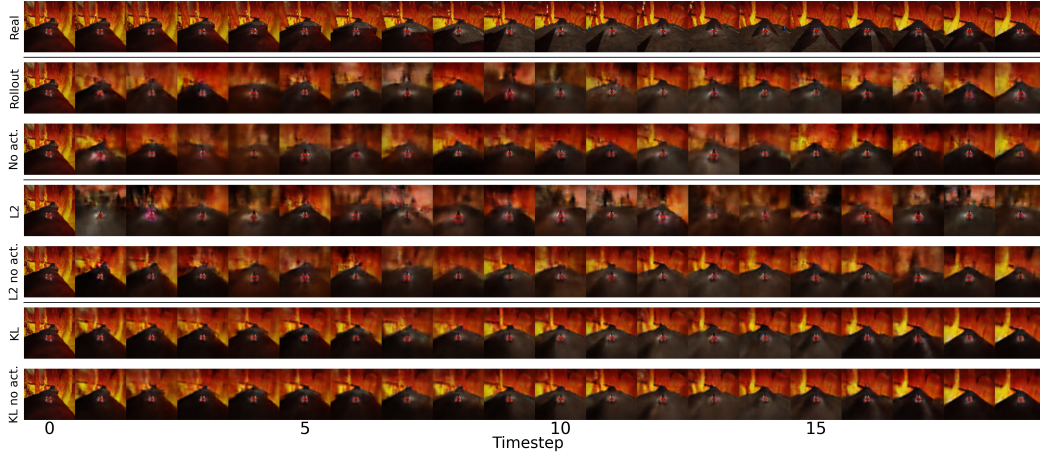


Figure 21: Effects of action conditioning on dynamics prediction in "fortmagma" track. First row reports the real sequence; after that, each pair of rows reports our three methods, respectively with and without action conditioning.

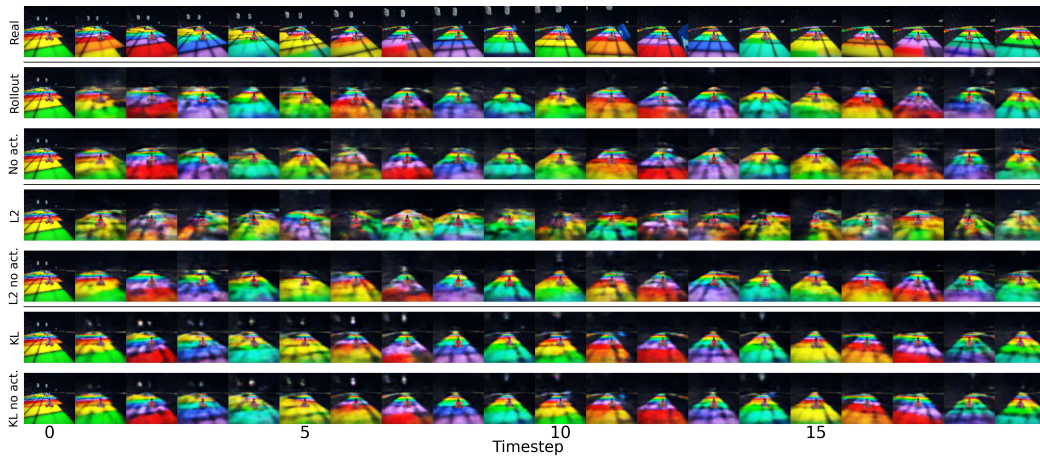


Figure 22: Effects of action conditioning on dynamics prediction in "snes_rainbowroad" track. First row reports the real sequence; after that, each pair of rows reports our three methods, respectively with and without action conditioning.

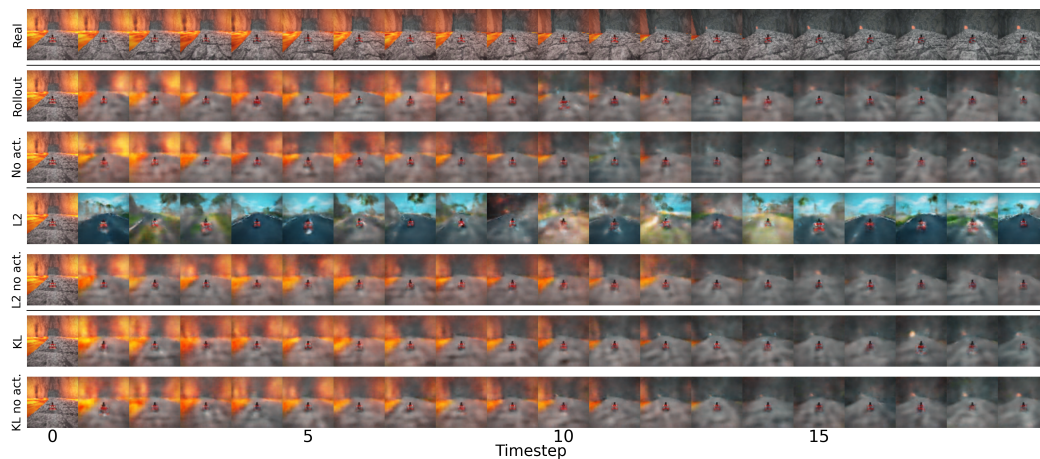


Figure 23: Effects of action conditioning on dynamics prediction in "volcano_island" track. First row reports the real sequence; after that, each pair of rows reports our three methods, respectively with and without action conditioning.

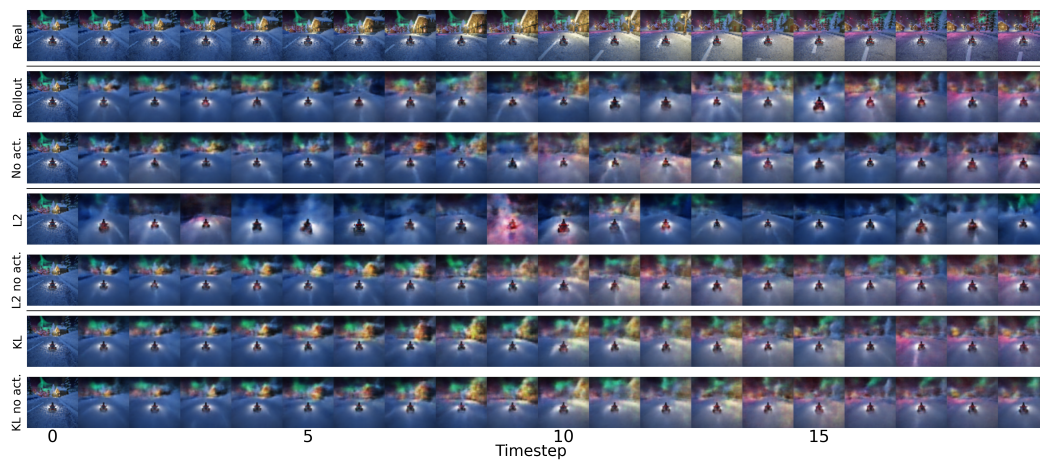


Figure 24: Effects of action conditioning on dynamics prediction in "snowmountain" track. First row reports the real sequence; after that, each pair of rows reports our three methods, respectively with and without action conditioning.

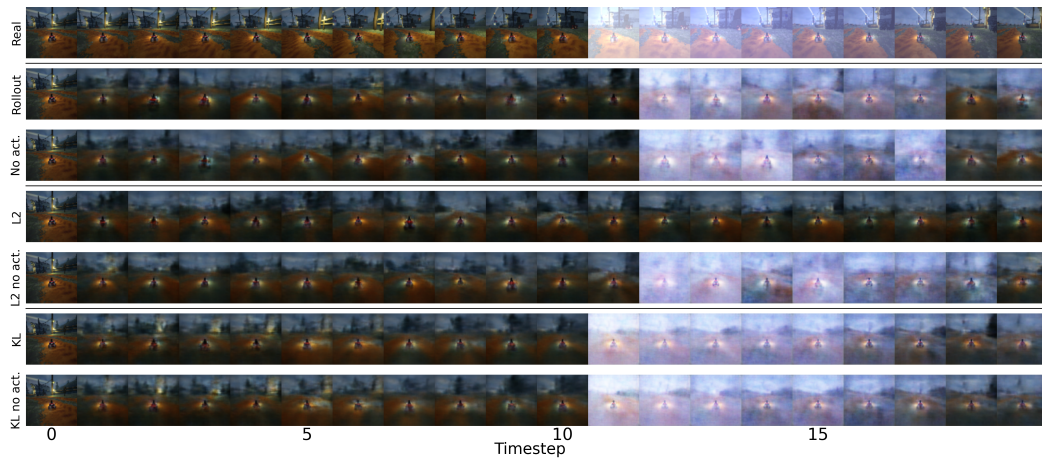


Figure 25: Effects of action conditioning on dynamics prediction in "lighthouse" track. First row reports the real sequence; after that, each pair of rows reports our three methods, respectively with and without action conditioning.