Improving Token-Based World Models with Parallel Observation Prediction

April 09, 2025

Reference

Lior Cohen, Kaixin Wang, Bingyi Kang, and Shie Mannor. Improving token-based world models with parallel observation prediction, 2024. URL https://arxiv.org/abs/2402.05643.

Overview

- REM [Cohen et al., 2024] builds on IRIS, following a $\mathcal{V}$-$\mathcal{M}$-$\mathcal{C}$ structure:
  - $\mathcal{V}$: Visual perception module compressing observations
  - $\mathcal{M}$: Predictive model capturing environment dynamics
  - $\mathcal{C}$: Controller learning to maximize return
- Uses replay buffer to store environment interaction data

$\mathcal{V}$ - Tokenizer

- Implemented as a VQ-VAE discrete auto-encoder
- Components:
  - Encoder: Maps input image $\mathbf{o}_t$ to latent vectors $(\mathbf{h}_t^1, \mathbf{h}_t^2, \cdots, \mathbf{h}_t^K)$
  - Embedding table: $\mathcal{E} = \{\mathbf{e}_i\}_{i=1}^N \in \mathbb{R}^{N \times d}$ with N trainable vectors
  - Tokens: $z_t^k = \arg\min_i \|\mathbf{h}_t^k - \mathbf{e}_i\|$
  - Decoder: Maps token embeddings back to reconstructed observation $\hat{\mathbf{o}}_t$
- Trained on frames sampled uniformly from replay buffer

$\mathcal{M}$ - World Model

- Models environment dynamics in latent token space
- Predicts three distributions at each step t:
  - Transition: $p(\hat{\mathbf{z}}_{t+1} | \mathbf{z}_1, a_1, \ldots, \mathbf{z}_t, a_t)$
  - Reward: $p(\hat{r}_t | \mathbf{z}_1, a_1, \ldots, \mathbf{z}_t, a_t)$
  - Termination: $p(\hat{d}_t | \mathbf{z}_1, a_1, \ldots, \mathbf{z}_t, a_t)$
- Uses code vectors $\mathcal{E}$ learned by tokenizer (not updated by $\mathcal{M}$)
- Maintains dedicated embedding tables for actions and special tokens

$\mathcal{C}$ - Controller & Chunkwise Computation

- Actor-critic controller trained in imagination
- Components:
  - Policy network π
  - Value function estimator V^{π}
- Training process:
  - Initialize with trajectory from replay buffer
  - Interact with world model for H steps
  - Sample actions from policy
  - World model generates rewards, terminations, and next tokens
  - Use trajectories to train agent
- Computational efficiency:
  - "Chunkwise" computation splits sequences into smaller chunks
  - Previous chunks summarized by recurrent state **S**
  - Enhanced by POP extension for efficient training

Retention Architecture

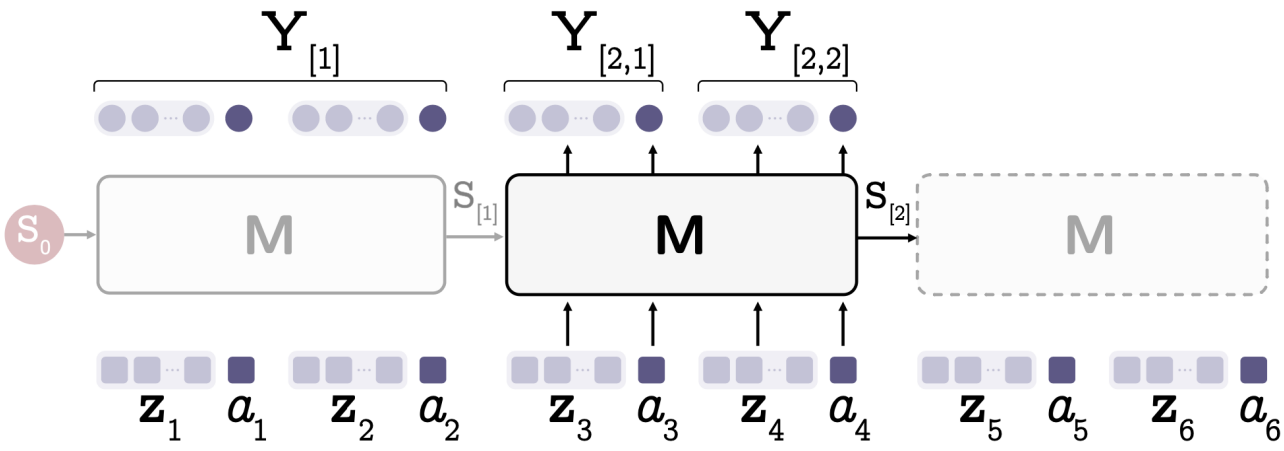

Figure: Retention mechanism with chunkwise computation.

Retention Formulation

- For token sequence $(x_1, x_2, \cdots, x_m)$ with embeddings $\mathbf{X} \in \mathbb{R}^{m \times d}$
- In RL context: sequence is token trajectory of observation-action blocks
- Split into chunks of B tokens where $B = c(K+1)$
- Retention output for i-th chunk:

$$\mathbf{Y}_{[i]} = \left(\mathbf{Q}_{[i]}\mathbf{K}_{[i]}^{\top} \odot \mathbf{D}\right)\mathbf{V}_{[i]} + (\mathbf{Q}_{[i]}\mathbf{S}_{[i-1]}) \odot \boldsymbol{\xi}$$

- Recurrent state update:

$$\mathbf{S}_{[i]} = (\mathbf{K}_{[i]} \odot \zeta)^{\top}\mathbf{V}_{[i]} + \eta^{B}\mathbf{S}_{[i-1]}$$

Retention Notation

- RetNet model consists of stacked layers with Retention mechanism
- Retention has dual form of recurrence and parallelism ("chunkwise")
  - Splits long sequences into smaller chunks
  - Parallel computation within chunks
  - Sequential recurrent form between chunks
- Information from previous chunks summarized by recurrent state $\mathbf{S} \in \mathbb{R}^{d \times d}$
- Retention mechanism formulation:
  - where the bracketed subscript $[i]$ is used to index the i-th chunk
  - $\mathbf{Q} = (\mathbf{X}\mathbf{W}_Q) \odot \Theta$
  - $\mathbf{K} = (\mathbf{X}\mathbf{W}_K) \odot \bar{\Theta}$
  - $\mathbf{V} = \mathbf{X}\mathbf{W}_V$
  - $\boldsymbol{\xi} \in \mathbb{R}^{B \times d}$ is a matrix with $\boldsymbol{\xi}_{ij} = \eta^{i+1}$
  - $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$ are learnable weights
  - η is an exponential decay factor
  - the matrix $\mathbf{D} \in \mathbb{R}^{B \times B}$ combines an auto-regressive mask with the temporal decay factor η
  - the matrices $\Theta, \bar{\Theta} \in \mathbb{C}^{m \times d}$ are for relative position embedding

World Model Imagination (Figure)

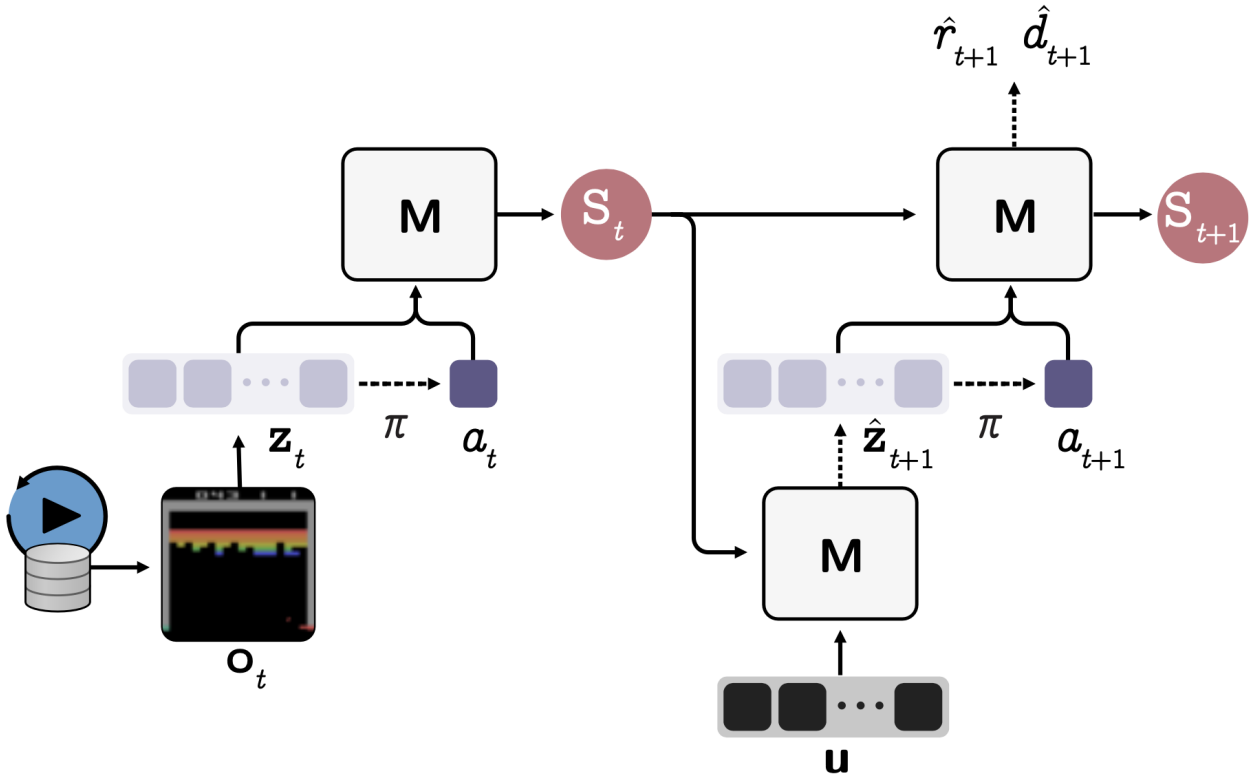

Figure: World model imagination.

World Model Imagination (1/2)

- Agent training relies entirely on world model imagination
- For trajectory generation, we need to predict:
  - Reward $\hat{r}_t$
  - Termination signal $\hat{d}_t$
  - K latent observation tokens $\hat{\mathbf{z}}_{t+1}$
- Predicting $\hat{\mathbf{z}}_{t+1}$ is the computational bottleneck
- In IRIS, token prediction is sequential:
  - Each token predicted one at a time
  - Requires KH sequential world model calls
  - Poor GPU utilization and long computation time

World Model Imagination (2/2)

- POP's solution: dedicated prediction tokens
- Maintains set of K prediction tokens $\mathbf{u} = (u_1, \ldots, u_K)$
- To generate $\hat{\mathbf{z}}_{t+1}$ in one pass:
  - Compute RetNet outputs from recurrent state $\mathbf{S}_t$
  - Use $\mathbf{u}$ as input sequence
  - Chunk size limited to single block ($K + 1$)
- Benefits:
  - Reduces world model calls from KH to $2H$
  - Eliminates dependency on number of tokens K
  - Improves scalability at expense of higher overall computation
- Follows trend of favoring scalability over raw computational cost

World Model Training (Figure)

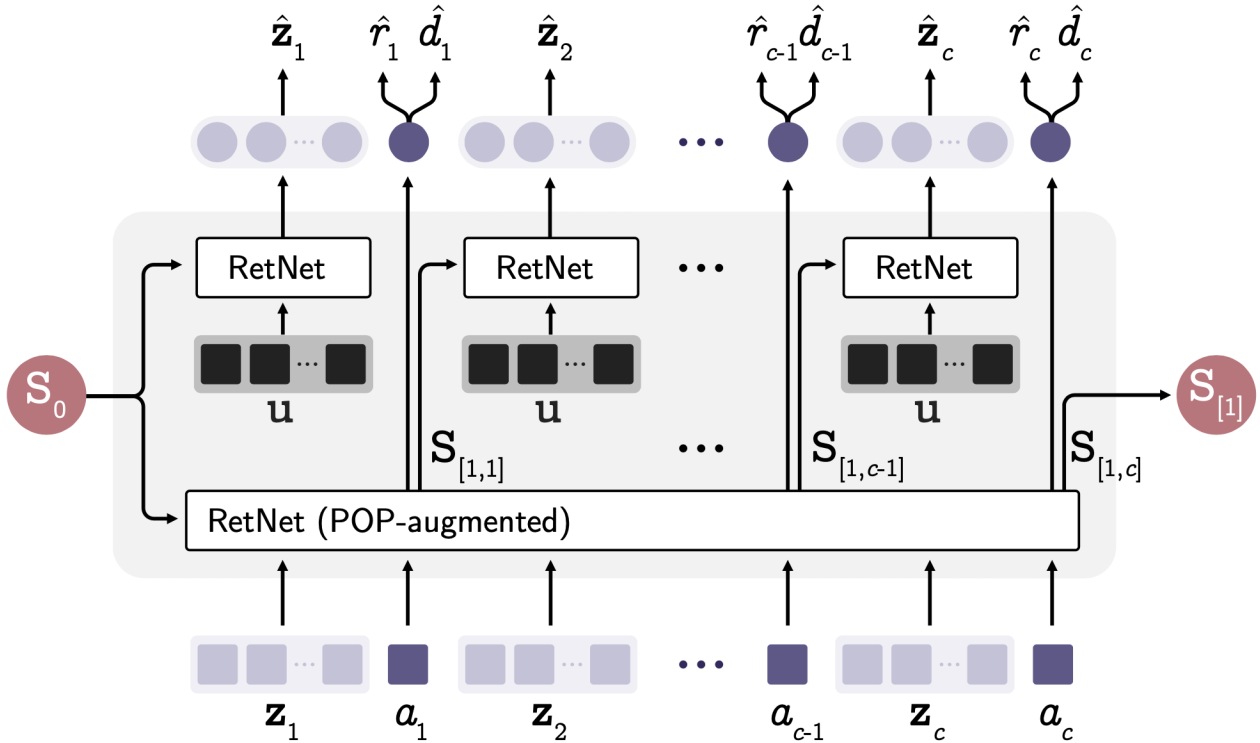

Figure: An illustration of the POP chunkwise forward algorithm (Alg. 1 and 2) for a single-layer model. During training, $\mathcal{M}$ computes the outputs of c observation-action blocks in parallel. Blue squares represent token inputs, while the corresponding RetNet outputs are denoted by circles. Each RetNet block represents a forward call to the same RetNet model. The bottom RetNet call uses our POP extension for computing the additional recurrent states at the end of every observation-action block (Alg. 2, lines 2-7). The top row of RetNet calls are batch-computed in parallel (Alg. 2, line 8). Finally, the output combines the observation token outputs produced by the top RetNet call with the rewards and termination outputs computed by the bottom one (Alg. 1, lines 7-9).

World Model Training (1/3)

- Challenge: POP requires special training data preparation
- For each time step t, model must predict $\mathbf{z}_t$ given:
  - Trajectory prefix $(\mathbf{z}_1, a_1, \ldots, \mathbf{z}_{t-1}, a_{t-1})$
  - Prediction tokens $\mathbf{u}$
- Problem: Cannot simply replace tokens with prediction tokens
  - Original tokens needed for future predictions
  - Standard approach not viable - requirements contradict
- Need efficient method to compute outputs for all time steps in parallel

World Model Training (2/3)

- Solution: Summarize trajectory prefixes into recurrent states
- For chunk $(\mathbf{z}_1, a_1, \ldots, \mathbf{z}_c, a_c)$:
  - $(\mathbf{z}_1, a_1)$ summarized into $\mathbf{S}_{[1,1]}$
  - $(\mathbf{z}_1, a_1, \mathbf{z}_2, a_2)$ summarized into $\mathbf{S}_{[1,2]}$
- Two-step computation to get all recurrent states $\mathbf{S}_{[i,j]}$:
  - Compute intermediate states $\tilde{\mathbf{S}}_{[i,j]}$ in parallel:

$$\tilde{\mathbf{S}}_{[i,j]} = \left(\mathbf{K}_{[i,j]} \odot \zeta\right)^\top \mathbf{V}_{[i,j]}$$

  - Compute final states sequentially with minimal overhead:

$$\mathbf{S}_{[i,j]} = \tilde{\mathbf{S}}_{[i,j]} + \eta^{K+1} \mathbf{S}_{[i,j-1]}$$

- Then predict all next observations from $(\mathbf{S}_{[i,j]}, \mathbf{u})$ tuples

World Model Training (3/3)

- Training process:
  - Sample trajectory segments of H steps from replay buffer
  - Process in chunks of c observation-action blocks
  - Produce modeled distributions using POP chunkwise forward
- Optimization:
  - Minimize cross-entropy loss for transitions and termination
  - For rewards: MSE loss (continuous) or cross-entropy (discrete)
- Key innovation: Extended RetNet to support batched computation with appropriate positional encoding
- Algorithm illustrated in Figure:
  - Bottom RetNet call computes additional recurrent states
  - Top row of RetNet calls batch-computed in parallel
  - Output combines observation token outputs with rewards and termination

Algorithm

Algorithm 1 RetNet POP Chunkwise Forward

Input: chunk size $1 \leq c \leq H$, token embeddings $\mathbf{X}_{[i]}$ of chunk i, per-layer recurrent states $\{\mathbf{S}^{l}_{[i-1]}\}_{l=1}^{L}$.
Initialize $\mathbf{A}^{0}_{[i]} \leftarrow \mathbf{X}$
Initialize $\mathbf{B}^{0}_{[i,1]}, \ldots, \mathbf{B}^{0}_{[i,c]} \leftarrow \mathcal{E}_{\mathbf{u}}, \ldots, \mathcal{E}_{\mathbf{u}}$
for $l = 1$ **to** L **do**
    $\mathbf{A}^{l}_{[i]}, \mathbf{B}^{l}_{[i]}, \mathbf{S}^{l}_{[i]} \leftarrow \text{POPLayer}(\mathbf{A}^{l-1}_{[i]}, \mathbf{B}^{l-1}_{[i]}, \mathbf{S}^{l}_{[i-1]}, i)$
end for
for $j = 1$ **to** c **do**
    $\mathbf{Y}_{[i,j]} \leftarrow \text{Concat}(\mathbf{B}^{L}_{[i,j]}, \mathbf{A}^{L}_{[i,j,K+1]})$
end for
Return $\mathbf{Y}, \{\mathbf{S}^{l}_{[i]}\}_{l=1}^{L}$

Algorithm

Algorithm 2 POPLayer Chunkise Forward

Input: Chunk latents $\mathbf{A}_{[i]}$, observation prediction latents $\mathbf{B}_{[i]}$, recurrent state $\mathbf{S}_{[i-1]}$, chunk index i.
$\mathbf{A}_{[i]} \leftarrow \text{Retention}(\mathbf{A}_{[i]}, \mathbf{S}_{[i-1]}, i)$ (Eqn. 4)
Compute $\tilde{\mathbf{S}}_{[i,1]}, \ldots, \tilde{\mathbf{S}}_{[i,c]}$ in parallel (Eqn. 6)
for $j = 1$ **to** c **do**
    $\mathbf{S}_{[i,j]} \leftarrow \tilde{\mathbf{S}}_{[i,j]} + \eta^{K+1}\mathbf{S}_{[i,j-1]}$ (Eqn. 7)
end for
$\mathbf{S}_{[i]} \leftarrow \mathbf{S}_{[i,c]}$
$\mathbf{B}_{[i,j]} \leftarrow \text{Retention}(\mathbf{B}_{[i,j]}, \mathbf{S}_{[i,j-1]}, [i,j])$ in parallel for $j = 1, \ldots, c$ (Eqn. 4)
Return $\mathbf{A}_{[i]}, \mathbf{B}_{[i]}, \mathbf{S}_{[i]}$