Cylindrical Lattice Embedding for Program Induction

Jinseo Shim(jinseo5892@gm.gist.ac.kr)
GIST(Gwanju Institute of Science and Technology)
데이터사이언스연구실 GIST Data Science Lab

AAAI-26 / IAAI-26 / EAAI-26

Introduction and Prior Work

Program Induction

Frameworks designed to learn programs from incomplete specification

Rather than manual coding by a human programmer, the AI induces the underlying logic required to transform given inputs into target outputs.

Compositional Reasoning

The process of solving new problems by systematically composing primitive operations into complex transformation.

Models must learn primitive interactions, perform inductive generalization, and refine complex compositional structure

Prior Work: PLAE (Program Lattice Auto Encoder)

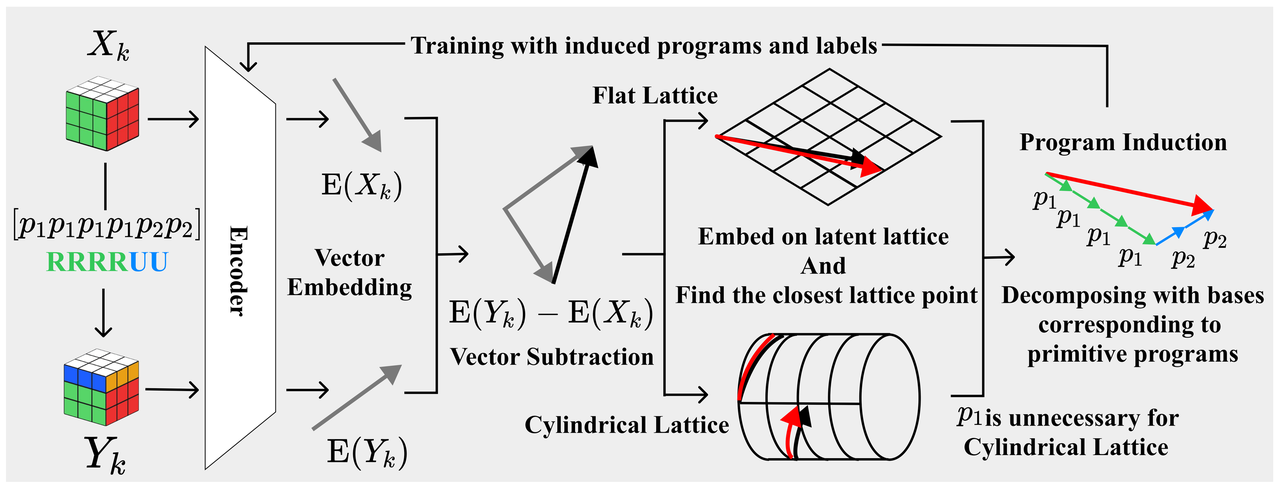
It embeds observations as vectors within a structured latent space, called a *program lattice*, **capturing geometric nature of compositional reasoning**

In program lattice, primitive programs acts as basis vectors, allowing any composed program to be expressed as a linear combination of bases

Program Induction Process

1. Inferrence: Mapping task data to a corresponding vector within the lattice space
2. Identification: Finding the closest lattice point to the inferred vector(CVP)
3. Decomposition: Expressing the resulting lattice point vector as a linear combination of bases vector which correspond to primitive programs

Limitation and Key Idea: Cylindrical Lattice Embedding



PLAE model mechanism on a flat program lattice(above) and a cylindrical lattice(below)

Limitation of PLAE

Since PLAE relies on a fixed, flat lattice structure, it can be suboptimal for certain problem domains. Specifically, invariant program-**sequences where actions return the system to its initial state**- cannot be effectively represented on a flat lattice.

This **"invariant program problem"** forces the model to generate redundant, longer program sequences, which reduces computational efficiency.

Approach

While preserving the properties of a flat lattice, I propose mapping the latent program space onto curved manifolds, specifically a **cylindrical manifold.**

Closed-Loop Representation: This geometry naturally represents invariant programs as closed loops that return to the initial state.

Interested in my work? Scan here for my portfolio and contact information.

Geometric Definition and Justification

Isometric Transformation

$$\forall a, b \in X,\ \ d_X(a,b) = d_Y(f(a), f(b))$$

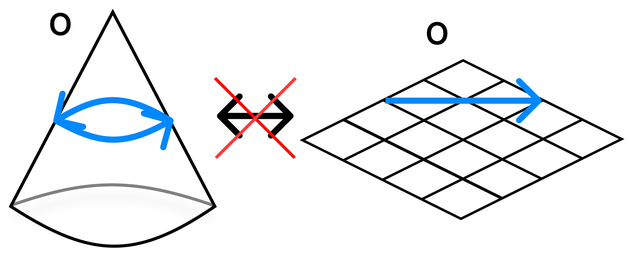
Isometric transformations are **distance-preserving mappings** between manifolds.

To maintain the compositionality of the lattice (where programs are linear combinations of basis vectors), the embedding manifolds must be isometric to the flat Euclidean lattice, called **developable surfaces.**

Manifold	Principal Curvatures (k_1, k_2)	$\mathbf{K}(k_1 * k_2)$
Plane	$k_1 = 0, \quad k_2 = 0$	0
Cylinder	$k_1 = 0, \quad k_2 = 1/R$	0
Cone	$k_1 = 0, \quad k_2 > 0$ (variable)	0^*
Tangent Dev.	$k_1 = 0, \quad k_2 \geq 0$ (variable)	0

Types of Developable Surfaces

Why a cylinder? (vs. a cone)

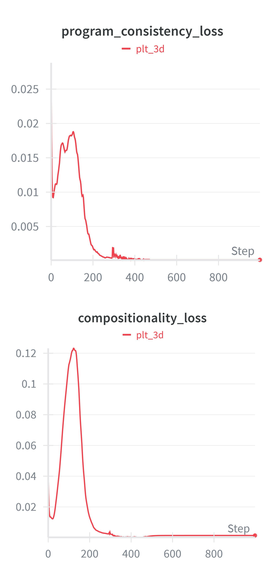


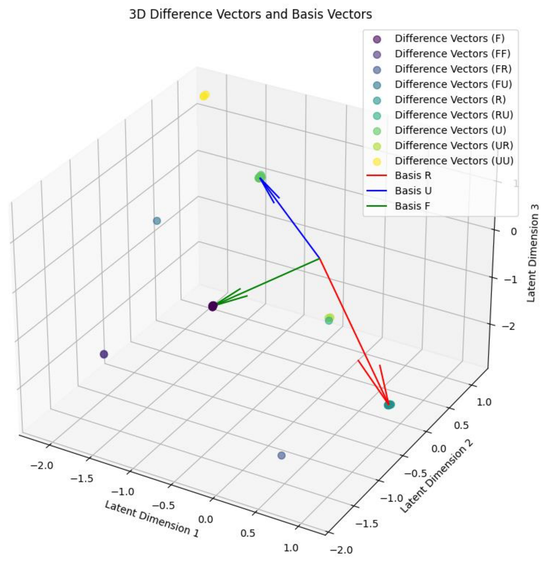
Topology: On a conical manifold, straight lines (geodesics) from a flat plane map to spirals rather than closed loops. Consequently, a cone lacks the necessary topology to map linear invariant sequences into periodic closed loops without disrupting the lattice structure.

Proof Concept: On a cone with apex O, any closed loop maintained at a constant distance from O cannot be expressed as a linear combination of program bases.

Implementation and Experimental Plan

Immediate Experimental Results: Conducted toy experiments using Rubik's Cube permutations.





Expected Findings: Comparative analysis highlights that cylindrical geometry effectively abstracts data with inherent invariants, improving sample efficiency in compositional program induction.

Loss Functions: Developed consistency loss and compositionality loss to enforce the geometric properties.

$$L_{\text{consist}} = \sum^{n} \{(E(y_i) - E(x_i)) - \bar{\Delta}\}^2 = \sum^{n} (\Delta_i - \bar{\Delta})^2$$

$$L_{\text{comp}} = \sum^{n} \{(E(y_i) - E(x_i)) - Bz\}^2 = \sum^{n} (\Delta_i - p^*)^2$$

(x_i, y_i) is input and output pairs

$E(x_i)$ and $E(y_i)$ are embedding vectors

$B = [p_1, p_2, \ldots, p_d] \quad z = [z_1 z_2 \ldots z_d]^T$ matrix consisting of bases vector and integer program sequence

Generalization Bottleneck: Despite training on primitive programs (e.g., R, U, F) and binary compositions (e.g., RU), the original PLAE fails to generalize to unseen sequences involving three or more primitives (e.g., RUF). This identifies a systemic extrapolation hurdle in lattice-based induction.

Future works and Research Questions

Non-Euclidean embedding spaces for compositional reasoning

On developable surfaces, program composition is represented by the vector addition of embedding vectors within the latent program lattice.
However, alternative representations of composition, such as rotations, can preserve compositionality on non-Euclidean (non-developable) manifolds.

Lattice structure within Spherical Manifolds: $\text{Lattice } L = \{g \cdot p_0 | g \in G\}$

$\text{SO(N) Group: } R = e^{i\theta L}$ where L is generator / $G = \{R_1^a R_2^b \ldots | a, b, \ldots \in \mathbb{Z}\} \subset SO(3)$

Lattice structure within Concical Manifolds: $\phi_m = m \cdot \frac{2\pi}{M} \quad \text{for } m \in \{0, 1, 2, \cdots, M-1\}$

$z_n = n \cdot d_z \quad \text{for } n \in \{0, 1, 2, \cdots, N\}$ with a constrain $z = \frac{\rho}{\tan \alpha}$

Research Questions and Future Interests

The current version of the **PLAE model relies on a fixed cylindrical embedding space.**
In the future, I aim to develop a model capable of **self-refining its latent embedding space** based on the properties of the input data for more robust compositional reasoning.
To achieve these goals, the following research questions are essential to address:

What is the underlying structure of compositional reasoning spanned by program primitives? What properties must this structure satisfy—invariance, invertibility, or equivariance?

How can we design a model that dynamically recognizes and manipulates its own program embedding space? Are there established architectures or prior works that support such meta-refinement?

Which mathematical frameworks are most suitable for formalizing these concepts? (e.g., Lie group discovery, geometric deep learning, or manifold learning)