Postgraduate Project

TetrisWorld: From World Models to Action Planning

Project description

World models aim to learn the dynamics of an environment directly from observed state transitions. Once trained, such a model can simulate possible futures without using the original environment rules. An important open question is whether a world model trained purely for prediction can subsequently be reused for decision-making without retraining its dynamics. This project investigates this question using Tetris as a controlled experimental environment. Tetris provides deterministic dynamics, an exact ground-truth simulator and a small discrete action space, making it possible to rigorously measure both prediction accuracy and planning reliability. The student will first implement a Tetris simulator and define a Markov state representation containing sufficient information to uniquely determine the next state, including the board configuration, active piece, position and rotation. The simulator will be used to construct a transition dataset, with deliberate sampling of rare but important events such as line clears, wall interactions and near top-out positions. A forward dynamics model will then be trained to predict the next state given the current state and action. The student will systematically investigate design choices such as state representation, network architecture, training objective and sampling strategy. Model performance will be evaluated not only using one-step accuracy but also through multi-step autoregressive rollouts, where small prediction errors may compound over time. The second part of the project will investigate whether the frozen world model can support action selection. Given a goal, such as clearing a line or reaching a target board configuration, the student will search possible action sequences by rolling them out through the learned model. Approaches may include model-predictive control, beam search or another suitable planning algorithm. An inverse dynamics model may also be explored as an extension.

The central research question is therefore:

> *To what extent can a world model trained only for forward prediction be reused, without retraining its dynamics, for reliable action planning?*

The project will compare predicted plans with execution in the exact Tetris simulator, allowing planning failures to be traced back to model error, search error or state-distribution shift. The models and datasets used in the project are intentionally small, allowing the complete project to be carried out on a standard desktop or laptop without dedicated high-performance computing resources.

Deliverables

- A literature review covering world models, learned simulators, forward and inverse dynamics, model-based reinforcement learning and planning with learned models.
- A Tetris simulator with a formally documented state representation, transition function and action space.
- A transition dataset with a deliberate strategy for sampling rare and decision-critical states.
- A trained forward dynamics model and systematic evaluation of key design choices.
- Ablation studies examining factors such as representation, model architecture, training objective or dataset composition.
- A playable neural Tetris demonstration in which the learned model replaces the original transition dynamics.
- Evaluation of one-step prediction accuracy and long-horizon rollout fidelity using exact-match and rule-violation metrics.
- A goal-conditioned planner operating on the frozen learned world model.
- Evaluation of planning success rate, search cost and robustness across different planning horizons.
- Analysis of the relationship between model prediction error and downstream planning performance.
- Source code, experiment scripts, datasets, results and documentation.
- A final MSc dissertation presenting the methodology, results, analysis and conclusions.

Possible extension: train an inverse dynamics or action-prediction model and compare it with explicit planning through the forward model.

Suggested timeline

- **Months 1–2:** Literature review, simulator design, state representation and dataset construction.
- **Months 2–3:** Forward dynamics model implementation and initial training.
- **Months 3–4:** Model refinement, ablations and multi-step rollout evaluation.
- **Months 4–5:** Planning algorithm implementation and world-model-based action selection.
- **Months 5–6:** Planning evaluation, error analysis, additional experiments and dissertation writing.

Equipment/Software required to carry out this project

- Standard personal computer or laptop: existing equipment, £0 additional cost.
- Python: open source, £0.
- PyTorch: open source, £0.
- NumPy, Matplotlib and Pygame: open source, £0.
- Git/GitHub: free educational use, £0.

Total additional equipment/software cost: £0.

All core experiments can be completed using CPU, Apple Silicon or an optional consumer GPU. Dedicated GPU or HPC resources are not required.

Skills developed in carrying out this project

- Understanding of world models and model-based decision-making.
- Forward and inverse dynamics modelling.
- Neural-network design and experimentation in PyTorch.
- Representation learning for structured state spaces.
- Dataset design and rare-event sampling.
- Autoregressive rollout analysis and distribution shift.
- Search and planning using learned dynamics.
- Model-predictive control and related planning concepts.
- Reinforcement-learning concepts including states, actions, transitions and goals.
- Rigorous experimental methodology and ablation studies.
- Analysis of interactions between prediction quality and downstream decision quality.
- Python programming, software engineering and reproducible experimentation.
- Scientific writing and presentation of research results.

Skills required to carry out this project

- Good programming experience, preferably in Python.
- Basic knowledge of machine learning and neural networks.
- Familiarity with basic probability, optimisation or algorithms.
- Willingness to work with PyTorch.
- Previous experience with reinforcement learning or world models is useful but not required.
- No GPU programming or specialist hardware experience is required.