School of Engineering

Centre for Artificial Intelligence (CAI)

Bachelor's thesis Computer Sciences

Shaping Latent Space by Architecture for Improved World Model Planning

Authors		Daniel Bucher Björn Wagner
Main supervisor		Prof. Dr. Thilo Stadelmann
Secondary supervisor		Pascal Sager
Date		04.06.2026

Imprint

Project: BSc-Thesis
Title: Shaping Latent Space by Architecture for Improved World Model Planning
Authors: Björn Wagner & Daniel Bucher
Date: June 4, 2026
Keywords: Artificial Intelligence, World Models
Copyright: Zurich University of Applied Sciences

Study program:
Computer Sciences, Bachelor
Zurich University of Applied Sciences

Supervisor 1:
Prof. Dr. Thilo Stadelmann
Zurich University of Applied Sciences
Email: thilo.stadelmann@zhaw.ch
Web: Link

Supervisor 2:
Pascal Sager
Zurich University of Applied Sciences
Email: pascal.sager@zhaw.ch
Web: Link

Abstract

For robots to manipulate objects in the real world, they must be able to predict the consequences of their actions before acting. World models address this by letting agents plan through simulating action consequences in a learned latent space. However, the visual features from foundation models like DINOv2 are optimized for recognition rather than planning. Their latent geometry is entangled: Euclidean distance fails to reflect the physical cost of moving between states, forcing sampling-based planners to waste compute in poorly structured regions. This thesis investigates whether this metric mismatch is best resolved implicitly through an architectural bottleneck or explicitly through geometric loss constraints. Building on DINO-WM, we introduce a lightweight projection layer on top of a frozen DINOv2 encoder. We evaluate this architecture against two explicit regularizations, a local action-sensitivity constraint and a global temporal constraint, on the contact-rich Push-T manipulation task. Using a diagnostic suite of PCA topology analysis, linear probing and rollout stability tracking, we show that the unconstrained bottleneck alone restructures the fractured feature space into a navigable manifold. This structural improvement raises the planning success rate of a reproduced DINO-WM baseline. Both explicit geometric constraints appear redundant or counterproductive in this setting, failing to improve manifold structure or planner success. From a practical standpoint, this points to lightweight projection layers as a simple, effective addition worth considering in world model pipelines built on frozen vision features.

Zusammenfassung

Damit Roboter in der realen Welt Objekte manipulieren können, müssen sie in der Lage sein, die Konsequenzen ihrer Handlungen vorherzusehen, bevor sie agieren. Weltmodelle (World Models) adressieren dies, indem sie Agenten durch die Simulation von Handlungsfolgen in einem erlernten latenten Raum planen lassen. Die visuellen Merkmale von Foundation-Modellen wie DINOv2 sind jedoch primär für die Erkennung und nicht für die Planung optimiert. Ihre latente Geometrie ist verworren: Die euklidische Distanz spiegelt nicht den physischen Aufwand wider, der für den Wechsel zwischen Zuständen erforderlich ist, was stichprobenbasierte Planer (Sampling-based Planners) dazu zwingt, in schlecht strukturierten Regionen unnötig Rechenleistung zu verschwenden. Diese Arbeit untersucht, ob diese metrische Diskrepanz am besten implizit durch einen architektonischen Flaschenhals (Bottleneck) oder explizit durch geometrische Verlustfunktionen (Loss Constraints) gelöst wird. Aufbauend auf DINO-WM führen wir eine leichtgewichtige Projektionsschicht über einem eingefrorenen DINOv2-Encoder ein. Wir evaluieren diese Architektur im Vergleich zu zwei expliziten Regularisierungen, einer lokalen handlungssensitiven Beschränkung (Action-Sensitivity Constraint) und einer globalen zeitlichen Beschränkung (Temporal Constraint), in der kontaktreichen Push-T Umgebung. Mithilfe einer Diagnosesuite, bestehend aus PCA-Analyse, Linear Probing und der Überwachung der Rollout-Stabilität, zeigen wir, dass allein der unbeschränkte Flaschenhals den fragmentierten Merkmalsraum in eine navigierbare Mannigfaltigkeit (Manifold) umstrukturiert. Diese strukturelle Verbesserung erhöht die Planungserfolgsrate einer reproduzierten DINO-WM-Baseline. Beide expliziten geometrischen Beschränkungen erweisen sich in diesem Kontext als redundant oder kontraproduktiv, da sie weder die Struktur der Mannigfaltigkeit noch den Erfolg des Planers verbessern. Aus praktischer Sicht deutet dies darauf hin, dass leichtgewichtige Projektionsschichten eine einfache und effektive Ergänzung darstellen, die in Weltmodell-Pipelines auf Basis eingefrorener visueller Merkmale in Betracht gezogen werden sollte.

Acknowledgements

We would like to thank our supervisors, Thilo Stadelmann and Pascal Sager, for their guidance and support throughout this project. Their feedback and insights were invaluable in shaping the direction of this work.

We also thank the Centre for Artificial Intelligence (CAI) at the Zurich University of Applied Sciences (ZHAW) for providing access to the compute cluster used for all experiments.

We would like to express our gratitude to everyone who took the time to read through our drafts. Your fresh eyes and valuable suggestions were essential in polishing the final text.

Finally we want to thank our loved ones for their patience and care.

Contents

Abstract iii

Zusammenfassung iv

Acknowledgements v

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Own Contributions . . . 3
- 1.4 Organization of Thesis . . . 4

2 Foundations 5
- 2.1 Introduction to Foundational Topics . . . 5
  - 2.1.1 Self-Supervised Learning & Representation Learning . . . 5
  - 2.1.2 Vision Transformers and Self-Supervised Distillation . . . 5
  - 2.1.3 World Models . . . 6
  - 2.1.4 Joint-Embedding Predictive Architectures . . . 7
  - 2.1.5 Latent Geometry & Manifold Hypothesis . . . 7
  - 2.1.6 Planning in Latent Spaces: CEM & Gradient-Based Methods . . . 8
- 2.2 Related Work . . . 9
  - 2.2.1 Pre-Trained Vision Encoders: DINOv2 . . . 9
  - 2.2.2 Planning on Vision Foundation Models: DINO-WM . . . 9
  - 2.2.3 Latent Projection: JEPA vs. Autoencoders . . . 9
  - 2.2.4 Architectural Validation in Predictive World Models: JEPA-WMS 10
  - 2.2.5 Recent Advances in Geometrically-Informed Latent Spaces . . . 10
  - 2.2.6 Smoothness and Input Sensitivity for Stable Planning . . . 11
  - 2.2.7 Temporal Constraints for Latent Planning . . . 11
  - 2.2.8 Quantifying Planning Efficiency in Non-Reconstructive Models 12
  - 2.2.9 Latent Space Analysis . . . 12

3 Methods 14
- 3.1 Base and Extended Encoder Architecture . . . 14
  - 3.1.1 DINO-WM Baseline . . . 14
  - 3.1.2 Encoder Projection Network . . . 14
  - 3.1.3 Predictor Integration . . . 16
  - 3.1.4 Update and Normalization . . . 16
- 3.2 Geometric Latent Space Constraints . . . 17
  - 3.2.1 Local Geometry: Action-Input Sensitivity . . . 17
  - 3.2.2 Global Geometry: Temporal Distance Metric . . . 18

4 Experimental Setup 20

4.1 Baseline Comparisons and Experimental Environment 20
4.1.1 Evaluated Models and Baselines 20
4.1.2 The Push-T Environment 21
4.2 Procedures and Implementation 22
4.2.1 Training Procedure 22
4.2.2 Planning Procedure 22
4.2.3 Implementation 23
4.2.4 Optimizer 23
4.2.5 Tuning 24
4.3 Evaluation 24
4.3.1 Training 25
4.3.2 Diagnostic Decoder 26
4.3.3 Planning 26
4.3.4 Manifold Analysis 26
5 Results **29**
5.1 Overall Task Performance 29
5.2 Decoder Reconstruction 30
5.3 Latent Topology and Rollout Stability 31
5.3.1 Latent Topology 31
5.3.2 Linear Probing 31
5.3.3 Rollout Stability 35
5.4 Training Behavior 36
5.5 Regularization Effects 37
5.5.1 Effect of Local Geometric Regularization 37
5.5.2 Effect of Global Geometric Regularization 38
6 Conclusion and Outlook **41**
6.1 Discussion 41
6.1.1 Implicit Regularization via the Unconstrained Bottleneck 41
6.1.2 The Ineffectiveness of Explicit Local Constraints 43
6.1.3 Metric Inflation and the Redundancy of Global Regularization . 44
6.2 Review of Thesis Objectives 45
6.3 Limitations 46
6.4 Future Work 47
A Appendix **48**
A.1 Local Geometry Constraint: Action-Space Smoothness 48
A.1.1 Method 48
A.1.2 Results 49
A.1.3 Discussion 51

B MPC Rollout Stability **52**

C Code and Readme Links **53**

D Declaration of Generative AI **54**

List of Figures **55**

List of Tables **58**

List of Abbreviations **59**

Bibliography **60**

To every W&B plot we never quite understood,
and the ones we understood but chose to ignore.
And to Dani's cat, who never understood any of it either.

Chapter 1

Introduction

To establish the framework of this thesis, this introduction outlines the fundamental tension between visual representation and physical planning in robotic agents. The following sections define the underlying problem statement, formulate the guiding research questions, and preview our primary contributions and the structural organization of the thesis.

1.1 Motivation

For artificial intelligence (AI) and robotics to operate in the real world, agents must possess a fundamental understanding of how their environment works (LeCun 2022). Humans do this intuitively: as originally proposed by Craik (1967), the brain constructs "small-scale models" of reality to simulate the potential outcomes of decisions before taking action (Hamrick 2019). For example, we can imagine what will happen if we push a glass off a table without physically executing the action.

In modern AI, this is realized with World Models (Ha and Schmidhuber 2018). Rather than relying strictly on physical interaction within the real environment, systems equipped with world models can simulate and evaluate plans entirely inside of a learned latent space. By predicting the consequences of actions within this abstract space, an agent can select the optimal sequence of actions before interacting with the physical environment (Hafner, T. P. Lillicrap, et al. 2020).

Building on foundational work in representation learning (Bengio, Courville, and Vincent 2013), recent breakthroughs have significantly elevated the potential of these models. Vision foundation models, such as DINOv2 (Oquab et al. 2024), have demonstrated an excellent ability to extract meaningful features from raw images. Architectures like the Joint-Embedding Predictive Architecture (JEPA) (LeCun 2022), which build upon the foundational principle of predictability minimization (also referred to as PMAX) (Schmidhuber 1992), have emerged as state-of-the-art frameworks for learning world dynamics (Assran et al. 2023). Unlike traditional models that try to reconstruct every single pixel of a future frame, JEPAs operate entirely in the latent space.

However, a key challenge remains in adapting these internal representations for actual robotic planning. While models like DINOv2 extract highly detailed visual features, they are optimized for image classification rather than step-by-step decision making. They often fail to capture the precise cause-and-effect relationships necessary for manipulation (Ma et al. 2023). As a result, their internal spaces can be highly

distorted (Arvanitidis, Hansen, and Hauberg 2018). In these raw visual spaces, the mathematical distance between two points rarely reflects the actual physical effort required to move between them, meaning a simple physical action might look like an unpredictable jump to the model (Wang et al. 2026). Furthermore, pure visual encoders tend to get distracted by prominent but functionally irrelevant detail, such as background textures or lighting shifts, which act as confusing noise for a planner trying to navigate the environment (Toso et al. 2026).

When applying decoder-free architectures like JEPA to physical dynamics, balancing structural stability and capacity becomes essential. Furthermore, to preserve the true scale of physical movements, these models often omit normalization layers (Maes et al. 2026). However, without these structural guardrails, the model's predictions are highly susceptible to "drift" or numerical divergence when simulating long sequences of actions. This introduces a fundamental geodesic-metric mismatch: the Euclidean distance between two embeddings fails to represent the true reachability or "action cost" in the physical environment (Arvanitidis, Hansen, and Hauberg 2018).

To close this gap, practitioners must determine whether the underlying metric mismatch is best resolved organically via an additional unconstrained encoding layer or explicitly through mathematical loss constraints.

1.2 Problem Statement

When the latent manifold is distorted or structurally unaligned with action dynamics, derivative-free planning heuristics like the Cross-Entropy Method (CEM) (Rubinstein 1999) demand computationally expensive, brute-force evaluations (Chua et al. 2018). Planners operate inefficiently in rugged landscapes because they are forced to waste sampling resources exploring disjointed or uninterpretable regions where the raw visual representation provides an unreliable signal for the underlying physics.

To isolate the root drivers of latent space regularity and planning efficiency, this thesis investigates the interplay between architectural bottlenecks and explicit geometric regularizations. We introduce a custom, unnormalized intermediate processing layer built on top of a frozen DINOv2 foundation (Oquab et al. 2024) within a JEPA framework (LeCun 2022). Using this architecture as a baseline structural bottleneck, we then evaluate the isolated impact of two independent geometric regularizations: an action sensitivity smoothness constraint addressing local topology and a temporal geodesic triplet loss constraint addressing global topology. By decoupling the implicit benefits of the architectural projection layer from the explicit effects of the geometric constraints, this work aims to clarify how to structure a navigable and stable planning manifold with minimal complexity, evaluating these approaches within the continuous control dynamics of the Push-T manipulation task (Chi et al. 2025).

Specifically, this work seeks to answer the following research question:

How can a visual input encoded latent space be restructured into a navigable manifold for non-reconstructive world models in continuous control tasks, using an unconstrained lightweight bottleneck layer versus a geometrically constrained bottleneck layer?

To answer this, we focus on two specific sub-questions:

1. What is the individual impact of an unconstrained bottleneck versus a geometrically constrained bottleneck on the stability and topological regularity of the learned latent manifold?

2. To what extent do explicit geometric constraints improve the convergence and execution success of a sampling-based planner in contact-rich manipulation, compared to the unconstrained bottleneck?

1.3 Own Contributions

The primary contribution of this thesis is a comparative evaluation of structural and geometric methods for organizing non-reconstructive world model manifolds within a contact-rich manipulation setting. The specific contributions are as follows:

- **A Trainable Architectural Bottleneck:** The design and implementation of an intermediate projection layer that connects a frozen visual encoder (DINOv2) to a joint predictive planning architecture (JEPA). This bottleneck organically filters out irrelevant visual noise by compressing the data, establishing a cleaner internal space for the planner to navigate.

- **Explicit Geometric Constraints:** The implementation of two independent mathematical constraints, applied to the new architectural bottleneck, designed to actively shape the model's internal map: a local action-sensitivity Lipschitz constraint ensuring that small physical actions result in predictable, bounded changes in the predictions and a global temporal constraint that aligns mathematical distances with passing time.

- **Latent-Space Diagnostic Framework:** The implementation of an evaluation framework that adapts probing and decoding techniques from established literature to our architecture. This framework measures prediction stability, structural quality directly within the internal map by using lightweight diagnostic decoders instead of computationally expensive, pixel-level image generation.

1.4 Organization of Thesis

This thesis is organized into six chapters, each addressing a specific phase of the research and evaluation process:

Chapter 1: Introduction provides the motivation for the work, defines the problem of latent space distortion in world models, and introduces the research questions.

Chapter 2: Foundations establishes the theoretical background required to understand the research, covering the evolution of world models, the mechanics of the Joint Embedding Predictive Architecture, and the principles of manifold learning.

Chapter 3: Methods details the technical design and mathematical formulations of the proposed framework, focusing on the intermediate encoder bottleneck and the local and global geometric constraints.

Chapter 4: Experimental Setup outlines the empirical benchmarking environment, model configurations, implementation details, and the diagnostic evaluation framework.

Chapter 5: Results presents a comparative evaluation of the bottleneck architectures alongside the ablation studies, analyzing both task performance and the internal structure of the learned latent representations.

Chapter 6: Conclusion and Outlook summarizes the key findings regarding how architectural structures mitigate geometric mismatch, outlines the core contributions of this work, and suggests potential avenues for future research.

Chapter 2

Foundations

This chapter establishes the theoretical and empirical background driving modern predictive world models. The first section reviews the foundational machine learning paradigms required to understand decoder-free latent planning. The second section contextualizes this project within recent literature, examining the trade-offs between architectural design choices and explicit geometric regularizations.

2.1 Introduction to Foundational Topics

To evaluate how a learned latent landscape can be optimized for downstream planning, it is first necessary to examine the core paradigms that drive modern world models.

2.1.1 Self-Supervised Learning & Representation Learning

Self-Supervised Learning (SSL) has revolutionized the field of machine learning by bypassing the bottleneck of human-annotated datasets. By formulating pretext tasks that predict hidden parts of the data using visible parts, models can learn rich structures directly from raw, unlabeled data (He et al. 2022). The core objective of this representation learning is to compress high-dimensional, complex observations into a compact and structured Latent Space (Bengio, Courville, and Vincent 2013). This paradigm is largely responsible for the massive success of Large Language Models (LLMs), which learn representations simply through next-token prediction (Brown et al. 2020). However, applying SSL to high-dimensional visual domains (such as images and videos) requires different architectures than text (LeCun 2022), leading to the development of contrastive frameworks (T. Chen et al. 2020) and Self-Supervised Vision Transformers (Caron et al. 2021; Oquab et al. 2024).

2.1.2 Vision Transformers and Self-Supervised Distillation

To extract robust initial representations from high-dimensional environments, modern architectures rely on self-supervised Vision Transformers (ViTs). Unlike traditional convolutional networks, ViTs process images by dividing them into a sequence of fixed-size spatial patches, applying self-attention mechanisms to capture complex global and local relationships (Dosovitskiy et al. 2021). To train these architectures without human-annotated labels, they are often optimized using a knowledge distillation framework. This framework consists of a student network parameterized by weights θ (fed augmented or masked images) and a teacher network parameterized by weights θ' (fed unmasked images) (Caron et al. 2021). To provide a stable learning

target, the teacher is not updated via standard backpropagation. Instead, building upon the Mean Teacher paradigm (Tarvainen and Valpola 2017), its weights at training step t, denoted as θ'_t, are updated using an Exponential Moving Average (EMA) of the student's weights θ_t (as illustrated in Figure 2.1):

$$\theta'_t = \alpha\theta'_{t-1} + (1 - \alpha)\theta_t$$

where $\alpha \in [0, 1]$ is a momentum parameter that strictly controls how slowly the teacher updates.

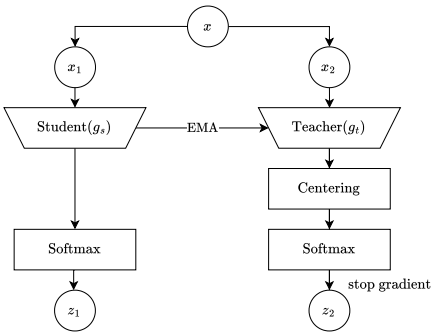

FIGURE 2.1: The self-distillation framework. The teacher network provides a stable learning target and is updated via an exponential moving average (EMA) of the student network's weights. Adapted from Caron et al. (Caron et al. 2021).

A primary challenge in this self-supervised paradigm is representation collapse, a failure mode where the network lazily outputs a constant vector for every image, ignoring the input entirely (Hadsell, Chopra, and LeCun 2006). While earlier methods relied on computationally expensive contrastive learning (using negative samples) (T. Chen et al. 2020) to force representations apart, architectures such as BYOL (Grill et al. 2020) demonstrated that the asymmetric EMA framework can independently prevent collapse in joint-embedding setups. Furthermore, empirical studies like MoCo v3 (X. Chen, Xie, and He 2021) validated that this momentum framework is crucial for stabilizing the training dynamics specifically for Vision Transformers. Alongside this architectural stability, modern distillation frameworks apply localized regularization methods to further refine the feature space while maintaining computational efficiency (Caron et al. 2021; LeCun 2022).

2.1.3 World Models

While Vision Transformers provide robust static features, denoted as $h_t = \text{enc}(x_t)$, planning in dynamic environments requires World Models to simulate temporal transitions (Ha and Schmidhuber 2018). World Models predict a future latent state $\hat{z}_{t+1}$ based on the current state z_t and an action a_t. Early world models employed generative architectures to predict future states by reconstructing them at the pixel level (Ha and Schmidhuber 2018; Hafner, T. P. Lillicrap, et al. 2020).

In contrast, modern approaches predict within a compressed, abstract latent space. Building on the historical foundations of early predictability minimization frameworks (Schmidhuber 1992) and heavily influenced by Yann LeCun's modern vision (LeCun 2022), these architectures discard the decoder. By doing so, the model focuses on learning the abstract mechanics relevant for planning and does not waste compute on task-irrelevant, visual details (such as a moving background).

2.1.4 Joint-Embedding Predictive Architectures

The realization of this decoder-free philosophy is the Joint-Embedding Predictive Architecture (JEPA) (Assran et al. 2023). A JEPA consists of three primary networks: an Online Encoder (student) E_θ, a Target Encoder (teacher) $E_{\bar{\theta}}$ and a Predictor P_ϕ (see Figure 2.2). The context encoder maps an observation x_1 into a stable latent representation $z_1 = E_\theta(x_1)$. Simultaneously, the target encoder maps a second observation x_2 to generate the ground-truth target $z_2 = E_{\bar{\theta}}(x_2)$. The predictor then uses the state z_1, conditioned on an input i, to forecast the target representation $\hat{z}_2 = P_\phi(z_1, i)$.

The network is trained by minimizing the objective loss $\mathcal{L}$ between this predicted state and the true target state, typically utilizing the squared Euclidean (L_2) distance:

$$\mathcal{L} = \|\hat{z}_2 - z_2\|_2^2$$

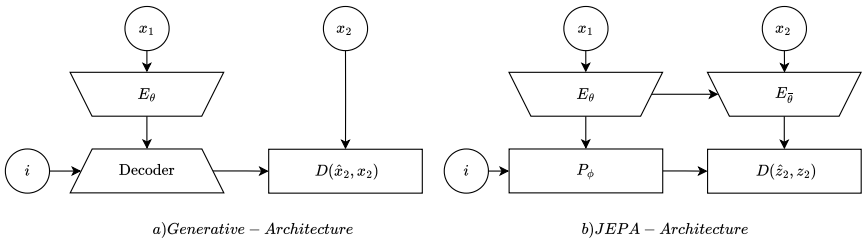

FIGURE 2.2: Comparison of world model architectures. Left (a): Generative architectures waste compute reconstructing pixel-level details via a decoder. Right (b): Joint-Embedding Predictive Architectures (JEPA) predict directly within an abstract latent space, bypassing the decoder. Adapted from Assran et al. (2023).

2.1.5 Latent Geometry & Manifold Hypothesis

Decision-making requires navigating the underlying latent geometry. Standard planners naively evaluate the distance between a current latent state z_A and a goal latent state z_B using the Euclidean (L_2) norm:

$$d_{L_2}(z_A, z_B) = \|z_A - z_B\|_2$$

However, this metric is highly misleading because physically valid states do not fill the latent space uniformly. Instead, they lie on a curved, lower-dimensional data manifold (Bengio, Courville, and Vincent 2013). Consequently, a straight Euclidean line frequently cuts through "empty" latent space (physically impossible states) or traverses straight through environmental obstacles. This phenomenon creates a geodesic-metric mismatch: in an unconstrained latent space, the mathematical distance between two embeddings fails to represent the true reachability or temporal "action cost" in the physical environment. For example, two states may share highly salient visual similarities, such as background illumination or static textures, causing them to sit close together in Euclidean space, while requiring a complex, long-horizon sequence of actions to transition between them physically. Therefore, the true transition cost is governed by the geodesic distance, the shortest path constrained entirely to the surface of this valid manifold (Arvanitidis, Hansen, and Hauberg 2018), seen in Figure 2.3.

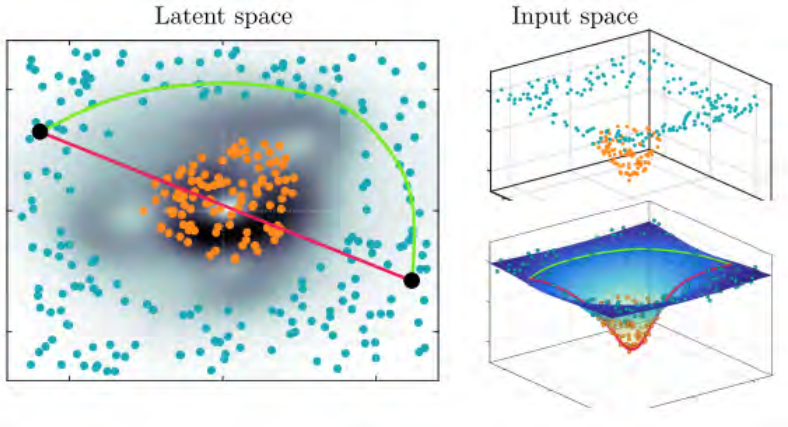

FIGURE 2.3: The Manifold Hypothesis in latent spaces. A straight Euclidean line (red) frequently cuts through "empty" latent space, corresponding to physically impossible states. In contrast, the true geodesic path (green) correctly follows the underlying geometric structure of the valid data manifold. Adapted from Arvanitidis et al. (Arvanitidis, Hansen, and Hauberg 2018).

2.1.6 Planning in Latent Spaces: CEM & Gradient-Based Methods

Given the transition $\hat{z}_{t+1} = P_\phi(z_t, a_t)$, planners seek an optimal action sequence to minimize the cost to a goal state z_g. To navigate the latent space, the Cross-Entropy Method (CEM) or Gradient-Based Planning are utilized.

CEM is a sampling-based optimizer (Rubinstein 1999; Chua et al. 2018). It calculates the performance of thousands of random action sequences by passing them through the world model. CEM then selects the top-performing sequences and refines them until they converge on an optimal path.

Alternatively, gradient-based planning uses the end-to-end differentiability of the JEPA architecture (Hafner, T. P. Lillicrap, et al. 2020). By backpropagating the transition cost directly through the frozen world model, it computes gradients to analytically optimize the action sequence.

The success of both planners is strictly bound by the latent geometry (Arvanitidis, Hansen, and Hauberg 2018). CEM will waste massive compute sampling physically impossible transitions in "empty" latent space, while gradient-based planners will receive deceptive L_2 gradients, causing them to collapse into local minima.

CEM can be deployed in two modes. In open-loop execution, the planner computes a full action sequence and executes it without replanning. In Model Predictive Control (MPC) (Chua et al. 2018), only the first action is executed before replanning from the new observation, making it more robust to prediction errors at the cost of additional computation.

2.2 Related Work

Building on the theoretical foundations established above, this section reviews the most relevant prior work this thesis builds upon and situates our contributions within the current landscape of world model research.

2.2.1 Pre-Trained Vision Encoders: DINOv2

Building upon the self-supervised distillation paradigms detailed in Section 2.1.2, this work utilizes DINOv2 as the foundational feature extractor (Oquab et al. 2024). DINOv2 is a state-of-the-art, self-supervised Vision Transformer specifically designed to produce highly robust, general-purpose visual features.

The primary advantage of DINOv2 within the context of world modeling is its out-of-the-box semantic richness. Because it was trained on a massive and curated dataset using the stable Teacher-Student EMA framework (bypassing the need for contrastive negative samples), it naturally captures deep object-level and spatial relationships. Building on the patch-based processing introduced in Section 2.1.2, DINOv2 maps each spatial patch of the raw input image to a high-dimensional feature vector, explicitly forming a sequence of dense "patch embeddings." By representing the environment as a sequence of these dense, localized embeddings, the frozen DINOv2 encoder can be seamlessly integrated into our pipeline, simplifying the processing of high-dimensional images so the subsequent architecture can focus entirely on learning the abstract temporal dynamics of the environment.

2.2.2 Planning on Vision Foundation Models: DINO-WM

While Section 2.2.1 detailed the underlying mechanics of the DINOv2 vision encoder, its specific application to dynamic control was recently formalized by Zhou et al. (2025) in the DINO World Model (DINO-WM) paper. By utilizing frozen DINOv2 features as the foundational representation for a JEPA-style dynamics model (detailed in Section 2.1.4), DINO-WM demonstrates that agents can successfully bypass pixel-level reconstruction and plan entirely within an abstract embedding space.

However, a critical limitation of the vanilla DINO-WM architecture is its reliance on visual representations optimized strictly for information retention rather than predictive planning. As established by Arvanitidis, Hansen, and Hauberg (2018), planning in such unconstrained spaces is computationally inefficient because the planner must navigate unpredictable latent dynamics.

This highlights a fundamental gap: To achieve efficient planning, the latent space must be explicitly structured and matched to the predictive dynamics. This limitation motivates the introduction of a dedicated, trainable latent encoder layer augmented with geometric constraints (detailed in Section 3.1.2), ensuring the manifold is explicitly optimized for action-conditioned prediction.

2.2.3 Latent Projection: JEPA vs. Autoencoders

Once the foundational visual features are extracted via DINOv2, a mechanism is needed to compress and project these features into a navigable latent space. Historically, world models utilized Autoencoder (AE) architectures, training a decoder to reconstruct future states at the pixel level (Ha and Schmidhuber 2018).

However, recent literature strongly advocates against reconstructive objectives for continuous control tasks. As argued by LeCun (2022), Autoencoders force the model to allocate significant representational capacity to planning-irrelevant details. By contrast, Joint-Embedding Predictive Architectures (JEPA) discard the decoder entirely, optimizing purely for the predictability of latent variables (Assran et al. 2023).

This architectural shift is empirically supported by ablation studies in the original DINO-WM literature (Zhou et al. 2025), which demonstrate that models trained via decoder-propagated reconstruction loss fall short of those utilizing pure latent-space prediction. Consequently, to bridge the metric mismatch, this work adopts a dedicated JEPA projection layer for the predictive bottleneck.

2.2.4 Architectural Validation in Predictive World Models: JEPA-WMS

To fully leverage the trainable encoder layer introduced above, the selection of the underlying optimization heuristics is critical. The foundational choices for this architecture are heavily informed by the recent comprehensive analysis conducted by Terver, Yang, et al. (2025), who systematically investigated the drivers of success in Joint-Embedding Predictive World Models (JEPA-WMs).

The JEPA-WM framework demonstrated that the Cross-Entropy Method (CEM) significantly outperforms gradient-based planners (GD) when navigating through non-reconstructive latent spaces (Terver, Yang, et al. 2025). Gradient-based approaches frequently fail due to the non-convex nature of the manifold, trapping the planner in local minima (Chua et al. 2018; Nagabandi et al. 2018). Furthermore, their empirical evaluations validated the superiority of specific vision foundation models, confirming that architectures leveraging pre-trained features (such as DINOv2) provide a more stable basis for planning than training visual backbones from scratch.

2.2.5 Recent Advances in Geometrically-Informed Latent Spaces

The recognized necessity of structuring these latent spaces using JEPA-based architectures has further been explored in recent publications.

For instance, Maes et al. (2026) introduced LeWorldModel (LeWM), establishing a stable, end-to-end predictive architecture directly from pixels, while Nam et al. (2026) proposed Causal-JEPA, which focuses on learning world models through object-level latent interventions to capture underlying physical causalities. Concurrently, approaches such as GeoWorld (Zeyu Zhang et al. 2026) utilize non-Euclidean manifolds to align latent topology with physical constraints, while alternative approaches leverage frozen foundation models to inherit rich, pre-trained spatial features without requiring explicit visual reconstruction (Huang et al. 2025; Toso et al. 2026).

However, a fundamental structural challenge persists across these architectures. As demonstrated by Li et al. (2026), a latent world model can achieve high predictive capability while remaining poorly aligned with downstream planning due to a spatial and temporal mismatch where Euclidean proximity fails to accurately reflect physical reachability. While existing frameworks attempt to resolve this mismatch using computationally heavy end-to-end re-training or multi-horizon supporting structures, they leave a notable gap regarding the viability of simpler alternatives. By explicitly decoupling an unconstrained structural bottleneck layer from isolated

local and global geometric constraints, this thesis directly tests whether complex loss engineering is strictly necessary for efficient, sampling-based trajectory optimization.

2.2.6 Smoothness and Input Sensitivity for Stable Planning

As established in Section 2.1.6, derivative-free planners like the Cross-Entropy Method (CEM) require a locally smooth latent landscape to converge efficiently (Chua et al. 2018). We previously noted that, if the latent manifold is highly non-linear or "rugged," a minor deviation in the planned action can result in an unpredictably large jump in the predicted future state, rendering the sampling heuristic prone to failure (Arvanitidis, Hansen, and Hauberg 2018).

To mitigate this instability, recent literature emphasizes the importance of enforcing localized Lipschitz-like continuity on transition models (Asadi, Misra, and Littman 2018). Within modern self-supervised learning, representational stability is typically achieved through architectural mechanisms like Exponential Moving Average (EMA) targets (Tarvainen and Valpola 2017) or explicit variance regularization (VICReg) (Bardes, Ponce, and LeCun 2022). However, these approaches primarily regularize the static state representation, rather than the active, action-conditioned transition dynamics.

This presents a gap for decoder-free architectures operating in contact-rich environments. Standard continuity constraints, such as global Lipschitz bounds (Asadi, Misra, and Littman 2018), assume that a large action magnitude must always correspond to a proportionally large latent shift. In practice, this global scaling fails to capture the nonlinear dynamics of physical control. For instance, a small action might initiate a massive environmental change, whereas a large action might result in minimal change. Consequently, there is a need for a regularization strategy that abandons global action correlation and instead explicitly bounds the predictor's sensitivity to localized action noise. By ensuring that small perturbations in the action space produce predictably bounded shifts in the latent representation, the model provides the exact local stability required by the CEM planner to safely sample and evaluate trajectories.

2.2.7 Temporal Constraints for Latent Planning

Another identified mismatch is between visual similarity and true physical distance over long planning horizons. Modern machine learning models use time-based rules to organize the latent space. In frameworks that do not rebuild images piece-by-piece, a "triplet" setup can be used to stretch and shape the internal landscape based on how far apart events occur in time (Myers et al. 2024; Wang et al. 2026).

This is done by picking three specific states from the data: an "anchor" state, a "positive" state that happens immediately next, and a "negative" state that occurs much later. The training setup penalizes the model if it accidentally places the distant negative state closer to the anchor than the immediate positive state. This process essentially straightens out the timeline within the model's internal map. It guarantees that as two states get further apart in actual time, their straight-line (L_2) distance inside the model grows larger (Wang et al. 2026). By making coordinate distance mirror true physical effort, CEM can accurately use simple distance calculations to estimate how many steps it will take to reach a target goal.

2.2.8 Quantifying Planning Efficiency in Non-Reconstructive Models

To objectively measure whether the proposed projection layer and geometric constraints successfully structure the latent space, we must define how planning efficiency is evaluated in modern, decoder-free architectures. The field offers several diverging directions for quantifying success.

The most direct interpretation of efficiency focuses on computational speed and resource minimization. For instance, Zhiwei Zhang et al. (2025) measure the explicit planning time per step, reporting significant speedups compared to reconstructive baselines. Another approach, by Nam et al. (2026), measures efficiency by the reduction in latent input size, demonstrating successful planning using only a fraction of the features required by standard patch-based architectures.

However, computational speed is irrelevant if the planner fails to reach the physical goal. That is why efficiency is ultimately weighted by task success. Success rates over fixed temporal horizons remain the standard benchmark (Terver, Yang, et al. 2025; Terver, Balestriero, et al. 2026; Zhiwei Zhang et al. 2025), with the foundational DINO-WM explicitly measuring success across a strict 25-step horizon (Zhou et al. 2025). Furthermore, recent work emphasizes the stability of these success rates across extended planning rollouts, evaluating how quickly the latent manifold degrades over time (Zeyu Zhang et al. 2026; Terver, Yang, et al. 2025). These success metrics are frequently supplemented with trajectory analysis, such as success weighted by path length and absolute trajectory error (Zhiwei Zhang et al. 2025) or by measuring the reduction in environment interactions required to achieve convergence (Sobal et al. 2025).

2.2.9 Latent Space Analysis

In decoder-free architectures like JEPA, evaluating the quality of the learned representations presents a unique challenge, as the model no longer produces human-interpretable pixel reconstructions (LeCun 2022). To objectively validate the structural integrity, geometric organization, and information capacity of these models without relying solely on downstream task performance, modern representation learning relies on a specialized suite of latent space analysis methods.

Principal Component Analysis (PCA) and Macro-Topology While neural networks project data into highly complex, non-linear manifolds, Principal Component Analysis (PCA) remains a state-of-the-art diagnostic tool for interpreting the global macro-structure of these abstract spaces. Recent literature in both representation learning and world models utilizes PCA to project high-dimensional latent vectors into an interpretable, lower-dimensional basis, exposing the true intrinsic dimensionality of the embeddings. By analyzing the variance explained across principal components, researchers can compute the *participation ratio* (effective dimensionality) of the latent space. This is a critical metric: a high participation ratio indicates a robust, well-distributed feature space, whereas a disproportionately low ratio is a primary indicator of representation collapse, where the network degenerates to utilizing only a few active dimensions (Grill et al. 2020; Caron et al. 2021).

Furthermore, by projecting and color-coding these latent states by their true physical environment coordinates, practitioners can visually and mathematically verify

whether the abstract embeddings have organically aligned into a continuous, navigable coordinate system. This explicitly proves whether the architecture successfully resolves the geodesic-metric mismatch (Arvanitidis, Hansen, and Hauberg 2018).

Diagnostic Probing and Decoders Because predictive world models intentionally sacrifice visual decoding to maximize planning efficiency, it is crucial to prove that the compressed latent bottleneck has not discarded critical environmental dynamics. A state-of-the-art approach to verify this without corrupting the main architecture is the use of post-hoc diagnostic decoders or linear probes (Assran et al. 2023). Both methods are trained entirely independently on the frozen latent representations, with stop-gradients ensuring no loss leaks back into the world model.

These two tools serve complementary diagnostic purposes. Visual decoders evaluate total information capacity. If a non-linear decoder can successfully reconstruct the physical scene, it proves the representation learning process retained the necessary spatial information. Conversely, linear probes evaluate geometric accessibility (Alain and Bengio 2016). By restricting the probe to a single linear layer, researchers can test whether specific environmental variables (such as spatial coordinates, object orientations, or applied actions) are exposed in the manifold. If a linear probe successfully extracts these variables, it indicates that the latent space is untangled and structured in a way that downstream planners like CEM can efficiently exploit.

Temporal Rollout and Sensitivity Analysis Finally, the practical utility of a world model is strictly bound by its temporal and localized stability. Topological consistency is typically evaluated by measuring latent drift over time, specifically by tracking the Euclidean (L_2) error between predicted latent states and true encoded states across an extended execution horizon (Zeyu Zhang et al. 2026; Terver, Yang, et al. 2025). This assesses rollout stability, confirming whether the internal manifold supports the stable, long-horizon "imagination" required by sampling-based planners like the Cross-Entropy Method (CEM). Coupled with localized sensitivity metrics which measure how bounded the latent space is against small action perturbations (Asadi, Misra, and Littman 2018) these combined techniques form a comprehensive, state-of-the-art evaluation framework for non-reconstructive world models.

Chapter 3

Methods

This chapter presents the architectural framework and mathematical formulations designed to restructure the latent space for trajectory planning. We outline the transition from a non-reconstructive baseline model toward our Extended Encoder Architecture, detailing the mechanics of the lightweight intermediate bottleneck layer. Finally, we define the explicit local and global geometric constraints implemented to actively shape the topology of the underlying manifold.

3.1 Base and Extended Encoder Architecture

To establish our architecture, this section first outlines the foundational DINO-WM baseline and subsequently introduces our Extended Encoder Architecture, which is designed to compress and restructure the latent space.

3.1.1 DINO-WM Baseline

The implementation for our baseline follows the architecture described by Zhou et al. (Zhou et al. 2025). This model utilizes a frozen DINOv2 encoder (Oquab et al. 2024) to extract high-level semantic features, which are then integrated with Proprioceptive and Action Encoder modules. A core design choice maintained in this baseline is the omission of visual reconstruction; since planning efficiency (detailed formally in Section 4.3.3) is the primary objective, the model avoids the computational burden of pixel-level decoding, focusing instead on transitions within the latent embedding space.

3.1.2 Encoder Projection Network

The first contribution of this work is the integration of a second encoder (Figure 3.1) following the JEPA architecture. As mentioned in Section 2.2.3, we favor the JEPA implementation since a reconstruction based Auto-Encoder does not inherently benefit abstract trajectory planning (LeCun 2022). This encoder gets the encoded patch embeddings (h_t) from the DINOv2 encoder as input and passes on the embeddings (z_t) to the predictor. It serves as an implicit regularizing encoder layer, as well as a base where we can add our explicit constraints, described in Section 3.2.

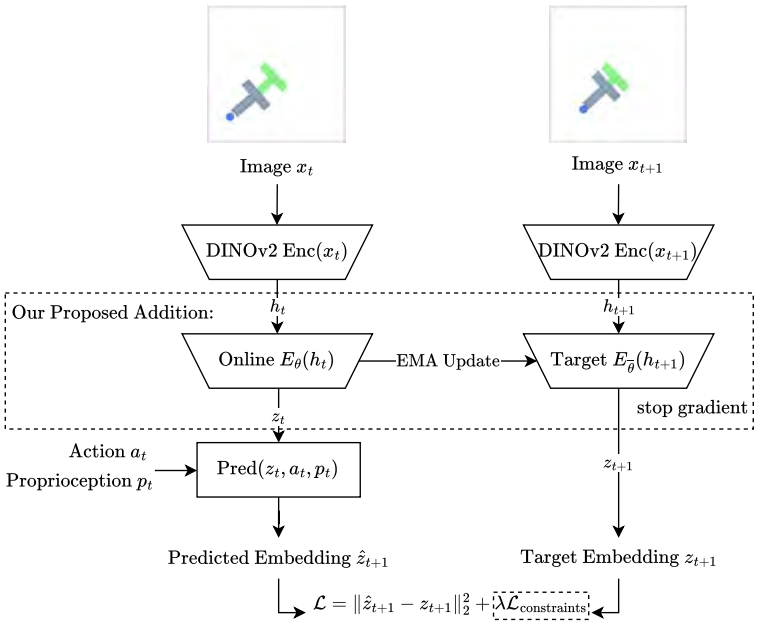

FIGURE 3.1: Architecture of the proposed JEPA-augmented DINO-WM. Images x_t and x_{t+1} are processed by frozen DINOv2 encoders to extract high-level features h_t and h_{t+1}. These features are passed into our proposed addition (dashed box), where the Online Encoder (E_θ) and Target Encoder ($E_{\bar{\theta}}$) map them to a latent space, yielding z_t and target embedding z_{t+1}. To prevent representation collapse without requiring negative samples, the $E_{\bar{\theta}}$ is updated via an Exponential Moving Average (EMA) of the E_θ weights, and a stop gradient is applied to its output. The predictor module takes the online embedding z_t, conditioned on the action a_t and proprioception p_t, to output the predicted embedding $\hat{z}_{t+1}$. The network is optimized by minimizing the Euclidean distance between the predicted and target embeddings, alongside our proposed constraints $\lambda\mathcal{L}_{\text{constraints}}$.

As illustrated in Figure 3.1, our architecture introduces a dual-encoder network as an intermediate processing layer between the frozen DINOv2 visual features and the predictor dynamics loop. The online and target encoders are implemented as identical 4-layer projection networks (including one input layer, two hidden layers, and one output layer), utilizing Gaussian Error Linear Unit (GELU) activations (Hendrycks and Gimpel 2016). GELU was selected to maintain architectural consistency with the Vision Transformers (ViTs) (Dosovitskiy et al. 2021) employed in the DINO-WM baseline.

This specific depth configuration was evaluated via a hyperparameter search, detailed in Section 4.2.5, and supported with the decoder findings in Section 5.2. Beyond optimization stability, this non-linear projection bottleneck was chosen to reduce the features into a smaller latent space (z_t), providing a lower-dimensional bottleneck for testing our geometric regularizations. This Online Encoder maps the intermediate representations (h_t, 14×14 patch-embeddings) extracted by the frozen DINOv2 Encoder from the current image x_t into this stable manifold, reducing the dimensionality by a factor of 2 per layer and allowing the network to restructure the latent landscape to optimize downstream planning dynamics.

3.1.3 Predictor Integration

To transition from our custom geometric manifold back into the standard dynamics loop, this compressed latent state z_t is fed directly into the baseline predictor network adopted from Dino-WM. The predictor takes this geometric embedding z_t and handles action conditioning by concatenating it with the corresponding action vector a_t and the proprioceptive state vector p_t and outputs the prediction z_{t+1}. Because the original Dino-WM predictor expects unconstrained visual features, inserting our trainable JEPA projection layer allows us to fundamentally reshape the geometry of the inputs z_t before the predictor executes its forward rollout. This ensures that, while the forward dynamics mechanics remain identical to the baseline, the underlying space through which it plans is strictly governed by our added encoder and constraints.

3.1.4 Update and Normalization

To provide a stable learning signal for the predictor, the target encoder $E_{\bar{\theta}}$ generates the target z_{t+1} from the future image x_{t+1}. This pathway is blocked by a stop-gradient operation: $E_{\bar{\theta}}$ receives no gradient updates via backpropagation and is instead updated as an Exponential Moving Average (EMA) of the online encoder's weights, adhering to stable, self-distillation training paradigms (Grill et al. 2020; Assran et al. 2023; Morales-Brotons, Vogels, and Hendrikx 2024).

As introduced in Section 2.1.2, self-supervised architectures often require explicit variance regularization, such as VICReg, to prevent representation collapse. However, we intentionally omit such variance-forcing loss terms from our objective. Because our encoder layers operate on top of semantically rich, pre-trained features from DINOv2, preliminary experiments indicated that applying standard variance regularization artificially inflated the latent distribution. This over-regularization disrupted the existing semantic structure, destabilizing the encoder-predictor dynamics and severely hindering convergence. Instead, we empirically observed that the asymmetric EMA update framework was fully sufficient to prevent representational collapse on its own.

Furthermore, explicit normalization layers (such as Batch Normalization or Layer Normalization) are intentionally omitted from the bottleneck layers. This omission is mathematically important for preserving the unconstrained latent feature magnitudes required by our additional geometric constraints. Enforcing a target-driven normalization operation inside the projection layers uniformly scales intermediate token states, which artificially flattens absolute geometric variations (Bardes, Ponce, and LeCun 2022).

Because our global temporal constraint (Section 2.2.7) relies on a triplet hinge loss to explicitly maximize the Euclidean distance between temporally distant states based on a fixed margin γ, preserving absolute metric scale is essential. While foundational Deep Metric Learning (DML) frameworks frequently utilize global L_2 vector normalization to stabilize the triplet loss domain (Schroff, Kalenichenko, and Philbin 2015; Hermans, Beyer, and Leibe 2017), activation-dependent batch standardization is omitted here to prevent mini-batch statistics from dynamically scaling or collapsing these absolute coordinate spaces.

Similarly, our action-space perturbation loss (Section 3.2.1) relies on measuring a deterministic, Lipschitz-bounded latent shift in response to small, localized action noise injections (δ) (Asadi, Misra, and Littman 2018). Incorporating activation-based normalization would dynamically shift gradient magnitudes based on fluctuating batch or spatial token statistics. In continuous control environments, this introduces a stochastic, step-to-step variation that breaks down the steady causal mapping required by trajectory optimization frameworks that plan via sampling-based heuristics, like the Cross-Entropy Method (CEM) (Chua et al. 2018; Hafner, T. Lillicrap, et al. 2019).

3.2 Geometric Latent Space Constraints

Standard self-supervised objectives prioritize feature predictability but do not inherently structure the geometry of the resulting manifold. To ensure the latent space is navigable for CEM, we introduce explicit regularizations targeting both the local smoothness and global temporal structure of the latent space.

3.2.1 Local Geometry: Action-Input Sensitivity

Building upon the theoretical need for Lipschitz-like continuity established in Section 2.2.6, our objective is to explicitly constrain the latent predictor to ensure local smoothness. A natural first approach is to bound the latent displacement by the magnitude of the applied action, under the assumption that larger commanded movements produce proportionally larger visual changes. However, this assumption did not have the desired effects on our predictions. We explore this formulation further in Appendix A.

Instead, we measure the sensitivity of the world model's prediction function directly with respect to its action input. A small, random perturbation $\delta \sim \mathcal{N}(0, \sigma^2 I)$ is applied to the actions. We then re-encode, predict and measure how much the predicted latent shifts. This enforces a local Lipschitz condition on the predictor as a function of its inputs:

$$\mathcal{L}_{\text{perturb}} = \frac{1}{B} \sum_{b=1}^{B} \max \left(0, \frac{\frac{1}{P} \sum_{p=1}^{P} \| \hat{z}_{b,p}^{\text{pert}} - \hat{z}_{b,p} \|_2}{\| \delta_b \|_2 + \varepsilon} - C \right)$$

Where:

- $\hat{z}_{b,p}$: The predicted latent for patch p of sample b under original history actions.
- $\hat{z}_{b,p}^{\text{pert}}$: The predicted latent for patch p of sample b when history actions are perturbed by δ_b.
- $\delta_b \sim \mathcal{N}(0, \sigma^2 I)$: Gaussian noise applied to the history actions of sample b.
- P: The total number of visual patches in the latent grid ($14 \times 14 = 196$).
- σ: The perturbation scale controlling the neighborhood size.
- ε, C: Numerical stability constant and margin threshold.

Because the perturbation δ is not a quantity from the environment, this formulation measures the sensitivity of the predictor to its action input directly, without depending on whether the pusher is currently in contact with the T-block (see the Push-T task in Figure 4.1).

3.2.2 Global Geometry: Temporal Distance Metric

As explained in Section 2.2.7 for planning heuristics to function accurately, the internal metric of the latent space must reflect the causal constraints of the physical environment. While local smoothness ensures stability between adjacent states, a global geometric structure is required to ensure that the Euclidean distance between any two points serves as a reliable proxy for reachability (Wang et al. 2026; Myers et al. 2024).

In an unconstrained latent space, two states that are temporally distant may end up mathematically close in L_2 distance simply because they share visual similarities, such as background textures (Wang et al. 2026). To rectify this, we introduce a Temporal Distance Metric regularization based on a triplet-margin objective (Myers et al. 2024).

Implementation and Formulation

To capture the global relationship across a sequence of length T, we first compute a global latent representation $z_{global,t}$ by applying a spatial mean-pooling operation over the visual patch embeddings:

$$z_{global} = \frac{1}{P} \sum_{p=1}^{P} z_{t,p}$$

where P is the total number of visual patches $(14 \times 14 = 196)$. This mean-pooling ensures the loss focuses on the global state transition rather than getting distracted by localized, patch-level noise (Zhou et al. 2025). We then define a temporal triplet consisting of an anchor (z_t), a positive sample (z_{t+1}) and a negative sample (z_{t+N}). The parameter N defines the "look-ahead" distance; if N is too small, the manifold might not be stretched enough to support long-horizon planning (Wang et al. 2026; Myers et al. 2024).

The temporal loss is formulated as a hinge loss that enforces a margin γ between the distance to the next state and the distance to the distant state:

$$\mathcal{L}_{temporal} = \frac{1}{T-N} \sum_{t=1}^{T-N} \max(0, \|z_t - z_{t+1}\|_2^2 - \|z_t - z_{t+N}\|_2^2 + \gamma)$$

The margin γ is essential to prevent the loss from collapsing to zero, forcing the model to actively separate temporally distant states in the embedding space (Myers et al. 2024).

Role in Planning Efficiency This constraint prevents the "fractured map" problem by penalizing representations where the manifold folds back on itself in a way that violates temporal causality (Wang et al. 2026). By shaping the global topology so that Euclidean distance increases as a function of temporal separation, we provide the CEM planner with a consistent signal for its "distance-to-goal" heuristic. This ensures the latent manifold functions as a navigable coordinate system rather than a disconnected set of visual features.

Chapter 4

Experimental Setup

This chapter details the empirical framework utilized to evaluate the proposed architectural modifications and geometric constraints. We define our evaluation models, outline the dynamics of the robotic manipulation environment, specify the implementation and hyperparameter configurations, and establish the multi-axis diagnostic suite used to analyze our results.

4.1 Baseline Comparisons and Experimental Environment

To evaluate our proposed modifications, this section outlines the experimental setup. We define the specific model configurations and control baselines used to isolate our contributions, and then introduce the Push-T environment, detailing its task dynamics, observation spaces, and success criteria.

4.1.1 Evaluated Models and Baselines

We compare our architecture against both the officially reported state-of-the-art and our own locally reproduced baseline. Constraints are evaluated against the Extended Encoder Architecture. The models are defined as follows:

- **Reported DINO-WM Baseline (Zhou et al. 2025):** As an absolute reference point, we consider the published performance metrics of the original DINO-WM architecture.
- **Reproduced DINO-WM Baseline:** To ensure a strictly fair comparison that isolates architectural improvements from hardware disparities or our specific efficiency optimizations, we re-trained the vanilla DINO-WM architecture from scratch on a local GPU cluster. This serves as our primary control model.
- **Extended Encoder Architecture:** To first evaluate whether any observed improvements are specifically derived from our geometric constraints and not from the implicit regularization introduced by the lightweight bottleneck architecture alone, we evaluate an unconstrained version of our model. This sets the bar for our constraints to surpass.
- **Local Constraint:** This is the first constrained variation of the Extended Encoder Architecture. It integrates the proposed local geometry loss on top of the standard prediction loss, introducing explicit regularization on top of the implicit regularization already provided by the Extended Encoder Architecture.

- **Global Constraint:** This is the second constrained variation of the Extended Encoder Architecture. It integrates the proposed global geometry loss on top of the standard prediction loss, introducing explicit regularization on top of the implicit regularization already provided by the Extended Encoder Architecture.

Comparing the Extended Encoder Architecture against our Reproduced DINO-WM ensures that any measured efficiency gains or performance boosts are strictly attributable to our architectural and algorithmic contributions rather than differences in the training environment.

4.1.2 The Push-T Environment

To evaluate our proposed architecture, we utilize the **Push-T** environment, introduced by Chi et al. (2025). We select this environment because it requires complex geometric reasoning and precise control. It is significantly more challenging than the Maze or Wall tasks, which are also used in DINO-WM (Zhou et al. 2025), but it remains computationally feasible.

As illustrated in Figure 4.1, the core objective of the task is to control a pusher (represented as a blue dot) to interact with a T-shaped block (gray) until it perfectly overlaps with a fixed target T zone (green). The task requires both the pusher and the T to match the target locations.

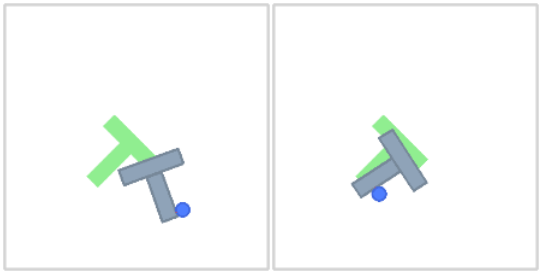

FIGURE 4.1: Visualization of the Push-T environment task (Chi et al. 2025). The left panel illustrates a start state, while the right panel shows an intermediate state after 25 steps.

We formalize the environment dynamics as follows:

- **Observation Space:** At each timestep t, the agent receives an image of the workspace, alongside a proprioceptive state vector representing the current (x, y) coordinates of the pusher. The dataset consists of 18500 samples generated from the original released expert trajectories with various levels of noise.
- **Action Space:** The action is a 2D vector specifying the target positional displacement for the pusher.
- **Episode Horizon:** The agent is given a strict maximum budget of 25 steps to complete the task.
- **Success Criteria:** An action sequence is considered successful if the T-block is within 20px of the goal position and within 20° of the goal angle.

4.2 Procedures and Implementation

Building on the architectural framework and environment dynamics established above, this section details the practical execution of our experiments. We outline the offline training loop for the predictive model, the CEM-based planning strategy used during inference, and the specific implementation details required to ensure reproducibility, including hardware, optimizers, and hyperparameter tuning.

4.2.1 Training Procedure

Our architecture is trained entirely on offline samples. During each training iteration, we sample mini-batches of transitions consisting of current observations, applied actions, and future observations (x_t, p_t, a_t, x_{t+1}). The training step proceeds as follows:

1. **Feature Extraction:** The frozen DINOv2 vision model extracts high-level feature maps h_t and h_{t+1} from the respective images.

2. **Latent Projection:** The online encoder E_θ maps h_t to the stable latent space z_t. Simultaneously, the target encoder $E_{\bar{\theta}}$ maps h_{t+1} to the target embedding z_{t+1}. A stop-gradient is applied to $E_{\bar{\theta}}$ which is updated separately.

3. **Forward Prediction:** The predictor P_ϕ takes the current latent state z_t, proprioceptive data p_t and the action a_t to compute the predicted future state $\hat{z}_{t+1}$.

4. **Optimization:** The network parameters of E_θ and P_ϕ are updated by minimizing the total loss, which is a weighted sum of the prediction loss (e.g., Euclidean distance between $\hat{z}_{t+1}$ and z_{t+1}) and our proposed geometric constraints.

5. **EMA Update:** Following the parameter update of E_θ, the weights of the $E_{\bar{\theta}}$ are updated via EMA, copying E_θ weights partially using the EMA Rate (τ), following standard self-distillation paradigms established in BYOL (Grill et al. 2020) and MoCo (X. Chen, Xie, and He 2021).

4.2.2 Planning Procedure

At inference time, the model weights are frozen, and the agent must use its learned latent space to achieve the goal state. We employ the Cross-Entropy Method (CEM) within a Model Predictive Control (MPC) framework to control the agent's behavior.

At each environment step t, the planning procedure executes the following routine (all environment hyperparameters are detailed in Table 4.1):

1. **Action Sampling:** The CEM planner samples N candidate action sequences of a fixed planning horizon H from a parameterized Gaussian distribution.

2. **Latent Rollout:** For each candidate action sequence, we use our trained JEPA predictor to roll out the predicted latent trajectory. This results in a total predicted temporal horizon of $H \times f = M$ steps, starting from the current encoded state z_t.

3. **Trajectory Scoring:** Each predicted trajectory is scored based on its latent distance (MSE) to the encoded representation of the target goal state z_{t+1}.

4. **Distribution Update:** The top K performing action sequences are selected, and their mean and variance are used to update the CEM sampling distribution. This process is iteratively refined over J planning iterations.

Once the CEM iterations reach a number of target optimization iterations J, the planner outputs the optimal action sequence of length H. To evaluate the robustness of our model, in accordance with the DINO-WM paper (Zhou et al. 2025), we use MPC and open-loop execution by adjusting m and I. In standard MPC execution, the agent executes only the first m actions before replanning. The new observation is then encoded, and the entire cycle is repeated until the task is successfully completed or the maximum step limit I is reached. In open-loop (non-MPC) execution, the agent directly executes all H actions and finishes.

4.2.3 Implementation

All models were trained and evaluated on a high-performance GPU cluster. The architectures and training loops were implemented using PyTorch. The primary compute nodes were equipped with NVIDIA H100 (80GB PCIe) and NVIDIA H200 (140GB SXM) GPUs, operating with bf16 mixed precision. Additional lightweight evaluations, such as planning runs, were distributed across nodes featuring NVIDIA A100 (40GB SXM) and V100 (32GB SXM) accelerators, which utilized standard fp16 mixed precision.

To ensure the reproducibility of our results, we detail the core hyperparameters utilized during the planning phase. Table 4.1 summarizes the configurations for the CEM planner on the Push-T task. These values were selected to maintain a comparison with the established baseline (Zhou et al. 2025).

TABLE 4.1: Environment-specific hyperparameters for the CEM planner on the Push-T task.

Notation	Description	Value
N	Number of sampled trajectories evaluated in parallel	300
H	Planning horizon	5
I	MPC iterations*	5 | 1
m	Number of actions to step in the environment*	1 | 5
K	Number of top trajectories selected in CEM	10
J	Number of optimization iterations	30
f	Frameskip	5
M	Total number of steps per planning episode	25

*Depends on using MPC or open-loop

In addition to the inference-time planning configurations, Table 4.2 details the hyperparameters utilized during the offline training phase of our JEPA architecture. These parameters were kept consistent across both the proposed constrained model and the unconstrained ablations to ensure fair comparison.

4.2.4 Optimizer

The JEPA encoder is trained with the optimizer AdamW (Loshchilov and Hutter 2019) at a learning rate of 6×10^{-4} and a (PyTorch-default) decoupled weight decay of 0.01, applied uniformly to all parameters of the online encoder, this is the framework

TABLE 4.2: Core training hyperparameters for the JEPA architecture and predictive world model.

Category	**Hyperparameter**	**Value**
Extended Encoder Architecture	Target Network EMA Rate (τ)	0.992
	Network Depth	4
	Encoder Optimizer	AdamW
	Encoder Learning Rate	6×10^{-4}
Local Constraint	Margin Threshold (C)	1
	Constraint Loss Weight	1
	Sigma	0.05
Global Constraint	Loss Margin (γ)	1
	Constraint Loss Weight	0.1
	Negative Sample Distance	5

default rather than an explicitly tuned value. The learning rate of 6×10^{-4} was chosen to compensate for a global batch size of 192 (tripled relative to the 64-sample default), broadly following square-root and batch-scaling heuristics for large-batch training (Krizhevsky 2014; Hoffer, Hubara, and Soudry 2017).

AdamW was preferred over Stochastic Gradient Descent (SGD) because the features coming from the DINOv2 visual encoder are highly complex and detailed (Oquab et al. 2024). For this type of data, adaptive optimizers like AdamW learn much faster and more reliably than standard fixed-step methods (Kingma and Ba 2015; Loshchilov and Hutter 2019). Additionally, AdamW separates its weight decay adjustment from the regular gradient updates. This separation is important because it stops the mathematical weights from being distorted by the fluctuating gradients of our extra geometric loss functions. This ensures that the resulting latent topology is driven by our total loss objective ($\mathcal{L} = \|\hat{z}_{t+1} - z_{t+1}\|_2^2 + \lambda \mathcal{L}_{\text{constraints}}$) instead of optimization artifacts.

4.2.5 Tuning

For our implementation, we had multiple hyperparameters that needed to be optimized. Specifically, our EMA learning rate, encoder depth, as well as Constraint specific hyperparameters. We landed at a middle ground of 0.992 after evaluating a range of [0.8, 0.999]. Similarly, we evaluated the amount of hidden layers used in E_θ and $E_{\bar{\theta}}$ using a range of [1, 5].

Our criteria for choosing these specific hyperparameters was an Optuna driven search (Akiba et al. 2019) with empirical evaluation using loss curves, prediction drift, and standard deviation metrics as our guiding factors, as well as selective planning runs to evaluate the findings. The Table 4.2 details our findings after our search.

4.3 Evaluation

To assess the quality of the trained models and systematically answer our research questions, the evaluation is conducted along three primary axes: the analysis of the

training dynamics, planning performance of the world model, and the evolution of the latent manifold.

4.3.1 Training

To gain deeper insights into the training dynamics, training loss curves and latent space divergence metrics are systematically compared across all configurations.

Additionally predictor rollout stability was logged as performance indicators and standard deviations of the batches were analyzed to ensure the latent representation was not collapsing.

To further evaluate the effectiveness of our local constraint, we developed a diagnostic script that computes three complementary metrics on the validation set across training checkpoints, each targeting a different aspect of the learned action-conditioned latent dynamics.

Smoothness ratio measures latent displacement between consecutive real observations relative to action magnitude:

$$\text{smooth_ratio} = \frac{\|z_{t+1} - z_t\|_2}{\|a_t\|_2}$$

Action sensitivity measures prediction discriminability across $K = 50$ randomly sampled actions, computed as the standard deviation of predicted latents:

$$\text{action_sensitivity} = \mathbb{E}_{\text{batch}}\left[\text{std}_k\left(z^{(k)}\right)\right]$$

Perturbation sensitivity measures the local Lipschitz behavior of the predictor under small action noise $\delta \sim \mathcal{N}(0, \sigma^2 I)$:

$$\text{perturb_ratio} = \frac{\|z_{\text{pred}}^{\text{pert}} - z_{\text{pred}}^{\text{base}}\|_2}{\|\delta\|_2}$$

Together, these metrics characterize whether the latent space is geometrically stable, action-discriminative, and locally smooth.

Temporal Distance Analysis

To measure the effects of the global constraint we additionally add a diagnostic plot measuring raw and normalized Euclidean latent distances between anchor z_t and distant states z_{t+k}. This is used to evaluate the effectiveness of the constraint compared to our unconstrained variant and probe how the learned latent space is organized along the time axis.

$$d(k) \;=\; \mathbb{E}_t\Big[\,\big\|z_t - z_{t+k}\big\|_2\,\Big], \qquad k = 1, \ldots, 10,$$

This is computed over 2000 validation slices of the PushT dataset. Latents are aggregated across patches with the same mean operator that the temporal triplet loss uses during training, so the curve reports exactly the quantity the loss acts on.

4.3.2 Diagnostic Decoder

To empirically verify that the reduction introduced by training the encoders filters out redundant information without discarding critical environmental dynamics, we employ a visualization decoder. This decoder is trained independently using a combination of Mean Squared Error (MSE) and a Visual Geometry Group (VGG) based perceptual loss (Johnson, Alahi, and Fei-Fei 2016). The post-hoc attachment ensures that gradients from the pixel-level reconstruction do not backpropagate into the main JEPA encoders. The decoder acts purely as a diagnostic probe to visualize the information capacity of the embedding space.

4.3.3 Planning

The planning performance is our main evaluation criterion. To directly address our research question, we build on the planning efficiency metrics surveyed in Section 2.2.8.

At inference time, the frozen model uses CEM (based on the findings in Section 2.2.4) within an MPC framework to plan exclusively in the learned latent space. At each environment step, the planner samples candidate action sequences, rolls out each through the predictor, and scores trajectories by their latent distance to the encoded goal state. The top performing sequences are used to iteratively refine the sampling distribution until convergence. In MPC execution, only the first actions are executed before replanning from a new real observation. In open-loop execution, the full sequence is executed without replanning.

Planning performance is evaluated along two axes. First, task success rates are reported under both MPC and open-loop execution settings, providing a direct measure of planning effectiveness. The planning horizon is capped at 25 steps, matching the evaluation protocol of the original DINO-WM benchmark (Zhou et al. 2025) and reflecting the average task complexity of the Push-T environment. Second, rollout stability is assessed by tracking the L2 distance between predicted and real encoded states over the execution horizon, alongside the initial L2 error, which reflects the representational gap between the online E_θ and target encoders $E_{\bar{\theta}}$.

4.3.4 Manifold Analysis

To isolate the quality of the learned representations from the planner's search strategy, we evaluate the latent space using two complementary methods: Principal Component Analysis (PCA) to visualize the global manifold topology, and linear probing to quantitatively assess the accessibility of the underlying environmental dynamics.

Principal Component Analysis To compare how our approaches restructure the latent manifold we perform a Principal Component Analysis (PCA) (see Section 2.2.9). This allows us to visualize a part of what the world-model encoder has learned. The PCA projects frame-level latents onto their first two principal components and colors each point by the ground-truth T-block position (x, y) from the simulator state. Specifically, for a batch of validation trajectories, every frame is encoded and the resulting patch tokens are aggregated into a single D-dimensional vector (D=96 for our Extended Encoder and D=384 for DINOv2). A full-rank PCA is then fitted on the resulting point cloud and the first two components are scattered.

This topological analysis addresses three structural requirements for downstream planning:

1. **Task-Relevant State Recoverability:** If coloring by x and y produces smooth, monotonic gradients along the leading PCs, the encoder has placed object position on a few dominant linear axes, which is exactly the kind of structure a latent-space planner can exploit. A colored cloud with no spatial structure indicates that position is either absent or entangled with directions PCA does not see first.

2. **Representational Concentration:** We report the explained variance of PC1/PC2 and the participation ratio $\mathrm{PR} = \left(\sum_i \lambda_i\right)^2 / \sum_i \lambda_i^2$, the effective number of dimensions carrying the variance. $\mathrm{PR} \approx 1$ signals representational collapse, $\mathrm{PR} \approx D$ a diffuse, isotropic spread, and intermediate values a useful concentration along a small number of axes.

3. **Architectural Comparison:** Because each run is plotted in its own column with identical axes and coloring, differences between configurations (e.g. the smoothness constraint, choice of predictor, training duration) become directly visible as changes in geometry and in $(\mathrm{PC1}, \mathrm{PC2}, \mathrm{PR})$, without committing to a particular downstream metric.

The visualization is therefore complementary to the planning-success metrics in Section 4.3.3 by identifying whether the latent space exposes the state variables the planner needs.

Linear Probing To decouple latent space quality from the planning heuristics of the Cross-Entropy Method, we employ a suite of evaluation probes, introduced in Section 2.2.9. These probes assess the representations generated by the DINO-WM baseline and the bottleneck architectures across four structural dimensions.

First, global and local geometry probes measure whether the latent manifold reflects the physical distances of the environment. The global geometry probe computes the linear correlation between latent Euclidean distance and physical state distance. The local k-nearest neighbors (Cover and Hart 1967) probe evaluates small-scale metric preservation by comparing latent neighborhoods against random samples for reference.

Second, state retention is evaluated to verify that the model preserves ground-truth physical variables within its compressed latent space. To measure this, a full pose probe applies linear regression to predict the exact 2D (x, y) pixel coordinates of both the pusher and the T-block, confirming whether the complete spatial configuration is linearly accessible. Simultaneously, an orientation probe utilizes ridge regression (Hoerl and Kennard 1970) to extract the $\sin(\theta)$ and $\cos(\theta)$ components of the T-block, assessing the retention of rotational dynamics.

Third, a temporal leakage probe tests for absolute time memorization. The probe measures whether a single latent frame contains enough information to linearly predict its exact time step within the episode. This type of memorization is undesirable because it introduces an optimization shortcut: Rather than learning the true physical dynamics of the environment, the model might simply correlate specific visual features with the progression of time to estimate task completion. By minimizing this temporal leakage, we verify that the model is forced to rely entirely on spatial understanding to make predictions.

Finally, an inverse dynamics probe evaluates transition linearity (Pathak et al. 2017). By predicting the applied action from two consecutive latent states, this probe tests whether the latent transitions maintain a strong relationship with the action space. As outlined in Section 2.1.6, this property is necessary for stable trajectory optimization.

Chapter 5

Results

This chapter presents the results across all model variants, examining both task performance and the internal structure of the learned latent representations for our discussion in Chapter 6.

5.1 Overall Task Performance

To evaluate the core performance of the proposed architectures, we first compare the final task success rates across all model variants. Table 5.1 summarizes the performance under both Model Predictive Control (MPC) and open-loop (without MPC) execution settings.

The baseline models establish a performance floor, with the Reproduced DINO-WM Baseline achieving a success rate of 0.92 with MPC and 0.88 without MPC. The introduction of the Extended Encoder Architecture yields a measurable performance increase, reaching the highest recorded success rates of 0.98 with MPC and 0.94 in the open-loop setting.

The Local Constraint model matches the Extended Encoder's performance under MPC (0.98) but exhibits a lower success rate in the open-loop setting (0.92). The Global Constraint model overlaps with our reproduced DINO-WM baseline, achieving an MPC success rate of 0.92 and for open-loop planning 0.92.

TABLE 5.1: Success Rate Comparison between the Models with MPC and open-loop.

Model	With MPC	Open-Loop
DINO-WM Baseline	0.90	0.86
Reproduced Dino-WM Baseline	0.92	0.88
Extended Encoder Architecture	0.98	0.94
Local Constraint	0.98	0.92
Global Constraint	0.92	0.92

Beyond final success rates, the efficiency of the planning process is evaluated by tracking convergence over the optimization horizon. Figure 5.1 illustrates the open-loop (Without MPC) success rate plotted against the number of Cross-Entropy Method (CEM) iterations.

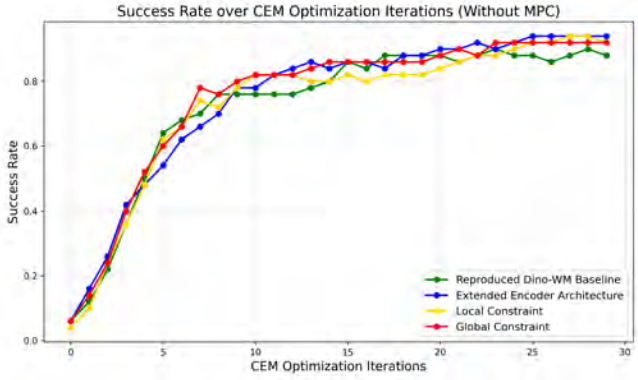

FIGURE 5.1: Success rate over CEM optimization iterations for each model variant without using MPC.

The early optimization phase (iterations 0 to 10) shows overlapping performance. In later stages of the CEM cycle, the Extended Encoder and Constraint models end up at a higher success rate.

5.2 Decoder Reconstruction

As visualized in Figure 5.2, the decoder successfully reconstructs the physical environment state from both the initial intermediate representations z_t (column 2) and the model's forward-predicted embeddings $\hat{z}_{t+1}$ (column 3) for all the models. For the Action-Space Smoothness Constraint, elaborated in Appendix A, the decoder failed to accurately reconstruct z_t and $\hat{z}_{t+1}$ (column 3) back into real images, as seen in Figure 5.3. For example, it occasionally omitted the blue pusher entirely or produced a gray "T" lacking sharp geometric definition.

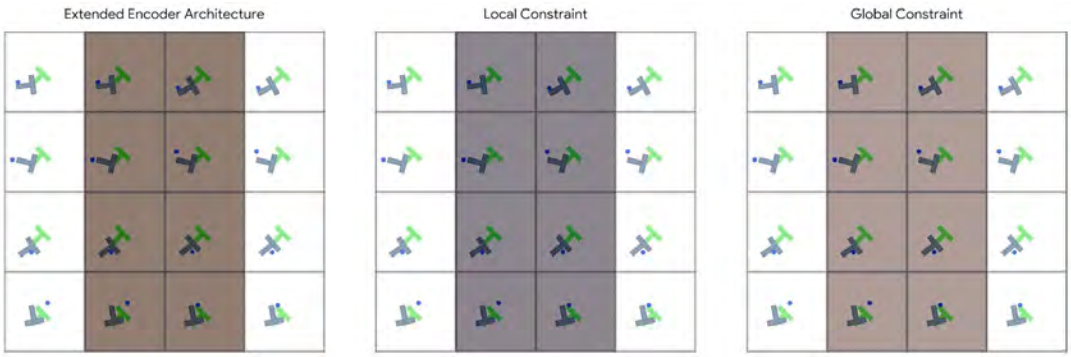

FIGURE 5.2: From left to right, the columns display: (1) The original observation x_t (2) The visual reconstruction decoded directly from the encoder's intermediate representation z_t (3) The visual reconstruction decoded from the model's forward-predicted embedding $\hat{z}_{t+1}$ and (4) The ground truth future observation x_{t+1}. Extended Encoder Architecture (left), Local Constraint (middle), Global Constraint (right).

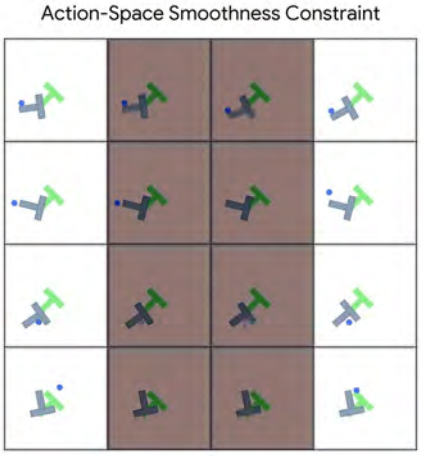

FIGURE 5.3: Decoder reconstructions of the Action-Space Smoothness Constraint, elaborated in Appendix A.

5.3 Latent Topology and Rollout Stability

To understand the performance differences reported in Section 5.1, we analyze the internal structure of the learned latent representations across all model variants.

5.3.1 Latent Topology

To understand the performance results detailed in Section 5.1, we analyze the Principal Component Analysis (PCA) as explained in Section 4.3.4.

Figure 5.4 compares the global latent topology of the raw Reproduced DINO-WM Baseline against our proposed unconstrained Extended Encoder Architecture.

To isolate how the addition of explicit regularizations affected the latent manifold, we expand our topological analysis to the constrained models via PCA (see Figure 5.5).

5.3.2 Linear Probing

To assess the structural properties and support our findings in the PCA analysis of the latent spaces, we applied the suite of linear probes introduced in Section 2.2.9 to the validation set across all model variants.

Global and Local Geometry The global geometry probe evaluates the linear correlation between physical state distance and Euclidean latent distance. A positive slope combined with a high correlation (r) indicates that physical separation proportionally scales into latent separation, whereas a flattened line signifies an entangled topology. Figure 5.6 reports these outcomes. The reproduced DINO-WM baseline exhibits a Pearson correlation of $r = 0.17$. The introduction of the unconstrained Extended Encoder Architecture increases this correlation to $r = 0.71$. The Local and Global Constraint models record correlations of $r = 0.61$ and $r = 0.59$.

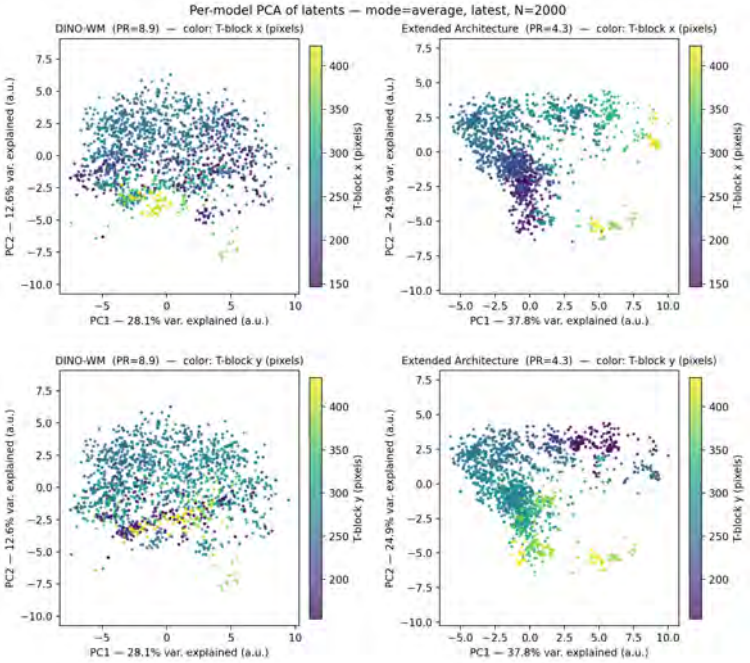

FIGURE 5.4: Per-model Principal Component Analysis (PCA) of latent spaces ($N = 2000$) color-coded by the physical T-block position. The left column displays the DINO-WM baseline, and the right column displays the Extended Encoder Architecture model. The top row maps spatial x-coordinates (pixels), while the bottom row maps spatial y-coordinates (pixels).

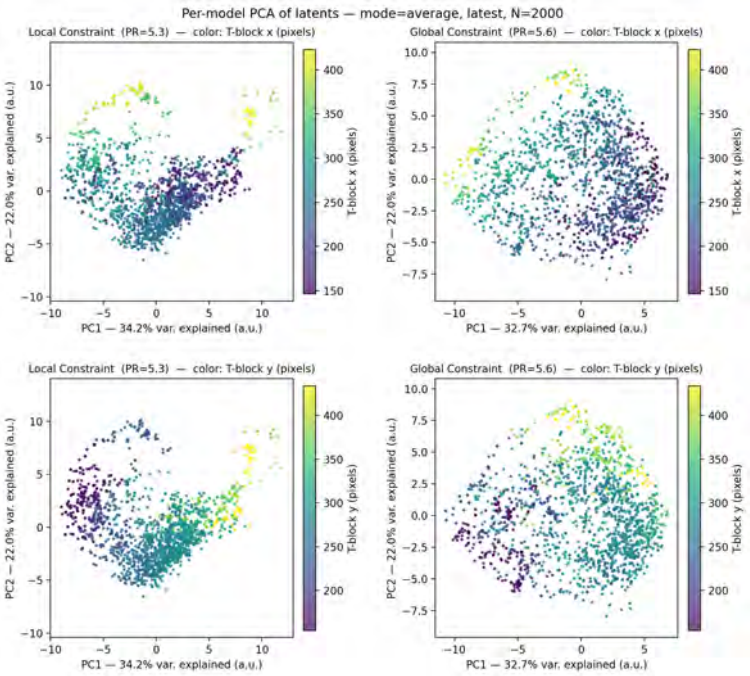

FIGURE 5.5: Per-model Principal Component Analysis (PCA) of constrained latent spaces ($N = 2000$) color-coded by physical T-block coordinates. The left column displays the Local Perturbation Constraint model, and the right column displays the Global Geodesic Constraint model. The top row maps spatial x-coordinates (pixels), while the bottom row maps spatial y-coordinates (pixels).

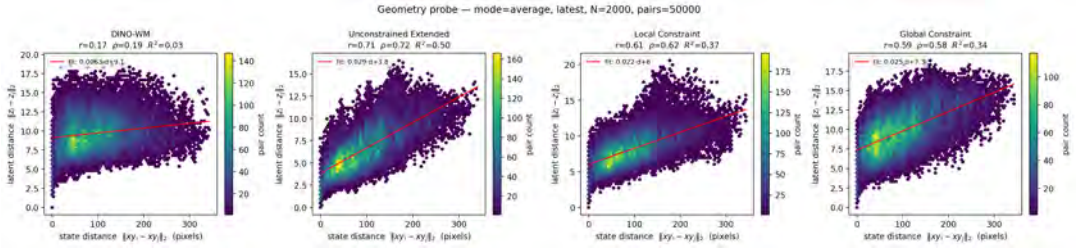

FIGURE 5.6: Global geometry probe comparing state distance (pixels) to latent distance across random validation frame pairs. Red lines indicate the linear regression fit. Panel titles report Pearson (r) and Spearman (ρ) correlation coefficients alongside the explained variance (R^2).

To evaluate local geometric stability, the k-nearest neighbors (k-NN) probe compares the mean physical state distance of the $k = 10$ latent nearest neighbors (blue) against a random distribution (grey) for reference. A ratio (μ_{k-NN}/μ_{random}) approaching 0 indicates strong local geometric preservation. Furthermore, a narrow distribution shifted left demonstrates uniform geometric quality, while a wide distribution overlapping the random distribution suggests inconsistent metric preservation. As shown in Figure 5.7, the DINO-WM baseline yields a local ratio of 0.11. The unconstrained Extended Encoder Architecture lowers this ratio to 0.07, while both the Local and Global Constraint models record a ratio of 0.08.

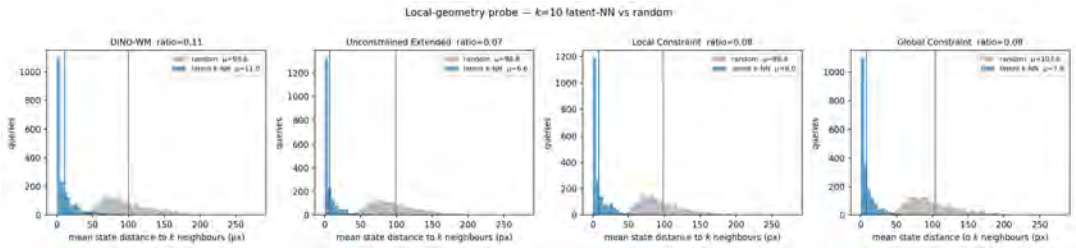

FIGURE 5.7: Local geometry probe, comparing the mean physical state distance of $k = 10$ latent nearest neighbors (blue) against a randomly sampled reference (grey). Histograms display the distance distribution across query frames, with vertical lines marking the distribution means. Panel titles report the ratio of these means (μ_{k-NN}/μ_{random}).

State Retention and Orientation State retention is evaluated using linear and ridge regression to extract coordinate and rotational data from the embeddings. The full pose probe generates a heatmap of R^2 scores for individual spatial and rotational variables, as shown in Figure 5.8. The orientation probe plots (Figure 5.9) predicted versus true physical angle θ, where a tighter alignment along the diagonal indicates higher rotational accuracy. The DINO-WM baseline maintains R^2 values at or above 0.82 across all state dimensions, with an orientation error of 7.2°. The bottlenecked architectures maintain comparable R^2 scores for spatial coordinates (x, y), but demonstrate varying reductions in rotational accuracy. Specifically, the R^2 for the $\sin\theta$ variable drops to 0.55 in the unconstrained model, with orientation errors increasing to 14.4° (Extended Encoder), 17.1° (Local), and 13.5° (Global).

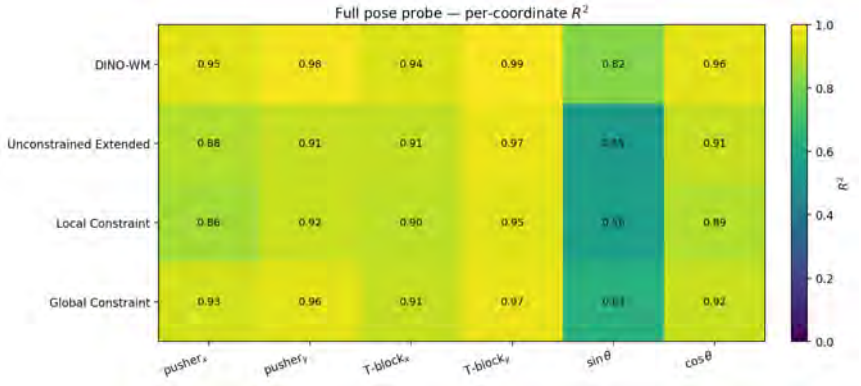

FIGURE 5.8: Full pose linear probe, displaying a heatmap of R^2 values for per-coordinate spatial and rotational retention across model variants.

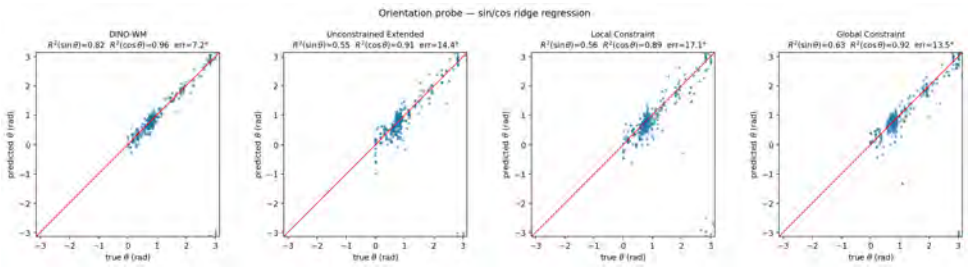

FIGURE 5.9: Orientation probe scatter plots comparing the true physical angle θ (in radians) against ridge regression predictions from the latent embeddings. Red lines indicate perfect prediction.

Temporal Leakage The temporal leakage probe measures whether a single latent frame contains enough information to linearly predict its exact time step within the recorded episode. The results are visualized as scatter plots, comparing the true normalized frame position against the predicted position. As explained in Section 4.3.4, a lower R^2 score is preferable because it indicates the model relies less on artificial temporal cues. Figure 5.10 reports these findings. The DINO-WM baseline yields a predictive R^2 of 0.90, indicating high temporal leakage. The Extended Encoder Architecture lowers this to an R^2 of 0.80. The Local Constraint model reports the lowest temporal predictability at $R^2 = 0.73$, while the Global Constraint model reports $R^2 = 0.82$.

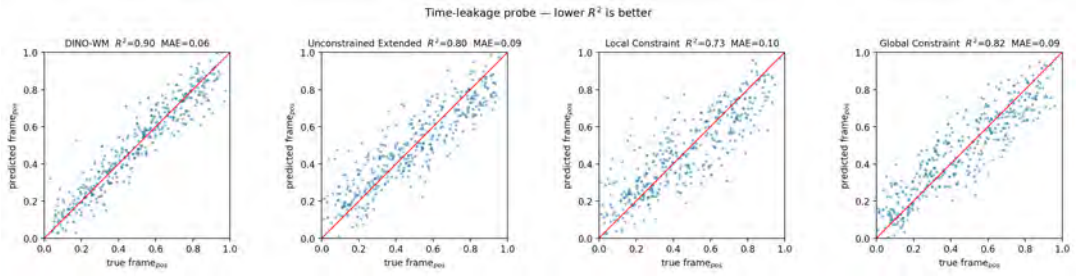

FIGURE 5.10: Temporal leakage probe scatter plots comparing the true normalized frame position within an episode to the predicted position. Red lines indicate perfect prediction. Panel titles report R^2 and Mean Absolute Error (MAE).

Inverse Dynamics The inverse dynamics probe evaluates transition linearity by predicting the applied action a_t given two consecutive latent states (z_t, z_{t+1}). The scatter plots compare the true action values against the predicted values for each action dimension. Tighter clustering along the diagonal yields a higher R^2 score. Figure 5.11 reports these accuracies. The DINO-WM baseline achieves R^2 scores of 0.70 and 0.86 for the two action dimensions, with the upper row corresponding to the horizontal target displacement (Δx) and the lower row to the vertical target displacement (Δy). The Global Constraint model records similar predictive accuracy with scores of 0.74 and 0.85. The unconstrained Extended Encoder Architecture (0.54, 0.62) and the Local Constraint model (0.52, 0.73) exhibit lower inverse dynamic predictability.

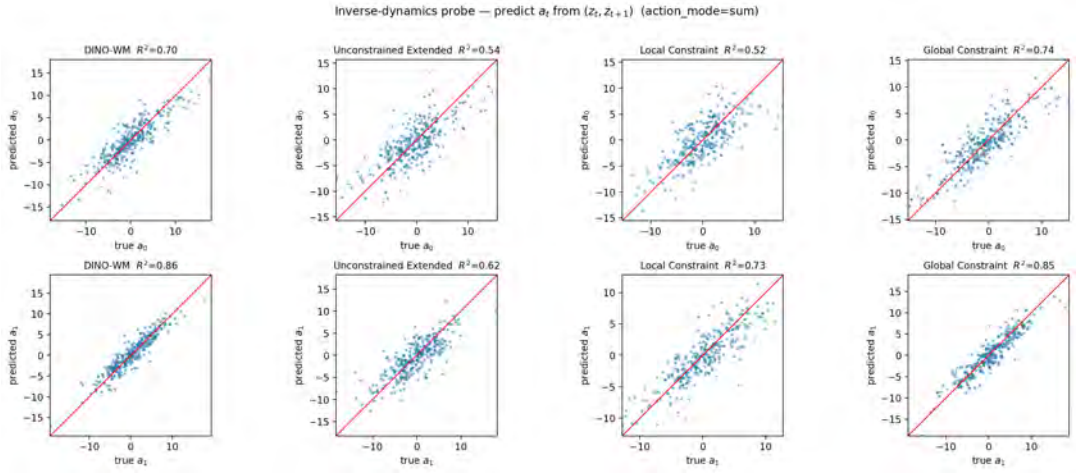

FIGURE 5.11: Inverse dynamics probe scatter plots predicting the applied action dimensions from consecutive latent transitions. The top row displays the horizontal target displacement (Δx), while the bottom row displays the vertical target displacement (Δy). True action values are plotted against predicted values, with red lines indicating perfect prediction.

5.3.3 Rollout Stability

Figure 5.12 quantifies the rollout stability by plotting the L_2 distance between the predicted latent states and the encoded real-world physical states over the 25-step (5 planning steps because frameskip is equal to 5) execution horizon without using MPC. The Initial L_2 Error says how much the encodings of the Online Encoder differ from the encodings of the Target Encoder. In the DINO-WM Baseline the same encoders (DinoV2) are used, so the Error is 0. Our Extended Encoder Architecture has the biggest initial L_2 error between encodings.

Even though the initial encodings exhibit a larger representational gap, the Extended Encoder Architecture maintains a significantly tighter error bound throughout the entire 5-step horizon when predictions are involved.

For completeness, the equivalent tracking error configurations evaluated under active Model Predictive Control (MPC) rollouts are provided in Appendix B.

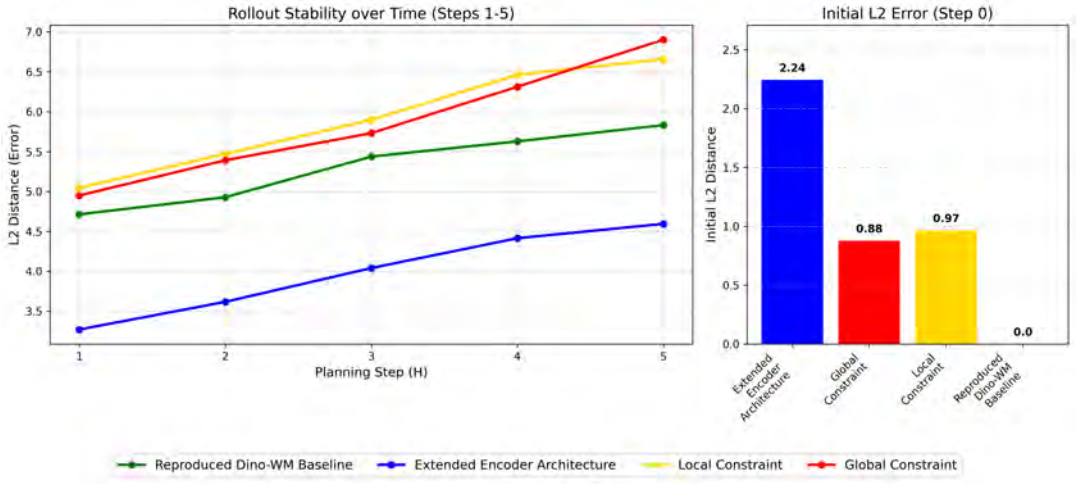

FIGURE 5.12: L_2 distance between predicted and real encoded states over the 5 planning steps (25 environment steps at frameskip 5) without using MPC. Step 0 reflects the gap between online and target encoder representations. Steps 1–5 reflect true prediction error, where a lower value indicates better rollout stability.

5.4 Training Behavior

Figure 5.13 compares the validation loss trajectories of the Reproduced DINO-WM Baseline and the Extended Encoder Architecture over their respective training runs.

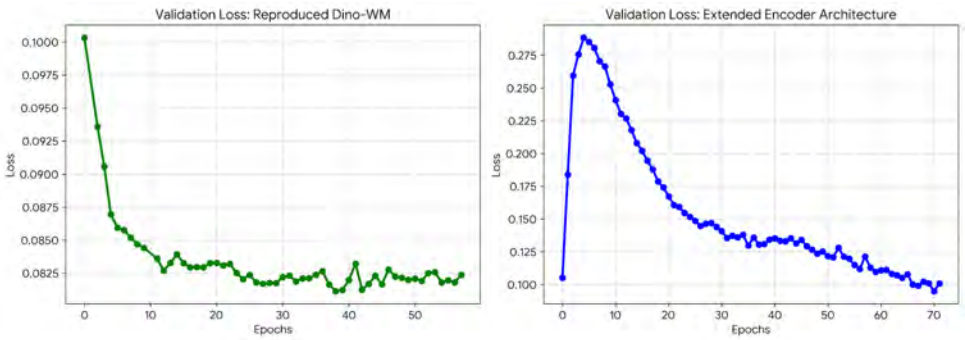

FIGURE 5.13: Validation loss curves for the Reproduced DINO-WM Baseline and the Extended Encoder Architecture training runs.

An additional evaluation demonstrated that the Extended Encoder Architecture already achieves a success rate of 0.80 after a single training epoch, though it required 70 epochs to reach 0.94.

Figure 5.14 compares the smoothness, action sensitivity and perturbation sensitivity of the Reproduced DINO-WM Baseline and the Extended Encoder Architecture over training. The two models exhibit differing trends across all three metrics. The DINO-WM Baseline shows constant smoothness, a low and mostly stable action sensitivity, alongside a higher and rising perturbation sensitivity. In contrast, the metrics for the Extended Encoder Architecture exhibit an initial peak during early epochs, followed by a gradual, steady decline. Notably, by the end of the training horizon, its action sensitivity converges toward the level established by the baseline model.

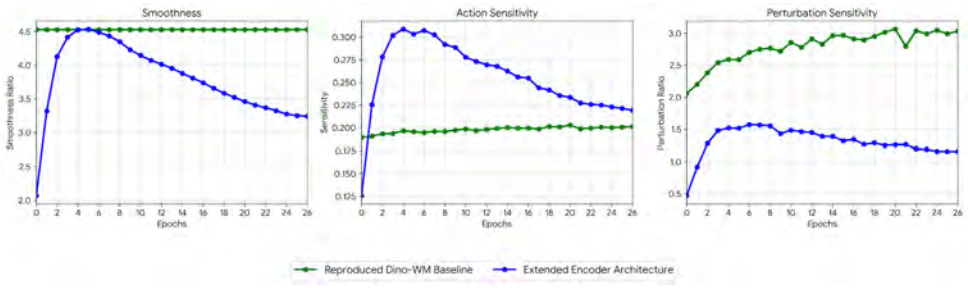

FIGURE 5.14: Smoothness ratio, perturbation ratio and action sensitivity over training epochs for the Reproduced DINO-WM Baseline and the Extended Encoder Architecture, illustrating the dynamics between a frozen and a trainable encoder.

5.5 Regularization Effects

Having established that the unconstrained Extended Encoder Architecture already improves upon the baseline, we now isolate the individual contribution of each explicit geometric constraint.

5.5.1 Effect of Local Geometric Regularization

Table 5.2 reports the success rate at each checkpoint. Performance remains stable at 0.90 through epochs 2 and 7, collapses to 0.64 at epoch 13, then partially recovers to 0.86 at epoch 14, despite the fact that both vision and local loss are only decreasing (as shown in Figure 5.15).

TABLE 5.2: Success Rate Comparison between the Local Constraint Epochs.

Model	Success Rate
Epoch 2	0.90
Epoch 7	0.92
Epoch 13	0.64
Epoch 14	0.86

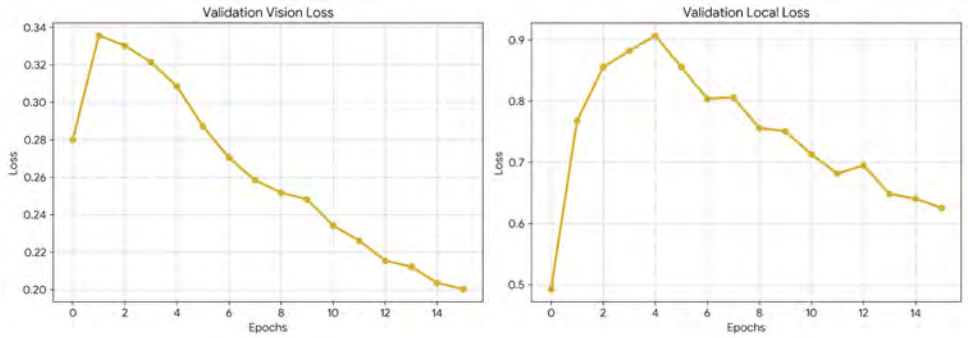

FIGURE 5.15: Validation vision loss (left) and validation local perturbation loss (right) over training epochs for the Local Constraint model. The perturbation loss magnitude is consistently larger than the vision loss throughout training.

Figure 5.12 shows a lower L2 error for the Extended Encoder Architecture across the 5-step rollout horizon when predictions are involved than the Local Constraint Model. In contrast, the Local Constraint Model has a much lower Initial L2 Error of 0.97, where the Extended Encoder Architecture has 2.24. The error accumulation (slope) of the Local Constraint is very similar to the one of the Extended Encoder Architecture.

The smoothness ratio, action sensitivity and perturbation sensitivity of the Local Constraint model (shown in Figure 5.16) are lower, but quite similar to those of the unconstrained Extended Encoder Architecture.

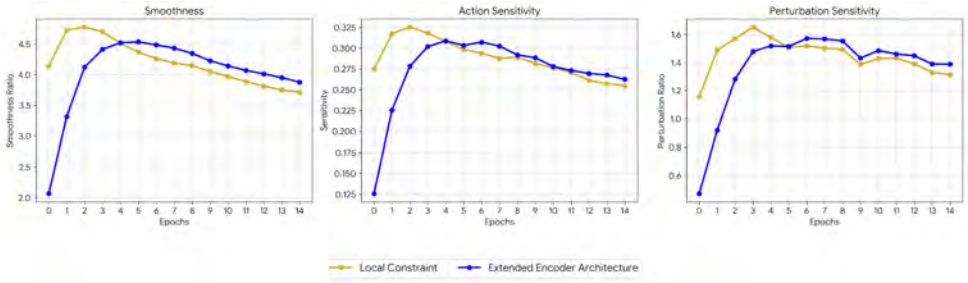

FIGURE 5.16: Smoothness (left), action sensitivity (middle), and perturbation sensitivity (right) over training epochs, comparing the Local Constraint model against the unconstrained Extended Encoder Architecture.

5.5.2 Effect of Global Geometric Regularization

To quantify the impact of the proposed architectural modifications and regularizations on the metric scale of the manifold, we plot the mean latent distance as a function of the temporal gap, as detailed in Section 4.3.1. Figure 5.17 displays the raw Euclidean distance metrics, where a vertical dotted line highlights the target horizon ($N = 5$), where the global constraint's hinge margin is applied. The absolute values of the plot are shown in Table 5.3.

TABLE 5.3: Temporal Distance Absolute Values

Model	**k=1**	**k=5**	**k=10**
DINO-WM	5.34	8.67	9.51
Extended Encoder Architecture	2.15	4.36	5.80
Global Constraint	3.44	7.37	8.97

The three model configurations exhibit distinct absolute numerical scales. The DINO-WM Baseline maintains the highest absolute distance profile across all evaluated intervals. The introduction of our unconstrained Extended Encoder Architecture shifts this distance baseline downward, establishing a tighter profile. The addition of the explicit Global Constraint alters this trajectory relative to the unconstrained Extended Encoder Architecture, scaling the absolute metrics upward across the entire horizon.

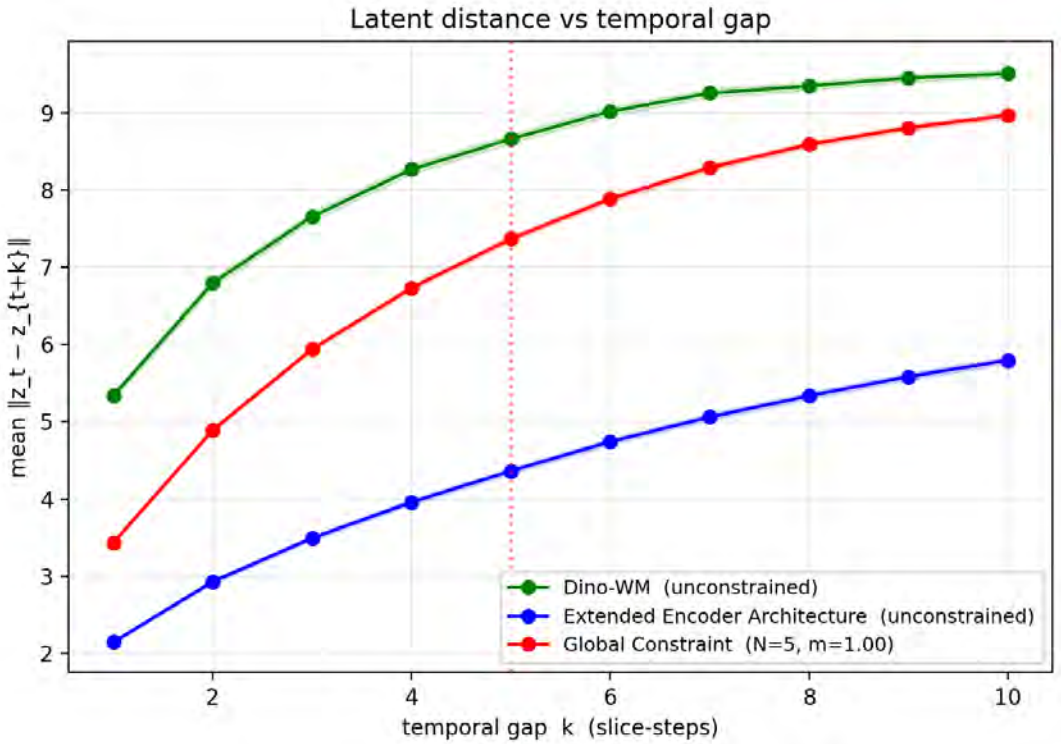

FIGURE 5.17: Mean raw Euclidean latent distance $\|z_t - z_{t+k}\|$ plotted against temporal gap k (slice-steps) for validation trajectory slices. The shaded bands show a standard error interval ($\frac{std}{\sqrt{N_{\text{samples}}}}$), and the vertical dotted line marks the alignment boundary ($N = 5$) for the global constraint.

When each curve is divided by its baseline value at $k = 1$ to evaluate the underlying topological shape (see Figure 5.18 with Table 5.4), the structural variance between the configurations changes. The DINO-WM baseline displays a flattened trajectory relative to the other two variants, expanding by a total factor of ~ 1.6 over the five-step horizon. In contrast, both the unconstrained Extended Encoder Architecture and the Global Constraint variant produce overlapping curves that scale upward at a similar rate, reaching a final scaling factor of ~ 2 at $k = 5$.

TABLE 5.4: Temporal Distance Normalized Values

Model	**k=1**	**k=5**	**k=10**
DINO-WM	1.0	1.62	1.78
Extended Encoder Architecture	1.0	2.03	2.70
Global Constraint	1.0	2.15	2.61

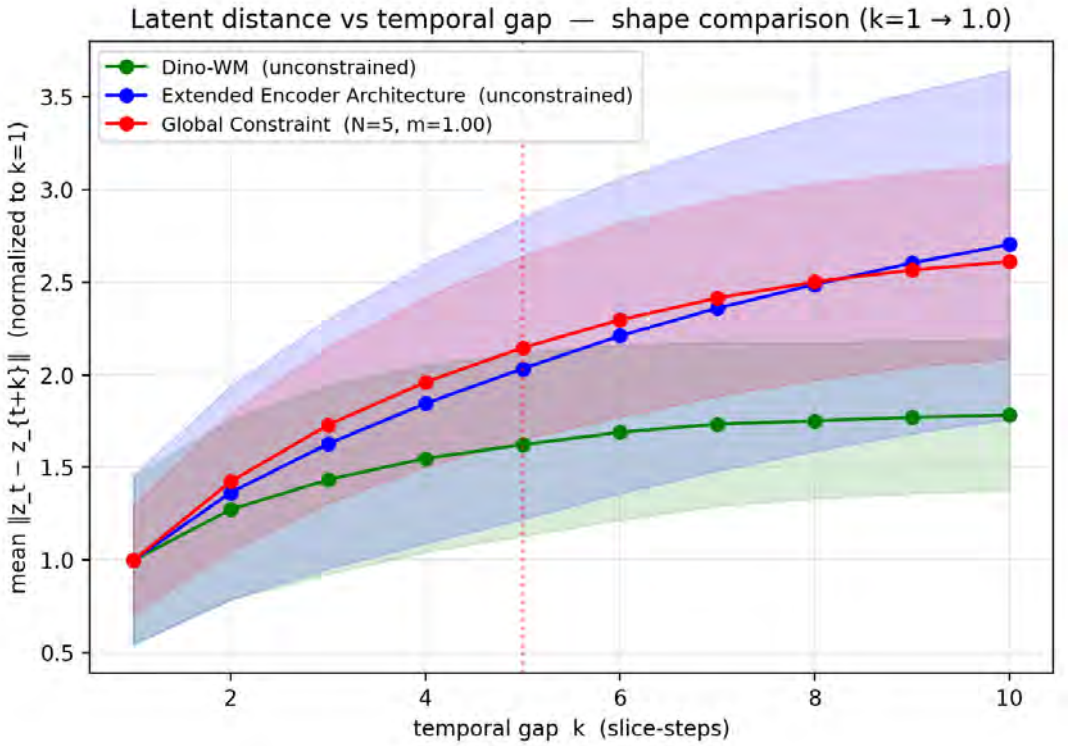

FIGURE 5.18: Normalized latent distance shape comparison against temporal gap k. Each curve is normalized by its relative baseline scale at $k = 1$, evaluating the geometric expansion factor of sequential frames relative to adjacent transitions.

To further examine the optimization behavior during training, Figure 5.19 displays the three main loss components for our Encoding Layer.

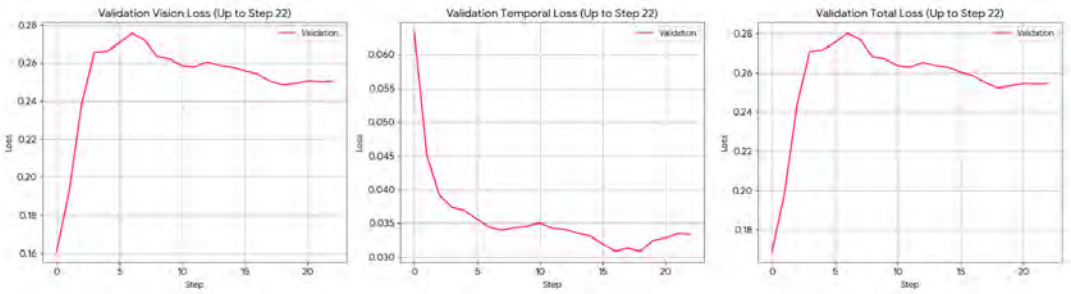

FIGURE 5.19: Left plot shows the Global Constraint MSE loss, center the temporal triplet loss and right is the combined loss together with proprioceptive loss (not included)

Chapter 6

Conclusion and Outlook

This chapter synthesizes the experimental findings and draws conclusions regarding the effectiveness of lightweight bottlenecks and explicit geometric constraints for latent world model planning. We first discuss the implications of each experimental finding, then revisit the original thesis objectives. Finally, we outline the limitations of the current experimental setup and identify concrete directions for future work.

6.1 Discussion

We now interpret the experimental results across all model variants, examining both the structural changes observed in the latent manifold and their consequences for downstream planning performance.

6.1.1 Implicit Regularization via the Unconstrained Bottleneck

The central finding of this thesis is that our unconstrained Extended Encoder Architecture as a restructuring layer, added on top of a frozen DINOv2 encoder, is sufficient to greatly improve planning performance in the Push-T environment over the Reproduced DINO-WM Baseline (Table 5.1).

The PCA analysis in Figure 5.4 provides a direct visualization, the bottleneck restructures the fractured DINOv2 feature space into a coordinate system where physical position correlates with the primary geometric axes. This restructuring is not the result of any explicit constraint, it emerges from the Extended Encoder Architecture itself, which acts as an implicit regularizer by filtering out task-irrelevant visual features, which are not relevant for predictions and reorganizing the latent space. This compression does not discard task-relevant information, as evidenced by the decoder in Figure 5.2 successfully reconstructing the physical environment state from the compressed latent embeddings.

The visual interpretation is mathematically confirmed by the linear probing results. The global geometry probe (Figure 5.6) demonstrates a drastic structural shift: the DINO-WM baseline exhibits an almost entirely entangled manifold ($r = 0.17$), whereas the unconstrained bottleneck explicitly aligns latent distances with physical state distances ($r = 0.71$). Also, the temporal leakage probe (Figure 5.10) reveals that the baseline model relies heavily on artificial temporal cues, memorizing absolute episode time ($R^2 = 0.90$). The introduction of the bottleneck acts as a strong temporal regularizer, stripping away this shortcut learning and forcing the model to rely on spatial dynamics.

However, the probing results also reveal the specific trade-offs introduced by this structural compression (Figure 5.8). While the DINO-WM baseline retains near-perfect rotational reproducibility and highly linear inverse dynamics, the bottleneck architectures compress away fine-grained rotational details (evidenced by the drop in $\sin\theta$ predictability in Figure 5.9) and feature less linear action-transitions (Figure 5.11). Predicting a_t from (z_t, z_{t+1}) via linear regression, the probe's lower R^2 scores reflect reduced linear decodability. This occurs because the bottleneck uses non-linear GELU activations and compression layers to optimize solely for state prediction error rather than action linearity. Still, the planning success rates indicate that for sampling-based heuristics like CEM, an unentangled global coordinate map is significantly more valuable than perfect local transition linearity. The planner can successfully tolerate a less linear action space as long as the global latent layout provides a reliable, monotonic gradient towards the target goal.

The training dynamics in Figure 5.13 reveal why this architectural restructuring is so effective over extended training horizons. While the DINO-WM baseline prematurely saturates around epoch 25, restricted by a frozen encoder that leaves the predictor without the capacity to improve further, the Extended Encoder Architecture continues to steadily improve through epoch 70. Notably, this sustained, long-term improvement follows a sharp initial spike in prediction loss. Rather than indicating a failure to learn, this early instability is a well-documented artifact of training momentum-based target networks. As demonstrated in MoCo v3 by X. Chen, Xie, and He (2021), architectures utilizing an EMA lag frequently exhibit such volatility before the target space successfully converges.

This divergence is further reflected in the diagnostic metrics shown in Figure 5.14. The frozen DINO-WM architecture exhibits a higher and rising perturbation sensitivity, alongside a low, static action sensitivity. This indicates that the predictor becomes locally sensitive to action noise but fails to develop globally discriminative action dynamics. Because the frozen DINOv2 features are not naturally organized around action-relevant variations, the predictor has no coherent structure to latch onto, forcing it to overfit to local noise within the feature space.

In contrast, the trainable Extended Encoder Architecture follows a markedly different trajectory that closely mirrors the early instability observed in the prediction loss. The initial peak across all three metrics is driven by the same momentum-encoder dynamics (X. Chen, Xie, and He 2021). As the randomly initialized online encoder rapidly expands the latent manifold, the lagging EMA target encoder creates a highly volatile, unstructured space. Because the latent space is highly disorganized during this early phase, any change in the action input causes the model's predictions to jump erratically. This chaotic jumping is exactly what drives the temporary spike in both action and perturbation sensitivity. However, as the target space stabilizes, the encoder and predictor co-adapt. This adaptation drives a steady decline in the metrics, compressing the representations to minimize prediction error. Ultimately, the encoder restructures the latent geometry into a smooth, navigable coordinate system, which proves sufficient for CEM planning, as demonstrated by the sustained performance improvements.

The 1-epoch diagnostic, reported in Section 5.4, confirms that this sustained improvement does not come from the predictor becoming more expressive. The model evaluated after a single epoch already achieves a strong success rate, indicating the

predictor's core transition function is learned rapidly. The continued gains, therefore, originate from the encoders reshaping the latent manifold into a more navigable geometry, giving the same predictor an increasingly well-structured space to operate in. Notably, this improvement persists despite the online and target encoders producing representations in different spaces, as reflected by the higher initial L2 error in Figure 5.12. Yet once predictions are involved, the Extended Encoder Architecture maintains a significantly tighter error bound than the baseline across the full rollout horizon, suggesting the predictor successfully adapts to bridge this representational gap.

Together, these two observations point to the same conclusion: in this setting, the quality of the latent space is the primary bottleneck for planning performance, not the capacity of the predictor. While the general benefit of restructuring the latent space was anticipated, the extent to which the bottleneck alone could close the performance gap (without any explicit geometric regularization) was unexpected.

This aligns with the broader argument by LeCun (2022) and Assran et al. (2023) that decoder-free architectures operating in abstract latent spaces are inherently more suitable for planning than reconstructive alternatives. However, for this specific environment and configuration, the results suggest that a lightweight architectural bottleneck may be sufficient to resolve the metric mismatch organically, without requiring explicit loss engineering as proposed in recent geometric regularization literature (Wang et al. 2026; Myers et al. 2024).

6.1.2 The Ineffectiveness of Explicit Local Constraints

While the Extended Encoder Architecture demonstrated that the bottleneck alone is sufficient to restructure the latent space, we investigated whether an explicit local smoothness constraint could further improve planning. Our findings suggest it could not and for two compounding reasons. First, the implicit regularization had already achieved a comparable level of local smoothness organically, as evidenced by the smoothness ratio and action sensitivity of the Local Constraint model closely tracking those of the unconstrained Extended Encoder Architecture (Figure 5.16). Second, despite having nothing left to improve geometrically, the dominant perturbation loss still exerted significant pressure on the encoder (Figure 5.15), while the analyzed latent manifold based on T-block position produced similar shapes in the most dominant dimensions (Figure 5.5). Ending up in a geometry that is neither smoother than the unconstrained bottleneck, nor better structured for CEM planning.

The linear probing results confirm this redundancy. The local k-nearest neighbors (k-NN) probe (Figure 5.7) demonstrates that the unconstrained bottleneck already achieves a highly stable local metric ratio of 0.07. The application of the explicit Local Constraint fails to improve upon this, yielding a ratio of 0.08. More critically, the inverse dynamics probe reveals that the Local Constraint makes state transitions much harder to predict (Figure 5.11). By forcing the predictor to satisfy the perturbation loss, the model sacrifices its ability to linearly map consecutive states to the applied action (R^2 dropping to 0.52 and 0.73). For a planner that relies on consistent action-to-state mappings, this degradation in inverse dynamics directly correlates with the observed drop in open-loop success rates.

This dynamic is reflected in the epoch-level success rates in Table 5.2, where performance rises to 0.92 through the early checkpoints before collapsing to 0.64 by

epoch 13 but increasing again later. This pattern suggests that the bottleneck initially establishes a useful latent geometry that supports planning, but the accumulating distortion from the perturbation loss introduces an unstable learning signal.

Together these findings indicate that explicit local geometric regularization is counterproductive in this setting, where the bottleneck has already resolved the smoothness problem the constraint was designed to address.

6.1.3 Metric Inflation and the Redundancy of Global Regularization

The Global Temporal Constraint represents a slight performance regression among our proposed changes to the DINO-WM architecture. It drives the open-loop success back down, closer to the baseline floor, from the newfound ceiling of 0.94 with the unconstrained Extended Encoder Architecture. The data demonstrates that this specific triplet loss configuration marginally underperformed relative to the unconstrained bottleneck, however these results do not rule out the viability of an explicit temporal regularization constraint broadly.

First, the normalized temporal scaling curves in Figure 5.18 indicate that the explicit global constraint is topologically redundant within this setup. The model optimized with the triplet hinge loss produces a nearly identical normalized curvature to the unconstrained architecture, within $\sim 5\%$ at the alignment horizon (k=5), demonstrating that the unconstrained bottleneck already develops a temporal gradient on its own. Together with the unnormalized temporal distance in Figure 5.17 we can see that the latent space is simply inflated by the margin distance without meaningful planning improvements or temporal encoding. This is further confirmed when examining the loss curves in Figure 5.19 where the temporal loss quickly falls and then stagnates, while the implicit regularization loss term remains dominant over the composite loss term.

When examining the Principal Component Analysis (Figure 5.5), focusing on the X and Y positions of the T-block, we observe a slight gradient effect compared to DINO-WM. However there is a less clear restructuring visible compared to the Extended Architecture and the Local Constraint. This could be due to the constraint pulling the manifold of the two primary dimensions into a more uniform shape when initial restructuring happens.

Although the Global Constraint successfully preserved highly linear inverse dynamics and matched the baseline performance ($R^2 = 0.85$ in Figure 5.11), this retention of transition linearity was insufficient to yield overall planning improvements. Because it failed to improve the underlying spatial geometry beyond the unconstrained bottleneck, recording a lower global geometry correlation of $r = 0.59$ (Figure 5.6), the constraint acts as a redundant mechanism in this configuration. Ultimately, these findings indicate that the explicit global regularization introduces additional loss complexity without providing tangible benefits to the CEM planner.

6.2 Review of Thesis Objectives

As defined in Section 1.2, this thesis investigates: **How can a visual input encoded latent space be restructured into a navigable manifold for non-reconstructive world models using an unconstrained lightweight bottleneck layer versus a geometrically constrained bottleneck layer?** To answer this, we first evaluate our two sub questions.

What is the individual impact of an unconstrained bottleneck versus a geometrically constrained bottleneck on the stability and topological regularity of the learned latent manifold?
The bottleneck introduced by the Extended Encoder Architecture restructured the latent manifold, as shown in the PCA plot and tighter rollout error bounds. The local constraint variant produced a comparable but less stable topology, with mid-training instability, despite similar smoothness metrics. The global constraint variant increased the absolute metric scale without meaningfully changing the normalized topological shape, which the unconstrained bottleneck had already established. This structural transformation was mathematically verified by the linear probing suite, which demonstrated that the unconstrained bottleneck organically aligns latent Euclidean distances with physical state realities, simultaneously acting as a strong regularizer against temporal memorization.

To what extent do explicit geometric constraints improve the convergence and execution success of a sampling-based planner compared to the unconstrained bottleneck?
Neither constraint improved upon the unconstrained bottleneck. The local constraint caused a performance drop at epoch 13, before partially recovering, suggesting the perturbation loss interfered with an already functional manifold. The global constraint reduced open-loop performance, confirming that scaling the metric does not translate into planning gains for the CEM planner.

Returning to our main question. The sub-questions together provide a coherent answer. The bottleneck changes the structure of the latent space in a way that benefits planning, and adding explicit constraints on top of this improves neither the manifold structure nor the planner success rate. This suggests that, for this setting, the metric mismatch identified in Section 1.2 is sufficiently resolved by the architectural bottleneck alone, and that explicit geometric loss terms are redundant when a trainable projection layer is already present. From a practical perspective, this suggests that a lightweight projection layer on top of any frozen vision encoder may be a worthwhile addition to world model pipelines.

6.3 Limitations

The contributions and findings summarized above are subject to a number of practical limitations that arose from available compute resources and time constraints. The following limitations should be considered when interpreting the results and inform the directions for future work outlined in Section 6.4.

Single environment All experiments were conducted exclusively on the Push-T environment, which limits the generalization of the findings. While the results demonstrate that the architectural bottleneck is effective in this setting, it remains an open question whether the same approach would yield comparable improvements in environments with different dynamics.

Bottleneck Dimensionality The dimensionality of the bottleneck projection was briefly evaluated in Section 4.2.5, but then fixed at 96 throughout all experiments, reducing the DINOv2 features by a factor of four. Whether this specific reduction factor is optimal was not further investigated.

Potentially Suboptimal Parameters A full grid search for hyperparameters was not feasible within the available compute and time budget, as a training run of 10 epochs required approximately 24 hours. The constraint weights and margins were therefore evaluated on a reduced dataset, meaning the reported constraint configurations may not represent the optimal settings, and better-tuned variants could potentially yield different results.

Single-seed Training The models were trained with a single random seed, meaning the reported results may be sensitive to weight initialization. Running multiple training seeds would establish whether the performance differences between model variants are consistent or fall within natural variance.

Planning Variance Similarly, the planning success rates are reported from a single evaluation run. Given that CEM is a stochastic, sampling-based method, running the planning procedure multiple times and reporting mean and standard deviation would provide a more reliable estimate of true task performance. However, informal repeated evaluations suggested variance of approximately 2% between runs, indicating that the relative ordering of model performance is unlikely to change substantially across evaluation runs.

Linearity of Manifold Diagnostics While the diagnostic suite utilized in this work encompasses PCA topology analysis, rollout stability tracking, and linear probing to provide a detailed evaluation of the latent space, it remains inherently limited to evaluating linearly accessible information (Alain and Bengio 2016). Although linear predictability is an appropriate proxy for the capabilities of standard planning heuristics like CEM, the neural manifold may contain highly structured and nonlinear geometric relationships that our current linear metrics and PCA visualizations cannot fully capture. Future research could employ nonlinear manifold estimation techniques to provide a more complete picture of the underlying topology.

6.4 Future Work

Beyond addressing the limitations identified in Section 6.3, the findings of this thesis open several promising directions for future research.

Constraint Finetuning on Extended Encoder Architecture The reported results in Section 6.1.2 suggest that training the encoder and predictor jointly with the local geometric constraint from the start allows the prediction loss to dominate early training, establishing a latent geometry before the constraints can meaningfully shape it. In contrast the results of Section 6.1.3 showed that the global constraint converges very quickly, leaving the vision loss solely dominant. A promising alternative for both constraints could be to first train the Extended Encoder Architecture to convergence, then finetune the encoder with the geometric constraints applied on top of the already established latent structure. This approach could prevent the constraints from disturbing a well-formed manifold and instead allow them to refine it.

Environmental Generalization Evaluating the unconstrained Extended Encoder Architecture on additional environments remains an important next step to assess the generalization of the findings. Testing the Extended Encoder Architecture on other environments would answer this open question. Testing the Local and Global Constraint on environments with smoother action spaces, such as the Maze or Wall tasks, would clarify whether the failure is specific to the dynamics of Push-T. And, introducing more complex environments with 3D dynamics would test the improvements of the unconstrained Extended Encoder Architecture.

Bottleneck Dimensionality The relationship between compression factor and implicit regularization effect remains unexplored. The participation ratios (PR) observed in Figure 5.4 and Figure 5.5 suggest that only a fraction of the 96 projected dimensions are actively used for positional encoding of the T-block, and the decoder reconstructions in Figure 5.2 demonstrate that the compressed latent space already preserves sufficient information for accurate reconstruction. Together these observations indicate that a higher compression factor may be achievable without sacrificing planning performance. This is further supported by Nam et al. (2026), who demonstrated successful planning using only a fraction of the features required by standard patch-based architectures. Future work should systematically vary the projection dimensionality to determine whether stronger compression produces more structured latent spaces, or whether there is an optimal trade-off between information preservation and geometric regularity.

Longer Planning Horizon for Tasks The current evaluation is limited to a 25-step planning horizon on a single manipulation task. As indicated by the Figure 5.12 the roll out stability achieved could be further tested with more complex tasks and longer planning horizons. This would test whether the effect of the Extended Encoder Architecture scales.

Other Foundation Models The implicit regularization observed in this work is specific to a DINOv2-based projection. It remains an open question whether the same effect would emerge when using other frozen vision foundation models.

Appendix A

Appendix

A.1 Local Geometry Constraint: Action-Space Smoothness

A.1.1 Method

Building upon the theoretical need for Lipschitz-like continuity established in Section 2.2.6, our objective is to explicitly constrain the latent predictor to ensure local smoothness. Empirical analysis of the environment (see Figure A.1) demonstrates that the applied action magnitudes vary significantly. The action itself is a 2D vector commanding the target position of the pusher in pixel space, so its magnitude represents the Euclidean distance the agent is commanded to travel in a single step. Under the assumption of roughly proportional dynamics (that larger commanded displacements tend to produce larger visual changes), action magnitude provides a scale for how much latent displacement to permit.

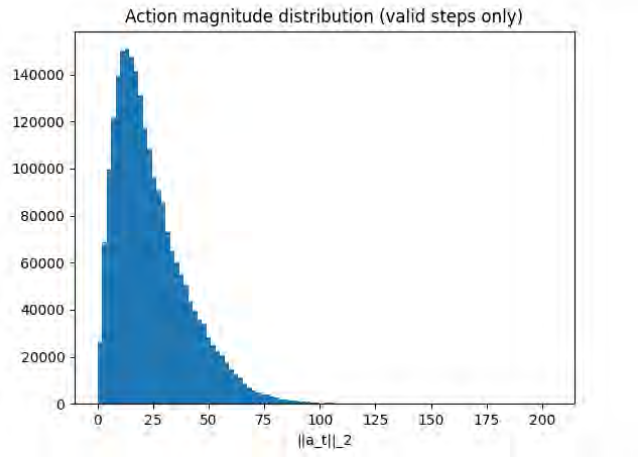

FIGURE A.1: Distribution of action magnitudes ($\|a_t\|_2$) for valid steps in the environment.

We implement the smoothness by explicitly capping how much the latent state is allowed to shift relative to the action applied, enforcing a strict upper bound across a sequence of length T:

$$\mathcal{L}_{\text{smooth}} = \frac{1}{T-1} \sum_{t=1}^{T-1} \max\left(0, \frac{\frac{1}{P}\sum_{p=1}^{P} \|z_{t+1,p} - z_{t,p}\|_2}{\|a_t\|_2 + \varepsilon} - C\right)$$

Where:

- $z_{t,p}$: The encoded latent representation at time t for spatial patch p, while $z_{t+1,p}$ is the representation at $t+1$.
- P: The total number of visual patches in the latent grid ($14 \times 14 = 196$).
- a_t: The action vector applied at time t.
- ε: A small numerical stability constant to prevent division by zero.
- C: The margin threshold, defining the maximum allowable ratio of latent change to action size.

By utilizing the $\max(0, \ldots)$ operation, the network gets penalized only when the ratio of latent displacement to action magnitude explicitly exceeds the predefined threshold C. If the latent change is naturally smaller than the threshold, the loss is zero. This enforces a Lipschitz-like condition between action space and latent space. A small movement should correspond to a small latent displacement, while a large movement is allowed to produce a proportionally larger one.

A.1.2 Results

The Action-Space Smoothness constraint achieves a success rate of 0.40 after a single training epoch, rising to a maximum of 0.64 at epoch 18 (Table A.1) without MPC. It scores a success rate of 0.68 with MPC using epoch 18.

TABLE A.1: Success Rate Comparison between the Action-Space Smoothness Constraint Epochs.

Epoch	**Success Rate**
Epoch 0	0.40
Epoch 5	0.38
Epoch 10	0.54
Epoch 18	0.64

The PCA analysis in Figure A.2 reveals a severe representational collapse, with a participation ratio of 1.9, substantially lower than the unconstrained Extended Encoder Architecture (4.3) and the DINO-WM Baseline (8.9). This indicates that the constraint forces the encoder to concentrate its representations into very few active dimensions.

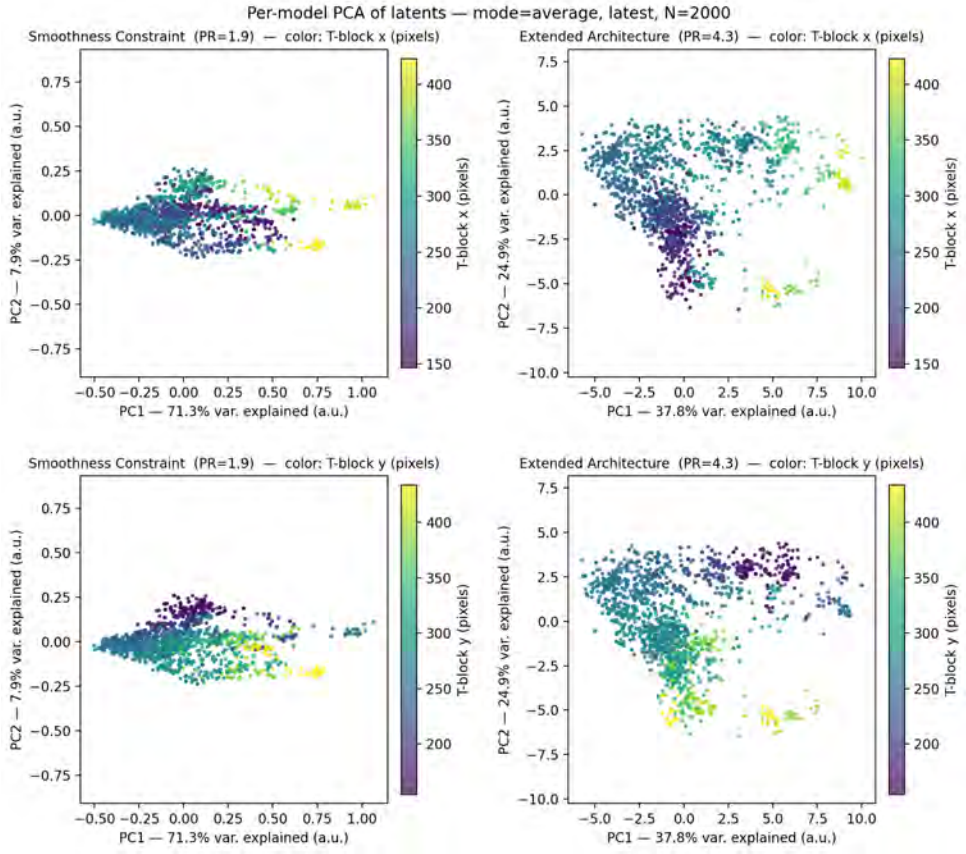

FIGURE A.2: PCA of latent spaces (N=2000) color-coded by the physical T-block position for the Local Action-Space Smoothness constraint (left) and the Extended Encoder Architecture (right).

Furthermore, both the smoothness ratio and action sensitivity fall significantly below the unconstrained Extended Encoder Architecture (as shown in Figure A.3).

Despite the severe representational collapse indicated by the participation ratio of 1.9, the model retains partial planning capability, suggesting the remaining active dimensions still encode some task-relevant structure.

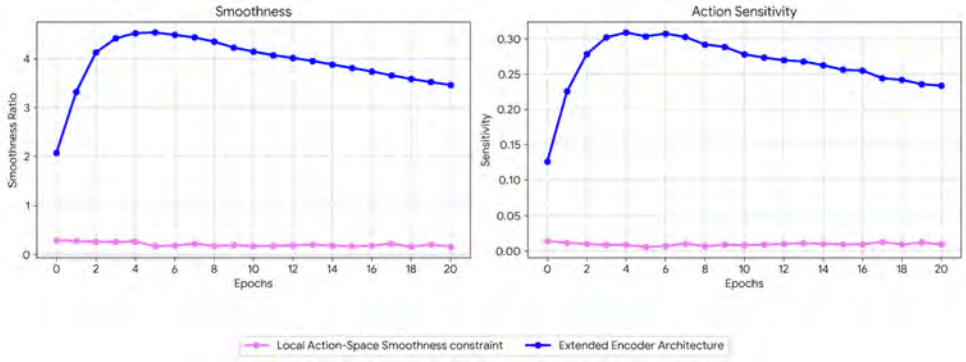

FIGURE A.3: Smoothness (left) and action sensitivity (right) over training epochs, comparing the Local Action-Space Smoothness constraint against the unconstrained Extended Encoder Architecture.

A.1.3 Discussion

The severe representational collapse observed in the PCA analysis A.2 is a direct consequence of the chosen loss formulation. This occurs because the constraint penalizes only the upper bound of the latent displacement ratio, giving the model an unintended shortcut: Rather than learning to calibrate latent transitions to action magnitude, the encoder learns to shrink all latent transitions uniformly, collapsing the manifold in the process.

While the constraint succeeds in reducing the smoothness ratio (as shown in Figure A.3), it does so through collapse, rather than calibration. This can be seen by the action sensitivity falling significantly below the unconstrained Extended Encoder Architecture in Figure A.3. The model produces nearly identical latent predictions regardless of the action taken, rendering it ineffective for planning. CEM relies on the world model being sensitive to action differences. Despite this, the model retains a partial planning capability of 0.64, suggesting the few remaining active dimensions still encode enough task-relevant structure to support basic navigation, but insufficient for reliable planning. This is also supported by the Figure 5.3, which shows that the reconstructions does not contain all the relevant information anymore.

This failure motivated the development of the perturbation-based local constraint presented in Section 3.2.1. This formulation explicitly prevents the collapse shortcut by penalizing insensitivity to action changes, rather than penalizing large transitions.

Appendix B

MPC Rollout Stability

Figure B.1 shows the 1-step prediction error at each MPC planning step. Under MPC, the planner replans from a real observation before each new imagined rollout, so prediction errors don't accumulate. The error is highest at step 1, where the scene is at its most uncertain, then drops sharply and stabilizes as the task converges toward the goal, because the scene becomes more static and CEM selects smaller actions. This pattern reflects decreasing task difficulty rather than model quality, which is why this metric is not used for model comparison in the main evaluation.

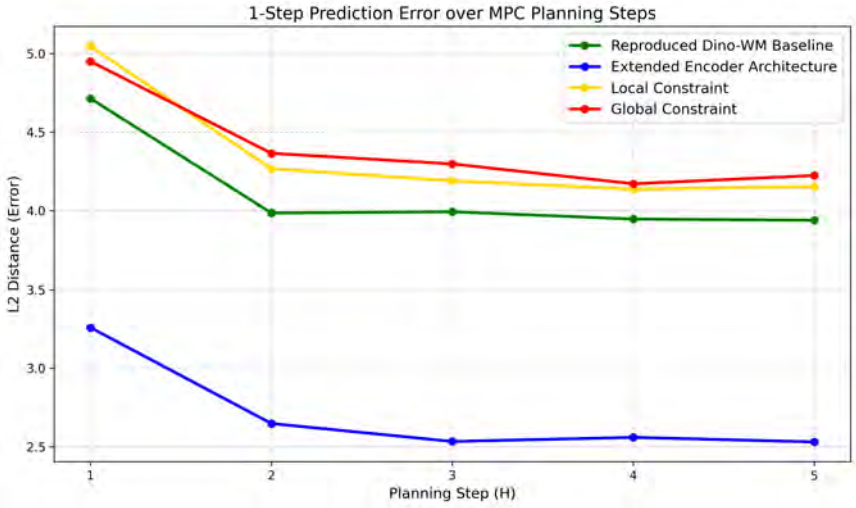

FIGURE B.1: 1-step prediction error (L_2 distance between predicted and real encoded states) at each MPC planning step across all model variants.

Appendix C

Code and Readme Links

Our repository is a fork of **DINO-WM**. The modified repository containing our work can be accessed under the following URL:

`https://github.com/Danbuch/dino-wm`

The summarized structure of the codebase, and in particular, the distinction between the functionality inherited from the original DINO-WM release and the extensions we developed for this thesis is described in our repository Readme:

`https://github.com/Danbuch/dino-wm/blob/main/README.md`

Unless a component is explicitly listed under *Our Contributions*, it is part of the original DINO-WM repository and should be attributed to the original authors Zhou et al. (2025).

The original project documentation is preserved here:

`https://github.com/Danbuch/dino-wm/blob/main/DINOWM_README.md`

Appendix D

Declaration of Generative AI

The following AI-Tools have been used in the process of this bachelors thesis:

- **Anthropic Claude (Chat & Code)**
  - **Coding & Development:** Utilized Claude Code for writing, refactoring, and debugging software components, as well as generating the project's README documentation.
  - **Formulation & Editing:** Utilized Claude Chat to refine textual formulations, perform comprehensive spell checking, and ensure stylistic consistency across chapters.
- **Google Gemini**
  - **Coding & Development:** Employed for code generation, syntax optimization, and programmatic troubleshooting.
  - **Formulation & Editing:** Used for drafting, sentence restructuring, proofreading and improving overall readability.
  - **Research & Literature Exploration:** Assisted in navigating academic topics and discovering relevant scientific literature.
- **Consensus AI**
  - **Research & Literature Exploration:** Utilized as a specialized research engine to identify relevant papers and support the foundational literature review.
- **GPTZero**
  - **Review & Integrity Checks:** Utilized for general paper review, simulating expert-level feedback, and conducting plagiarism and citation verification.
- **DeepL Write**
  - **Grammar & Spellchecking:** Employed for spellchecking, identifying grammatical errors, and refining overall language mechanics.

List of Figures

2.1 The self-distillation framework. The teacher network provides a stable learning target and is updated via an exponential moving average (EMA) of the student network's weights. Adapted from Caron et al. (Caron et al. 2021). 6
2.2 Comparison of world model architectures. Left (a): Generative architectures waste compute reconstructing pixel-level details via a decoder. Right (b): Joint-Embedding Predictive Architectures (JEPA) predict directly within an abstract latent space, bypassing the decoder. Adapted from Assran et al. (2023). 7
2.3 The Manifold Hypothesis in latent spaces. A straight Euclidean line (red) frequently cuts through "empty" latent space, corresponding to physically impossible states. In contrast, the true geodesic path (green) correctly follows the underlying geometric structure of the valid data manifold. Adapted from Arvanitidis et al. (Arvanitidis, Hansen, and Hauberg 2018). 8

3.1 Architecture of the proposed JEPA-augmented DINO-WM. Images x_t and x_{t+1} are processed by frozen DINOv2 encoders to extract high-level features h_t and h_{t+1}. These features are passed into our proposed addition (dashed box), where the Online Encoder (E_θ) and Target Encoder ($E_{\bar{\theta}}$) map them to a latent space, yielding z_t and target embedding z_{t+1}. To prevent representation collapse without requiring negative samples, the $E_{\bar{\theta}}$ is updated via an Exponential Moving Average (EMA) of the E_θ weights, and a stop gradient is applied to its output. The predictor module takes the online embedding z_t, conditioned on the action a_t and proprioception p_t, to output the predicted embedding $\hat{z}_{t+1}$. The network is optimized by minimizing the Euclidean distance between the predicted and target embeddings, alongside our proposed constraints $\lambda\mathcal{L}_{\text{constraints}}$. 15

4.1 Visualization of the Push-T environment task (Chi et al. 2025). The left panel illustrates a start state, while the right panel shows an intermediate state after 25 steps. 21

5.1 Success rate over CEM optimization iterations for each model variant without using MPC. 30
5.2 From left to right, the columns display: (1) The original observation x_t (2) The visual reconstruction decoded directly from the encoder's intermediate representation z_t (3) The visual reconstruction decoded from the model's forward-predicted embedding $\hat{z}_{t+1}$ and (4) The ground truth future observation x_{t+1}. Extended Encoder Architecture (left), Local Constraint (middle), Global Constraint (right). 30

5.3 Decoder reconstructions of the Action-Space Smoothness Constraint, elaborated in Appendix A. 31
5.4 Per-model Principal Component Analysis (PCA) of latent spaces ($N = 2000$) color-coded by the physical T-block position. The left column displays the DINO-WM baseline, and the right column displays the Extended Encoder Architecture model. The top row maps spatial x-coordinates (pixels), while the bottom row maps spatial y-coordinates (pixels). 32
5.5 Per-model Principal Component Analysis (PCA) of constrained latent spaces ($N = 2000$) color-coded by physical T-block coordinates. The left column displays the Local Perturbation Constraint model, and the right column displays the Global Geodesic Constraint model. The top row maps spatial x-coordinates (pixels), while the bottom row maps spatial y-coordinates (pixels). 32
5.6 Global geometry probe comparing state distance (pixels) to latent distance across random validation frame pairs. Red lines indicate the linear regression fit. Panel titles report Pearson (r) and Spearman (ρ) correlation coefficients alongside the explained variance (R^2). 33
5.7 Local geometry probe, comparing the mean physical state distance of $k = 10$ latent nearest neighbors (blue) against a randomly sampled reference (grey). Histograms display the distance distribution across query frames, with vertical lines marking the distribution means. Panel titles report the ratio of these means ($\mu_{k-NN} / \mu_{random}$). 33
5.8 Full pose linear probe, displaying a heatmap of R^2 values for per-coordinate spatial and rotational retention across model variants. . . . 34
5.9 Orientation probe scatter plots comparing the true physical angle θ (in radians) against ridge regression predictions from the latent embeddings. Red lines indicate perfect prediction. 34
5.10 Temporal leakage probe scatter plots comparing the true normalized frame position within an episode to the predicted position. Red lines indicate perfect prediction. Panel titles report R^2 and Mean Absolute Error (MAE). 34
5.11 Inverse dynamics probe scatter plots predicting the applied action dimensions from consecutive latent transitions. The top row displays the horizontal target displacement (Δx), while the bottom row displays the vertical target displacement (Δy). True action values are plotted against predicted values, with red lines indicating perfect prediction. . 35
5.12 L_2 distance between predicted and real encoded states over the 5 planning steps (25 environment steps at frameskip 5) without using MPC. Step 0 reflects the gap between online and target encoder representations. Steps 1–5 reflect true prediction error, where a lower value indicates better rollout stability. 36
5.13 Validation loss curves for the Reproduced DINO-WM Baseline and the Extended Encoder Architecture training runs. 36
5.14 Smoothness ratio, perturbation ratio and action sensitivity over training epochs for the Reproduced DINO-WM Baseline and the Extended Encoder Architecture, illustrating the dynamics between a frozen and a trainable encoder. 37

5.15 Validation vision loss (left) and validation local perturbation loss (right) over training epochs for the Local Constraint model. The perturbation loss magnitude is consistently larger than the vision loss throughout training. 37
5.16 Smoothness (left), action sensitivity (middle), and perturbation sensitivity (right) over training epochs, comparing the Local Constraint model against the unconstrained Extended Encoder Architecture. . . . 38
5.17 Mean raw Euclidean latent distance $\|z_t - z_{t+k}\|$ plotted against temporal gap k (slice-steps) for validation trajectory slices. The shaded bands show a standard error interval ($\frac{std}{\sqrt{N_{\text{samples}}}}$), and the vertical dotted line marks the alignment boundary ($N = 5$) for the global constraint. . . . 39
5.18 Normalized latent distance shape comparison against temporal gap k. Each curve is normalized by its relative baseline scale at $k = 1$, evaluating the geometric expansion factor of sequential frames relative to adjacent transitions. 40
5.19 Left plot shows the Global Constraint MSE loss, center the temporal triplet loss and right is the combined loss together with proprioceptive loss (not included) 40

A.1 Distribution of action magnitudes ($\|a_t\|_2$) for valid steps in the environment. 48
A.2 PCA of latent spaces (N=2000) color-coded by the physical T-block position for the Local Action-Space Smoothness constraint (left) and the Extended Encoder Architecture (right). 50
A.3 Smoothness (left) and action sensitivity (right) over training epochs, comparing the Local Action-Space Smoothness constraint against the unconstrained Extended Encoder Architecture. 50

B.1 1-step prediction error (L_2 distance between predicted and real encoded states) at each MPC planning step across all model variants. . . 52

List of Tables

4.1 Environment-specific hyperparameters for the CEM planner on the Push-T task. 23
4.2 Core training hyperparameters for the JEPA architecture and predictive world model. 24

5.1 Success Rate Comparison between the Models with MPC and open-loop. 29
5.2 Success Rate Comparison between the Local Constraint Epochs. 37
5.3 Temporal Distance Absolute Values 38
5.4 Temporal Distance Normalized Values 39

A.1 Success Rate Comparison between the Action-Space Smoothness Constraint Epochs. 49

List of Abbreviations

AI	**A**rtificial **I**ntelligence
AE	**A**uto**e**ncoder
BYOL	**B**ootstrap **Y**our **O**wn **L**atent
CEM	**C**ross-**E**ntropy **M**ethod
DINO-WM	**DINO W**orld **M**odel
DML	**D**eep **M**etric **L**earning
EMA	**E**xponential **M**oving **A**verage
GELU	**G**aussian **E**rror **L**inear **U**nit
GD	**G**radient **D**escent
JEPA	**J**oint-**E**mbedding **P**redictive **A**rchitecture
k-NN	**k N**earest **N**eighbors
LLM	**L**arge **L**anguage **M**odel
MAE	**M**ean **A**bsolute **E**rror
MPC	**M**odel **P**redictive **C**ontrol
MSE	**M**ean **S**quared **E**rror
PCA	**P**rincipal **C**omponent **A**nalysis
PR	**P**articipation **R**atio
SGD	**S**tochastic **G**radient **D**escent
SSL	**S**elf-**S**upervised **L**earning
VICReg	**V**ariance-**I**nvariance-**C**ovariance **R**egularization
ViT	**Vi**sion **T**ransformer
VGG	**V**isual **G**eometry **G**roup
WM	**W**orld **M**odel

Bibliography

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama (2019). "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *KDD* (cit. on p. 24).

Alain, Guillaume and Yoshua Bengio (2016). "Understanding Intermediate Layers Using Linear Classifier Probes". In: *arXiv preprint arXiv:1610.01644* (cit. on pp. 13, 46).

Arvanitidis, Georgios, Lars Kai Hansen, and Søren Hauberg (2018). "Latent Space Oddity: On the Curvature of Deep Generative Models". In: *ICLR* (cit. on pp. 2, 7–9, 11, 13).

Asadi, Kavosh, Dipendra Misra, and Michael Littman (2018). "Lipschitz Continuity in Model-Based Reinforcement Learning". In: *ICML* (cit. on pp. 11, 13, 17).

Assran, Mahmoud, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas (2023). "Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture". In: *CVPR* (cit. on pp. 1, 7, 10, 13, 16, 43).

Bardes, Adrien, Jean Ponce, and Yann LeCun (2022). "VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning". In: *ICLR* (cit. on pp. 11, 16).

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (cit. on pp. 1, 5, 7).

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). "Language Models are Few-Shot Learners". In: *NeurIPS* (cit. on p. 5).

Caron, Mathilde, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin (2021). "Emerging Properties in Self-Supervised Vision Transformers". In: *ICCV* (cit. on pp. 5, 6, 12).

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2020). "A Simple Framework for Contrastive Learning of Visual Representations". In: *ICML* (cit. on pp. 5, 6).

Chen, Xinlei, Saining Xie, and Kaiming He (2021). "An Empirical Study of Training Self-Supervised Vision Transformers". In: *ICCV* (cit. on pp. 6, 22, 42).

Chi, Cheng, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song (2025). "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion". In: *International Journal of Robotics Research* 44 (cit. on pp. 2, 21).

Chua, Kurtland, Roberto Calandra, Rowan McAllister, and Sergey Levine (2018). "Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models". In: *NeurIPS*. Vol. 31 (cit. on pp. 2, 8, 10, 11, 17).

Cover, Thomas and Peter Hart (1967). "Nearest Neighbor Pattern Classification". In: *IEEE Transactions on Information Theory* 13.1 (cit. on p. 27).

Craik, Kenneth James Williams (1967). *The Nature of Explanation*. Vol. 445. CUP Archive (cit. on p. 1).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *ICLR* (cit. on pp. 5, 15).

Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko (2020). "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning". In: *NeurIPS* (cit. on pp. 6, 12, 16, 22).

Ha, David and Jürgen Schmidhuber (2018). "Recurrent World Models Facilitate Policy Evolution". In: *NeurIPS* (cit. on pp. 1, 6, 9).

Hadsell, Raia, Sumit Chopra, and Yann LeCun (2006). "Dimensionality Reduction by Learning an Invariant Mapping". In: *CVPR*. Vol. 2 (cit. on p. 6).

Hafner, Danijar, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (2019). "Learning Latent Dynamics for Planning from Pixels". In: *ICML* (cit. on p. 17).

Hafner, Danijar, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi (2020). "Dream to Control: Learning Behaviors by Latent Imagination". In: *ICLR* (cit. on pp. 1, 6, 8).

Hamrick, Jessica B (2019). "Analogues of Mental Simulation and Imagination in Deep Learning". In: *Current Opinion in Behavioral Sciences* 29 (cit. on p. 1).

He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick (2022). "Masked Autoencoders Are Scalable Vision Learners". In: *CVPR* (cit. on p. 5).

Hendrycks, Dan and Kevin Gimpel (2016). "Gaussian Error Linear Units (GELUs)". In: *arXiv preprint arXiv:1606.08415* (cit. on p. 15).

Hermans, Alexander, Lucas Beyer, and Bastian Leibe (2017). "In Defense of the Triplet Loss for Person Re-Identification". In: *arXiv preprint arXiv:1703.07737* (cit. on p. 16).

Hoerl, Arthur and Robert Kennard (1970). "Ridge Regression: Biased Estimation for Nonorthogonal Problems". In: *Technometrics* 12.1 (cit. on p. 27).

Hoffer, Elad, Itay Hubara, and Daniel Soudry (2017). "Train Longer, Generalize Better: Closing the Generalization Gap in Large Batch Training of Neural Networks". In: *NeurIPS* (cit. on p. 24).

Huang, Yuhang, Jiazhao Zhang, Shilong Zou, Xinwang Liu, Ruizhen Hu, and Kai Xu (2025). "LaDi-WM: A Latent Diffusion-based World Model for Predictive Manipulation". In: *arXiv preprint arXiv:2505.11528* (cit. on p. 10).

Johnson, Justin, Alexandre Alahi, and Li Fei-Fei (2016). "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *ECCV* (cit. on p. 26).

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *ICLR*. Ed. by Yoshua Bengio and Yann LeCun (cit. on p. 24).

Krizhevsky, Alex (2014). "One Weird Trick for Parallelizing Convolutional Neural Networks". In: *arXiv preprint arXiv:1404.5997* (cit. on p. 24).

LeCun, Yann (2022). "A Path Towards Autonomous Machine Intelligence". In: *OpenReview* (cit. on pp. 1, 2, 5, 6, 10, 12, 14, 43).

Li, Wenyuan, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama (2026). "Predictive but Not Plannable: RC-aux for Latent World Models". In: *arXiv preprint arXiv:2605.07278* (cit. on p. 10).

Loshchilov, Ilya and Frank Hutter (2019). "Decoupled Weight Decay Regularization". In: *ICLR* (cit. on pp. 23, 24).

Ma, Yecheng Jason, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang (2023). "VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training". In: *ICLR* (cit. on p. 1).

Maes, Lucas, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero (2026). "LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels". In: *arXiv preprint arXiv:2603.19312* (cit. on pp. 2, 10).

Morales-Brotons, Daniel, Thijs Vogels, and Hadrien Hendrikx (2024). "Exponential Moving Average of Weights in Deep Learning: Dynamics and Benefits". In: *arXiv preprint arXiv:2411.18704* (cit. on p. 16).

Myers, Vivek, Chongyi Zheng, Anca D. Dragan, Sergey Levine, and Benjamin Eysenbach (2024). "Learning Temporal Distances: Contrastive Successor Features Can Provide a Metric Structure for Decision-Making". In: *ICML* (cit. on pp. 11, 18, 43).

Nagabandi, Anusha, Gregory Kahn, Ronald S Fearing, and Sergey Levine (2018). "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning". In: *ICRA* (cit. on p. 10).

Nam, Heejeong, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero (2026). "Causal-JEPA: Learning World Models through Object-Level Latent Interventions". In: *arXiv preprint arXiv:2602.11389* (cit. on pp. 10, 12, 47).

Oquab, Maxime, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski (2024). "DINOv2: Learning Robust Visual Features without Supervision". In: *Transactions on Machine Learning Research* 2024 (cit. on pp. 1, 2, 5, 9, 14, 24).

Pathak, Deepak, Pulkit Agrawal, Alexei Efros, and Trevor Darrell (2017). "Curiosity-Driven Exploration by Self-Supervised Prediction". In: *ICML* (cit. on p. 28).

Rubinstein, Reuven (1999). "The Cross-Entropy Method for Combinatorial and Continuous Optimization". In: *Methodology and Computing in Applied Probability* 1.2 (cit. on pp. 2, 8).

Schmidhuber, Jürgen (1992). "Learning Factorial Codes by Predictability Minimization". In: *Neural Computation* 4.6 (cit. on pp. 1, 6).

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *CVPR* (cit. on p. 16).

Sobal, Vlad, Wancong Zhang, Kynghyun Cho, Randall Balestriero, Tim Rudner, and Yann LeCun (2025). "Learning from Reward-Free Offline Data: A Case for Planning with Latent Dynamics Models". In: *arXiv preprint arXiv:2502.14819* (cit. on p. 12).

Tarvainen, Antti and Harri Valpola (2017). "Mean Teachers Are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Deep Learning Results". In: *NeurIPS*. Vol. 30 (cit. on pp. 6, 11).

Terver, Basile, Randall Balestriero, Megi Dervishi, David Fan, Quentin Garrido, Tushar Nagarajan, Koustuv Sinha, Wancong Zhang, Mike Rabbat, Yann LeCun, and Amir Bar (2026). "A Lightweight Library for Energy-Based Joint-Embedding Predictive Architectures". In: *arXiv preprint arXiv:2602.03604* (cit. on p. 12).

Terver, Basile, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun (2025). "What Drives Success in Physical Planning with Joint-Embedding Predictive World Models?" In: *arXiv preprint arXiv:2512.24497* (cit. on pp. 10, 12, 13).

Toso, Leonardo F, Davit Shadunts, Yunyang Lu, Nihal Sharma, Donglin Zhan, Nam H Nguyen, and James Anderson (2026). "Learning Invariant Visual Representations for Planning with Joint-Embedding Predictive World Models". In: *arXiv preprint arXiv:2602.18639* (cit. on pp. 2, 10).

Wang, Ying, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren (2026). "Temporal Straightening for Latent Planning". In: *arXiv preprint arXiv:2603.12231* (cit. on pp. 2, 11, 18, 19, 43).

Zhang, Zeyu, Danning Li, Ian D. Reid, and Richard I. Hartley (2026). "GeoWorld: Geometric World Models". In: *arXiv preprint arXiv:2602.23058* (cit. on pp. 10, 12, 13).

Zhang, Zhiwei, Hui Zhang, Kaihong Huang, Chenghao Shi, and Huimin Lu (2025). "Efficient Image-Goal Navigation with Representative Latent World Model". In: *arXiv preprint arXiv:2511.11011* (cit. on p. 12).

Zhou, Gaoyue, Hengkai Pan, Yann LeCun, and Lerrel Pinto (2025). "DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning". In: *ICML* (cit. on pp. 9, 10, 12, 14, 18, 20, 21, 23, 26, 53).