Learning in Infants using Intrinsically Motivated Goal Conditioned Reinforcement Learning

Dissertation submitted in partial fulfilment for the award of the degree

Master of Technology in Computer Science

by

T I Darsan

Roll No.: CS2435
M.Tech, 2nd year

Under the supervision of
Prof. Jochen Triesch, Dr. Malay Bhattacharyya

INDIAN STATISTICAL INSTITUTE

June, 2026

CERTIFICATE

This is to certify that the work presented in this dissertation titled “Learning in Infants using Intrinsically Motivated Goal Conditioned Reinforcement Learning”, submitted by T I Darsan, having the roll number CS2435, has been carried out under the supervision of Prof. Jochen Triesch at FIAS and Dr. Malay Bhattacharyya as an internal supervisor at ISI Kolkata in partial fulfilment for the award of the degree of Master of Technology in Computer Science during the session 2024-26 in the Computer and Communication Sciences Division, Indian Statistical Institute. The contents of this dissertation, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Jochen Triesch
Professor, Neuroscience Department
Frankfurt Institute of Advanced Studies(FIAS), Kolkata

Dr. Malay Bhattacharyya
Associate Professor, Machine Intelligence Unit
Associate Member, Centre for Artificial Intelligence and Machine Learning
Associate Member, Technology Innovation Hub on Data Science, Big Data Analytics, and Data Curation
Indian Statistical Institute, Kolkata

Acknowledgements

First and foremost, a massive thank you to my supervisor at FIAS, Prof. Jochen Triesch. Thank you for always being in my corner, offering incredible guidance, and quite literally being there whenever I needed a hand. Working with you has been an incredible experience. I also want to give a special thanks to Salar and Francisco for their contributions to the project and for always jumping in to help me out whenever I needed a hand.

I want to say a huge thank you to my supervisor at ISI Kolkata, Dr. Malay Bhattacharyya. Malay sir, thank you so much for taking me on as your student and patiently guiding me through the twists and turns of this whole process.

To my family: thank you from the bottom of my heart for backing me when I decided to go for a second Master's degree. It meant the world to me that you understood that this was exactly what I needed to do.

This journey wouldn't have been half as bright without the amazing people around me. Shoutout to my friends at ISI Kolkata for making the last year and a half so incredibly fun, and to the FIAS crew for keeping the work days lively and full of good times. A big thanks to my friends at IISER Kolkata for the weekend escapes, and to my IISER Pune friends for helping me build my foundational base in the first place—and, of course, for spending quality time in DB with me and helping make my Europe trip happen!

Date: 10-06-2026

T I Darsan
Roll No.: CS2435
M.Tech, 2nd year
Indian Statistical Institute

Abstract

Traditional artificial intelligence models learn by passively digesting large datasets. In contrast, human infants discover skills by actively interacting with their bodies and environments without explicit external rewards. This thesis introduces the Composer Architecture, a machine learning framework designed to mimic this autonomous, open-ended development. The Composer architecture operates in a multi-stage loop, the latent model using Contrastive Learning Through Time (CLTT) to compress high-dimensional raw data from visual, proprioceptive, and touch sensors into a low-dimensional space. To preserve data relationships and prevent topological collapse, a Softmax activation forces these latent representations to lie smoothly on a probability simplex. A goal-conditioned reinforcement learning policy then trains on this space by targeting randomly sampled one-hot goals. We evaluated the architecture on the MIMo platform, a highly realistic simulation of an 18-month-old child embedded in the MuJoCo physics engine. Testing progressed from primitive shapes to complex multi-joint control channels on the robot. On a single-finger setup, the architecture mapped latent extrema directly to full extension and flexion, while five-finger trials isolated whole-hand opening and closing configurations. To scale control, a hierarchical extension called Hi-Composer successfully coordinates complex limb motions by routing high-level commands as sub-goals to localized lower-level finger policies. Finally, full-body exploration benchmarks on a rollover task validated the "thin pancake" hypothesis. Compared to white-noise walks, the Composer architecture expands the agent's spatial reach by 134 percent while successfully restricting local exploration to an organized, lower-dimensional submanifold. This demonstrates that self-generated latent goals effectively guide open-ended exploration into structured movements.

Contents

List of Figures

List of Tables

1 Introduction

Human infant development is an autonomous, open-ended discovery process driven by internal mechanisms rather than explicit external rewards. Traditional reinforcement learning paradigms rely heavily on hand-crafted extrinsic feedback, failing to capture the biological reality of early cognitive growth. To construct a more accurate computational model of infant development,we introduce the framework of composer which takes into account the Intrinsically Motivated Goal-Conditioned Reinforcement Learning. This intrinsic approach closely mirrors how biological infants navigate high-dimensional environments, progressively organizing spontaneous movements into structured, intentional behaviors through play-like exploration. A similar approach can be found in the paper by Yordanova et al. [3].

Scaling these models to human-like complexity, however, introduces a large combinatorial explosion of goals that normal, flat reinforcement learning architectures cannot manage. To solve this scalability challenge, we introduce the hierarchical version of Composer which is called Hi-Composer. Operating on the principle of compositionality, it decomposes complex target behaviors into primitive behavioral modules. Early on, an infant focuses on isolating basic motor primitives, such as flexing individual fingers or tracking objects visually. Driven by intrinsic motivation, the composer architecture systematically binds these learned primitives into coordinated behavioral sequences which can be further used for complicated tasks.

1.1 The MIMo Model

To validate intrinsically motivated models, it is essential to have an accurate platform in terms of morphology and sensory complexity that matches those of human beings. The Multi-Modal Infant Model (MIMo) illustrated in Figure 1.1 has been designed by Mattern et al. [1]. MIMo is an open-source architecture developed under the MuJoCo simulation environment as a means to bring together machine learning and cognitive neuroscience. MIMo mimics the anthropometric characteristics of an 18-month-old child with a detailed set of 5-fingered, soft hands that allow for realistic interactions with objects and their physical properties.

MIMo has a wide range of sensory parameters under four main sensory modalities:

- **Vision:** Stereoscopic vision using two virtual cameras placed at the head.
- **Vestibular System:** A 3-axis accelerometer and gyroscope situated in the skull to deliver orientation, balance, and acceleration feedback.

- **Proprioception:** Continuous tracking of joint positions ($qpos$) and joint velocities ($qvel$) across the model's articulated structure.
- **Somatosensation (Touch):** A specialized virtual skin mapping thousands of individual pressure sensors across the entire 3D mesh, simulating human tactile feedback during manipulation and self-contact.

This high level of multi-modal fidelity makes MIMo an exceptional tool for computational cognitive neuroscience experiments. Researchers can use the model to simulate developmental coordination disorders or sensory atypicalities by intentionally degrading visual feeds, reducing tactile sensitivity, or restricting joint ranges.[4] Such experiments provide valuable insight into how the brain integrates disparate sensory streams to form internal egocentric spatial maps, and how cortical representations reorganize during active physical development.

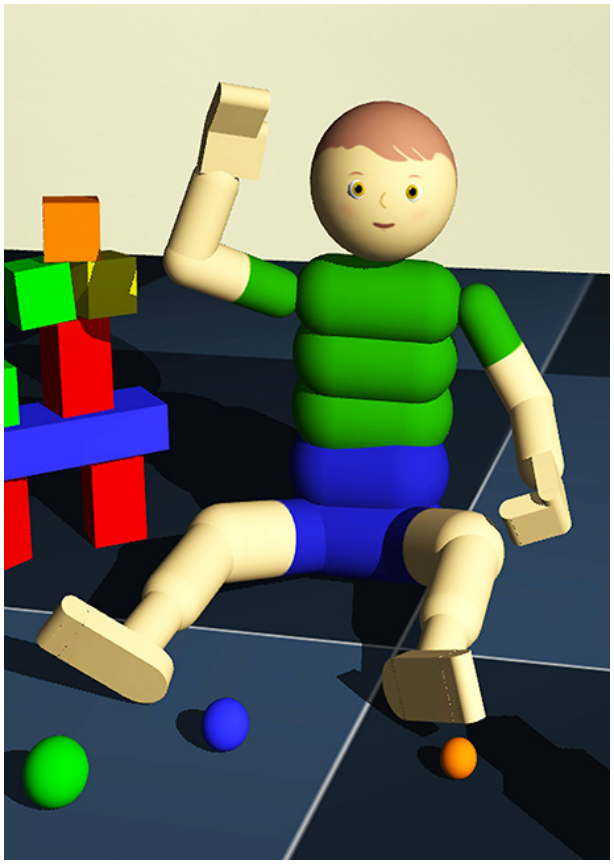


Figure 1.1: The MIMo play-room environment used for embodied interaction and developmental learning experiments.[1]

While the primary objective of deploying Composer on MIMo is to cultivate open-ended exploration, the platform is equally effective at mastering foundational motor milestones through simple external feedback. For small tasks, such as learning to

stand up from a seated position with hands glued to a (as shown in Figure 1.2), complex reward engineering is unnecessary; the policy's external reward function ($\mathscr{R}_t$) can be modeled simply as a function of head height:

$$\mathscr{R}_t = h_{\text{head}}(t) - \alpha \|\tau_t\|^2 \tag{1}$$

where $h_{\text{head}}(t)$ represents the vertical Z-coordinate of the head relative to the ground plane, and $\alpha\|\tau_t\|^2$ acts as a regularization penalty against excessive control torques (τ_t) to ensure smooth, natural movements. Other examples for such small tasks could be reaching for an object, catching a ball or just learning to roll.

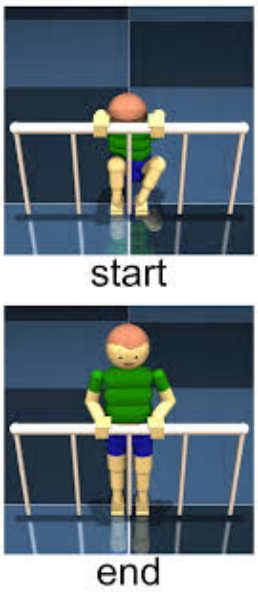


Figure 1.2: The MIMo standing task experiment.

2 Preliminaries

2.1 Latent Representations

The term "latent representation" as expalined in [5], denotes the process of compressing observations into a lower-dimensional feature space that contains the fundamental components of the data, such as the basic features and semantic structures present in high-dimensional raw data such as raw images from a camera or touch sensor data from multiple layers. In complicated settings, raw observations include a lot of noise and redundancy, such as background patterns or slight changes in light, which may easily overload a learning machine. Mapping raw observations to a latent space allows an agent or a world model to disregard irrelevant information and conduct its plans, predictions, and policy learning in an elaborate feature space.

2.1.1 Mathematical Definition

Mathematically, we define an encoder function f_ϕ parameterized by weights ϕ that maps an observation x from a high-dimensional observation space $\mathscr{X}$ to a lower-dimensional latent space $\mathscr{Z}$:

$$z = f_\phi(x) \tag{2}$$

Where $x \in \mathscr{X}$ (e.g., $\mathbb{R}^{C\times H\times W}$ for multi-modal images) and $z \in \mathscr{Z}$ (e.g., $\mathbb{R}^d$ where $d \ll C \times H \times W$). If the system uses a generative framework like an Autoencoder to reconstruct an approximation of the original state $\hat{x}$, a decoder function g_θ is utilized:

$$\hat{x} = g_\theta(z) = g_\theta(f_\phi(x)) \tag{3}$$

2.1.2 Contrastive Learning Through Time (CLTT)

For achieving these representations via a biologically plausible approach, this research employs Contrastive Learning Through Time (CLTT), a framework proposed by Schneider et al. [6], which is motivated from studies done on the field of developmental psychology regarding how human babies form invariance representations without label information in their environment, specifically in accordance with these principles:

- **No Artificial Augmentations:** While previous forms of contrastive learning have used artificial data augmentations such as random cropping or color jittering, CLTT uses natural transitions in time.
- **The "Close in Time, Will Align" Principle:** The core heuristic of CLTT is that successive observations experienced close together in continuous time are highly

likely to belong to the same underlying object or semantic concept.

- **Positive Pairs from Temporal Streams:** Positive pairs are generated procedurally by taking consecutive or near-consecutive frames (x_t, x_{t+k}) from an unsegmented temporal viewing sequence experienced during interaction.
- **Negative Pairs from Different Contexts:** Negative pairs are drawn from entirely separate sequences or distant timesteps, representing structurally distinct states or different objects.
- **Objective Optimization:** The encoder is trained using a contrastive InfoNCE loss function. This forces the representations of positive pairs to cluster closely together in the latent space $\mathcal{Z}$, while pushing negative pairs far apart:

$$\mathcal{L}_{\text{CLTT}} = -\log \frac{\exp(\text{sim}(z_t, z_{t+k})/\tau)}{\sum_j \exp(\text{sim}(z_t, z_j)/\tau)} \tag{4}$$

  where $\text{sim}(\cdot)$ represents a similarity metric and τ is a temperature parameter.
- **Application in the Composer Architecture:** By learning embeddings via CLTT, our composer architecture gains a stable, low-dimensional, and noise-resilient mapping of the environment. Here, we particularly disregard fast changing information like object orientation while slowly changing information (object identity) is selectively retained.

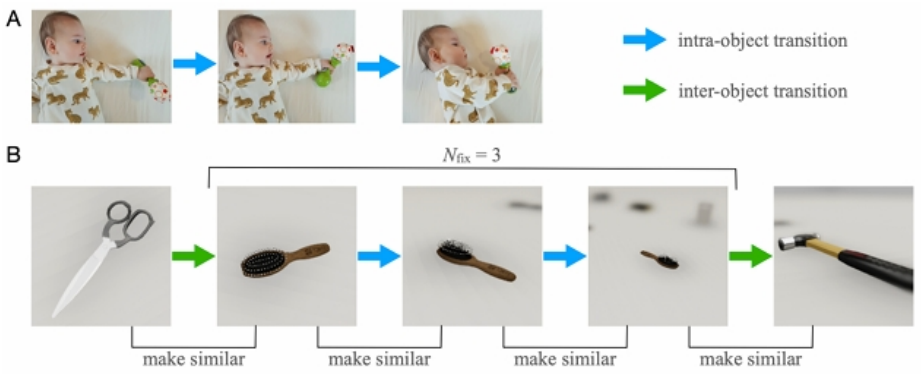


Figure 2.1: A. Infants learn about objects during extended interactions, where they experience different views of an object before directing their attention elsewhere. B. CLTT approach mimics the essence of such interactions. Latent representations of successive views (both intra-object and inter-object transitions) are made more similar.[2]

2.2 Reinforcement Learning

The foundational paradigm governing an agent's physical interactions in this work is Reinforcement Learning (RL)[7]. Formally structured as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, RL models an agent navigating a state space $\mathcal{S}$ by taking actions from an action space $\mathcal{A}$. At each discrete timestep t, the agent observes the current state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to its policy $\pi(a_t|s_t)$ as shown in Figure 2.2.

The environment then updates to a new state s_{t+1} governed by the transition probability function $\mathcal{P}(s_{t+1}|s_t, a_t)$, and issues a scalar reward signal $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$. The central objective of the learning policy is to find parameters that maximize the expected cumulative discounted reward over a temporal horizon, formalized as:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})\right] \tag{5}$$

where $\gamma \in [0,1)$ is the discount factor balancing intermediate rewards against long-term returns, and $\tau = (s_0, a_0, s_1, a_1, \ldots)$ represents an interaction trajectory sample.

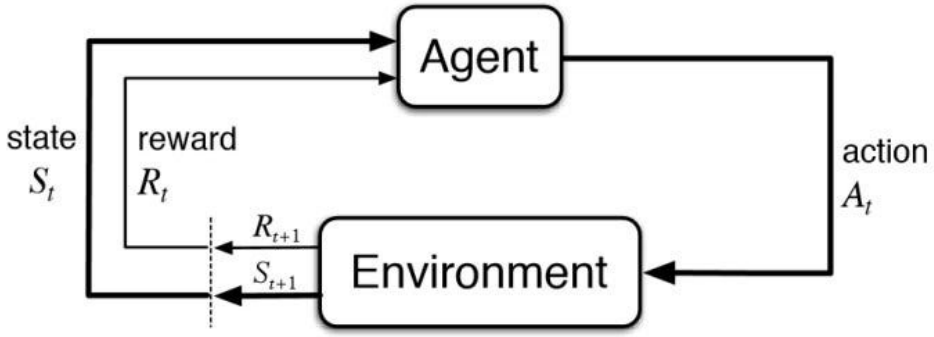


Figure 2.2: Reinforcement Learning

2.2.1 Proximal Policy Optimization (PPO)

To achieve stable policy optimization within high-dimensional sensorimotor environments, we employ Proximal Policy Optimization (PPO), an on-policy policy gradient method introduced by Schulman et al. [8]. Standard policy gradient methods often suffer from devastating policy degradation. This is when a parameter update steps too far outside stable regions. PPO resolves this instability by enforcing a specialised objective function that penalizes changes that move the updated policy too far from the old policy.

This update mechanism computes a probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and then optimizes the clipped objective function:

$$\mathscr{L}^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{6}$$

where $\hat{A}_t$ is the estimated generalized advantage value at timestep t, which dictates whether a taken action performed better or worse than the baseline average. The hyperparameter ϵ (typically configured between 0.1 and 0.2) serves to structurally clip the objective, removing the incentive for $r_t(\theta)$ to migrate outside the protective interval $[1-\epsilon, 1+\epsilon]$. This makes PPO highly data-efficient and robust for complex body controllers like the multi-jointed infant architecture.

2.2.2 Goal-Conditioned Reinforcement Learning

Standard reinforcement learning focuses on finding a policy that maximizes a single, fixed reward function for a specific task. In contrast, Goal-Conditioned Reinforcement Learning (GCRL) generalizes an agent's capabilities by training a single policy to achieve a infinite variety of distinct targets or states (goals) across the environment. This paradigm closely mimics an infant practicing multiple self-generated milestones during play-like exploration. [9]

In the GCRL framework, both the policy π and the reward function $\mathscr{R}$ are explicitly parameterized by a goal variable $g \in \mathscr{G}$, where the goal space $\mathscr{G}$ is often mapped directly to the learned latent space $\mathscr{Z}$. Instead of choosing actions based solely on the current state s_t, the goal-conditioned policy takes the target goal g as an explicit input:

$$a_t \sim \pi(a \mid s_t, g) \tag{7}$$

The environment computes a reward based on how close the agent's achieved state mapping $\phi(s_{t+1})$ is to the desired goal g. This is frequently expressed as a sparse reward function, which evaluates to 0 only if the agent successfully reaches the goal within a threshold radius ϵ_g, and -1 otherwise:

$$\mathscr{R}(s_t, a_t, s_{t+1}, g) = \begin{cases} 0 & \text{if } \|\phi(s_{t+1}) - g\| \leq \epsilon_g \\ -1 & \text{otherwise} \end{cases} \tag{8}$$

Alternatively, when dense feedback is preferred to guide early policy updates, it can be expressed as the negative Euclidean distance between the achieved state representa-

tion and the goal within the latent space:

$$\mathscr{R}(s_t, a_t, s_{t+1}, g) = -\|\phi(s_{t+1}) - g\|_2 \tag{9}$$

By optimizing this objective across a wide distribution of goals, the agent learns a universal policy capable of navigating between arbitrary states. When driven by intrinsic motivations, the GCRL framework allows the agent to autonomously sample goals from its CLTT-structured latent space, systematically building an organized repertoire of sensorimotor primitives.

3 The Composer Architecture

The main contribution of this thesis is the *Composer Architecture*. This framework is designed for open-ended skill discovery and sensory abstraction in developmental robotics. The architecture combines representation learning using temporal contrastive methods with goal-conditioned policy execution. A full overview of the core parts, structural rules, training loops, and scaling features of the Composer framework is presented below.

3.1 Core Components of the Architecture

The system splits the problem of high-dimensional physical interaction into two main, connected parts:

- **Latent Model (Encoder Network):** The network takes high-dimensional raw observations $x \in \mathscr{X}$ (for example, vision, joint angles, touch, etc., from MIMo) and compresses it down to a low-dimensional space. Here, it is important that the output of the last layer of this network is defined using the *Softmax* activation function rather than the common Rectified Linear Unit (ReLU):

$$z = \text{Softmax}\big(W_{\text{out}} \cdot h + b_{\text{out}}\big), \quad z \in \mathbb{R}^{d-1} \tag{10}$$

  where h is the output of the previous layer. In this way, the obtained latent code z lies strictly within the standard simplex Δ^{d-1}.

  The use of the Softmax activation function rather than ReLU can be understood as a necessity because of shape-preserving. Otherwise, in case of applying common ReLU activations, all points from large volumes (e.g., all third quadrant in the two-dimensional case) will be mapped to the origin point, while others will be squeezed at the vertices of the simplex face.

- **The Goal-Conditioned PPO Policy:** This part is responsible for turning abstract states into actual joint movements. The state vector given to the policy at each step is created by concatenating the current latent state vector z_t and a target goal vector g:

$$s_{\text{policy}} = [z_t \,\|\, g] \tag{11}$$

  Goals are represented as one-hot vectors that match the corners or vertices of the simplex. To encourage exploration, a new goal is picked randomly from the goal space $\mathscr{G}$ every 100 training iterations.

3.2 Reward Specification and Policy Learning

Instead of using rewards from outside the environment, the policy's reward function is completely internal. The reward $\mathscr{R}_t$ measures the distance between the target one-hot goal g and the current latent state z_t using a squared difference calculation:

$$\mathscr{R}_t = -\|g - z_t\|_2^2 \tag{12}$$

This reward acts as an internal guide for Proximal Policy Optimization (PPO) [8]. It trains the robot's body movements so that its current position matches the selected corner of the simplex.

3.3 The Continuous Training Loop

The training of the Composer architecture is split into a one-time initialization phase followed by a repeating optimization loop. This setup allows the agent to build an initial baseline and then continuously refine both its latent model and policy.

3.3.1 Initialization Phase (Done Once)

At the very beginning of training, the framework sets up the initial networks through two sequential steps:

1. **Phase 1: Initial Latent Model Training:** A data buffer is created by running a completely random walk (motor babbling) across MIMo's joints. These collected multi-modal sensory sequences are used to train the initial latent space based on Contrastive Learning Through Time (CLTT) [6] rules with a temporal InfoNCE loss.

2. **Phase 2: Initial Policy Grounding:** The initial Latent Model weights are frozen. The goal-conditioned PPO loop runs for the first time to practice reaching the randomly chosen one-hot goals, which connects physical movements to the abstract corners of the simplex.

3.3.2 The Core Repeating Loop (Repeated for N Iterations)

Once initialization is complete, the framework enters a repeating loop where the latent model and the PPO policy are continually retrained and updated together:

1. **Phase 3: Targeted Data Collection for Map Refinement:** To collect data that improves the boundaries of the representation, a new buffer is filled by making

the current policy walk directly from one corner of the learned simplex manifold to other corners.

2. **Phase 4: Latent Model Retraining and Blending:** The latent model is retrained on this new data buffer. To prevent the model from forgetting earlier representations, the newly updated network weights (Ω_{new}) are merged with the previous weights (Ω_{old}) using a linear mix:

$$\Omega_{\text{updated}} = (1-\lambda)\Omega_{\text{old}} + \lambda\Omega_{\text{new}} \tag{13}$$

where $\lambda \in (0,1)$ controls the update rate. Alternatively, the architecture can simply reload the previous weights as starting parameters before fine-tuning with the new data.

3. **Phase 5: Policy Retraining:** The updated latent model is frozen again, and the goal-conditioned PPO policy is retrained using the newly refined state embeddings to ensure the movements accurately track the updated corners of the simplex space.

3.3.3 Algorithmic Refinement and Emergent Manifold Extrema

By repeating this optimization loop over many iterations, an essential result appears: *points that are pushed to map onto the outermost corners of the low-dimensional latent simplex match up with extreme physical postures and actions in the high-dimensional original space.*

For example, if the system only looks at the joints of a single finger on the MIMo model, the optimization pushes the ends of a 1D simplex line to match exactly with full finger opening and full finger closing.

We get complex hand patterns when we do this for all five fingers. If the latent space is limited to a small 2D space, the corners represent behaviors like opening and closing the whole hand. To make these whole-hand movements work well in low dimensions, we add a small coupling or mechanical constraint between adjacent fingers.

3.4 Experimental Evolution

We do step-by-step experiments to evaluate the exploration, finally, in MIMo models:

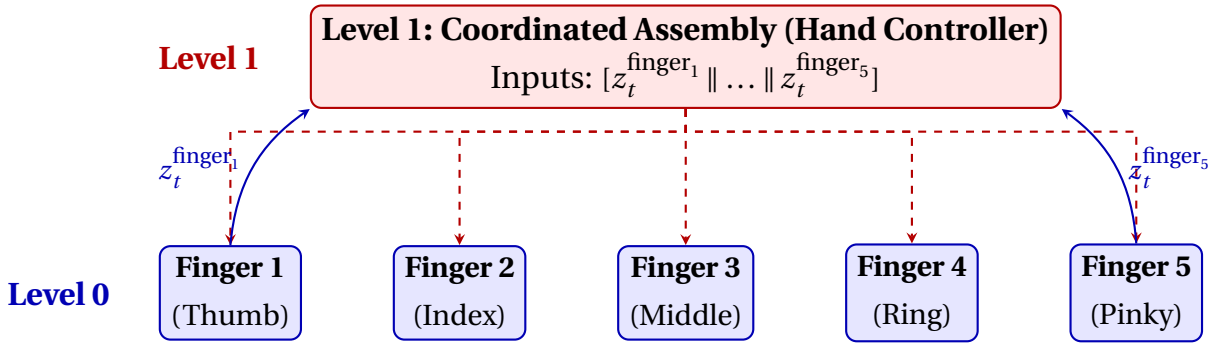
1. **Simple World Models:** First, we test the architecture on simple world models of cuboid, ellipsoid and dumbbell.

2. **MIMo platform:** Moving to complex robot bodies, the model is first tested on a single finger case and then on the full 5-finger hand case using the MIMo platform.

3. **Exploration Efficiency Analysis:** We track and compare whether the Composer architecture explores a larger volume of the state space compared to standard methods (like White noise walk or random walk) in high dimensions.

The final goal of this project is to apply this setup to the full body of the MIMo model. Because the entire MIMo has many moving joints and thousands of touch points, standard flat models run into training limits. To fix this dimensionality problem, we introduce a hierarchical version called the *Hi-Composer*.

3.5 Hierarchical Scaling: The Hi-Composer Architecture



While the standard Composer works well for one limb or a few joints it is hard to use for the body of the MIMo model. The full MIMO body has moving joints and lots of touch points on its virtual skin. When we put all these inputs into one model, training gets slow and unstable. To solve this we created a version called Hi-Composer. The Hi-Composer breaks the problem into parts by using the Composer across many connected levels or hierarchies. This way, the Composer handles each part of the MIMo's body separately.

To illustrate how the Hi-Composer operates, we consider a two-layer setup for controlling a hand, structured as follows:

- **Level 0 (Local Component Level):** This level is where we have blocks for small parts of the body. For example each of the five fingers on MIMo's hand has its block at this level. Each finger has its small latent model that helps to simplify the information about the joints in that finger. This model works with a policy that helps to control the movements of that specific finger.

- **Level 1 (Coordinated Muscle Level):** This level acts as a supervisor that coordinates the lower-level parts. Here, a single Level 1 block represents the entire left hand. It does not look at the raw finger joints directly. Instead, it observes the hand as a whole by reading the outputs of the level below it.

The training and execution pipeline for this hierarchical structure follows a bottom-up representation step followed by a top-down control loop:

3.5.1 Multi-Level Representation Mapping

Firstly, the low-level elements are trained according to the Composer procedures to develop latent spaces individually. After training the five-finger models is completed, we develop a state space for the hand at Level 1. The state vector for Level 1, represented as $X_t^{\text{Level 1}}$, is generated by combining the five latent vectors obtained from the finger models of Level 0:

$$X_t^{\text{Level 1}} = \left[z_t^{\text{finger}_1} \parallel z_t^{\text{finger}_2} \parallel z_t^{\text{finger}_3} \parallel z_t^{\text{finger}_4} \parallel z_t^{\text{finger}_5} \right] \tag{14}$$

The process of training a latent model at Level 1 begins with performing an exploratory walk through this composite space. In the first stage, each finger achieves its own goal randomly, with different duration and with no connection between these changes of goal. These data are gathered in another buffer and processed by the contrastive learning procedure to obtain a compact latent vector for the whole hand at Level 1, represented as z_t^{hand}.

3.5.2 Hierarchical Policy Execution Loop

Once the representations are set up, the goal-conditioned PPO policy at Level 1 is trained using a top-down control loop. The step-by-step process runs as follows:

1. **Master Goal Selection:** Just like the standard Composer, a master goal vector $g^{\text{Level 1}}$ is drawn randomly every 100 iterations as a one-hot vector. This represents a desired abstract configuration for the whole hand.

2. **Action Routing as Sub-Goals:** The policy at Level 1 takes the current hand representation z_t^{hand} and the master goal $g^{\text{Level 1}}$ as inputs. Instead of outputting raw joint forces, the actions chosen by the Level 1 policy serve directly as the target goals for the level below:

$$a_t^{\text{Level 1}} = \left[g^{\text{finger}_1}, g^{\text{finger}_2}, g^{\text{finger}_3}, g^{\text{finger}_4}, g^{\text{finger}_5} \right] \tag{15}$$

3. **Lower-Level Physical Execution:** The five individual Level 0 finger's policies receive these sub-goals from Level 1. They execute their own local policies to meet these targets, which moves the actual physical joints in the MuJoCo engine.

4. **Feedback and Reward Updates:** As the fingers move, they change the local latent vectors at Level 0. This instantly updates the concatenated state at Level 1, causing the overall latent vector at Level 1 z_t^{hand} to move. The Level 1 policy is then optimised using PPO to minimise the squared difference between the master hand goal and its current latent vector:

$$\mathscr{R}_t^{\text{Level 1}} = -\| g^{\text{Level 1}} - z_t^{\text{hand}} \|_2^2 \tag{16}$$

The Hi-Composer allows MIMo to learn simple, isolated movements at the lowest level, while the higher levels learn to corporate those movements into complex, coordinated actions. By doing so, we open a path to building complex architecture that can control complex spaces like the full body of MIMo.

4 Results

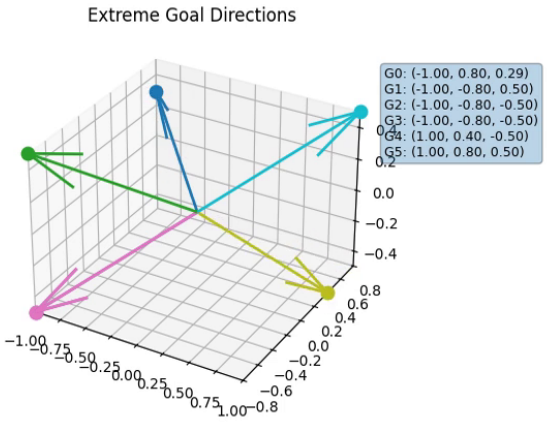
This section presents the empirical findings evaluated across several experimental environments. We begin with primitive geometric models like the cuboid one and progress incrementally to the high-dimensional control surfaces of the Multi Modal Infant Model (MIMo) platform.

4.1 Manifold Alignment in Primitive World Models

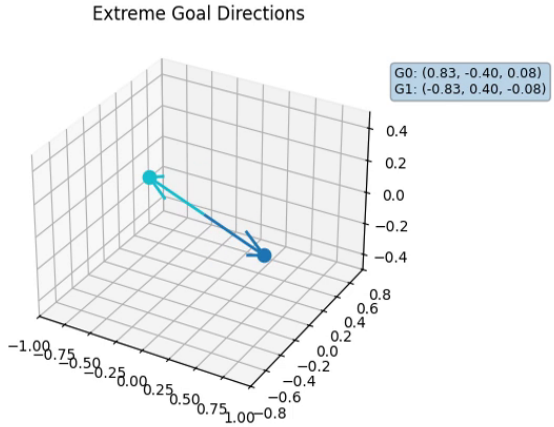
The initial check was that if the corner points in the latent space align with the extreme points in the original high-dimensional world. Testing was conducted within three world models: a cuboid, an ellipsoid, and a dumbbell environment, as shown in Figure 4.1. The latent network with a non-linearity like softmax outputs values on a standard simplex Δ^{d-1}, successfully mapping random trajectories into structured representations.

After training, the architecture finds the extreme points. For both the cuboid and the ellipsoid, the model naturally searches until it finds the absolute end of the diagonals as extreme points. When tried in the dumbbell environment, the learned extreme points settle at opposite ends of the two separate lobes. We can see how the reinforcement learning policy reacts during execution: every time the target goal switches, the policy guides the agent across the narrow bridge to get to the opposite lobe.

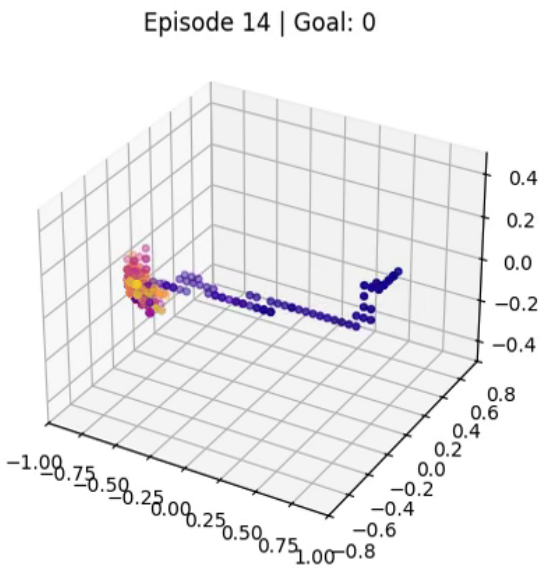
This ability to autonomously find and alternate between two extremes using goal conditioning has practical implications for more complex movements. When scaling this setup to the full-body MIMo platform, we can directly leverage this dual-extremum goal mapping to handle developmental milestones like the classic rolling task. Here, the policy can coordinate the entire body to smoothly transition back and forth between two highly stable terminal states: rolling from a prone position to a supine position, or vice versa. This transition elegantly demonstrates how primitive geometric representations can form the foundation for complex, real-world motor control.



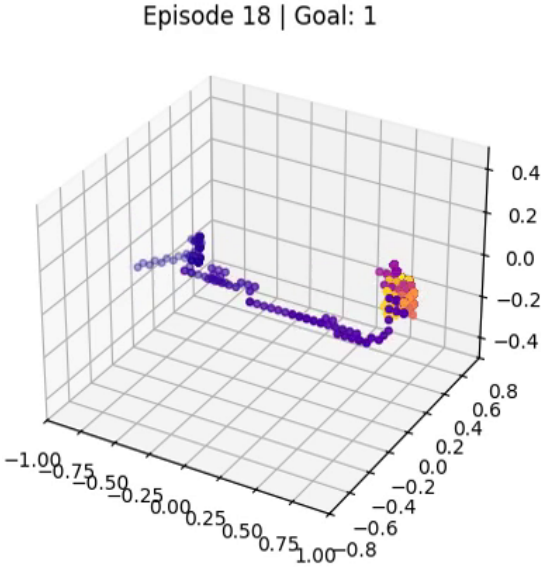
(a) Cuboid Latent Mapping Extremas



(b) Ellipsoid Latent Mapping Extremas (Latent dimension=2)



(c) Dumbbell Space Exploration with Goal 0



(d) Dumbbell Space Exploration with Goal 1

Figure 4.1: Mapping of Sensorimotor Extrema onto Latent Probability Simplices across Diverse Geometric World Models. The Composer architecture successfully groups maximum elongated diagonals and boundary extrema as steady extreme points on the learned latent manifold.

4.2 Emergent Extrema in Single-Finger Posture Control

Following geometric validation, the Composer architecture was applied to the motor control space of the MIMo model, starting with an isolated individual finger configuration. The joint actuation spaces of human-like fingers are highly coupled and nonlinear.

When trained through the multi-stage optimization loop, the system successfully mapped continuous physical configurations onto a low-dimensional latent line. The emergent extrema of the learned manifold mapped directly to the absolute physiological limits of the joint structure. The outer vertices of the latent space corresponded perfectly to the state of full finger extension (completely open) and full finger flexion (completely closed). This transition from a random initialization to a neatly bounded motor primitive demonstrates that the architecture can autonomously isolate core structural degree-of-freedom trajectories during play-like exploration.

4.3 Discovery of Basis Hand Configurations in Multi-Digit Spaces

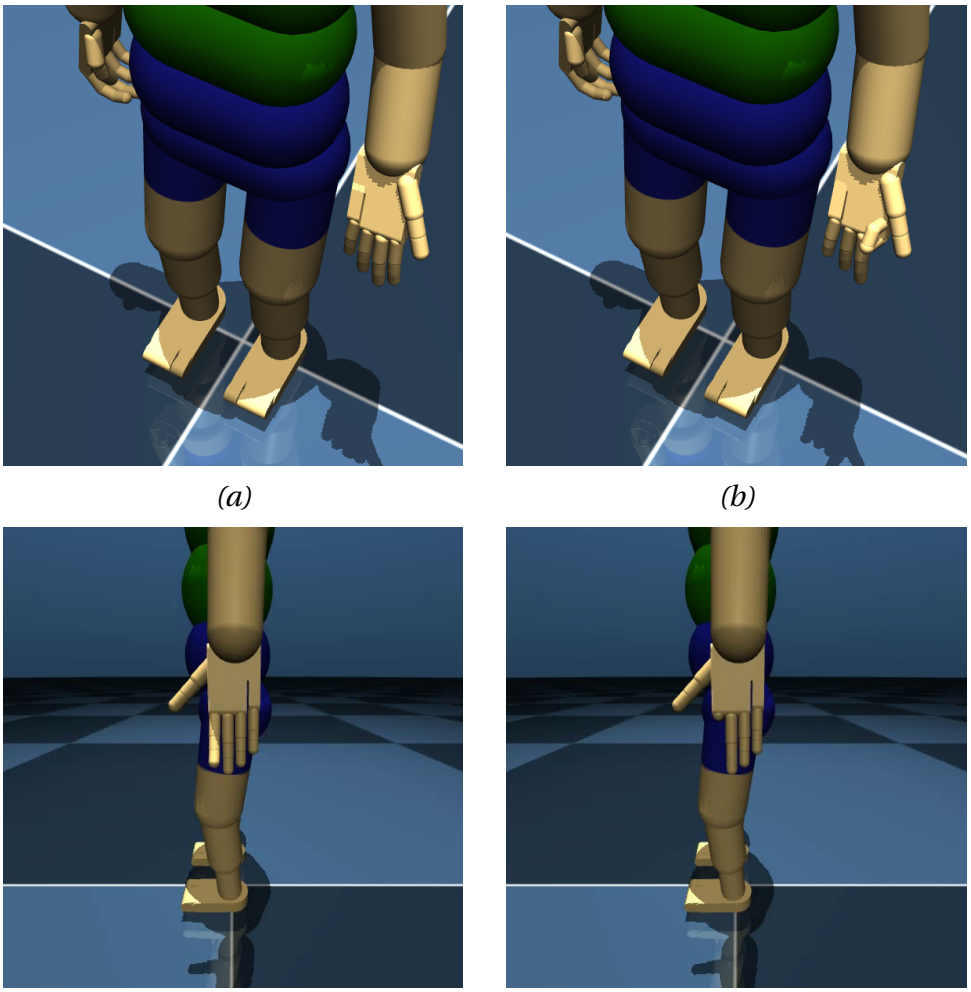
The architecture was then expanded to encompass the joint combinations of a complete 5-finger unified hand. When controlling all five digits simultaneously, the size of the continuous action space scales significantly, creating highly redundant coordinate surfaces.

To achieve clean coordination patterns within a lower latent space, a soft mechanical coupling factor was introduced by Salar between adjacent finger joint models. Under these structural settings, the Composer architecture discovers a basis set of hand configurations at the corners of simplex. With enough mechanical coupling between the fingers, some of the corner points became open and closed hand. This highlights the framework's ability to compress highly redundant joint arrays into meaningful macro-actions.

4.4 Hierarchical Fingertip Coordination via Hi-Composer

To scale control across complex, multi-level structural tasks, the hierarchical variant, Hi-Composer, was evaluated on a top-down hand control objective. Level 0 blocks were configured to independently manage the individual joint mechanics of each of the five fingers, while the higher Level 1 block observed the hand as a unified coordinate space by receiving the concatenated latent vectors of the lower layer.

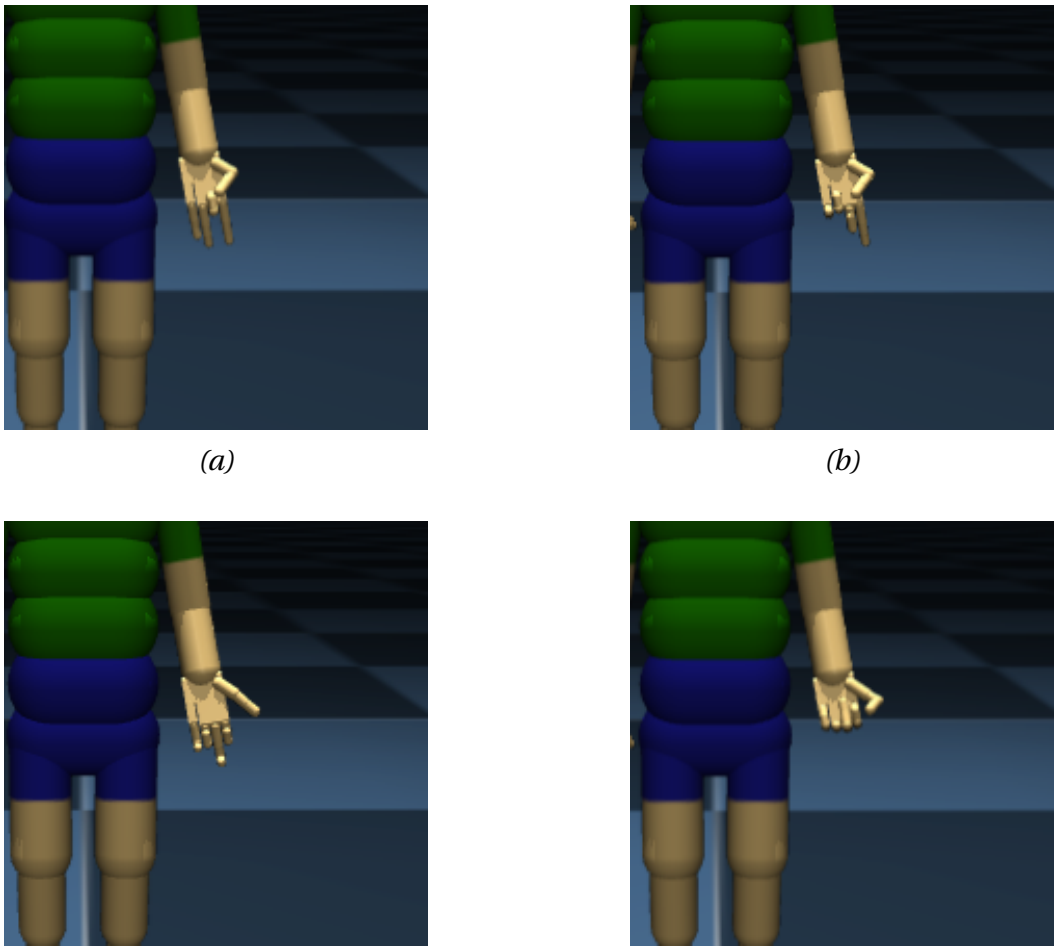
The execution trials demonstrate that the Hi-Composer successfully achieves abstract behavioral coordination across separate digits. Instead of forcing a single model



(a) *(b)*

(c) *(d)*

Figure 4.2: Emergent Manifold Extremes for a Single Actuated Digit viewed from multiple perspectives: (a, c) full finger extension (open configurations) and (b, d) full finger flexion (closed configurations). The trained goal-conditioned policy isolates full finger opening and full finger closing as the terminal boundary states of the low-dimensional latent space.



(a) *(b)*

(c) *(d)*

Figure 4.3: Mapping of the complete five-finger action space into four distinct hand postures at the outer edges of the learned latent space: (a) Configuration 1, (b) Configuration 2, (c) Configuration 3, and (d) Configuration 4.

to compute raw joint torques for dozens of independent channels, the Level 1 policy successfully routes composite action signals that serve as target sub-goals for the Level 0 controllers. This multi-level interaction allows the fingers to execute synchronized paths, mapping high-level goals into fluid, cooperative physical executions while maintaining separate representations at local levels.

4.5 Sensorimotor State-Space Coverage and the Thin Pancake Hypothesis

To evaluate how well the full framework explores a complex space, we tested the Composer architecture against multiple open-loop noise schemes on a full-body rollover task using the MIMo platform. The testing and the evaluation were done by Salar. In this setup, the infant model lies on the floor and can freely move its body using a continuous action space of 46 dimensions. We tracked a 98-dimensional vector containing joint positions and velocities. This allowed us to measure actual body movements and shapes, and also ignore how far the robot simply slid or rolled across the floor.

The experiment compared eight different conditions: the full Composer architecture ($C1$), three version variants with single components removed ($C2$: no contrastive learning; $C3$: no simplex normalization; $C4$: no one-hot goals), and four pure noise patterns ($C5$: white noise; $C6$: pink noise; $C7$: baby noise; $C8$: Ornstein-Uhlenbeck noise) [10]. Three geometric metrics were used for analysis: a greedy sampling method to estimate the total cloud [2], participation ratio to see how global variance spreads across axes, and an intrinsic dimension algorithm ($TwoNN$) [11]. The intrinsic dimension is calculated by taking two nearest neighbours of each point and then taking the ratio of those distances. Average of those ratios provide an idea for the intrinsic dimension of a dataset. The numerical results across all seeds and conditions are detailed in Table 1.

Table 1: Sensorimotor Space Coverage Metrics at 1,000,000 Environment Steps

ID	Exploration Strategy	Diameter	Participation Ratio	TwoNN Intrinsic Dim
$C1$	COMPOSER (Full)	326.86 ± 46.43	15.87 ± 0.81	27.28 ± 1.19
$C2$	COMPOSER (CLTT off)	300.32 ± 23.59	14.97 ± 0.87	29.27 ± 1.19
$C3$	COMPOSER (Simplex off)	354.16 ± 61.14	14.11 ± 0.81	31.08 ± 1.49
$C4$	COMPOSER (Goal off)	334.88 ± 36.96	14.58 ± 1.06	33.93 ± 0.37
$C5$	White Noise	139.85 ± 11.51	11.61 ± 0.04	30.99 ± 0.17
$C6$	Pink Noise	351.60 ± 18.07	19.95 ± 0.13	24.18 ± 0.20
$C7$	Baby Noise	344.31 ± 33.81	17.32 ± 0.07	28.67 ± 1.12
$C8$	Ornstein-Uhlenbeck (OU)	177.58 ± 25.54	14.58 ± 0.07	18.16 ± 0.18

The data strongly supports the “thin pancake” hypothesis. This idea suggests that an efficient explorer should not expand blindly into a massive, uniform ball. Instead, it should sweep out a wide area while keeping its movements focused on a lower-dimensional, highly organized path.

The Composer architecture had a big impact, doing much better than the standard method of using random white-noise actions in two important areas. It allowed the agent to move around a much larger area, with a total diameter of 326.86, compared to the small area of 139.85 achieved with white noise, which is a huge 134 percent increase in how much space the agent could explore. This difference was not just a coincidence, but a real and significant improvement, with every single test run using Composer reaching farther than every single test run using white noise. At the same time, Composer was able to make the agent's movements more efficient and organized, reducing the complexity of its movements to an intrinsic dimension of 27.28, which is about 4 units simpler than the messy and disorganized movements created by white noise.

The study also showed an interesting difference between how thick or thin the movements were in a small area, and how much they varied overall. Composer's movements were thin and structured in a small area, but when looking at the bigger picture, it moved around more than white noise. This makes sense when you look at how they behave: white noise moves around it's starting position, so its movements are all clustered together. So it's overall variance is low. Even though its small movements are messy and all over the place, they don't spread out very far. While this is the case for white noise, Composer moves its different body parts in a coordinated way, stretching out in specific directions. This spreads out its movements across the bigger picture, but keeps each small movement simple and straightforward.

Removing individual parts out of the architecture highlighted exactly what each component contributes. The internal goal mechanism proved to be the most critical piece for maintaining structure. When the policy receives random, uncorrelated goals each episode, its updates often pull in opposing directions. The movements average out, which fills up more dimensions unnecessarily. Turning off simplex normalisation increased the local dimension by nearly 4 units, while removing the contrastive learning updates increased it by 2 units. Interestingly, the overall diameter was highly robust to these changes. The version without simplex normalisation actually reached the largest diameter of all, because its unnormalized structure allowed for unbounded rewards. This was exploited by the policy to force extreme physical extensions.

What's interesting is that the baseline tests ended up with a surprising outcome - the open-loop pink noise actually did better than expected. It managed to create the

thinnest and widest pancake shape out of all the conditions that were tested. It even performed slightly better than the full Composer architecture in this particular setup. The pink noise was able to reach a maximum diameter of 351.60. It also kept the local dimension relatively low at 24.18. This suggests that the specific frequency spectrum of pink noise somehow naturally matches the physics of how our limbs move continuously. The motor system can integrate these smooth signals over short timescales, producing natural, sweeping oscillations without exceeding structural limits. This is a pretty significant finding, and it could have implications for how we think about movement and motor control.

The Composer architecture is able to figure out these coordinated paths all by itself through training, and it's really good at getting close to the optimal geometric properties without any help. This ability to learn on its own is very important because it gives us a starting point for more complicated tasks, like moving objects to some place, where you can't just rely on random background noise to get things just right - you need to be precise and in control.

5 Conclusion and Future Work

5.1 Conclusion

This thesis presented the design, implementation, and evaluation of the Composer Architecture which is a developmental machine learning framework modeled after the autonomous, open-ended learning behaviors seen in human infants. It is tested and used in MIMo, a mujoco based physics model. By combining unsupervised representation learning with goal-conditioned reinforcement learning, the framework successfully addresses the sample inefficiency and training instability challenges which are common in high-dimensional continuous control platforms like the MIMo itself.

The results we get from simple world models, like in the cuboid case, tell us that the extreme points in the simplex are mapped to extreme points in the world models also. When applied to the complex MIMo infant model, the architecture successfully isolated functional motor primitives, such as the full opening and closing of a single finger and a complete 5-finger hand. And through extensive benchmarking on the full-body rollover task, we validated the thin pancake hypothesis. The Composer architecture allows further exploration of the large dimensional space but also restricts itself to a small dimensional pancake like submanifold. These findings suggest that an internal, self-generated goal mechanism is essential for guiding open-ended exploration into low-dimensional spaces.

5.2 Future Work

There are still many areas where the Composer architecture can be improved. Even though it has already shown it can handle sensorimotor learning and exploring spaces really well, there's still a lot to discover and work on.

1. **Action-Aware Representation Learning:** Let's think about how we can improve the way we learn representations. Right now, we're using a method called Contrastive Learning Through Time (CLTT) to create a latent space based on how things change over time. But there's a new idea that's worth exploring: using Action Aware Self-Supervised Learning (AA-SSL) frameworks. By taking into account the actions the agent is taking, we might be able to create more organised and structured latent spaces. This could help us better understand how specific actions cause changes in the environment. For example, if we know what actions lead to a certain outcome, we can use that information to improve our representation learning. This approach has the potential to yield more informative and useful representations. It is an area worth investigating further.

2. **Scaling via Deep Hierarchies:** The initial validation of the Hi-Composer variant focused on coordinating local finger primitives to control a unified hand. We should be able to scale this up in future. The whole body should be treated as one level with a lot of other levels and parts, like each finger on the last level.

3. **Complex Developmental Tasks:** The framework should be tested on a wider range of computational cognition experiments. Key target environments include complex full-body coordination tasks like actively rolling over from a prone to a supine position, as well as the mobile kicking [12] [13], which is a classic paradigm in developmental psychology for studying infant learning. To really test how well the Composer architecture works, it should be used in standard MuJoCo robotics environments, not just the MIMo model made for infants. This means trying it out on classic robots with many joints, like robotic hands that can grasp things or four-legged robots learning to walk. By doing this, we can see if the framework can be used with different kinds of robots, both in industry and academia, and if it can be controlled effectively. This will help us understand if the Composer architecture is versatile and can be used in many different situations, making it a valuable tool for robotics.

4. **Alternative Algorithmic Pipelines and Goal Sampling:** Finally, future studies should look into goal-conditioned reinforcement learning algorithms other than Proximal Policy Optimization (PPO). Examples are off-policy maximum-entropy variants [14] or deterministic actor-critics [15]. Additionally, rather than relying strictly on random uniform goal sampling from the simplex vertices, implementing active goal-selection methods that are based on competence progress or learning gains significantly improve sample efficiency during open-ended play.

References

[1] D. Mattern, P. Schumacher, F. M. López, M. C. Raabe, M. R. Ernst, A. Aubret, and J. Triesch, "Mimo: A multimodal infant model for studying cognitive development," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 4, pp. 1291–1301, 2024.

[2] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical computer science*, vol. 38, pp. 293–306, 1985.

[3] M. Yordanova and V. V. Hafner, "Guided reinforcement learning for continuous sensorimotor learning in dynamic environments," in *IOP Conference Series: Materials Science and Engineering*, vol. 1343, no. 1. IOP Publishing, 2026, p. 012020.

[4] F. M. López, M. Lenz, M. G. Fedozzi, A. Aubret, and J. Triesch, "Mimo grows! simulating body and sensory development in a multimodal infant model," in *2025 IEEE International Conference on Development and Learning (ICDL)*. IEEE, 2025, pp. 1–6.

[5] J. Crabbé, Z. Qian, F. Imrie, and M. van der Schaar, "Explaining latent representations with a corpus of examples," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 154–12 166, 2021.

[6] F. Schneider, X. Xu, M. R. Ernst, Z. Yu, and J. Triesch, "Contrastive learning through time," in *Svrhm 2021 workshop@ neurips*, 2021.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[9] B. Eysenbach, T. Zhang, S. Levine, and R. Salakhutdinov, "Contrastive learning as goal-conditioned reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[10] O. Eberhard, J. Hollenstein, C. Pinneri, and G. Martius, "Pink noise is all you need: Colored noise exploration in deep reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[11] E. Facco, M. d'Errico, A. Rodriguez, and A. Laio, "Estimating the intrinsic dimension of datasets by a minimal neighborhood information," *Scientific Reports*, vol. 7, 09 2017.

[12] X. Xu and J. Triesch, "Infants and mobiles: Developing an understanding of cause and effect," *Developmental Science*, vol. 29, no. 4, p. e70211, 2026.

[13] ——, "From kicking to causality: Simulating infant agency detection with a robust intrinsic reward," 2025. [Online]. Available: https://arxiv.org/abs/2507.15106

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018. [Online]. Available: https://arxiv.org/abs/1801.01290

[15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.

[16] M. G. Fedozzi, F. Rea, G. Sandini, J. Triesch, and A. Sciutti, "Canalizing babbling: Development-inspired goal sampling for visuo-motor learning," in *2025 IEEE International Conference on Development and Learning (ICDL)*. IEEE, 2025, pp. 1–6.

[17] F. M. López, H. Kanazawa, O. Fiala, Y. Balashov, V. Marcel, L. Rustler, M. Lenz, D. Kim, Y. Kuniyoshi, J. Triesch *et al.*, "Simulating infant first-person sensorimotor experience via motion retargeting from babies to humanoids," *arXiv preprint arXiv:2604.27583*, 2026.

[18] F. M. López, B. E. Shi, and J. Triesch, "Efficient coding in active perception: A developmental perspective on autonomous control," 2026.

A Mathematical Properties of the Softmax Activation

Let $\mathbf{h} = [h_1, h_2, \ldots, h_d]^T \in \mathbb{R}^d$. The Softmax transformation $\sigma(\mathbf{h}) : \mathbb{R}^d \to \mathbb{R}^d$ is defined component-wise as:

$$\sigma(\mathbf{h})_i = z_i = \frac{\exp(h_i)}{\sum_{j=1}^{d} \exp(h_j)} \quad \forall i \in \{1, 2, \ldots, d\} \tag{17}$$

Since $\exp(h_i) > 0$, it follows that $z_i > 0$ and:

$$\sum_{i=1}^{d} z_i = \frac{\sum_{i=1}^{d} \exp(h_i)}{\sum_{j=1}^{d} \exp(h_j)} = 1 \tag{18}$$

Thus, $\mathbf{z}$ is strictly constrained to the $(d-1)$-dimensional standard geometric simplex:

$$\Delta^{d-1} = \left\{ \mathbf{z} \in \mathbb{R}^d : \sum_{i=1}^{d} z_i = 1,\ z_i \geq 0\ \forall i \right\} \tag{19}$$

B Properties of the Standard Simplex

Notation. Let $\vec{0}_k \in \mathbb{R}^k$ denote the all-zero vector, and $\vec{1}_k \in \mathbb{R}^k$ the all-one vector. When the dimension is clear, we omit the subscript.

The $(k-1)$-dimensional standard simplex is defined as

$$S_{k-1} = \left\{ \vec{x} \in \mathbb{R}^k : \sum_{i=1}^{k} x_i = 1,\ x_i \geq 0\ \forall i \right\}. \tag{20}$$

Euclidean norm. For any $\vec{x} \in S_{k-1}$,

$$0 < \|\vec{x}\|_2 \leq 1. \tag{21}$$

The strict positivity follows from $\|\vec{x}\|_2 \geq 0$ and the fact that $\vec{0}_k \notin S_{k-1}$.
To show the upper bound, note that

$$\|\vec{x}\|_2^2 = \sum_{i=1}^{k} x_i^2.$$

Since $x_i \geq 0$ and $\sum_{i=1}^{k} x_i = 1$, we have $x_i \in [0,1]$, implying $x_i^2 \leq x_i$. Therefore,

$$\|\vec{x}\|_2^2 = \sum_{i=1}^{k} x_i^2 \leq \sum_{i=1}^{k} x_i = 1,$$

and hence $\|\vec{x}\|_2 \leq 1$.

Equality condition. Equality $\|\vec{x}\|_2 = 1$ holds if and only if

$$\sum_{i=1}^{k} x_i^2 = \sum_{i=1}^{k} x_i,$$

which is equivalent to

$$\sum_{i=1}^{k} x_i(1 - x_i) = 0.$$

Since each term $x_i(1-x_i) \geq 0$, the sum is zero only if each term is zero. Hence for every i, either $x_i = 0$ or $x_i = 1$. Together with $\sum_{i=1}^{k} x_i = 1$, this implies that exactly one component equals 1 and all others are zero. These are precisely the vertices of the simplex. ■

Center point. Define the center of the simplex as

$$\vec{x}_c(k) = \frac{1}{k}\vec{1}_k \in S_{k-1}.$$

Its norm is

$$\|\vec{x}_c(k)\|_2 = \frac{\sqrt{k}}{k} = \frac{1}{\sqrt{k}} \xrightarrow[k\to\infty]{} 0. \tag{22}$$

Thus, in high dimensions, the center approaches the origin.
Moreover, $\vec{x}_c(k)$ is the closest point in the simplex to the origin. To see this, write any $\vec{x} \in S_{k-1}$ as

$$\vec{x} = \vec{x}_c(k) + \vec{\delta}, \quad \text{with } \sum_{i=1}^{k} \delta_i = 0.$$

Then

$$\vec{\delta} \cdot \vec{x}_c(k) = \frac{1}{k}\sum_{i=1}^{k} \delta_i = 0,$$

so $\vec{\delta}$ is orthogonal to $\vec{x}_c(k)$. By the Pythagorean theorem,

$$\|\vec{x}\|_2^2 = \|\vec{x}_c(k)\|_2^2 + \|\vec{\delta}\|_2^2 \geq \|\vec{x}_c(k)\|_2^2,$$

with equality if and only if $\vec{\delta} = \vec{0}$.

Remark. The norm $\|\vec{x}\|_2$ is smallest at the center and increases as $\vec{x}$ moves toward the vertices, reaching 1 at the corners. Hence, maximizing $\|\vec{x}\|_2$ (or $\|\vec{x}\|_2^2$) encourages points to move from the center toward the vertices of the simplex.

C Local Intrinsic Dimension Estimation via TwoNN

The Two-Nearest Neighbors (TwoNN) algorithm is a non-parametric statistical method designed to estimate the intrinsic dimension ($\mathscr{D}$) of a dataset. The fundamental advantage of this framework is its reliance on minimal neighborhood information, which allows it to isolate the dimensional properties of a local manifold independently of global density variations or complex topological curvature.

The core geometric assumption underlying the TwoNN algorithm is that within a sufficiently small neighborhood surrounding any given data point, the data points are drawn from a locally uniform distribution, meaning the local density remains approximately constant. Let $\mathscr{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ represent a dataset containing N independent samples embedded in an ambient coordinate space of dimension D. For each individual data point $\mathbf{x}_i$, we compute its Euclidean distances to its closest neighbors. We define:

- $r_{i,1}$: The Euclidean distance from $\mathbf{x}_i$ to its **first nearest neighbor**.
- $r_{i,2}$: The Euclidean distance from $\mathbf{x}_i$ to its **second nearest neighbor**.

Using these distances, we define the scaling ratio μ_i for each point in the dataset as the quotient of the second and first neighbor distances:

$$\mu_i = \frac{r_{i,2}}{r_{i,1}} \quad \text{where } \mu_i \in [1, \infty) \tag{23}$$

Assuming the local neighborhood can be modeled as a uniform distribution within a $\mathscr{D}$-dimensional hypersphere, the probability density function $P(\mu)$ for the independent scaling ratios follows a strict power-law relationship:

$$P(\mu) = \mathscr{D}\, \mu^{-(\mathscr{D}+1)} \tag{24}$$

Integrating this probability density function from the boundary condition 1 to a given threshold value t yields the theoretical cumulative distribution function (CDF):

$$F(t) = P(\mu < t) = \int_1^t \mathscr{D}\, \mu^{-(\mathscr{D}+1)} d\mu = 1 - t^{-\mathscr{D}} \tag{25}$$

To estimate the intrinsic dimension $\mathscr{D}$ from an empirical dataset, the computed ratios are sorted in ascending order such that $\mu_{(1)} \leq \mu_{(2)} \leq \cdots \leq \mu_{(N)}$. We map each sorted ratio to its empirical coordinate using a standard continuous empirical distribution

function with a continuity correction:

$$F_i = \frac{i - 0.5}{N} \tag{26}$$

By applying a logarithmic transformation to the theoretical cumulative distribution function, we can decouple the intrinsic dimension parameter $\mathscr{D}$ from the geometric ratio variable:

$$1 - F(t) = t^{-\mathscr{D}} \tag{27}$$

$$\log(1 - F(t)) = -\mathscr{D}\log(t) \tag{28}$$

$$-\log(1 - F_i) = \mathscr{D}\log(\mu_i) \tag{29}$$

Let $x_i = \log(\mu_i)$ and $y_i = -\log(1 - F_i)$. This expression describes a linear relationship that passes strictly through the origin ($y = \mathscr{D}x$). Consequently, the intrinsic dimension $\mathscr{D}$ can be analytically calculated by applying a linear least-squares regression constrained to a zero intercept