Pythia: An Edge First Agent for State Prediction in High-Dimensional Environments

Andreas Karatzas, Iraklis Anagnostopoulos, *Member, IEEE*

***Abstract*—Modern deep learning agents usually operate in low-dimensional environments. They process pixel input, don't offer insights into their thought process, and require significant power and computational resources. These characteristics make them inapplicable for embedded devices. In this letter, we present Pythia, an edge-first framework that uses latent imagination to handle complex environments efficiently and envision future agent states. It utilizes a VQ-VAE to reduce the high-dimensional features into a low-dimensional space, making it ideal for modern embedded devices. Moreover, Pythia offers human interpretable feedback and scales well with respect to the design space. Pythia surpassed the other state-of-art models in prediction accuracy on both intrinsic and extrinsic metrics.**

***Index Terms*—Latent Imagination, Edge Inference, Explainability, Embedded Deep Learning, Self Attention**

I. Introduction and Related Work

Modern Reinforcement Learning (RL) agents map states to actions to maximize a numerical signal, i.e., the reward. In deep RL, the model that acts upon the environment can capitalize on the accurate representation of future states [1]. This is found in *model-based* RL, where *world models* envision future states of the world to solve an environment. A world model enables an agent to plan ahead explicitly and to act by "imagining" the long-term outcomes of its actions [2], thus adding behavioral flexibility and offering *feedback on their reasoning process* [3]. Such feedback could enable explainable agents at the edge. For example, we refer to wildfire mitigation and evacuation planning use case. Specifically, reliable agents can be deployed in real-world settings to plan the evacuation in case of fire [4]. Most reinforcement learning agents cannot provide human feedback on their actions; however, a model-based RL agent can output an estimation of the future environment state in pixel space, thereby enhancing interpretability. While many works attempt to improve the prediction accuracy of world models, they mostly require computationally expensive planning mechanisms, thus introducing new challenges and prohibiting their usage in more demanding tasks. They process raw pixel features, and when coupled with a large action space, the complexity of managing the combined state-action space increases exponentially. These limitations render such methods inapplicable for embedded devices, which are resource-constrained. To that end, latent-based models could aid in tackling the *curse of dimensionality* by mapping the input image to a compressed one.

A. Karatzas and I. Anagnostopoulos are with the School of Electrical, Computer and Biomedical Engineering, Southern Illinois University, Carbondale, IL 62901 USA.

Corresponding author: Andreas Karatzas (andreas.karatzas@siu.edu).

Regarding latent-based models, authors in [5] propose an auto-encoder consisting of fully connected layers. However, the number of parameters grows exponentially, rendering this approach impractical in high-dimensional pixel spaces. A similar approach is described in [6], in which the authors use a variational auto-encoder [7] to capture patterns in the latent space. Towards designing agents that can operate in complex environments, authors in [8] propose several deep neural networks that mainly consist of LSTM cells and convolutional layers. However, the convolutional encoder used in their framework to extract features is efficient only in low-dimensional environments. Authors in [9] are the first to introduce a curiosity-driven model. Still, the environment observations are 42×42 images. Authors in [10] train an ensemble of Generative Adversarial Networks (GANs) to estimate the agent's future states. However, this framework performs well in low-dimensional environment space.

In this paper, we present Pythia, an *edge-first* framework based on latent imagination that can replace standard world models and boost the accuracy of long-term future state prediction in high-dimensional RL environments. The contributions of our work are manyfold: ❶ Pythia is an edge-first world model that operates in latent space, reducing computational and power requirements, thus boosting run-time performance. ❷ Pythia provides human interpretable feedback by reconstructing its predictions back into pixel space. ❸ Pythia is tested on a high-dimensional and complex open-world environment that spawns many possible goals.

II. Proposed Methodology

Figure 1 depicts an architectural overview of Pythia, which consists of: **(i)** an encoder, which transforms the input image from pixel to latent space; **(ii)** a $2D$ conditional U-Net, which envisions the next state in latent space; and **(iii)** a decoder, which transforms the latent features back to human-interpretable pixel features. Pythia takes as input a sequence of consecutive captured frames and an action vector associated with the last frame. First, Pythia encodes the input pixel elements into a latent tensor. Then, this tensor is combined with the input action vector to yield one final latent tensor representing Pythia's encoded prediction of the agent's next state. Finally, Pythia reconstructs the pixel matrix out of the predicted latent features.

Encoder: The first module of Pythia is the encoder module from a Vector Quantized Variational Auto-Encoder (VQ-VAE) [11]. An Auto-Encoder is a type of neural network that compresses its input into a lower dimensional space, the *latent space*. In Pythia, the encoder transforms the input image from

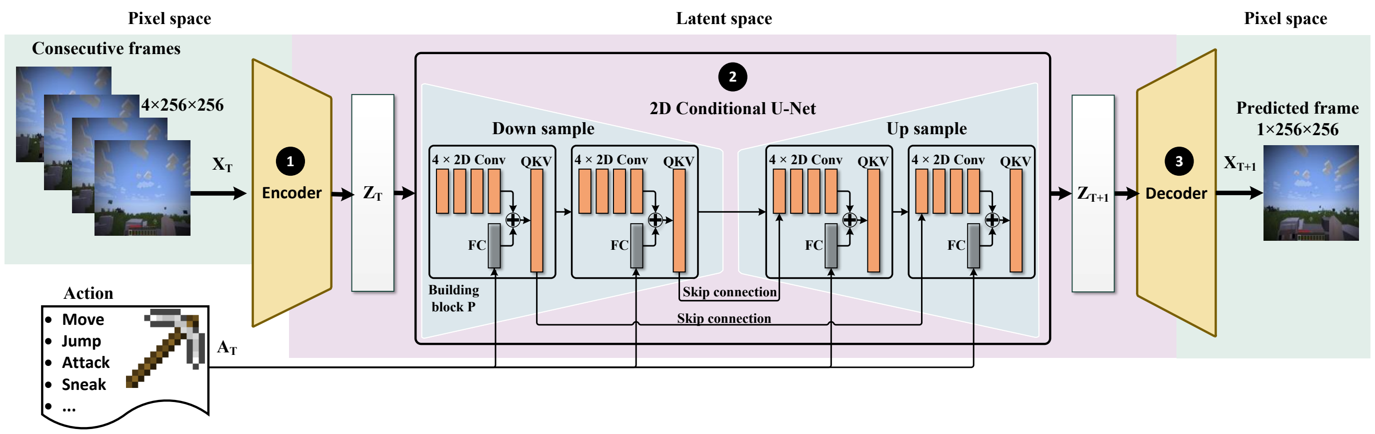

Fig. 1. Architectural overview of Pythia. Pythia employs an encoder, a $2D$ conditional U-Net, and a decoder.

pixel space (X_t) into latent space (Z_t). In particular, Pythia applies vector quantization, i.e., the continuous latent vectors [7] are mapped to a finite set of vectors. Quantization suits the model for tasks that benefit from discrete representations, such as computer vision tasks. The encoder maps the pixel input X_t to a sequence of discrete latent variables Z_t using (**i**) 3 convolutional layers to extract the high-level features; and (**ii**) 18 residual layers to capture the lower-level features. We choose a low layer count to address the limited computational resources and power constraints commonly encountered in embedded devices. We iteratively input our encoder module with 4 consecutive frames of size 256×256 to yield a latent tensor of size $16 \times 64 \times 64$. We also apply frame skipping because few changes happen between two neighboring frames. Regarding the number of stacked and skipped frames, we build upon the findings of [12] and set them equal to 4.

2D Conditional U-Net: The second module of Pythia is a $2D$ Conditional U-Net model [13]. In Pythia, we employ a U-Net variant, namely $2D$ Conditional U-Net, to process Z_t, which represents the observation of the agent at time t in the latent space, and predict Z_{t+1}, which is the observation at the next timestep. To accurately transpose the latent features Z_t into the latent features Z_{t+1} for time step $t+1$, we feed into the $2D$ Conditional U-Net the action vector a_t of time step t. Our model comprises primary blocks P to process, which enable it to synergistically correlate tensor data (Z_t) with vector data (a_t). Each block P is composed of: (**i**) a stream of four 2D convolutional layers with group normalization and GELU activation, which process tensor data (Z_t); (**ii**) a stream of a SiLU activation and a Fully Connected (FC) layer, which transposes the vector data (a_t) into an embedding; (**iii**) a layer that sums the output of the two streams; and (**iv**) a self-attention mechanism (QKV) with 4 heads and GELU activation. Our $2D$ Conditional U-Net has 4 building blocks P, equally distributed amongst a down- and an up-sampling module.

Decoder: As aforementioned, the latent space is a compressed space that aids in the reduction of the computational requirements that Pythia projects to the embedded device. Therefore, it is not an interpretable space. The decoder is responsible for the human-interpretable feedback by reconstructing the dynamic latent data Z_{t+1} back in pixel space X_{t+1}. The architecture of the decoder module is symmetric to the architecture of the encoder in the VQ-VAE. Returning to pixel space is an important feature of Pythia since it provides human interpretable insight and addresses black-box architectures. Our VQ-VAE decoder module achieves that by transforming the latent prediction from our $2D$ Conditional U-Net to an image.

III. Experimental Evaluation

We assess the efficiency of Pythia against three widely deployed models: (**i**) U-Net [13], which features a contracting path to capture pixel features and an expanding path that manipulates these features; (**ii**) VAE [7], using ResNet-18 modules; and (**iii**) VQ-VAE2 [14], similar to the VQ-VAE employed in Pythia. These three models satisfy the computational constraints of modern embedded devices and can function as world models for model-based RL agents. We also evaluated other world models [15], [16] that predict the next environment state based on a sequence of frames and an agent's action trajectory. However, their computational requirements made them unfit for embedded devices. Regarding the training of all models, we utilized the dataset from the BASALT (Benchmark for Agents that Solve Almost-Lifelike Task) competition 2022 [17]. To the best of our knowledge, BASALT is the only dataset with unique complexity due to the open-world setting while providing information regarding the agent's actions, thus inspiring research in *imitation learning* in open-world environments. BASALT provides $13,928$ videos of Minecraft gameplay split into four tasks: (**i**) `FindCave`: The agent searches for a cave ($1,399$ videos); (**ii**) `MakeWaterfall`: The agent builds a waterfall ($2,833$ videos); (**iii**) `CreateVillageAnimalPen`: The agent builds an animal pen containing several animals ($5,466$ videos); and (**iv**) `BuildVillageHouse`: The agent builds a new house ($4,230$ videos). Each video comprises 640×360 RGB frames at 20 per second (FPS). We down-scaled the frames to 256×256 while applying frame skipping and stacking [12] to reduce the training complexity. This maintains the vast state space while aiding the training process of our framework. The environment properties inherit those of an open world, which means that the entropy of the different agent episodes is expected to be high. After pre-processing the frames, each observation input has a size of $4 \times 256 \times 256$.

TABLE I
THE COMPLETE ACTION SPACE FOR EACH FRAME IN THE DATASET.

Action name	Span	Description
ESC	Discrete(2)	Returns to the menu and then back to the game (key ESC)
Attack	Discrete(2)	Start/Stop attacking/destroying (mouse 1)
Move on x-axis	Discrete(4)	Forward (W), backward (S), stand still (no key press), or press W and S at the same time
Move on y-axis	Discrete(4)	Strafe left (A), Strafe right (D), stand still (no key press), or press A and D at the same time
Hotbars	Discrete(2^9)	Toggle a hotbar (9 in total, keys 1-9)
Inventory	Discrete(2)	Toggle the inventory (key E)
Jump	Discrete(2)	Trigger a character jump (key SPACE)
Pick item	Discrete(2)	Trigger the pick action (mouse 2)
Sneak	Discrete(2)	Trigger the sneak/crouch action (key SHIFT)
Sprint	Discrete(2)	Trigger the sprint action (key CTRL)
Swap hands	Discrete(2)	Swapping hands of the character (key F)
Use	Discrete(2)	Use item/place block (mouse 3)
Camera	Discrete(360^2)	Rotate the camera 360° in x- and y-axis (mouse movement)

Additionally, Pythia also utilizes action vectors described in Table I. Each frame is accompanied by one such action vector, compiling an action space of more than half a billion possible combinations. So, combining the size of the input $(4 \times 256 \times 256)$ with the vast action space, Pythia, to the best of our knowledge, is the first agent to ❶ operate on such a diverse and high-dimensional state-action space while ❷ maintaining low computational overhead. To further reduce our framework's power and compute requirements, we leveraged PyTorch Automatic Mixed Precision Package to quantize our model's parameters to 16-bit floating precision and fine-tune it for 5 epochs using 20% of the training dataset. We selected the Adam optimizer and configured the learning rate with `StepLR` scheduler to reduce the learning rate by half after 10 epochs, to introduce smaller updates to the weights and aid in better convergence and model performance as training progresses. We compiled a dataset of $50,000$ state-action pairs and trained Pythia for 20 epochs. We maintained the criterion function for the VQ-VAE module of Pythia and chose $L2$ for the $2D$ Conditional U-Net. We kept the original published criteria functions for the U-Net, the VAE, and the VQ-VAE2.

Quantitative Evaluation: We compare Pythia against: (**i**) U-Net; (**ii**) VAE; and (**iii**) VQ-VAE2. For this purpose, we sampled a sequence of consecutive images from our test set and fed it to each model to predict the agent's trajectory. Since our goal is to dive as deep as possible into the agent's future states, we performed the following steps: (**i**) Initially, we fed each model with four consecutive 256×256 frames from the original test set (e.g., F^1_{true}, F^2_{true}, F^3_{true}, and F^4_{true}); (**ii**) Then, we captured each model's prediction (e.g., F^5_{pred}); (**iii**) We used this frame as input (e.g., F^2_{true}, F^3_{true}, F^4_{true}, and F^5_{pred}), to get the next prediction (e.g., F^6_{pred}); (**iv**) We repeated this process 50 times. After the first four predictions, each model operated solely on the predicted and reconstructed frames.

Figure 2 depicts the performance of Pythia, U-Net, VAE, and VQ-VAE2 for 50 consecutive reconstructed frames on 4 different metrics: (**i**) Mean Absolute Error (MAE) is a metric that captures the model's prediction accuracy in regression

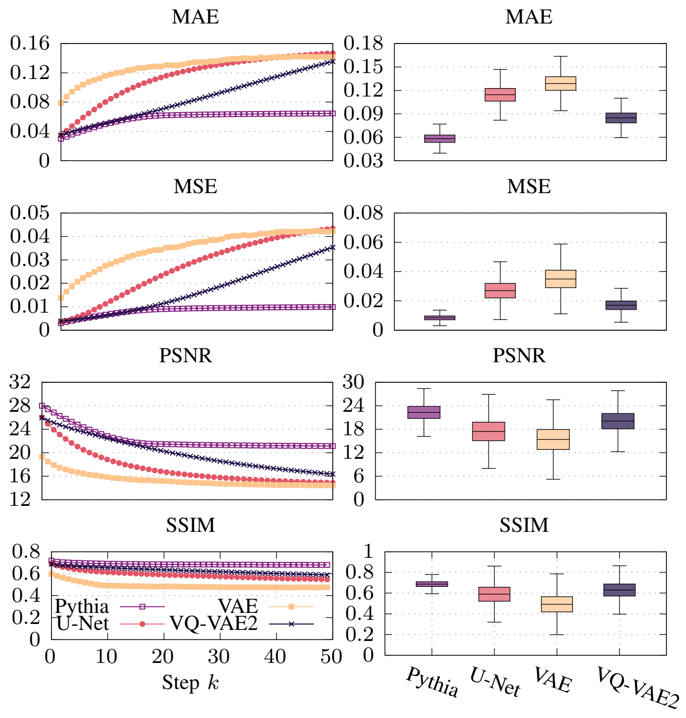

Fig. 2. Evaluation of Pythia, U-Net, VAE, and VQ-VAE2, on predicting future agent states under $\approx 30,000$ unseen images. Initially, each model is fed with 4 consecutive frames. **In the left column**, the models envision the agent's state for $k = 50$ steps ahead. **In the right column**, we provide insight into the models' reliability in terms of error mean and standard deviation.

tasks. (**ii**) Mean Squared Error (MSE) is widely used in regression problems. It behaves leniently towards low errors and aggressively towards high errors. (**iii**) Peak Signal to Noise Ratio (PSNR) is a variant of MSE. It is an extrinsic evaluation metric and evaluates Gaussian blurriness and white noise. Higher PSNR equals better image quality. (**iv**) Structural Similarity Index Measure (SSIM) measures the similarity concerning human perception [18]. SSIM is defined in the $[0, 1]$ range, and as SSIM increases, the image quality improves.

Pythia was the only model that accurately predicted the agent's future states. In contrast, the other models mainly produced predictions almost identical to the input frames without capturing any changes caused by the agent's actions. In particular, the accumulation of the reconstruction differences resulted in inaccurate predictions over time. This can be seen in the long-term predictions of U-Net, VAE, and VQ-VAE2. VAE performed the worst, indicating that conventional convolutional architectures are inadequate for reconstructing large inputs. VQ-VAE2 kept reconstructing the last frame of the input (i.e., $F^5_{pred} \equiv F^4_{true}$), leading to error curves that never converge.

Accuracy: Pythia outperformed U-Net, VAE, and VQ-VAE2 with significant margins in terms of MAE. On average, Pythia achieved a superiority of 96.1%, $2.21\times$, and 45.3% over U-Net, VAE, and VQ-VAE2, respectively. This performance advantage is higher for long-term predictions ($k \geq 30$), as the improvements increase to $2.19\times$, $2.21\times$, and 81%. Regarding MSE, Pythia again surpassed U-Net, VAE, and VQ-VAE2 by $3.19\times$, $4.13\times$, and $2.1\times$, respectively. Extrinsic evaluation metrics complemented these findings. Pythia achieved a 28.1%, 45.2%,

TABLE II
RUN-TIME PERFORMANCE COMPARISON.

Module	**Throughput** $\left(\frac{frames}{sec}\right)$		**Power** (W)	
	AGX	*Orin*	*AGX*	*Orin*
Pythia	18.5	47.4	15.4	7.4
U-Net	17.1	23.9	22.9	12.8
VAE	9.7	14.2	22.1	12.9
VQ-VAE2	16.8	38.2	15.1	9.2

and 11.4% improvement over U-Net, VAE, and VQ-VAE2, respectively, in terms of PSNR. For long-term predictions, the respective improvements increased to 39.3%, 46.1%, and 22.4%. Additionally, Pythia exhibited a higher average SSIM, outperforming U-Net, VAE, and VQ-VAE2 by 16.5%, 39.7%, and 9.3%, respectively. Overall, Pythia associates the action vector with the agent observations and can thus accurately predict the agent's trajectory.

Reliability: Regarding MAE, Pythia and VQ-VAE2 do not demonstrate big fluctuations, with Pythia achieving, on average, 31.7% better MAE. Pythia increases that advantage in MSE with 52.9% better MSE on average compared to the VQ-VAE2. The increase in performance regarding MSE is attributed to Pythia's ability to avoid large prediction errors. The standard deviation is augmented regarding PSNR for all frameworks. Still, Pythia maintains its advantage with 11% better PSNR on average compared to VQ-VAE2. Finally, regarding SSIM, Pythia has a standard deviation of only 0.035, thus further validating the claim of small prediction error. On the contrary, VQ-VAE2 has a standard deviation of 0.087, rendering it unreliable for long-term predictions.

Runtime Evaluation: To assess Pythia's compute and power requirements, we utilized the NVIDIA AGX Xavier board and the NVIDIA Orin Developer Kit. The AGX features an 8-core ARM CPU, a 512-core Volta GPU, and 32GB of high-speed memory, while the Orin is equipped with a 6-core ARM CPU, a 1024-core Ampere GPU, and 8GB of high-speed memory. Table II depicts the achieved throughput in terms of FPS and corresponding power consumption on each board. Power consumption includes both GPU and CPU power. Pythia achieved the highest throughput over both boards among all the methods. This validates our design choices, such as using a lighter residual encoder-decoder block and fewer attention heads in our $2D$ Conditional U-Net. Additionally, weight quantization contributed to Pythia's exceptional runtime performance, particularly in the case of the Orin board. Regarding power consumption, Pythia achieved $\sim 15W$ and $7.4W$ on the AGX and Orin, respectively, considerably less than U-Net and VAE. Pythia falls in second place to VQ-VAE2 when deployed on AGX due to the attention mechanisms in our $2D$ Conditional U-Net, but exhibits $\sim$ 20% power reduction when deployed to the Orin due to newer GPU architecture that leverages mixed precision to deliver more efficient performance. Overall, Pythia strikes a balance between hardware requirements and prediction performance.

IV. Conclusion and Future Work

In this paper, we introduced Pythia, a framework that leverages latent imagination to effectively handle high-dimensional environment spaces and provide accurate predictions of future agent states. Pythia is designed to have a low resource footprint, enabling deployment on modern embedded devices. Pythia primarily focuses on the observation sequence over the input action vector for frame prediction, leading to some inaccuracies, especially during fast movements. This issue likely stems from the action vector's inadequate representation. Plans include enhancing the action vector with a higher-dimensional space. Furthermore, Pythia's tendency to forget previously seen objects will be addressed by upgrading our diffusion U-Net into a multi-modal transformer to improve memory.

Acknowledgments

This work is supported by grant NSF CCF 2324854.

References

[1] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.
[2] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International conference on machine learning*. PMLR, 2019, pp. 2555–2565.
[3] S. Racanière *et al.*, "Imagination-augmented agents for deep reinforcement learning," *Advances in neural information processing systems*, 2017.
[4] A. Tapley, M. Dotter, M. Doyle, A. Fennelly, D. Gandikota, S. Smith, M. Threet, and T. Welsh, "Reinforcement learning for wildfire mitigation in simulated disaster environments," *arXiv preprint arXiv:2311.15925*, 2023.
[5] B. C. Stadie *et al.*, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv preprint arXiv:1507.00814*, 2015.
[6] M. Watter *et al.*, "Embed to control: A locally linear latent dynamics model for control from raw images," *Advances in neural information processing systems*, 2015.
[7] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[8] K. Gregor *et al.*, "Shaping belief states with generative environment models for rl," *Advances in Neural Information Processing Systems*, 2019.
[9] D. Pathak *et al.*, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017.
[10] H. Charlesworth and G. Montana, "Plangan: Model-based planning with sparse rewards and multiple goals," *Advances in Neural Information Processing Systems*, 2020.
[11] A. Van Den Oord *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, 2017.
[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature*, 2015.
[13] O. Ronneberger *et al.*, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015.
[14] A. Razavi *et al.*, "Generating diverse high-fidelity images with vq-vae-2," *Advances in neural information processing systems*, 2019.
[15] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.
[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[17] "Basalt competition 2022," https://minerl.io/basalt/.
[18] Y. Lu, "The level weighted structural similarity loss: A step away from mse," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.