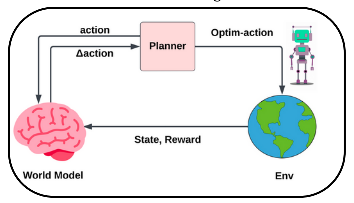
Deep Learning Architectures as Learned World Models for Reinforcement Learning Agents

World Models

Reinforcement learning is commonly broken down into two main frameworks: Model-Free RL and Model-Based RL

Source: Gradient-based Planning with World Models



- **Model-Free RL** learns action dynamics by experience in an an environment and rewards that serve as noisy labels.
- **Model-Based RL** learns dynamics of the environment itself and uses that to aid its learning speed and inference performance.

Model-based RL increases learning speed by learning the environments dynamics via the interactions between an agent and the environment. Then, simulated trajectories can be generated to **replace costly environment sampling**. The learned dynamics, or the **world model**, can be used to search the state space at inference time in order to improve performance.

So why are model-based algorithms explored less?

A good world model satisfies **three main properties**:

1. Sample Efficient
2. Lightweight
3. Correct

These three properties form a Venn diagram with very few models in the intersection. The few models that do satisfy these conditions are **limited by their expressive capabilities or by their complexity**.

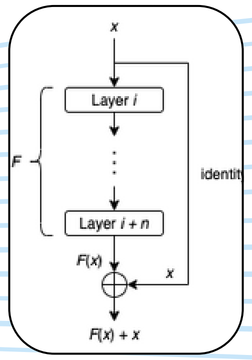
Recently, variants of neural networks have taken the world by storm, largely due to their function approximation qualities. In this study we used a novel approach to compare two popular neural network architectures (RNN's and Residual Connections) with NCPS, which derived from these two models collectively. The author of NCPS claims the model to **capture the cause and effect of data** making it a theoretically strong fit in it's capacity to model intricate dynamics.

Architectures

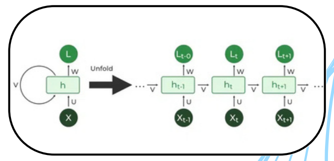
Residual Connection

Residual Connection models **only predict the change from one state to the next**. We claim this is a simpler problem than creating the next state from scratch.

In practice, this is done by adding its output (the prediction for the change in state) to the input state to get a prediction for the next state.



Recurrent Network (RNN)

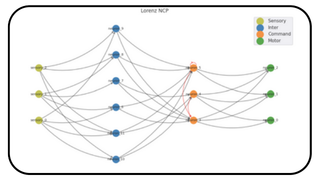


RNN are a type of neural network designed for sequential data, where past information influences the current output via a **hidden state** that maintains context across time steps.

Many world models in previous works have used derivatives on RNN architectures, such as dreamerV3.

Neural Circuit Policies (NCPS)

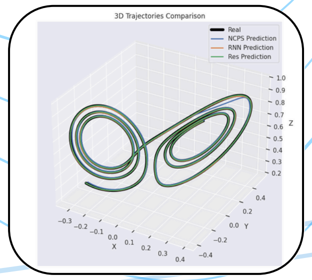
Fundamentally derived from NeuralODE's (and by extension Residuals) and recurrent networks, **these models promise to identify cause-and-effect patterns in data**, theoretically tackling a separate problem that that of predictive inference.



These models, developed by *Hasani, R., et al. (2022)*, are derivatives of the foundational work of continuous-time neural networks which train neural networks as **approximate derivatives** and solving over temporal data by **integrating** that neural network over time, or **closed form approximating** that integration. We can then add sparse connections to certain circuits in the network in a way which mimics biological networks, with the consequence of increased interpretability.

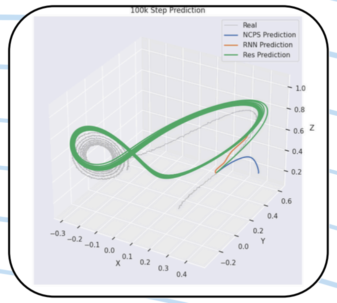
Experiments

Lorenz System



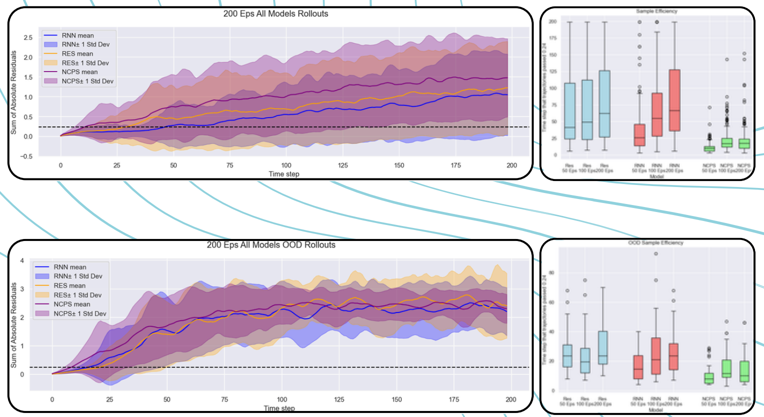
Lorenz Systems are examples of **chaotic systems**, a classification of dynamical systems where small perturbations in inputs lead to large changes in output. The graph on the left outlines a dynamical system with three models trained on it's trajectory.

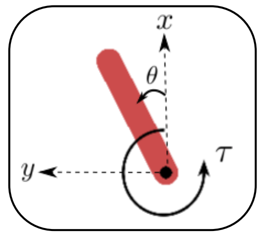
Lorenz systems display, due to their chaotic nature, dynamics which are difficult to approximate with neural networks. While the models can understand next step prediction quite well, **the underlying dynamics that cause lobe-switching behavior is not often well approximated.** The plot on the right showcases this, with the multistep predictions of NCPS and Recurrent Networks falling into stable points. Meanwhile, **the residual showcases the lobe switching characteristics that are more closely align with the underlying system.**
We hypothesize this is due to the **models attempting to learn separate problems:** predicting the residual vs the underlying cause-and-effect dynamics.



Pendulum

The **state space** of pendulum consists of x and y positions, and angular velocity, and the **action** applies a torque to the pendulum. Pendulum's simple yet continuous dynamics make it ideal for testing an architectures as world models. Models trained random trajectories predict new random trajectories, and trajectories generated by an agent (Out of Distribution). The horizontal line at **0.24 depicts the average residual of a dummy model** that returns its input state (the identity). A dummy model that selects a random state from a set of random trajectories gets an average score of y=2.





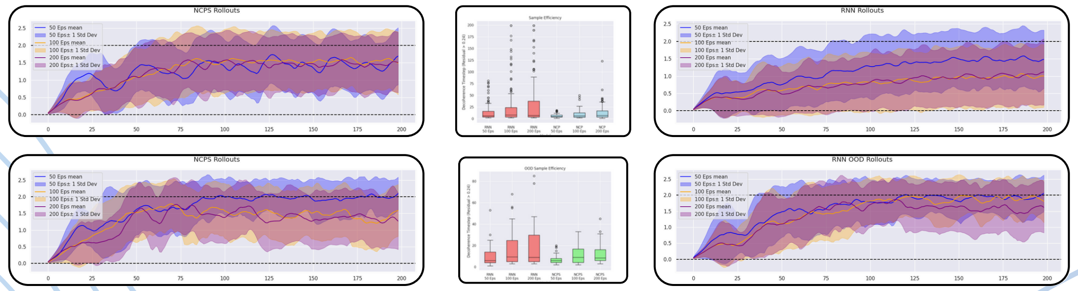
https://gymnasium.farama.org/environments/classic_control/pendulum/

The boxplots describe at what point the sum absolute residual for a single trajectory exceeds 0.24. That is, when the identity would give a closer prediction. The longer a predicted trajectory lasts with a low residual, the better.

For in distribution trajectories, RNN give the lowest mean sum of absolute residuals. However, the boxplots show that the Residual Connection produces a very similar number of trajectories that stay low, while having better performance with fewer samples.

Non-Markovian Pendulum

This environment is identical to the above Pendulum, except **we hide the velocity**. This makes the environment non-markovian, which means that there is insufficient information in a single state to predict the next one. Residual Connection models are ommitted as they have no temporal aspect to learn how the state changes over time. With less information, **the predicted paths of all models diverge from the ground truth much faster.** The RNN performs better, maintaining more predicted trajectories closer to the ground truth than the NCPs.



Future Work

We plan to Integrate these methods into model based learning with PPO agents and compare their agents respective decision-making performance. A larger issue is forming a hypothesis for why NCPS is under performing in our benchmarks, and identifying a set of environments better suited for NCPS. We would also like to increase our number of compared architectures and environments.