Structured World Understanding: Integrating Object-Centric Learning with Model-Based Reinforcement Learning

Master's Thesis: Summer Semester 2025
Master thesis by Your Full Name (Student ID: 1234567)
Date of submission: 24.10.2025

1. Review: Jannis Blüml
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science

Institute of Computer Science

Machine Learning Research Group

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Your Full Name, die vorliegende master thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 24.10.2025

Y. Name

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

List of Abbreviations

RL	Reinforcement Learning
MBRL	Model-Based Reinforcement Learning
MFRL	Model-Free Reinforcement Learning
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
PPO	Proximal Policy Optimization
A3C	Asynchronous Advantage Actor-Critic
DQN	Deep Q-Network
RSSM	Recurrent State Space Model
KL	Kullback-Leibler
MSE	Mean Squared Error
RGB	Red-Green-Blue
GPU	Graphics Processing Unit
JAX	Just After eXecution

1. Introduction

Reinforcement Learning as one of the three flavors in Machine Learning among Supervised and Unsupervised Learning follows the approach of having an agent performs actions in an environment and receiving rewards for certain actions. So unlike Supervised Learning for example there is no ground truth from the which the agent can learn from. Among numerous distinctions within Reinforcement Learning there is the difference between Model Based Reinforcement Learning and Model Free Reinforcement Learning. The Latter focuses on the learning of only the Agent which is deployed in the environment (or learns from samples). Thus the agent will have no explicit way of predicting the next state of the environment, although sufficiently large for example neural networks might create some implicit model of the world. Model based Reinforcement learning augments the architecture by a Model that is predicting the next state of the world but also still trains an agent. Although model free approaches usually are less complex since they only have to learn one model, the agent they also have significant disadvantages.

Model Free Learning approaches often struggle with sample efficiency and generalization to new tasks. Small changes in the environment can lead to significant drops in performance. Moreover, agents can learn misaligned policies that exploit loopholes in the environment rather than achieving the intended goals. These challenges highlight the need for more structured and interpretable learning methods. So Object Centricity and Model Based approaches both have the goal of improving abstract reasoning by developing a more abstract understanding of the world and improving sample efficiency. 1.3 Research Questions and Objectives This thesis aims to investigate the potential benefits of integrating object-centric learning into model-based reinforcement learning frameworks to address these challenges by harvesting the strengths of both paradigms. This thesis is therefore guided by the following research questions:

1. How do object-centric world models compare to pixel-based world models regarding prediction accuracy and sample efficiency?

2. Can an actor-critic framework be effectively trained on simulated rollouts from object-centric world models?
3. What is the impact of different world model architectures (such as LSTM-based versus alternative designs) on the overall performance of object-centric model-based reinforcement learning in structured environments?
4. Does the object-centric inductive bias in world models improve robustness to changes in the environment?

Within this thesis we propose to develop a world model that is running on top of object centric states provided by the JAXAtari framework. This world model will be based on an LSTM architecture and will be trained to predict the next state of the environment given the current state and action. An actor-critic framework based on the DreamerV2 architecture will be integrated with the world model to learn policies from imagined rollouts. The training pipeline will involve alternating between updating the world model and optimizing the policy based on simulated experiences. The performance of the proposed approach will be evaluated in the Pong environment from the Atari benchmark suite, comparing it against pixel-based baselines and assessing its sample efficiency and robustness.

The remainder of this thesis is structured as follows: Chapter 2 provides background information on reinforcement learning, model-based approaches, and object-centric representations. Chapter 3 details the methodology, including the world model architecture, actor-critic integration, and training pipeline. Chapter 4 presents the experimental setup, results, and analysis. Chapter 5 discusses the findings, challenges, and implications of the results. Finally, Chapter 6 concludes the thesis and outlines directions for future work.

2. Background and Related Work

2.1. Reinforcement Learning

(TODO (MDP's, model-free, model-based als Teil davon), "Dreamer Architecture"(vgl. auch dreamerv1, dreamerv2, dreamerv3, evtl. PlaNet), Öbject Centric RL"(als Teil davon auch object-centric model-based RL)

Reinforcement Learning (RL) is a machine learning paradigm that involves training agents to make sequential decisions by interacting with an environment. At its core, RL is often formalized using Markov Decision Processes (MDPs), where the environment's dynamics are defined by states, actions, transition probabilities, and reward functions. RL can be broadly categorized into model-free and model-based approaches.

In model-free RL, the agent learns a policy directly from interactions with the environment without constructing an explicit model of the world. By contrast, model-based RL incorporates a world model to predict future states or rewards, enabling more sample-efficient training by simulating rollouts in the constructed model. This abstraction allows agents to generalize better and potentially handle unseen situations more effectively.

2.2. Dreamer Architecture

The Dreamer architecture represents a family of model-based RL algorithms, namely DreamerV1, V2, and V3, that focus on combining world models with actor-critic methods for policy and value learning. Dreamer utilizes latent dynamics models to predict environment behavior in a compact latent space, enabling the agent to train policies on imagined rollouts. Notable predecessors like PlaNet introduced the concept of latent space dynamics, which Dreamer extends by incorporating actor-critic frameworks.

DreamerV2 builds on DreamerV1 by improving stability and scalability to discrete action spaces, commonly used in Atari benchmarks. DreamerV3 further generalizes the approach to handle diverse tasks with limited hyperparameter tuning.

Object-Centric Reinforcement Learning (OCRL) emphasizes representing an environment as a collection of interacting entities, rather than processing the entire environment as raw inputs, such as pixels. This paradigm improves sample efficiency, generalization, and interpretability by focusing on objects and their relationships.

Object-centric model-based RL combines the abstraction of object representations with predictive world models. By leveraging structured object-centric states, agents can reason about dynamic interactions in a more robust and interpretable manner compared to pixel-based representations.

There are multiple way to to train a world model. As mentioned above already you can predict latent states or directly predict future observations. There are also multiple different architectures which can do the job. From a simple MLP over LSTMs to Transformers there have been many different approiaches to that same problem. Tes

DreamerV2 [3] is a model-based RL algorithm.

Explain in general what dreamer does what are the components and how do they play together

The λ-target is defined recursively as:

$$V_t^\lambda = \hat{r}_t + \hat{\gamma}_t \cdot \begin{cases} (1-\lambda) v_\xi(\hat{z}_{t+1}) + \lambda V_{t+1}^\lambda & \text{if } t < H \\ v_\xi(\hat{z}_H) & \text{if } t = H \end{cases} \tag{2.1}$$

where $\hat{r}_t$ represents the predicted reward, $\hat{\gamma}_t$ is the predicted discount factor, $v_\xi(\hat{z}_t)$ is the critic's value estimate, and $\lambda = 0.95$ controls the weighting between immediate and future rewards. This formulation creates a weighted average of n-step returns, where longer horizons receive exponentially decreasing weights.

In my case I dont have z but simply the observation itself. So no latent space.

The actor network in DreamerV2 employs a sophisticated loss function that combines multiple gradient estimators to achieve both learning efficiency and convergence stability. The actor aims to maximize the same λ-return targets used for critic training, incorporating intermediate rewards directly rather than relying solely on terminal value estimates.

The combined actor loss function is formulated as:

$$\mathcal{L}(\psi) = \mathbb{E}\left[\sum_{t=1}^{H-1}\Big[-\rho \ln p_\psi(\hat{a}_t|\hat{z}_t)\,\mathrm{sg}(V_t^\lambda - v_\xi(\hat{z}_t)) \quad \text{[REINFORCE]} \right. \tag{2.2}$$

$$-(1-\rho)V_t^\lambda \quad \text{[Dynamics Backprop]} \tag{2.3}$$

$$\left. -\eta\mathcal{H}[a_t|\hat{z}_t] \quad \text{[Entropy Regularization]}\Big]\right. \tag{2.4}$$

The loss function incorporates three distinct components:

REINFORCE Gradients: REINFORCE algorithm, which maximizes the log-probability of actions weighted by their advantage values. The advantage is computed to be the difference between the lambda-return and the critic's estimate, with gradients stopped around the targets (denoted by sg stop gradient) to prevent interference with critic learning

Entropy Regularization: The third term encourages exploration by maximizing the entropy of the action distribution. The entropy coefficient η controls the trade-off between exploitation and exploration, with higher values promoting more diverse action selection.

The weighting parameter ρ determines the relative contribution of REINFORCE versus straight-through gradients. For discrete action spaces like Atari, DreamerV2 typically uses $\rho = 1$ (pure REINFORCE) with $\eta = 10^{-3}$, while continuous control tasks benefit from $\rho = 0$ (pure dynamics backpropagation) with $\eta = 10^{-4}$.

The critic network in DreamerV2 serves as a value function approximator that estimates the expected discounted sum of future rewards from any given latent state. This component is essential for both the λ-target computation and providing baseline estimates for the REINFORCE algorithm, making it a cornerstone of the learning process.

The critic is trained using temporal difference learning with the λ-targets as regression targets. The loss function is formulated as a squared error between the critic's predictions and the computed λ-returns:

$$\mathcal{L}(\xi) = \mathbb{E}\left[\sum_{t=1}^{H-1}\frac{1}{2}\Big(v_\xi(\hat{z}_t) - \mathrm{sg}(V_t^\lambda)\Big)^2\right] \tag{2.5}$$

where $v_\xi(\hat{z}_t)$ represents the critic's value estimate for latent state $\hat{z}_t$, and $\mathrm{sg}(V_t^\lambda)$ denotes the λ-target with stopped gradients. The gradient stopping prevents the critic's learning

from interfering with the target computation, maintaining the stability of the temporal difference updates.

Several key design choices enhance the critic's learning efficiency:

Target Network Stabilization: Following the approach used in Deep Q-Networks, DreamerV2 employs a target network that provides stable targets for critic learning. The target network is a delayed copy of the critic parameters, updated every 100 gradient steps. This approach prevents the rapid changes in the critic from destabilizing the learning targets.

Trajectory Weighting: The loss terms are weighted by cumulative predicted discount factors to account for episode termination probabilities. This weighting ensures that states likely to lead to episode endings receive appropriate emphasis during training.

Compact State Representation: Unlike traditional value functions that operate on high-dimensional observations, the DreamerV2 critic leverages the compact latent states $\hat{z}_t$ learned by the world model. This representation provides several advantages: reduced computational complexity, better generalization across similar states, and improved learning efficiency due to the structured nature of the latent space.

The critic architecture consists of a multi-layer perceptron with ELU activations and approximately 1 million trainable parameters. The network outputs a single scalar value representing the expected return from the input state, enabling efficient batch processing of imagined trajectories during training.

2.3. Object-Centric Reinforcement Learning

(TODO mention object-centric model-based RL)

- **High Dimensionality:**
- **Poor Generalization:**
- **Lack of Semantic Understanding:**
- **Sample Inefficiency:**
- **Sensitivity to Noise and Distractors:**

In contrast to pixel-based methods, object-centric learning approaches leverage understanding of objects to create an abstract representation of the environment. This mimics human perception, which naturally segments scenes into distinct entities and focuses on their interactions [4]. Object-centric learning represents a paradigm shift in how reinforcement learning agents process and understand their environments.

Therefore object centricity provides several advantages over pixel based methods.

Compositional Understanding: Object-centric representations naturally excel at treating input features as compositions of distinct entities. Making them understand that certain features correspond to the velocity and position of a ball and others to a paddle. This compositionality allows agents to reason about individual objects and their interactions.

Improved Generalization: By focusing on objects rather than pixel patterns, agents can generalize better across visually different but similar scenarios. For example agents that understands the concept of a "ballän an abstract obect can use this knowledge in environments where the ball has a different shape or appearance in general. (SOURCE)

Enhanced Interpretability: Object-centric representations provide naturally a better interpretability then pixel based representations since the agent can reason better that it perceives things as objects rather than "random"pixels.

Sample Efficiency: The structured nature of object-centric representations often leads to more sample-efficient learning. By working with compact, meaningful features rather than high-dimensional pixel data, agents can learn policies with fewer rollouts. THis is possible because the object space is already some kind of latent space which is usually the way things work (SOURCE)

A significant development in this field is the introduction of OCAtari (Object-Centric Atari) by Delfosse et al. [2]. OCAtari extends the widely-used Arcade Learning Environment (ALE) by providing resource-efficient extraction of object-centric states for Atari 2600 games. THis framework fixes a gap, where despite growing interest in object-centric approaches, no standardized benchmark existed for evaluating such methods on the popular Atari domain.

The OCAtari framework works by two ways of extracting object states. It can either extract the object states directly from the emulator’s RAM or by using template matching on the rendered frames. The RAM-based extraction is more efficient and accurate. Directly having those extracted object states allows for significantly faster training times and also supports researches in making comparable findings since everyone can use the same object states.

The importance of object-centric approaches in Atari environments is esspecially present in light of pixel-based methods in these domains. Delfosse et al. demonstrated that deep RL agents without interpretable object-centric representations can learn misaligned policies even in simple games like Pong. [1] This misalignment problem strengthens the argument for using object-centric understanding rather than attempting to retrofit interpretability onto pixel-based systems.

(TODO TALK ABOUT JAXATARI)

Object-centric representations naturally lend themselves to relational reasoning, where agents must understand not just individual objects but also the relationships and interactions between them. This capability is crucial for complex decision-making in multi-object environments where the optimal policy depends on understanding how different entities influence each other.

Relational reasoning in reinforcement learning encompasses several key aspects:

Spatial Relationships: Understanding the relative positions of objects and how spatial configurations affect optimal actions. For example, in Pong, the relationship between the paddle position and ball trajectory determines the appropriate movement strategy.

Temporal Relationships: Tracking how object relationships evolve over time and predicting future interactions. This temporal aspect is particularly important for planning and anticipatory behavior.

Causality: (SOURCE) Recognizing cause-and-effect relationships between actions and object state changes. This understanding enables more sophisticated planning and can help avoid unintended consequences.

The integration of relational reasoning with model-based approaches offers significant potential for improving agent performance. By incorporating relational structure into world models, agents can make more accurate predictions about future states and plan more effectively. This integration forms a core component of our proposed approach, where object-centric world models explicitly encode relational information to enhance both prediction accuracy and policy learning.

3. Methodology

3.1. World Model Architecture

3.1.1. LSTM-Based World Model (PongLSTM)

explain LSTM explain why is might be good for world models in general instead of models that only take the last state into account

3.1.2. Alternative Architectures Explored

Talk about MLP, Transformer, other RNNs (TODO) erwähnen dass ich sie ausprobiert habe und warum ich sie nicht genommen habe (Was ich nicht probiert habe in Future work oder related work)

3.1.3. State Normalization and Stability

why normalization helps in RL in general and why it was used here (TODO im Background erwähnen dass es Normalisierung in RL gibt und kurz reinschreiben dass ich das gemacht habe)

3.2. Actor-Critic Integration

Explain how the actor-critic framework is integrated with the object-centric world model. Discuss any modifications made to the standard actor-critic approach to accommodate the object-centric representations.

3.2.1. DreamerV2-Style Actor-Critic

Explain the Dreamer approach and how it differs which is mainly in the latent space

3.2.2. Policy Learning in Imagined Rollouts

how the policy is learned in imagined rollouts from the world model

3.2.3. Lambda-Return Computation

provide the formular here and briefly explain it

3.3. Training Pipeline

explain when the worldmodel gets retrained and how sampling works

3.3.1. World Model Training and experience collection

(TODO Pseudo CODE mit Abbildung) explain how the initial experience for the world model is collected and how it can be updated during later training stages

3.3.2. Policy Optimization

explain how the policy is optimized and how often

4. Experiments and Results

(QUESTION Soll ich hier Experimente 1,2,3 etc. auflisten und die dann auf die Forschungsfragen mappen z.B. Experiment 1 10 Schritte unrollst und real vs model vs oc-model bilder -> vielleicht hier einfach mein OC State Rendern und dann irgendeinen Bild Loss wählen, Dafür muss ich aber ein pixel based world model erstmal haben (einfach ein MLP hinzimmern?)) (Normalen Dreamer zum laufen bringen auch für RQ3 Farbe vom Ball ändern) (Ablation Studies: Unterschiedliche Architekturen, Unterschiedliche Reward Funktionen)

4.1. Experimental Setup

* Experimental Design: Baseline, Comparisons, Eval Metrics * Environment and Setup: Pong Environment Characteristics, Object-Centric State Description * Hyperparameters and Configuration: Model Architectures, Training Regimes * Hardware and Implementation Details: Computational Resources, Software Frameworks * Evaluation Protocol: Training and Testing Procedures, Statistical Analysis

4.2. World Model Performance

4.2.1. Prediction Accuracy Analysis

Answer here : (RQ1) How does the prediction accuracy of object-centric world models compare to pixel-based world models in terms of state transition prediction and long-term rollout quality in the Pong environment?

4.2.2. Long-Term Rollout Quality

Answer here : (RQ1) How does the prediction accuracy of object-centric world models compare to pixel-based world models in terms of state transition prediction and long-term rollout quality in the Pong environment?

4.2.3. Model Stability Assessment

Answer here : (RQ1) How does the prediction accuracy of object-centric world models compare to pixel-based world models in terms of state transition prediction and long-term rollout quality in the Pong environment?

4.3. Policy Learning Results

4.3.1. Sample Efficiency Comparison

Answer here : (RQ1) To what extent does integrating object-centric representations with model-based RL improve sample efficiency compared to model-free baselines and pixel-based model-based approaches?

4.3.2. Final Performance Evaluation

Answer here : (RQ1) To what extent does integrating object-centric representations with model-based RL improve sample efficiency compared to model-free baselines and pixel-based model-based approaches?

4.3.3. Learning Curve Analysis

Answer here : (RQ1) To what extent does integrating object-centric representations with model-based RL improve sample efficiency compared to model-free baselines and pixel-based model-based approaches?

4.4. Ablation Studies

4.4.1. Impact of Object-Centric Representations

Answer here : (RQ2) How effectively can the DreamerV2-style actor-critic framework learn optimal policies when trained on imagined rollouts from object-centric world models?

4.4.2. Architecture Component Analysis

Answer here : (RQ3) What is the impact of different world model architectures (LSTM-based vs. alternatives) on the overall performance of object-centric model-based RL in structured environments?

4.4.3. Reward Function Design Effects

Answer here : (RQ3) What is the impact of different world model architectures (LSTM-based vs. alternatives) on the overall performance of object-centric model-based RL in structured environments?

4.4.4. Real vs. Model Comparison

Answer here : (RQ2) How effectively can the DreamerV2-style actor-critic framework learn optimal policies when trained on imagined rollouts from object-centric world models?

4.4.5. Policy Behavior Visualization

Answer here : (RQ2) How effectively can the DreamerV2-style actor-critic framework learn optimal policies when trained on imagined rollouts from object-centric world models?

5. Discussion

* Challenges during training, what decreases model performance

* Analysis of Results - Strengths of the Proposed Approach - Limitations and Challenges - Comparison with Existing Methods

* Technical Insights - World Model Design Choices - Training Stability Issues - Reward Engineering Importance

* Implications for Object-Centric RL - Benefits of Integration - Generalization Potential - Scalability Considerations

* Future Research Directions - More Complex Environments - Improved Object Discovery - Multi-Object Scenarios

6. Conclusion and Future Work

- Summary of Contributions
- Key Findings
- Limitations and Future Work
- Final Remarks

References

[1] Quentin Delfosse u. a. *Interpretable Concept Bottlenecks to Align Reinforcement Learning Agents*. 2024. arXiv: 2401.05821 [cs.LG]. URL: https://arxiv.org/abs/2401.05821.

[2] Quentin Delfosse u. a. „OCAtari: Object-Centric Atari 2600 Reinforcement Learning Environments“. In: 2024. arXiv: 2306.08649 [cs.LG]. URL: https://arxiv.org/abs/2306.08649.

[3] Danijar Hafner u. a. „Dream to Control: Learning Behaviors by Latent Imagination“. In: *International Conference on Learning Representations (ICLR)*. 2020.

[4] Li Nanbo, Cian Eastwood und Robert B. Fisher. *Learning Object-Centric Representations of Multi-Object Scenes from Multiple Views*. 2021. arXiv: 2111.07117 [cs.CV]. URL: https://arxiv.org/abs/2111.07117.

A. Implementation Details

A.1. Code Structure and Organization

Provide an overview of the project's codebase, including the main modules, their purposes, and how they interact.

A.2. Hyperparameter Sensitivity Analysis

Discuss the impact of varying key hyperparameters on model performance and training stability.

A.3. Additional Experimental Results

Include supplementary experimental results that support the main findings, such as extended comparisons or detailed metrics.

B. Technical Specifications

B.1. Hardware Requirements

List the hardware specifications required to reproduce the experiments, including GPU/CPU details and memory requirements.

B.2. Software Dependencies

Detail the software libraries, frameworks, and versions used in the implementation.

B.3. Reproducibility Guidelines

Provide step-by-step instructions for reproducing the experiments, including setup, data preparation, and execution.

C. Supplementary Figures and Tables

Include additional figures and tables that complement the main text, such as visualizations of training curves or ablation study results.