Reinforcement Learning with Complex (valued) Memories

Sathya Kamesh Bhethanabhotla[*]
University of Amsterdam
s.k.bhethanabhotla@uva.nl

Efstratios Gavves
University of Amsterdam

André Biedenkapp[*]
Karlsruhe Institute of Technology
biedenkapp@kit.edu

Abstract

Partially observable environments pose a fundamental challenge in deep reinforcement learning, requiring agents to compress temporal information from observations and maintain a memory to make effective decisions. While there exist many approaches ranging from gated recurrence to attention mechanisms and model-based RL, the search for effective representational techniques that can capture long-term dependencies remains an active area of research. In this work we revisit Unitary recurrent networks (uRNNs) [Arjovsky et al., 2016, Jing et al., 2017a], that demonstrated superior gradient flow and associative recall, expressing the recurrence and the hidden state in a complex vector space. Their norm preserving unitary dynamics enable information propagation through long sequences. To this end, we propose three different versions of uRNNs as drop-in replacements for recurrent PPO architectures, and demonstrate that the simple recurrence and the added degree of freedom from the phase of the complex representations enable significant gains over baselines on several memory-improvable tasks, including continuous control. We further explore how to preserve the phase information of the complex hidden state for a *phase-aware* policy by drawing a parallel to how quantum states are measured. With our methods reaching up to 2-3$\times$ the reward in environments like rocksample and Craftax compared to the baselines, this work points towards an exciting new direction of representations for RL and the problem of partial observability.

1 Introduction

The human mind fundamentally relies on memory to function effectively in our partially observable world. When navigating through an environment, we do not have access to complete information about our surroundings at any given moment. Instead, we must integrate observations over time, maintaining an internal representation of relevant information from the past to inform our decisions and actions. Memory is what makes action possible. Without it, we could neither track a ball in motion, recall where we left our keys hours earlier nor any other task that unfolds over time. In our effort to understand and replicate the mechanics of sequential decision-making, we have turned to frameworks like reinforcement learning [RL; Sutton and Barto, 1998], developing methods that can emulate these aspects of agency.

Deep RL has addressed partial observability [Åström, Karl Johan, 1965, Kaelbling et al., 1998] and memory in a variety of ways in the past decades of work [see, e.g. Lambrechts et al., 2022,

[*]Work done partially at the University of Freiburg.

19th European Workshop on Reinforcement Learning (EWRL 2026).

Ni et al., 2022]. We have also seen the development of various environments that can be used to test different aspects of these problems. From early work on value-based deep learning methods like DQN [Mnih et al., 2015] and the recurrent DRQN [Hausknecht and Stone, 2015] playing Atari games [Bellemare et al., 2013], over policy gradient methods like Trust Region Policy Optimization (TRPO) [Schulman et al., 2017a] and Proximal Policy Optimization (PPO) [Schulman et al., 2017b], to transformer-based methods [Chen et al., 2021] we have seen a lot of progress in the field. Further, model-based approaches such as Dreamer [Hafner et al., 2025], PETS [Chua et al., 2018] or Recall2Imagine [Samsami et al., 2024] build an internal models of the world and use those to plan, enabling them to tackle partial observability. With techniques like gated recurrence [Hochreiter and Schmidhuber, 1997], attention [Vaswani et al., 2017], and state space models [Gu and Dao, 2024], the search for effective representational techniques to compress state information over time remains an active area of research. Despite this progress, the field still struggles with learning instabilities, for example caused by vanishing gradients or the dependence on short horizons.

A decade ago, Unitary Evolution Recurrent Neural Networks [uRNNs; Arjovsky et al., 2016] offered a fresh approach to addressing the challenges of memory and gradient flow in recurrent architectures. uRNNs operate in a complex-valued vector space, using complex-valued parameters and hidden states. The central insight of uRNNs is that constraining the recurrent dynamics to be *unitary*, a norm-preserving transformation that ensures stable evolution of the hidden state over arbitrarily long sequences, and aids associative recall. This extension to the complex domain provides an additional degree of freedom in the form of a phase, allowing for richer representational capacity. In this work, we propose this class of recurrence as a representation paradigm for RL agents acting in partially observable tasks, requiring long term memory. We show how different variants of this recurrent class can be straightforwardly integrated into the recurrent PPO method. Instead of learning real-valued representations these drop-in replacements enable learning complex-valued representations. We empirically show that such complex-valued representations already achieve a significantly better performance across a range of memory improvable tasks.

While complex-valued representations already enable PPO to learn better policies, we further aim to leverage the phase information directly. We propose a novel approach to propagate the phases captured in the complex hidden state to a discrete stochastic policy. By allowing the phases of the complex parameters to interfere constructively and destructively through a Born-rule like sampling [Born, 1984], akin to collapsing a wavefunction, we highlight a potentially deeper connection between policies and quantum states. Evaluating this *Phase Aware* policy shows competitive performance to other baselines over tasks like T-maze, battleship and craftax. Particularly in long-horizon tasks, our approach outperforms canonical recurrent architectures. Our contributions are:

I) We propose a recurrent PPO approach that uses the uRNN as the recurrent cell;

II) We extend the uRNN architecture to enable an input-dependent unitary transformation matrix;

III) We propose a way to interpret the policy from a quantum information perspective, and introduce a completely complex-valued policy head with a sampling method inspired by the Born rule in quantum mechanics;

IV) We evaluate the proposed methods on a range of partially observable tasks, and show that the uRNN architecture is capable of solving these tasks and outperforms the baselines.

2 Background

2.1 Partial Observability, Memory and the problem setting

In many real-world applications, the state is only partially observable, formalized as a Partially Observable Markov Decision Process (POMDP) [Åström, Karl Johan, 1965]. A POMDP augments the standard Markov Decision Process with an observation model and is described by the tuple $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma)$, where $\mathcal{S}$ is the set of states and $\mathcal{A}$ the set of actions; $T(s_{t+1} \mid s_t, a_t)$ is the state transition distribution; $R(s_t, a_t)$ is the reward function; Ω is the set of observations and $O(o_t \mid s_t)$ the observation model; $\gamma \in [0, 1)$ is the discount factor.

Unlike the fully observable case, the agent never accesses the underlying state s_t directly; at each step it receives only an observation $o_t \in \Omega$ emitted from s_t through O. A single observation is in general non-Markovian, in that it does not on its own summarize the information needed to act optimally, so

a policy $\pi(a_t \mid o_t)$ conditioned on the latest observation alone is insufficient. To recover the optimal policy, the agent must instead estimate the latent state from the history of observations seen so far, $o_{1:t} = (o_1, o_2, \ldots, o_t)$, compressing this history into a representation sufficient for action selection. This introduces the need for agents to maintain memory of past observations, and it is the partially observable setting we operate in throughout this work.

The simplest approach is frame stacking [Bellemare et al., 2013], also employed by Mnih et al. [2015] in DQN for Atari, where the last four frames were stacked as input. However, this captures dependencies only within the fixed window. Recurrent networks address this by compressing temporal information into a learned hidden state. DRQN [Hausknecht and Stone, 2015] replaced frame stacking with LSTM layers that process single frames while maintaining a hidden state, matching DQN's performance while being more robust to partial observability.

Recurrent architectures still face limitations: sequential processing prevents parallelization and fixed-size hidden states bottleneck information flow. GTrXL [Parisotto et al., 2020] stabilized Transformers for RL through gating mechanisms. Decision Transformer [Chen et al., 2021] reframed RL as sequence modeling by conditioning on desired returns.

2.2 Unitary Evolution Recurrent Neural Networks

Recurrent neural networks suffer from well-known stability issues: when the recurrent weight matrix has eigenvalues with magnitude not equal to one, the hidden state dynamics either contract or blow up exponentially, leading to vanishing and exploding gradients [Bengio et al., 1994]. Gated architectures such as LSTMs [Hochreiter and Schmidhuber, 1997] mitigate this through learned gating mechanisms, but still lack an inductive bias to bound the gradients.

The introduction of *Unitary Evolution Recurrent Neural Networks* (uRNNs) by Arjovsky et al. [2016] directly targeted this gap. The central insight is that if the recurrent transition is constrained to be *unitary*, then the hidden state evolves through a norm-preserving transformation at every time step. A unitary matrix $U \in \mathbb{C}^{n \times n}$, satisfying $UU^\dagger = U^\dagger U = I$, is the complex analog of an orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ (for which $QQ^\top = I$). Refer to B for an involved introduction.

Unitary Recurrent Dynamics. A uRNN maintains a hidden state $h_t \in \mathbb{C}^n$ that evolves as:

$$h_{t+1} = \sigma(Uh_t + V_x x_t), \tag{1}$$

where $U \in \mathbb{C}^{n \times n}$ is a unitary matrix, V_x maps real-valued inputs into the complex hidden space, and σ is the modReLU activation:

$$\sigma_{\text{modReLU}}(z) = \begin{cases} (|z| + b)\frac{z}{|z|} & \text{if } |z| + b \geq 0, \\ 0 & \text{if } |z| + b < 0. \end{cases} \tag{2}$$

Efficient Parameterization. The unitary matrix is decomposed into structured factors:

$$U = D_3 \, R_2 \, \mathcal{F}^{-1} \, D_2 \, \Pi \, R_1 \, \mathcal{F} \, D_1, \tag{3}$$

where D_1, D_2, D_3 are diagonal phase matrices, $\mathcal{F}$ is the DFT, R_1, R_2 are Householder reflections, and Π is a fixed random permutation. Each component is unitary by construction, giving $O(n \log n)$ complexity. Gradient updates are performed on unrestricted parameters, guaranteeing unitarity without projection. uRNNs demonstrated remarkable performance on associative recall tasks, outperforming LSTMs with a fraction of the parameters. This makes the architecture appealing for RL, where memory-related tasks require integrating information over long sequences.

3 Related Work

Benchmarking Memory and Reinforcement Learning Works such as Hausknecht and Stone [2015], Parisotto et al. [2020], Chen et al. [2021] presented methods relying on recurrence and transformers to handle partial observability. Notably, Ni et al. [2023] showed that transformers can be effective in capturing memory but are unable to assign credit correctly to past actions. Memory Gym [Pleines et al., 2023] and POPGym [Morad et al., 2023] provide benchmark suites of partially observable environments requiring memory. RLBenchNet [Smirnov and Gu, 2025] presents an evaluation suite of PPO agents across various architectures.

In our work, we use the POBAX benchmark [Tao et al., 2025] for our experiments, baselines and comparison. POBAX a benchmark for a suite of partially observable and memory improvable tasks written in JAX [Bradbury et al., 2018], including tasks ranging from vector input continuous control to complicated image based tasks like Crafter [Hafner, 2021]. They compare the performance of a GRU based recurrent PPO and a Lambda Discrepancy PPO [Allen et al., 2024] method against a memoryless and a full state baseline.

Model-Based Reinforcement Learning Model-Based RL relies on learning a world model [Ha and Schmidhuber, 2018] that can represent the state space of partially observable environments. These models can learn the dynamics of the environment in terms of this compressed latent state $p(s_{t+1}|s_t, a_t)$, enabling planning and off policy RL methods like SAC [Haarnoja et al., 2017], and TD-MPC2 [Hansen et al., 2024]. Most notable are PlaNet and Dreamer [Hafner et al., 2019, 2025], employing RSSM-based GRU cells. Recall2Imagine [Samsami et al., 2024] proposed a world model based on S4 [Gu et al., 2021] for long-term memory. Interestingly, S4 and Mamba [Gu and Dao, 2024] are based on the same kind of recurrence as the uRNN, differing only in the applied non-linearity.

Complex-Valued Neural Networks Neural networks with complex parameters have been explored for many years. Amin et al. [2011] showed how Wirtinger calculus enables gradient descent for complex-valued networks. Follow-up works [Trabelsi et al., 2018, Sarroff et al., 2015, Geuchen and Voigtlaender, 2023] demonstrated capabilities across domains. For a comprehensive overview, see Bassey et al. [2021].

Successors of uRNNs Several works addressed the limitation that the original uRNN's parameterization does not span the entire unitary group. Wisdom et al. [2016] proposed full-capacity unitary RNNs on Stiefel manifolds. Mhammedi et al. [2017] parameterized real orthogonal matrices via Householder reflections (oRNN). Helfrich et al. [2018] introduced scoRNN via the scaled Cayley transform. Jing et al. [2017b] extended unitary RNNs with gating in GORU, addressing the lack of an explicit forget mechanism. Later, the LRU [Orvieto et al., 2023] was introduced as a linear RNN with complex hidden states, which was the extended in Elelimy et al. [2024] for real-time RL. We use the original uRNN architecture as a starting point for exploring unitary transformations in RL.

In this work, we particularly concentrate on the method presented in [Jing et al., 2017a], a tuneable version of the uRNN composed of unitary transforms on alternate dimensions, that can span the whole unitary space for a given dimensionality. We explore this construction as an additional method for complex-valued recurrence, owing to its simple design and tuneable expressivity.

4 Unitary Recurrent Networks for PPO

We equip a PPO agent with a unitary recurrent cell, hypothesizing that a complex hidden state evolved by norm-preserving transformations provides a stable and expressive memory for partially observable control. Complex data types, forward and backward passes are fully supported in JAX and PyTorch, which we rely on, for building the method (see Appendix A).

4.1 Integrating the uRNN into Proximal Policy Optimization

We build on the recurrent PPO implementation of POBax [Tao et al., 2025], where the base recurrent cell is a GRU [Cho et al., 2014]. A task-specific encoder maps each observation o_t to a feature vector x_t, which drives a recurrent cell whose hidden state h_t feeds separate policy and value heads.

We replace this GRU with a unitary recurrent cell carrying a complex hidden state $h_t \in \mathbb{C}^n$, where n matches the GRU baseline's per-environment hidden size (Table 2), leaving the rest of the agent unchanged (Figure 1). We consider two parameterizations of the unitary transition U. In its original form [Arjovsky et al., 2016], the recurrence follows Equation (1) with U structured as in Equation (3): U is constant across time and the input enters only through V_x. Native `complex64` support allows a much simpler implementation than the original; the components are:

- D_1, D_2, D_3: Diagonal matrices parameterized with learnable phases: $D_{j,j} = e^{iw_j},\ w_j \in \mathbb{R}$.
- R_1, R_2: Reflection matrices with learnable $v_1, v_2 \in \mathbb{C}^n$: $R_i = I - 2v_k v_k^*/\|v_k\|^2$.
- V_x: A `complex64` linear layer mapping the feature vector to $\mathbb{C}^n$ (no bias).
- modReLU: Implemented as per Equation (2).

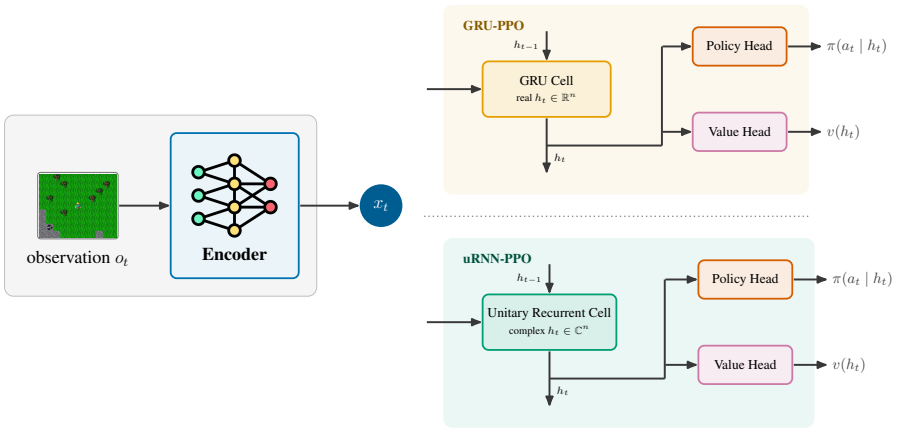

Figure 1: **Unitary Recurrent PPO architectures.** The Unitary Recurrent cell can be simply dropped-in for any standard recurrent cell. We show how the different variants can be constructed with complex hidden states, and later, fully complex policies.

The policy and value networks receive $2n$ input features (real and imaginary parts of the hidden state). This construction, due to its time/step independent recurrent dynamics, takes up far fewer parameters than a GRU cell, even though the complex parameters use twice the floating point space.

Parameter Initialization. We follow the original initialization [Arjovsky et al., 2016]: D phases from $\mathcal{U}(-\pi, \pi)$, reflection vectors from $\mathcal{U}(-1, 1)$ for both components, V_x via Glorot initialization [Glorot and Bengio, 2010], and modReLU bias initialized to 0.

Hidden State Initialization. We propose initializing the hidden state with a constant complex vector such that $\|h_0\|_2 = 1$:

$$h_0[k] = \frac{1}{\sqrt{2n}}(1+i) \quad k \in \{1, 2, \ldots, n\} \tag{4}$$

This is equivalent to an equal superposition state in an n-dimensional eigenbasis (Appendix D), pointing to an interesting interpretation explored in Section 6.

4.2 Tunable Unitary Recurrence with EUNNs

The original parameterization (Equation (3)) is efficient but spans only a subset of the unitary group [Wisdom et al., 2016]. As a second method we adopt the Efficient Unitary Neural Network (EUNN) of Jing et al. [2017a], replacing the fixed FFT structure with a product of L layers whose expressivity is directly tunable:

$$U_{\text{EUNN}} = D\, F_L F_{L-1} \cdots F_1, \tag{5}$$

where D is a diagonal phase and each F_k is block-diagonal in $H/2$ independent 2×2 unitary rotations, each acting on a pair of hidden dimensions (a, b):

$$\begin{pmatrix} h'_a \\ h'_b \end{pmatrix} = \begin{pmatrix} e^{i\phi}\cos\theta & -e^{i\phi}\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} h_a \\ h_b \end{pmatrix}, \tag{6}$$

with per-pair angles θ, ϕ. Consecutive layers alternate which dimensions are paired via a cyclic shift, mixing information across the full state. The capacity L tunes expressivity: small L restricts U to a structured subspace, while $L = H$ provably spans the full unitary group $U(H)$. The recurrence is again Equation (1) with $U = U_{\text{EUNN}}$, using learnable, input-independent angles.

4.3 Conditioning the Unitary Transformation on the Observation

While a constant U ensures stable gradient flow and is parameter-efficient, we hypothesize that an observation-dependent $U(x_t)$ can improve representational capacity. This mirrors the selectivity of

modern state-space models, where the transition is made input-dependent [Katharopoulos et al., 2020, Gu and Dao, 2024, Yang et al., 2025, 2024, Siems et al., 2025].

We propose the ICuRNN (Input Conditional Unitary RNN) that conditions U on the input x_t:

- D_1, D_2, D_3: Now parameterized with a real-valued linear layer: $D_j = \text{Linear}(x_t)$.
- R_1, R_2: Parameterized with a `complex64` linear layer: $v_k = \text{ComplexLinear}(x_t)$.

The remaining components (V_x, hidden state initialization) are unchanged. While this adds more parameters, we hypothesize that it allows the policy to learn what to remember faster.

5 Do uRNNs improve memory capabilities?

5.1 Environments

We evaluate on seven tasks from the POBax suite [Tao et al., 2025], each partially observable but solvable with the right memory, spanning discrete recall, noisy state tracking, spatial mapping, and continuous control:

T-Maze. The agent receives a cue at the start of a corridor indicating which direction to turn at the far end, but the cue is absent from the observation thereafter. It must carry a single bit across the entire corridor, isolating long-horizon recall. We use a corridor of length 75 (`tmaze_75`) to make it a true test of long term memory capability, compared to the original benchmark's 10.

RockSample. A rover on an $n \times n$ grid must sample good rocks and avoid bad ones, using a noisy sensor whose accuracy decays with distance. Solving it requires integrating repeated noisy checks to track the latent quality of each rock. We evaluate the $(11, 11)$ and $(15, 15)$ grid size configurations.

Battleship. The agent fires on a 10×10 board with hidden ships and observes only a hit or miss each step, without past outcomes retained in the observation. It must build and maintain an internal map of where it has already fired and what has been hit.

Masked Walker and HalfCheetah. Continuous-control locomotion tasks (Brax) in which only velocity features are observed and all positional state is masked. The agent must integrate the velocity history to recover position, testing memory in a high-dimensional continuous control setting.

Craftax (No Inventory). A pixel-based open-world survival task originally proposed in Hafner [2021] and made harder by Matthews et al. [2024]. The version we run is a futher modified version by Tao et al. [2025], in which the inventory panel is cropped from the observation. The agent must remember its collected resources and progress through the achievement tree from visual input alone, requiring higher memory capabilities than the original Craftax task. We downsample the observation image size, just as described in the POBAX benchmark.

5.2 Experimental Setup

Our implementations of the unitary recurrent networks are built on top of the POBAX benchmark [Tao et al., 2025], written entirely in JAX. We compare against two baselines from POBAX: the GRU recurrent PPO agent, and PPO-LD, which augments recurrent PPO with the λ-discrepancy memory objective based on a second value function approximator [Allen et al., 2024]. All architectural details apart from the RNN, including last action concatenation are kept here. Each uRNN variant is a drop-in replacement of the GRU cell, leaving the encoder, heads, and PPO objective untouched. We evaluate the three unitary parameterizations introduced in Section 4:

1. **uRNN**: original parameterization with a constant transition U (Section 4.1).
2. **EUNN**: tunable parameterization with capacity $L = 2$ (Section 4.2).
3. **ICuRNN**: input-conditioned transition $U(x_t)$ (Section 4.3).

All uRNN variants use a complex hidden state matching the GRU baseline's hidden size, and use the same encoder, policy and value heads. For the EUNN variant, we only evaluate the simplest tuning setting, with $L = 2$, as an initial demonstration given the original work [Jing et al., 2017c] shows that this setting is already comparable with traditional recurrent cells. We leave the investigation of the effects of tuning to future work. All evaluations are done for the same total steps as in the POBAX benchmark, averaged over 5 seeds, except for craftax, where we limit the runs to 100M steps.

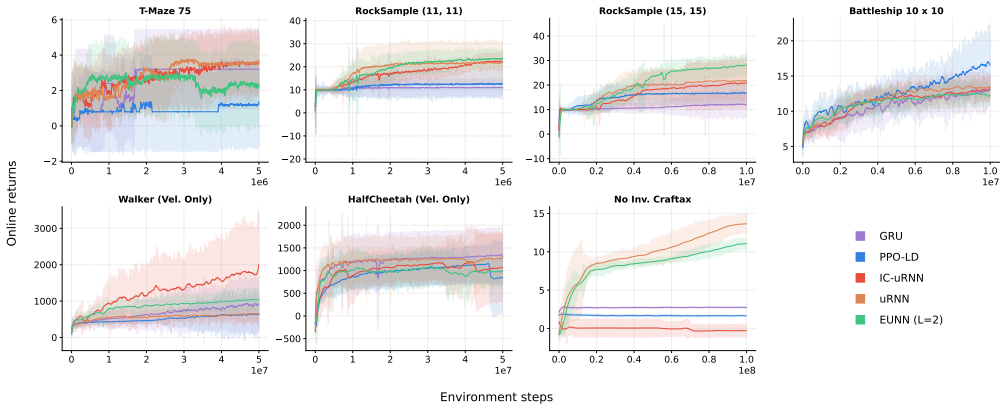

Figure 2: Reward curves for the 3 uRNN variants and the baselines across 7 environments from the POBAX [Tao et al., 2025] benchmark

Hyperparameters We evaluate the baselines GRU and PPO-LD on the same tuned hyperparameter settings in POBAX. We observe that the complex valued parameters in our methods require a much smaller and separate learning rate than the real valued ones. We keep the network sizes, hidden state sizes, discount factor and number of parallel environments identical to the baselines, but find that a larger number of steps (256) per PPO update works well for the uRNNs. The optimal GAE lambda, learning rates and entropy coefficient were found through sweeps over a grid, and the details per environment can be found in App. E. We do not tune the settings for the No Inventory Craftax environment, given its large number of parallel environments and compute constraints.

5.3 Results

The results of the evaluations across the 7 tasks on the 3 unitary recurrent variants are visualized in Fig. 2. Across the suite, the three unitary variants collectively outperform both the GRU and PPO-LD baselines on a majority of tasks, frequently by a wide margin, and remain competitive on the rest.

The clearest gains appear on the memory-heavy discrete tasks. On both RockSample configurations all three uRNN variants pull well above the baselines, which plateau near a return of 12: on the $(11, 11)$ grid the original uRNN reaches the highest returns, roughly $2\times$ the GRU and PPO-LD plateaus, with EUNN and IC-uRNN following, while on the larger $(15, 15)$ grid EUNN fares the best. The largest gains are on No-Inventory Craftax run with downsampled observations, where uRNN and EUNN far exceed the two GRU baselines[1]. IC-uRNN is the exception here, failing to make progress on this task despite being the strongest variant elsewhere. We attribute this case to insufficient tuning, given compute budget constraints.

The continuous-control and remaining tasks are more even. On the velocity-only Walker, IC-uRNN stands out with a substantial lead over all other methods, suggesting that conditioning the unitary transition on the observation is particularly useful for integrating velocity history, whereas on HalfCheetah all methods perform comparably. On T-Maze-75, the memory-capable all the methods converge to similar returns, comfortably above the PPO-LD baseline, with EUNN learning fastest early in training. Overall, the complex-recurrent cells deliver consistent and often large improvements on memory-intensive tasks, with different parameterizations excelling on different environments.

6 Phase Aware Policies: A Quantum Information Perspective

In the preceding sections we demonstrated that complex-valued representations provide a richer source for learning policies in POMDP settings. However, we believe that naïvely treating uRNN cells as a simple drop-in replacement for canonical recurrent units leaves untapped potential in the expressivity of the complex-valued representations. That is, our approach so far has treated both

[1]We follow the same observation cropping and downsampling as described in the POBAX [Tao et al., 2025] paper for all our evaluations but were unable to replicate their results on the Craftax environment

the real and imaginary parts of the hidden state as real-valued inputs to the policy head. However, a complex vector encodes more than just that - the different dimensions of the vector have different phases and these get lost in the conversion detailed above. This leads us to ask the question: *Can the phases of the hidden state encode relevant information towards the actions an optimal policy should take at each step?* In the quest of instilling "phase awareness" into our policies, we draw a parallel to concepts from Quantum Information Theory.

Quantum states are primarily represented vectors in a complex eigen basis, that is, they can be expressed as a linear combination of a set of orthogonal eigenvectors. A 'measurement' in this basis, *collapses* the state onto one of the eigenvectors, with a probability determined by the Born rule [Born, 1984]. A introduction to these concepts and notation used below is included in Appendix D.

This probabilistic nature of measurement, so fundamental to physical quantities and phenomena, sparks a comparison to the policy we are learning here. Consider a stochastic policy $\pi(a_t \mid s_t)$ over a *discrete* action space $\mathcal{A}$. This policy at any time step t can be expressed as a quantum state vector:

$$|\pi_t\rangle = \sum_{a \in \mathcal{A}} c_t^a \, |a\rangle \,, \quad c_t^a \in \mathbb{C} \tag{7}$$

Drawing on the Born rule (Appendix D), the probability of observing action a is:

$$\pi(a_t \mid s_t) = |\, \langle a | \pi_t \rangle \,|^2 = |c_t^a|^2 \tag{8}$$

Given the complex hidden state $h_t \in \mathbb{C}^n$ from the uRNN, we propose learning the action coefficients directly:

$$|\pi(s_t)\rangle = \sum_{a \in \mathcal{A}} \hat{c}_\theta^a(h_t) \, |a\rangle \,, \quad \hat{c}_\theta^a(h_t) \in \mathbb{C}, \; h_t \in \mathbb{C}^n \tag{9}$$

To this end, we replace the real-valued policy head with a complex-valued one with the ModReLU activation. The action coefficients $\hat{c}_\theta^a(h_t)$ are unnormalized. To apply the Born rule while maintaining compatibility with softmax, we output $\log(|\hat{c}_\theta^a(h_t)|^2)$ as logits:

$$\pi(a_t \mid s_t) = \frac{\exp(log(|\hat{c}_\theta^a(h_t)|^2))}{\sum_{a \in \mathcal{A}} \exp(log(|\hat{c}_\theta^a(h_t)|^2))} = \frac{|\hat{c}_\theta^a(h_t)|^2}{\sum_{a \in \mathcal{A}} |\hat{c}_\theta^a(h_t)|^2} \tag{10}$$

Interference in fully complex policies Because the action coefficients are a phase-preserving transformations (ModReLU preserves phase) of the hidden state, for action a:

$$\log\left(|\hat{c}_\theta^a(h_t)|^2\right) = \log\left(|w^a[0]h_t[0] + w^a[1]h_t[1] + \cdots + w^a[n-1]h_t[n-1]|^2\right) \tag{11}$$

where $w^a \in \mathbb{C}^n$ is the weight vector for action a. Since both weights and hidden states are complex, the phases interfere constructively or destructively (Equation (12)). Because the hidden state contains information from past time steps and the current observation, this interplay allows for potentially a much richer representation space for the policy.

Unitarity Furthermore, the evolution of quantum states through time, are also constrained by Unitary transformations. While our recurrent dynamics are not exactly unitary, owing to the addition of the observation dependent vector, this connection between concepts of reinforcement learning and quantum information theory is noteworthy.

6.1 Experiments: Phase Aware Policy uRNNs

Following the methodology described above, we integrate the fully complex policy head with the Born rule based sampling into the 3 previously specified uRNN variants, to have: QuRNN, QICuRNN and QEUNN. As we described the method for discrete action spaces, we limit our evaluation of the QuRNN variants to exclude the Walker and HalfCheetah environments. All other settings and hyperparameters (Appendix E) stay the same as Section 5.2.

Results. The reward curves for the three phase-aware variants are shown in Figure 3, alongside the GRU baseline and the best base uRNN method from Section 5 for clarity. Overall, the phase-aware policies are competitive with their base uRNN counterparts on most tasks. The performance is

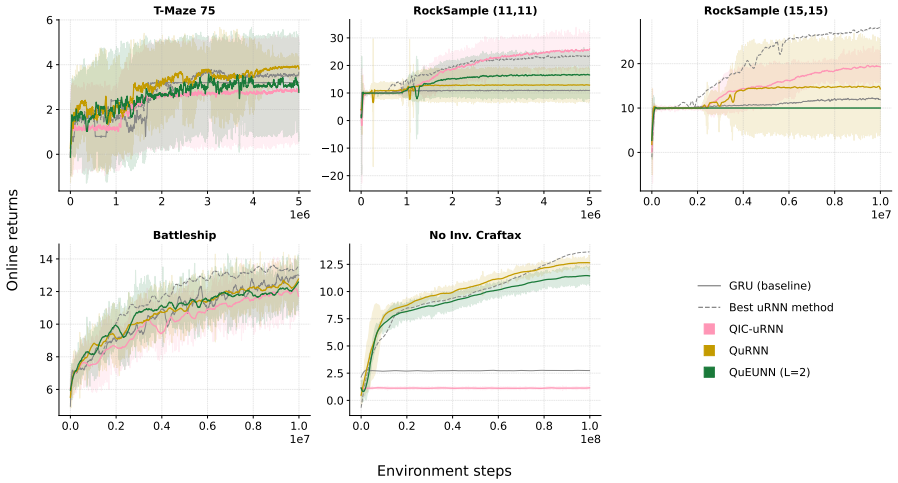

Figure 3: Reward curves for the 3 Phase Aware QuRNN variants plotted with a GRU baseline and the best uRNN method on 5 environments with discrete action spaces

strongest again on No-Inventory Craftax, where QuRNN and QuEUNN reach close to $5\times$ the GRU return and track the best base uRNN, and on RockSample, where QIC-uRNN slightly exceeds the best base uRNN variant. On T-Maze-75 and Battleship evironments, all methods end up tightly clustered around the GRU baseline. The exception mirrors the base results: QIC-uRNN fails to learn on Craftax, just as IC-uRNN did, while remaining one of the stronger variants on RockSample. These results indicate that preserving the phase through to action sampling retains the benefits of the complex-valued recurrence without sacrificing performance relative to the real-valued policy heads.

7 Discussion and Conclusion

In this work, we revisited unitary evolution recurrent neural networks as a representation for reinforcement learning in partially observable environments. We integrated their complex-valued, norm-preserving recurrence into recurrent PPO as a drop-in replacement for the GRU cell, and studied three parameterizations of the unitary transition. We then proposed a phase-aware policy head that samples actions through a Born-rule mechanism, preserving the complex representation through to action selection. The strong performance of our methods across the POBAX benchmark are evidence that complex-valued representations and unitary dynamics are a promising new direction for representation learning in RL, rather than a niche architectural choice. The connection to quantum information theory further hints at a framework bridging the two fields, as previously explored in a different context by Dong et al. [2008]. We believe these findings motivate a deeper investigation of complex-valued states for memory and partial observability in RL

Limitations. We do not explore the hyperparameter space in depth. Given the known sensitivity of PPO to hyperparameters [Eimer et al., 2023, Huang et al., 2022], more thorough exploration is needed. Our experiments are limited to part of the POBAX benchmark; evaluation on benchmarks like Memory Gym [Pleines et al., 2023] and Memory Maze [Pasukonis et al., 2022] would strengthen the argument. While we demonstrate the performance of the base version of the EUNN architecture [Jing et al., 2017c], we only evaluate the simplest tuning setting as a preliminary set of experiments, and we hope to understand the impact of the tuning parameter, and the performance of the method when accessing larger unitary spaces in the future.

Future Work. Looking to the future, we hope to extend this work in two main directions. The dual representation of the complex hidden state naturally motivates an exciting question: *what do the phases of the hidden states encode throughout the rollout?* We hope to investigate this mechanistically in upcoming work. A natural extension is to apply complex-valued representations and unitary transformations to model-based RL, where the norm-preserving property becomes central to state transitions in learned world models. More diverse environments and deeper analysis of training dynamics are also important directions.

Acknowledgments

André Biedenkapp is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 572775489. The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG. Efstratios Gavves and Sathya Kamesh Bhethanabhotla are supported by the European Union's Horizon Europe research and innovation programme under grant agreement number 101214398 (ELLIOT).

References

Martín Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, JMLR Workshop and Conference Proceedings. JMLR.org, 2016.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*. PMLR, 2017a.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Åström, Karl Johan. Optimal Control of Markov Processes with Incomplete State Information I. *Journal of Mathematical Analysis and Applications*, 10, 1965.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101:99–134, 1998.

Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. Recurrent networks, hidden states and beliefs in partially observable environments. *Transactions on Machiene Learning Research*, 2022.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free RL can be a strong baseline for many pomdps. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *Proceedings of the International Conference on Machine Learning, ICML 2022*, Proceedings of Machine Learning Research, pages 16691–16723. PMLR, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposia, Arlington, Virginia, USA, November 12-14, 2015*. AAAI Press, 2015.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017b.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Proceedings of the Annual Conference on Neural Information Processing Systems 2018 (NeurIPS 2018)*, pages 4759–4770, 2018.

Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. In *The Twelfth International Conference on Learning Representations*, 2024.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8), 1997.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024.

Max Born. Zur quantenmechanik der stoßvorgänge (1926). In *Die Deutungen der Quantentheorie*, pages 48–52. Springer, 1984.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphaël Lopez Kaufman, Aidan Clark, Seb Noury, Matthew Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*. PMLR, 2020.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5, 1994.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in RL? Decoupling memory from credit assignment. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Memory gym: Partially observable challenges to memory-based agents. In *The Eleventh International Conference on Learning Representations*, 2023.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. POP-Gym: Benchmarking partially observable reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Ivan Smirnov and Shangding Gu. Rlbenchnet: The right network for the right reinforcement learning task, 2025.

Ruo Yu Tao, Kaicheng Guo, Cameron Allen, and George Konidaris. Benchmarking partial observability in reinforcement learning with a suite of memory-improvable domains. *The Reinforcement Learning Journal*, 2025.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

Cameron Allen, Aaron Kirtland, Ruo Yu Tao, Sam Lobel, Daniel Scott, Nicholas Petrocelli, Omer Gottesman, Ronald Parr, Michael L. Littman, and George Konidaris. Mitigating partial observability in sequential decision processes via the lambda discrepancy. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. 2017.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565, 2019.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Md. Faijul Amin, Muhammad Ilias Amin, A. Y. H. Al-Nuaimi, and Kazuyuki Murase. Wirtinger calculus based gradient descent and levenberg-marquardt learning algorithms in complex-valued neural networks. In Bao-Liang Lu, Liqing Zhang, and James Kwok, editors, *Neural Information Processing*, pages 550–559. Springer Berlin Heidelberg, 2011.

Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018.

Andy M. Sarroff, Victor Shepardson, and Michael A. Casey. Learning representations using complex-valued nets, 2015.

Paul Geuchen and Felix Voigtlaender. Optimal approximation using complex-valued neural networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, 2023.

Joshua Bassey, Lijun Qian, and Xianfang Li. A survey of complex-valued neural networks, 2021.

Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. volume 29, 2016.

Zakaria Mhammedi, Andrew D. Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. 2017.

Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform. 2018.

Li Jing, Caglar Gulcehre, John Peurifoy, Yichen Shen, Max Tegmark, Marin Soljačić, and Yoshua Bengio. Gated orthogonal recurrent units: On learning to forget, 2017b.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023.

Esraa Elelimy, Adam White, Michael Bowling, and Martha White. Real-time recurrent learning using trace units in reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*. PMLR, 2010.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *Proceedings of ICLR*, 2025.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Proceedings of NeurIPS*, 2024.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear rnns via householder products, 2025.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2024.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2017c.

Daoyi Dong, Chunlin Chen, Hanxiong Li, and Tzyh-Jong Tarn. Quantum reinforcement learning. *Trans. Sys. Man Cyber. Part B*, 38(5), 2008.

Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`.

Jurgis Pasukonis, Timothy Lillicrap, and Danijar Hafner. Evaluating long-term memory in 3d mazes. *arXiv preprint arXiv:2210.13383*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., 2019.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Cambridge, 2010.

A Complex Forward and Backward Passes

When uRNNs were first proposed, PyTorch [Paszke et al., 2019] and TensorFlow [Abadi et al., 2015] did not offer reliable complex data types, so researchers had to write custom layers for initialization, normalization, and backpropagation. Works like [Trabelsi et al., 2018] still existed to describe the mechanics of complex-valued neural networks, defined activations, layers and normalization techniques around them, while demonstrating their applicability in various domains.

Today the situation is much simpler. Both PyTorch and JAX [Bradbury et al., 2018] include built-in complex dtypes (for example `torch.complex64` and `jax.numpy.complex64`) and their autograd systems support Wirtinger calculus automatically [Amin et al., 2011]. This means a complex forward or backward pass can be written just like its real-valued counterpart: declare parameters in a complex dtype and the framework handles gradient propagation.

In this work, we rely heavily on these frameworks for a much simpler and straightforward implementation of complex-valued neural networks.

B Complex Algebra and Unitary Matrices

An important idea in this work is the extension of linear algebra to the complex space. Complex numbers and complex-valued representations provide an additional degree of freedom in the form of a phase.

A complex number $z = x + iy$ can also be represented as $z = re^{i\theta}$ with $r = |z|$ and $\theta = \tan^{-1}(y/x)$. The product of two complex numbers can be seen as rotations in phase space: $z_1 z_2 = r_1 r_2 e^{i(\theta_1+\theta_2)}$.

An important interaction is the squared norm of the sum of two complex numbers:

$$|z_1 + z_2|^2 = |z_1|^2 + |z_2|^2 + 2|z_1||z_2| \cos(\theta_2 - \theta_1) \tag{12}$$

This shows that the squared norm depends on the phase difference, a property we will exploit later.

A complex vector space $\mathbb{C}^n$ is a Hilbert space where the inner product takes the form $\langle a, b\rangle = a^* b$, with $(\cdot)^*$ denoting conjugate transpose.

Now for matrices in a Hilbert space, the unitary property is defined as follows:

Definition B.1. A matrix $U \in \mathbb{C}^{n\times n}$ is *unitary* if $U^\dagger U = I$, where $U^\dagger$ denotes the conjugate transpose. For any vector $x \in \mathbb{C}^n$: $\|Ux\|_2 = \|x\|_2$.

All eigenvalues of a unitary matrix lie on the complex unit circle ($|\lambda| = 1$).

Householder Reflections. A Householder reflection (Fig. 4), characterized by a matrix in complex space is $H = I - 2\, vv^*/(v^*v)$, $v \in \mathbb{C}^n$. All Householder matrices are unitary (see Appendix C), and composing multiple reflections generates a rich family of unitary transformations efficiently. This idea forms the core of how we can parameterize unitary recurrent systems.

C Proof for the unitarity of Householder Matrices

In B, we introduced the Householder reflection as a way to parametrize unitary matrices. We now provide the proof for the unitarity of Householder matrices.

Theorem C.1. Let $v \in \mathbb{C}^n$ be a nonzero vector. The Householder matrix defined by

$$H = I - \frac{2vv^\dagger}{v^\dagger v}$$

is unitary, i.e., $H^\dagger H = I$.

Proof. We verify that $H^\dagger H = I$ by direct computation.

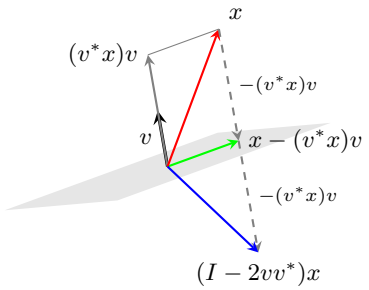

Figure 4: **Geometric interpretation of a Householder reflection.** A vector x is reflected across the hyperplane orthogonal to v to produce Hx, where $H = I - 2vv^*/(v^*v)$ is the Householder matrix. The reflection flips the parallel component while preserving the perpendicular component.

First, we compute the Hermitian conjugate of H:

$$\begin{aligned} H^\dagger &= \left(I - \frac{2vv^*}{v^*v}\right)^\dagger \\ &= I^\dagger - \frac{2}{v^*v}\left(vv^*\right)^\dagger \\ &= I - \frac{2}{(v^*v)^*}\left(v^*\right)^* v^* \\ &= I - \frac{2}{v^*v}vv^* \\ &= H, \end{aligned}$$

where we used the fact that v^*v is real and positive (hence $(v^*v)^* = v^*v$), and $\left(v^*\right)^* = v$.

Thus, H is Hermitian. Now we verify unitarity:

$$\begin{aligned} H^\dagger H &= H^2 \\ &= \left(I - \frac{2vv^*}{v^*v}\right)\left(I - \frac{2vv^*}{v^*v}\right) \\ &= I - \frac{2vv^*}{v^*v} - \frac{2vv^*}{v^*v} + \frac{4vv^*vv^*}{(v^*v)^2} \\ &= I - \frac{4vv^*}{v^*v} + \frac{4v(v^*v)v^*}{(v^*v)^2} \\ &= I - \frac{4vv^*}{v^*v} + \frac{4vv^*}{v^*v} \\ &= I, \end{aligned}$$

where in the second-to-last line we used the fact that v^*v is a scalar, so $v(v^*v) = (v^*v)v$.

Therefore, $H^\dagger H = I$, which proves that H is unitary. □

D Quantum Information Theory and Measurements

This section serves as a brief introduction to the concepts of quantum information theory and measurements, which we will use to motivate a alternative perspective from which this work and the underlying Unitary Recurrence can be viewed, in Section 6. While this section only covers the concepts and relationships relevant to this work, a more comprehensive treatment of the complex linear algebra and quantum mechanical principles can be found in [Nielsen and Chuang, 2010].

Throughout this section, we use Dirac notation: a ket $|\psi\rangle$ denotes a column vector, while a bra $\langle\phi|$ denotes a row vector. A braket $\langle\phi, \psi\rangle$ denotes the inner product $\langle\phi|\,|\psi\rangle$, and a ketbra $|\psi\rangle\,\langle\phi|$ denotes the outer product.

A quantum state provides a complete description of an isolated physical system.

Definition D.1. In a finite-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^d$, any pure state is represented as a unit vector

$$|\psi\rangle = \sum_{k=1}^{d} c_k |k\rangle ,$$

where $\{|k\rangle\}$ is an orthonormal basis and the coefficients $c_k \in \mathbb{C}$ satisfy $\sum_k |c_k|^2 = 1$.

The definition of basis is the same as in real vector spaces, where any set of orthogonal vectors that span the space can be a basis. This state vector expansion is not merely a coordinate representation: it encodes all physically accessible information about the system.

A state vector written in this form as a linear combination of the basis eigenstates, is said to be in a *superposition* of the eigenstates. The coefficients c_k are the *amplitudes* of the state vector in the basis eigenstates, and if all the amplitudes c_k are equal, then this superposition is a *equal superposition state*, meaning the state vector is equally likely to be in any of the basis eigenstates.

D.1 Operators and State Transformations

In quantum mechanics, transformations of quantum states are represented by *Operators*, which are square matrices acting on the Hilbert space. An operator $A \in \mathbb{C}^{d\times d}$ transforms a state $|\psi\rangle$ to $A|\psi\rangle$, analogous to how a matrix multiplies a vector in linear algebra. Operators can represent various physical processes: rotations, measurements, or time evolution.

For a measurement to be physically meaningful, the corresponding operator must be *Hermitian* (as defined in B). Hermitian operators have real eigenvalues and form a complete orthonormal eigenbasis, making them suitable for representing physical observables, or quantities that can be measured, such as energy or position. When we measure an observable represented by a Hermitian operator M, we are effectively projecting the state onto one of M's eigenstates.

D.2 Measurement and Probabilistic Outcomes

A measurement in the eigenbasis $\{|k\rangle\}$ of a Hermitian operator is inherently probabilistic. Upon measurement, the outcome λ_k (corresponding to eigenstate $|k\rangle$) is observed with probability

$$p(k) = |\langle k, \psi\rangle|^2 = |c_k|^2,$$

as prescribed by the Born rule, and the post-measurement state collapses to $|k\rangle$. The stochasticity arises not from noise but from the fundamental quantum mechanical property of projecting the state onto an eigenbasis of the measured observable. This projection is a linear operation that transforms the state from a superposition $|\psi\rangle = \sum_k c_k |k\rangle$ to a single basis state $|k\rangle$.

D.3 Time Evolution and Unitary Dynamics

The time evolution of a closed quantum system is governed by the Schrödinger equation, which is a linear differential equation that describes how the state of a quantum system changes over time.

$$i\hbar \frac{d}{dt} |\psi\rangle = H |\psi\rangle ,$$

where H is the Hamiltonian, a Hermitian operator encoding the system's energy structure. Its formal solution is

$$|\psi(t)\rangle = U(t) |\psi(0)\rangle , \qquad U(t) = e^{-\frac{i}{\hbar}Ht}.$$

where $U(t)$ is the time evolution operator. Because H is Hermitian, the operator $U(t)$ is unitary, ensuring that the norm of the state is preserved during evolution. This norm preservation mirrors the dynamical behaviour of unitary recurrent updates used in machine learning: the global "energy" of the representation remains constant, and information is carried forward through rotations in complex space rather than amplification or decay.

E Hyperparameters

We organize the hyperparameters into three tables. Table 1 lists the PPO settings that are held fixed across the two baselines (GRU, PPO-LD) and all six uRNN variants. Table 2 lists the per-environment settings that are fixed across all methods but vary by environment (hidden size, parallel environments, training budget, and discount). Table 3 reports the per-environment best hyperparameters of the two baselines, found by sweeping over a grid.

Table 1: Hyperparameters held fixed across all baselines and all six uRNN variants. The rollout length T is the only PPO setting we change relative to the baselines (see note).

Hyperparameter	**Value**	**Note**
Optimizer	Adam	
Update epochs	4	per PPO update
Minibatches per update	4	
PPO clip ϵ	0.2	
Value loss coefficient c_v	0.5	
Max gradient norm	0.5	global-norm clipping
Learning-rate anneal	linear $\to 0$	applied to the real parameter group
Action concatenation	yes	previous action a_{t-1} appended to o_t
Rollout length T	128 / 256	128 for baselines, 256 for uRNN variants; Craftax uses 64
Seeds	5	

Table 2: Per-environment settings, fixed across all methods. The complex hidden state of each uRNN variant has the same dimension n as the GRU baseline's hidden state. Training budgets follow the POBAX benchmark [Tao et al., 2025]; Craftax is capped at 10^8 steps.

Environment	**Hidden size n**	**Parallel envs**	**Total steps**	γ
T-Maze (corridor 75)	32	4	5.0×10^6	0.99
RockSample $(11, 11)$	256	8	5.0×10^6	0.99
RockSample $(15, 15)$	512	16	1.0×10^7	0.999
Battleship (10×10)	512	32	1.0×10^7	1.0
Masked Walker	256	4	5.0×10^7	0.99
Masked HalfCheetah	256	4	5.0×10^7	0.99
Craftax (no inventory)	512	256	1.0×10^8	0.99

Table 3: Per-environment best hyperparameters for the baselines, selected by a grid sweep, exactly following [Tao et al., 2025]

	GRU-PPO			**PPO-LD**				
Environment	lr	ent.	λ_0	lr	ent.	λ_0	λ_1	β
T-Maze (corridor 75)	2.5×10^{-3}	0.01	0.7	2.5×10^{-4}	0.01	0.95	0.95	0.25
RockSample $(11, 11)$	2.5×10^{-3}	0.2	0.7	2.5×10^{-3}	0.2	0.5	0.5	0.25
RockSample $(15, 15)$	2.5×10^{-3}	0.2	0.7	2.5×10^{-3}	0.2	0.1	0.95	0.5
Battleship (10×10)	2.5×10^{-3}	0.05	0.7	2.5×10^{-3}	0.05	0.1	0.95	0.5
Masked Walker	2.5×10^{-4}	0.01	0.95	2.5×10^{-4}	0.01	0.95	0.95	0.5
Masked HalfCheetah	2.5×10^{-4}	0.01	0.9	2.5×10^{-5}	0.01	0.95	0.7	0.25
Craftax (no inventory)	2.5×10^{-4}	0.01	0.5	2.5×10^{-4}	0.01	0.1	0.95	0.25

E.1 Per-environment sweeps for the uRNN variants

We tune four hyperparameters per environment for our six variants: the real and complex learning rates (lr, complex lr), the entropy coefficient, and the GAE coefficient λ_0. All remaining settings

are shared with the baselines (Tables 1 and 2); the second GAE coefficient $\lambda_1 = 0.5$ and the λ-discrepancy weight are not used ($\beta = 0$, single critic). For each environment we report the search space (left) and the best setting per variant (right). For **Craftax** we do not tune the uRNN variants: we reuse the baseline settings of Table 3.

Table 4: **T-Maze (corridor 75).** Search space (left) and best per-variant settings (right).

Parameter	**Values swept**
complex lr	$10^{-6}, 5\times10^{-6}, 10^{-5}, 5\times10^{-5}, 10^{-4}$
entropy	0.005, 0.01, 0.05
λ_0	0.8, 0.9, 0.95
lr	2.5×10^{-4}, 2.5×10^{-3}

Variant	complex lr	ent.	λ_0	lr
uRNN	10^{-6}	0.005	0.95	2.5×10^{-4}
EUNN	5×10^{-5}	0.01	0.8	2.5×10^{-4}
ICuRNN	5×10^{-5}	0.01	0.95	2.5×10^{-4}
QuRNN	10^{-5}	0.005	0.95	2.5×10^{-3}
QEUNN	10^{-4}	0.005	0.8	2.5×10^{-4}
QICuRNN	5×10^{-6}	0.01	0.9	2.5×10^{-3}

Table 5: **RockSample** $(11, 11)$**.** Search space (left) and best per-variant settings (right).

Parameter	**Values swept**
complex lr	10^{-6}, 10^{-5}, 8×10^{-5}
entropy	0.075, 0.1, 0.15, 0.2
λ_0	0.7, 0.8, 0.95
lr	2.5×10^{-4}, 2.5×10^{-3}

Variant	complex lr	ent.	λ_0	lr
uRNN	8×10^{-5}	0.075	0.7	2.5×10^{-3}
EUNN	10^{-6}	0.15	0.7	2.5×10^{-3}
ICuRNN	10^{-6}	0.075	0.7	2.5×10^{-3}
QuRNN	8×10^{-5}	0.075	0.7	2.5×10^{-3}
QEUNN	8×10^{-5}	0.1	0.7	2.5×10^{-3}
QICuRNN	8×10^{-5}	0.15	0.7	2.5×10^{-4}

Table 6: **RockSample** $(15, 15)$**.** Search space (left) and best per-variant settings (right).

Parameter	**Values swept**
complex lr	10^{-6}, 10^{-5}, 8×10^{-5}
entropy	0.075, 0.1, 0.15, 0.2
λ_0	0.7, 0.8, 0.95
lr	2.5×10^{-4}, 2.5×10^{-3}

Variant	complex lr	ent.	λ_0	lr
uRNN	10^{-5}	0.075	0.7	2.5×10^{-3}
EUNN	10^{-6}	0.1	0.7	2.5×10^{-3}
ICuRNN	8×10^{-5}	0.15	0.7	2.5×10^{-4}
QuRNN	8×10^{-5}	0.1	0.7	2.5×10^{-3}
QEUNN	10^{-6}	0.075	0.8	2.5×10^{-3}
QICuRNN	8×10^{-5}	0.2	0.7	2.5×10^{-4}

Table 7: **Battleship** (10×10)**.** Search space (left) and best per-variant settings (right).

Parameter	**Values swept**
complex lr	10^{-6}, 10^{-5}, 8×10^{-5}
entropy	0.01, 0.05, 0.1
λ_0	0.6, 0.7, 0.8, 0.9, 0.95
lr	2.5×10^{-3}

Variant	complex lr	ent.	λ_0	lr
uRNN	8×10^{-5}	0.01	0.9	2.5×10^{-3}
EUNN	10^{-6}	0.01	0.7	2.5×10^{-3}
ICuRNN	8×10^{-5}	0.01	0.95	2.5×10^{-3}
QuRNN	10^{-5}	0.01	0.8	2.5×10^{-3}
QEUNN	10^{-5}	0.01	0.8	2.5×10^{-3}
QICuRNN	10^{-5}	0.01	0.9	2.5×10^{-3}

Table 8: **Masked Walker.** Search space (left) and best per-variant settings (right). The Born-rule (Q) variants are defined for discrete action spaces only and are not evaluated here.

Parameter	**Values swept**
complex lr	10^{-6}, 10^{-5}, 8×10^{-5}
entropy	0.005, 0.01, 0.05, 0.1
λ_0	0.8, 0.9, 0.95
lr	2.5×10^{-4}

Variant	complex lr	ent.	λ_0	lr
uRNN	8×10^{-5}	0.01	0.95	2.5×10^{-4}
EUNN	10^{-5}	0.01	0.95	2.5×10^{-4}
ICuRNN	8×10^{-5}	0.005	0.95	2.5×10^{-4}

Table 9: **Masked HalfCheetah.** Search space (left) and best per-variant settings (right). The Born-rule (Q) variants are defined for discrete action spaces only and are not evaluated here.

Parameter	**Values swept**
complex lr	10^{-6}, 10^{-5}, 8×10^{-5}
entropy	0.005, 0.01, 0.05, 0.1
λ_0	0.8, 0.9, 0.95
lr	2.5×10^{-4}

Variant	complex lr	ent.	λ_0	lr
uRNN	8×10^{-5}	0.005	0.9	2.5×10^{-4}
EUNN	10^{-6}	0.005	0.9	2.5×10^{-4}
ICuRNN	8×10^{-5}	0.01	0.8	2.5×10^{-4}