Jafar: An Open-Source Genie Reimplemention in Jax

Timon Willi [*1] **Matthew Jackson** [*1] **Jakob Foerster** [1]

Abstract

We introduce Jafar, an open-source Jax reimplementation of Genie, a foundational world model. Genie was the first world model trained in an unsupervised manner on unlabelled internet data. We follow the reproducibility study in the appendix of Genie and evaluate our reimplementation on the CoinRun environment from the procgen suite. Jafar is implemented in JAX and produces visually consistent video generation and can be trained on a single GPU. Jafar can be found at https://github.com/flairox/jafar.

1. Introduction

Recent advancements in generative models have dramatically enhanced our ability to create coherent and contextually relevant outputs across different modalities (Vaswani et al., 2017; Radford et al., 2018; 2019; Brown et al., 2020). In natural language processing, large language models achieved remarkable proficiency in generating human-like text, benefitting from both architectural innovations and the expansion in model scale. Similarly, vision models improved dramatically, producing visually stunning images that are both diverse and detailed, thanks to similar improvements in model design and training methodologies (Ramesh et al., 2021; Hong et al., 2022). Building on these successes, the progression towards video generation represents a natural evolution of generative capabilities (Hu et al., 2023). The Genie model (Bruce et al., 2024) extends the frontiers of generative models into the dynamic and interactive domain of video. By generating sequences of frames interactively, Genie facilitates the creation of generative interactive experiences, marking a significant milestone in the synthesis of temporal and visual data.

Reimplementing Genie is crucial for several reasons. Firstly, the capability to control video generation and develop world models holds significant interest within the research community, promising to advance both theoretical insights and practical applications. Secondly, the high computational costs associated with video generation have historically restricted accessibility for academic labs with limited resources. Lastly, the absence of publicly available source code for the original Genie model necessitates a reimplementation to enable broader experimentation and validation of the reported capabilities.

In this work, we present "Jafar," a reimplementation of the Genie model in JAX (Bradbury et al., 2018; Heek et al., 2023), adhering closely to the reproducibility guidelines outlined in Bruce et al. (2024)'s appendix. We enhance the model's reproducibility, reporting multiple useful training metrics. We confirm that Jafar can be efficiently trained on a single GPU, and we demonstrate its capability to generate environments using the CoinRun benchmark from the ProcGen suite (Summerville et al., 2018). Furthermore, we contribute to the community by releasing our work as an open-source repository, facilitating further research and development in this field.

2. Background

Genie is built from three components: (1) a video tokenizer, which compresses video patches to a discrete latent space, (2) a latent action model (LAM), which learns an unsupervised conditioning token to represent the transition between frames, and (3) a dynamics model, which conditions on past video tokens and latent actions to autoregressively predict the next frame.

Video Tokenizer Genie uses a VQ-VAE (Van Den Oord et al., 2017) to compress video frames $\boldsymbol{x}_{1:T} = (x_1, x_2, \cdots, x_T) \in \mathbb{R}^{T\times H\times W\times C}$ into discrete tokens $z_{1:T} = (z_1, z_2, \cdots, z_T) \in \mathbb{Q}^{T\times D}$. This is parameterized by a causal spatiotemporal (ST) transformer (Xu et al., 2020), containing layers that attend *spatially* (i.e. to other tokens from the same frame) as well as *temporally* (i.e. to tokens from past frames in the same spatial position) in order to autoregressively generate video tokens.

Latent Action Model In order to learn controllable video generation from unlabelled data, Genie uses a LAM. Given

[*]Equal contribution [1]FLAIR, University of Oxford, Oxford, United Kingdom. Correspondence to: Timon Willi <timon.willi@eng.ox.ac.uk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

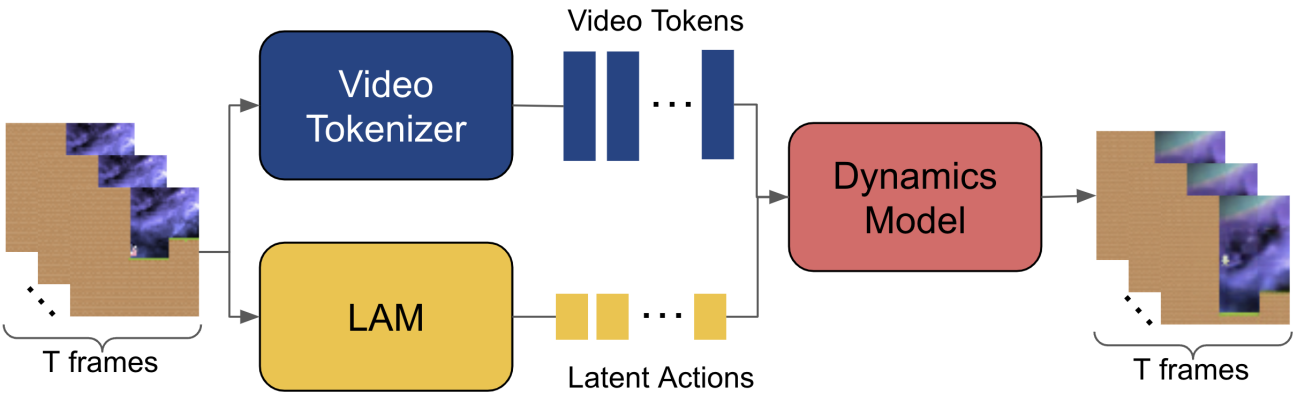

Figure 1. Genie's works as follows. A sequence of input frames are fed into the tokenizer and latent action model (LAM) respectively. The tokenizer and the LAM output a token or latent action respectively per frame. The tokens and latent actions are combined through additive conditioning and fed into a dynamics model. The dynamics model predicts the next frame tokens. At inference, these next frame tokens are fed into the tokenizer decoder to generate images.

a sequence of frames $\mathbf{x}_{1:T}$ and the following frame x_{T+1}, the LAM outputs a latent action token $\tilde{a} \in \mathbb{R}^N$ representing the transition to x_{T+1}. The LAM then conditions on $\mathbf{x}_{1:T}$ and $\tilde{a}$ to predict the next frame x_{T+1}, using reconstruction error to train the entire model. In doing so, the model is trained to compress the sequence such that the latent action contains all additional information required to generate the next frame given previous frames.

Dynamics Model The final component of Genie is the dynamics model, which uses the pretrained video tokenizer and LAM to condition on past video tokens and latent actions, with which it autoregressively predicts the next video token. The model uses MaskGIT (Chang et al., 2022), minimising cross-entropy loss between the predicted and real next token, $\tilde{z}_{T+1}$ and z_{T+1}. At test time, latent actions for an input frame or sequence of frames are frozen and the dynamics model sequentially generates future frames.

3. Jafar

3.1. Data Collection

We followed the instructions in Appendix F1 in Bruce et al. (2024). We collect 10M transition from the CoinRun environment from the Procgen suite (Cobbe et al., 2020). CoinRun is a 2D platformer with visually diverse levels and a 15-dimensional action space. Specifically, we collect 10,000 trajectories of length 1000. We sample seeds between values of 0 and 10,000 for each trajectory.

3.2. Training Details

We implemented Jafar in JAX (Bradbury et al., 2018), using Flax (Heek et al., 2023) as our neural network library.

We use the hyperparameters reported in Appendix F2 and F3 in Bruce et al. (2024), with two exceptions. First, we use an NVIDIA L40 GPU instead of a TPU. Second, we use a patch size of 16 instead of 4 for the tokenizer. We found that a patch size of 4 slows down training significantly. We were thus not able to train the tokenizer for 300k steps within 3 days with a patch size of 4. Setting the patch size to 16 keeps the maximum required amount of memory the same as with patch size 4. However, we hypothesise that the resulting smaller token vectors are more optimized for a GPU architecture, leading to less loads from the L2 cache. We leave it for future work to investigate this discrepancy.

As instructed in the Appendix, we train the LAM and the dynamics model for 200k steps. Furthermore, we normalize the inputs to a range $[0, 1]$ for the tokenizer and the LAM.

All models take approximately 34.5GB of memory on NVIDIA L40s, as opposed to the 16GB reported in the Appendix. We leave it to future work to investigate this discrepancy, as 34.5GB allows Jafar to run on a wide range of commercially available GPUs.

4. Results

The reproducibility case study documented in Appendix F (Bruce et al., 2024) does not report any quantitative or qualitative results. We can therefore make no performance comparisons to the original genie, except for visual comparisons. However, to ensure reproducibility of our work, we report results for Jafar, providing training curves and qualitative results to facilitate reproducibility.

Tokenizer For the tokenizer we identify 6 helpful metrics. The training loss, the structural similarity index (SSIM), the

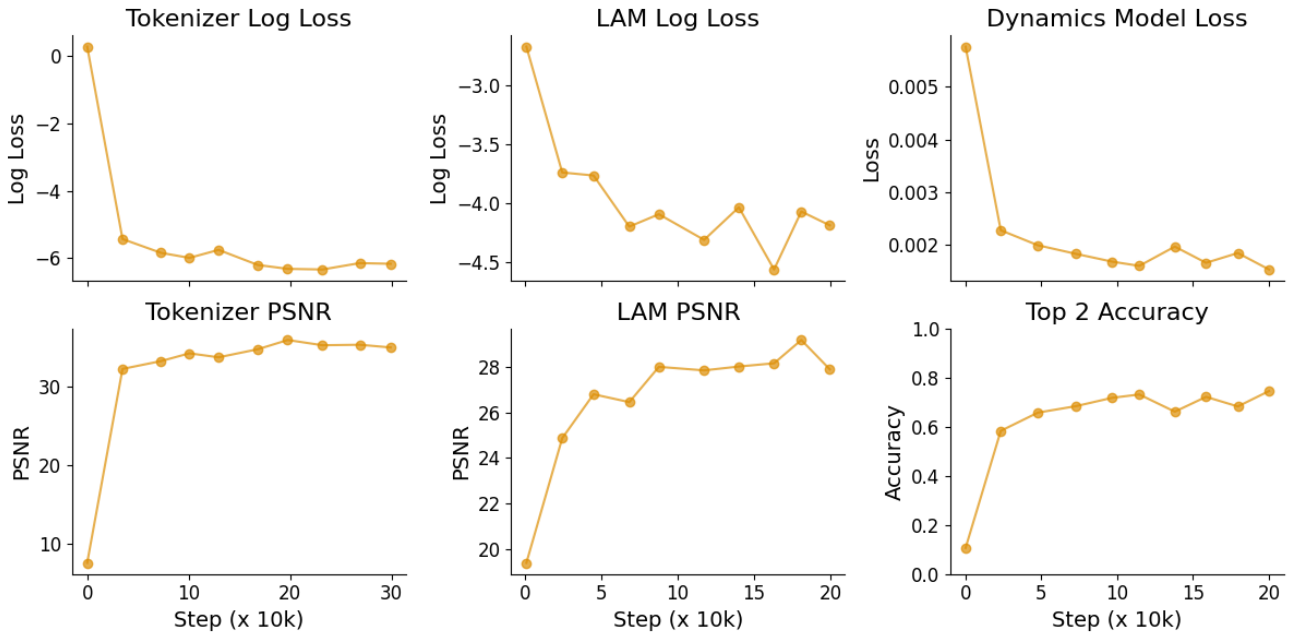

Figure 2. We report results for the tokenizer in the leftmost column, LAM in the center column, and dynamics in the rightmost column. The original Genie paper does not report losses for the reproducibility study. For the tokenizer, we note that we achieve a similar PSNR than Genie ($\sim$ 36), as reported in Appendix C.2, Table 6. There are no PSNR results reported for the LAM. For the dynamics model we also report the Top 2 Accuracy, describing the percentage of correctly predicted tokens within the top 2 logits, across a sequence of frames.

peak signal-to-noise ratio (PSNR), the entropy of the VQ-VAE embeddings, the code usage of the VQ-VAE and the VQ-VAE loss, as reported in Figure 2 and Figure 4. Note that the tokenizer achieves a PSNR of 35.7, similar to the PSNR values reported in Appendix C.2, Table 6 in Bruce et al. (2024). We also achieve a high SSIM of 0.94, suggesting visually sound reconstructions. We show example reconstructions of the tokenizer in Figure 3. For the VQ-VAE, we report high entropy at 0.76 and relatively low code usage at 0.57 in comparison to the LAM code usage at 1.0.

LAM The results for our LAM model are shown in Figure 2 and Figure 5. For our LAM, we report a PSNR of 28.5, which is lower than the PSNR or the tokenizer. We also report high perplexity and entropy values, i.e., 0.82 and 0.89 respectively, suggesting that the LAM struggles in learning distinct latent actions.

Dynamics Model We report the dynamics model results in Figure 2. The percentage of correctly predicted tokens within the top 2 logits, across a sequence of frames, i.e., the top 2 accuracy, is high at 0.79, suggesting that the dynamics model predicts visually faithful frames. We provide an example sequence of generated frames in Figure 6. First, note that we sample a visually consistent sequence of frames, where the background and environment stay the same across the frames. Second, the agent appears to be digging itself into a hole. Digging is not an action in the CoinRun environment, so Jafar never saw such a sequence. We hypothesise that the down movement is more prevalent in the dataset than left or right movements. Thus, Jafar learned that it would be more likely for an agent to move further down if already in a down movement and adapt the environment accordingly. Further investigations are needed to understand this behaviour.

5. Related Work

Genie falls under the broad class of action-conditioned sequence models, termed *world models* (Oh et al., 2015; Ha & Schmidhuber, 2018a;b). Recently, world models surged in popularity due to algorithmic advances and increased compute, achieving impressive results in self-driving with GAIA-1 (Hu et al., 2023), as well as in robotics with UniSim (Yang et al., 2023) and RT-2 (Brohan et al., 2023).

However, world models have a long history in model-based reinforcement learning (RL), which uses them to take optimal actions in the environment under consideration. Typically, models are leveraged in one of two ways: by planning over a finite horizon to estimate the next optimal action (Schrittwieser et al., 2020), or by rolling out the target policy in order to generate synthetic training experi-

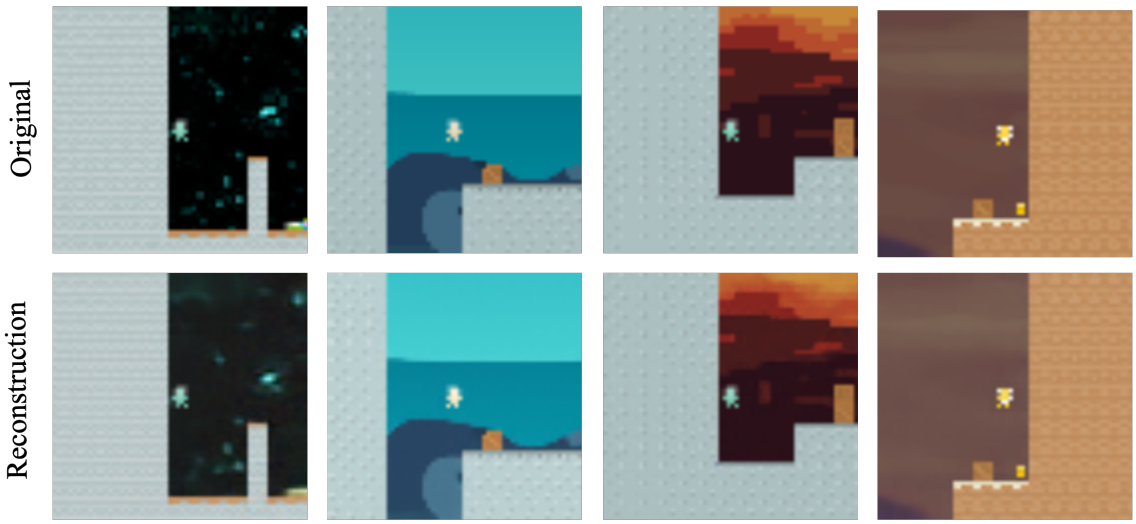

Figure 3. We demonstrate that our tokenizer reconstructs input images with high visual quality across diverse backgrounds.

Table 1. Genie performance metrics for the tokenizer, LAM, and dynamics model.

Component	Metric	Value
Tokenizer	Log-Loss	-2.8
	PSNR	35.8
	SSIM	0.95
	VQ Code Usage	0.57
	VQ Norm. Entropy	0.77
LAM	Log-Loss	-1.7
	PSNR	28.5
	VQ Code Usage	1.0
	VQ Norm. Entropy	0.89
	VQ Norm. Perplexity	4.9
Dynamics Model	Log-Loss	-2.8
	Top-1 Accuracy	0.55
	Top-5 Accuracy	0.84
	Top-16 Accuracy	0.92

ence (Hafner et al., 2019). Equally, world models are a key component in many predominant offline RL methods, where out-of-sample generalisation is an issue. Many methods learned *conservative* models to mitigate distribution shift (Yu et al., 2020; Kidambi et al., 2020), with recent work analysing these approaches (Lu et al., 2021; Sims et al., 2024) and leveraging diffusion models with policy guidance to improve generation quality (Jackson et al., 2024).

Genie's problem setting has parallels to procedural content generation (Summerville et al., 2018; Risi & Togelius, 2020; Liu et al., 2021, PCG). In contrast to RL, PCG studies the generation of engaging game content for a human user, rather than effective training data for a policy or a mechanism for planning. Also in contrast to traditional RL settings, Genie learns a world model from *unlabelled* training data. This requires a LAM (Edwards et al., 2019; Ye et al., 2022; Schmidt & Jiang, 2023), which learns a controllable parameterization of the environment dynamics. In the semi-supervised setting, where a small subset of action-labelled experience is provided, inverse dynamics models have been used to generate synthetic labels for a larger dataset (Baker et al., 2022).

6. Conclusion

We present "Jafar", an open-source implementation of Genie, a method for controllable generation of video from unlabelled training data. Training on a single GPU, we demonstrate coherent and controllable videos of the CoinRun environment. By releasing our implementation and all training details for Jafar, we provide a basis for academic and low-budget labs to meaningfully contribute to the field of controllable video generation. For example, can Jafar be extended to multi-agent environments (Willi et al., 2022; Rutherford et al., 2024)? Does Jafar demonstrate similar few-short learning capabilities as large language models? What are further architectural improvements to make Jafar more capable or efficient? Can we learn policies in Jafar's generative environments that transfer to the original environments?

References

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

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Bruce, J., Dennis, M., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., et al. Genie: Generative interactive environments. *arXiv preprint arXiv:2402.15391*, 2024.

Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Edwards, A., Sahni, H., Schroecker, Y., and Isbell, C. Imitating latent policies from observation. In *International conference on machine learning*, pp. 1755–1763. PMLR, 2019.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018a.

Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018b.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. Flax: A neural network library and ecosystem for JAX, 2023. URL http://github.com/google/flax.

Hong, W., Ding, M., Zheng, W., Liu, X., and Tang, J. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

Hu, A., Russell, L., Yeo, H., Murez, Z., Fedoseev, G., Kendall, A., Shotton, J., and Corrado, G. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

Jackson, M. T., Matthews, M. T., Lu, C., Ellis, B., Whiteson, S., and Foerster, J. Policy-guided diffusion, 2024.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21810–21823. Curran Associates, Inc., 2020.

Liu, J., Snodgrass, S., Khalifa, A., Risi, S., Yannakakis, G. N., and Togelius, J. Deep learning for procedural content generation. *Neural Computing and Applications*, 33(1):19–37, 2021.

Lu, C., Ball, P. J., Parker-Holder, J., Osborne, M. A., and Roberts, S. J. Revisiting design choices in offline model-based reinforcement learning. *arXiv preprint arXiv:2110.04135*, 2021.

Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. Action-conditional video prediction using deep networks in atari games. *Advances in neural information processing systems*, 28, 2015.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Risi, S. and Togelius, J. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8):428–436, 2020.

Rutherford, A., Ellis, B., Gallici, M., Cook, J., Lupu, A., Ingvarsson, G., Willi, T., Khan, A., Schroeder de Witt, C., Souly, A., et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 2444–2446, 2024.

Schmidt, D. and Jiang, M. Learning to act without actions. *arXiv preprint arXiv:2312.10812*, 2023.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Sims, A., Lu, C., and Teh, Y. W. The edge-of-reach problem in offline model-based reinforcement learning, 2024.

Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A. K., Isaksen, A., Nealen, A., and Togelius, J. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270, 2018.

Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Willi, T., Letcher, A. H., Treutlein, J., and Foerster, J. Cola: consistent learning with opponent-learning awareness. In *International Conference on Machine Learning*, pp. 23804–23831. PMLR, 2022.

Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G.-J., and Xiong, H. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

Yang, M., Du, Y., Ghasemipour, K., Tompson, J., Schuurmans, D., and Abbeel, P. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

Ye, W., Zhang, Y., Abbeel, P., and Gao, Y. Become a proficient player with limited data through watching pure videos. In *The Eleventh International Conference on Learning Representations*, 2022.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

A. Appendix

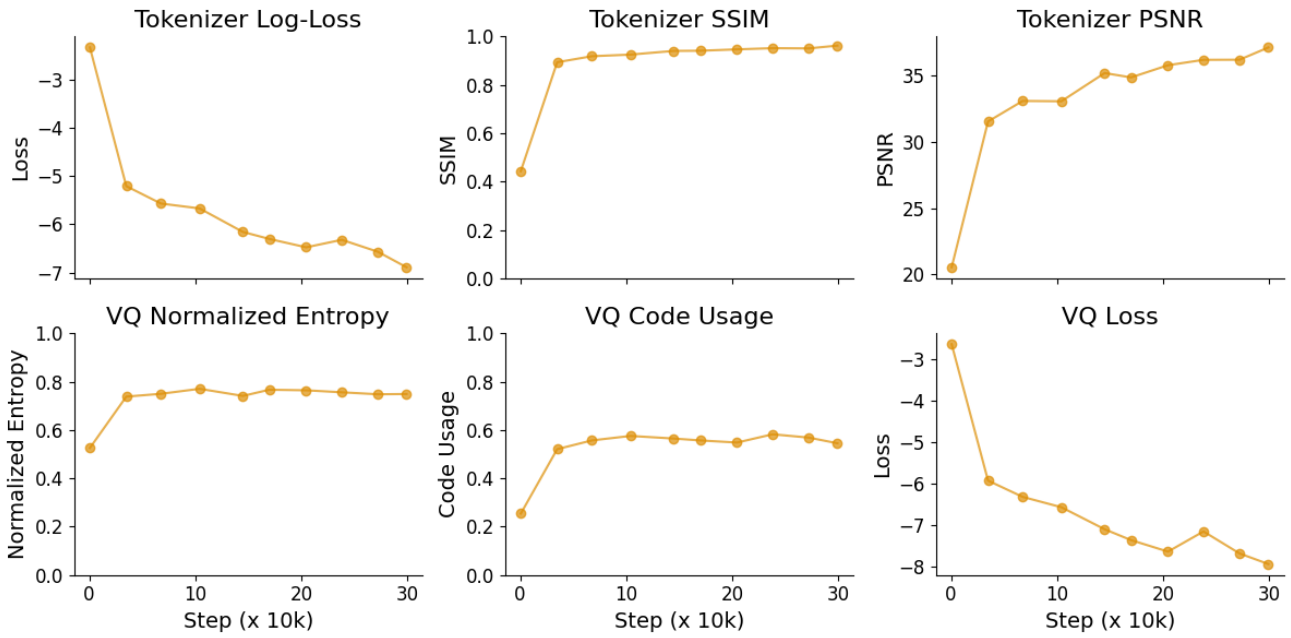

Figure 4. We report several metrics to reproduce our tokenizer performance. The original genie paper does not report such results, except for PSNR in Appendix C.2, Table 6, reporting values around 36, which we match. We also note that VQ Code Usage is fairly low compared to the code usage in the LAM. We have yet to investigate the role of code usage in the tokenizer.

Component	Parameter	Value
Architecture	num_layers	8
	d_model	512
Sampling	temperature	1.0
	maskgit_steps	25

Table 2. Dynamics model parameters as reported in Appendix F3, Table 17 in the original paper

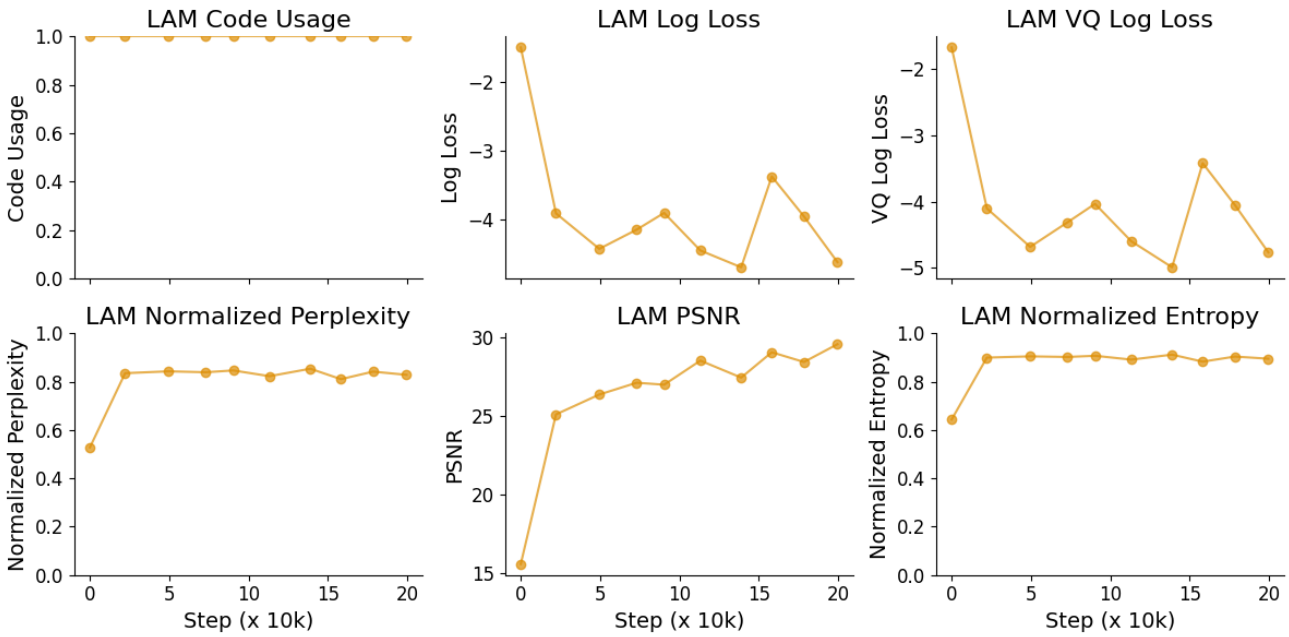

Figure 5. We report several metrics to reproduce our LAM performance. The original genie paper does not report such results. We note that VQ Code Usage is high at 1.0, suggesting that all 6 dimensions are being used to encode latent actions. The PSNR is around 30, which is lower than the PSNR for the tokenizer.

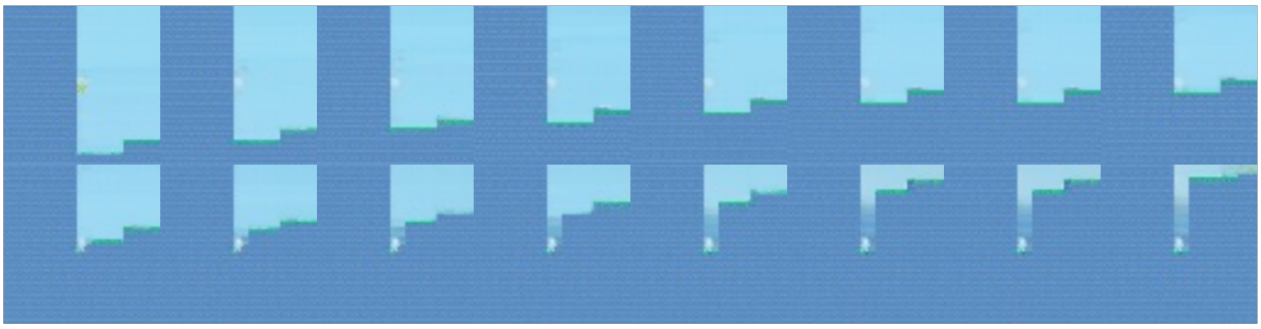

Figure 6. In this example, we sample 16 frames, starting at the top left. The agent starts in the air and moves down. As it hits the floor, the agent starts digging a hole, an action that is not present in CoinRun. Note that the environment stays consistent across frames.

Component	Parameter	Value
Encoder	num_layers	8
	d_model	512
	num_heads	8
Decoder	num_layers	8
	d_model	512
	num_heads	8
Codebook	num_codes	6
	latent_dim	32

Table 3. LAM hyperparameters as reported in Appendix F.3, Table 16. We use the same hyperparameters for our training.

Component	Parameter	Value
Encoder	num_layers	8
	d_model	512
	num_heads	8
Decoder	num_layers	8
	d_model	512
	num_heads	8
Codebook	num_codes	1024
	patch_size	16
	latent_dim	32

Table 4. Tokenizer hyperparameters as reported in Appendix F.3, Table 15. We use the same hyperparameters for our training, except for patch size, where we use 16 instead of 4.

Parameter	Value
max_lr	3e-4
min_lr	3e-4
β_1	0.9
β_2	0.9
weight_decay	1e-4
warmup_steps	10k

Table 5. Optimizer hyperparameters for the tokenizer model as reported in Appendix C.2, Table 8.

Parameter	Value
max_lr	3e-5
min_lr	3e-6
β_1	0.9
β_2	0.9
weight_decay	1e-4
warmup_steps	5k

Table 6. Optimizer hyperparameters for the LAM and dynamics model as reported in Appendix C.2, Table 9.