

Unveiling the Feature Geometry of Robot Foundation Models via Nyström NCut

Ningze Zhong
University of Pennsylvania
`zhong666@seas.upenn.edu`

Kyle Zhang
University of Pennsylvania
`kyle100@seas.upenn.edu`

Ily Rafaeli
University of Pennsylvania
`ilyr@seas.upenn.edu`

Content

1 What is a VLA?

2 Nyström NCut

3 VLA Encoder Extraction

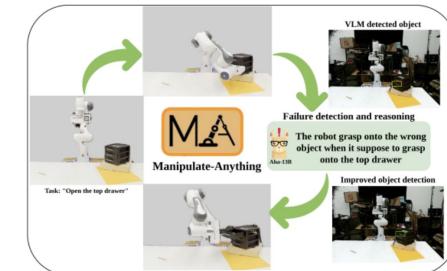
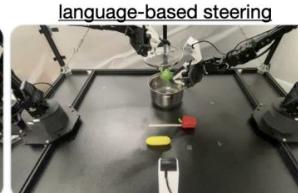
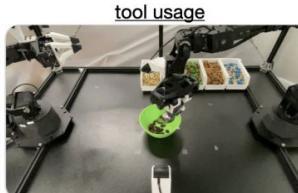
4 Dataset Preparation

5 Findings

6 Future Work

1 What is a VLA?

- VLA: Vision-Language-Action model
- Uses the latent space of an LLM to comprehend generalised robot tasks
- Allows robots to be instructed with words, and to produce lingual outputs
- Can be combined with backend ML model for correction of failed actions [1][2]
- We focus on exploring OpenVLA and OpenPi 0.5 [3][4]



2 Nyström NCut – Our Use Case

Problem:

VLAs are Black Boxes
– hard to debug!

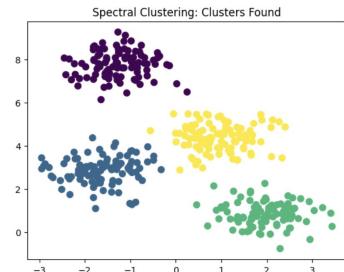
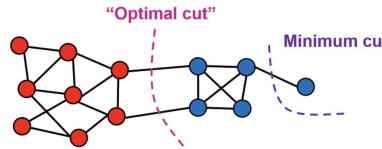
Proposition:

Use NCut to visualise
hidden layers [5]

Result:

Humans may infer the
reasoning & semantic
understandings of VLA

- What might we find?
- Visual clustering of image in latent space:
 - What objects are recognised? How are they segmented?
 - Where is the VLA paying attention?
- Visual clustering of language in latent space:
 - How do words in the instructions related to the environment which the robot sees?



2 Nyström NCut – How to do visual debugging

Index - Nyström Normalized Cuts PyTorch

Nyström Normalized Cuts PyTorch

Overview

What is Ncut?

How Fast is Ncut?

Tutorials

Tutorial 1: Color Visualization

Tutorial 2: Discrete NCut

Tutorial 3: Hierarchy in NCut

How Ncut Works

Basic Ncut

Nyström Ncut (Complexity)

Nyström Ncut (Quality)

k-way Ncut

M-space Coloring

Tutorials (advanced)

Feature Alignment

Custom Model

Inference and Extrapolation

Tutorials (misscs)

Gradient Computation

Channel Tracing and Attribution

Mixing Resolutions

Usage Examples

Model Gallery

Out-of-Distribution Adaptation

Blogs

API References

Methods

INTRODUCTION

This documentation series guides you through the mathematical foundations and practical implementations of Normalized Cuts. You will learn about exact methods for small graphs, linear-time approximations for million-scale datasets, and specialized techniques for feature alignment and visualization.

What you'll learn

- Spectral Clustering:** The math behind Normalized Cuts.
- Scalability:** How to handle large datasets efficiently with Nyström approximation.
- Analyze Quality:** How balanced sampling improve segmentation quality.
- Advanced Partitioning:** Multi-way cuts and discrete optimization.
- Visualization:** Tools for interpreting high-dimensional embeddings.

Core Modules

1. Basic NCut

Master the foundational implementation of Normalized Cuts and spectral clustering. Understand the graph Laplacian, eigenvector computations, and the core mathematics behind the algorithm.

[Start learning →](#)

Feature Alignment

TL;DR
Problem: Feature spaces from different models or layers are incompatible. Solution: Convert absolute features to relative affinity features ($A_{ij} = \text{softmax}(x_i, x_j)$). Key Insight: Use **BF** Auto-scaling to dynamically tune σ so the affinity matrix has a constant mean density, making the alignment robust to scaling and outliers.

Motivation

Deep learning models (e.g., CLIP, DINO, Stable Diffusion) learn robust visual concepts, but represent them in non-interpretable embedding spaces. Even within the same model, different layers operate in distinct coordinate systems. Direct comparison (e.g., $\|x_i - x_j\|_2$) distance between these spaces is mathematically meaningless.

Feature Alignment maps these disjoint representations into a common space where semantic correspondence is preserved. This enables cross-model visualization, layer-wise analysis, and valid distance computation.

Inspired by [Representational Similarity Analysis \(RSA\)](#), we leverage the insight that while absolute coordinates vary, the relative geometry of concepts remains consistent.

Quick Start

```
from ncut import predictor
import torch

# Initialize the predictor with a specific model configuration
predictor = NcutoffPredictor(model="clip", device="cuda:0")

# Load an image
img = Image.open("Image/clip/00000000000000000000000000000000.jpg")

# Create a tensor image
img_t = torch.tensor(img).permute(2, 0, 1).float()

# Compute the aligned relative space
aligned_t = predictor.refAffinity(img_t)

# Now valid to compare aligned_A and aligned_B using standard metrics
```

Results

Alignment reveals consistent semantic structures across models.

Case Study: SAM vs. DINO

SAM (Segment Anything Model) focuses on boundaries and edges. DINO captures high-level semantic parts.



Tutorial 2: Discrete NCut

This tutorial explores the Discrete Normalized Cut (DNCut) approach using the [Ncutoff](#) algorithm. Unlike standard NCut, which provides continuous eigenvectors representations, Discrete NCut explicitly partitions data into distinct categories. By converting continuous eigenvectors into binary labels, the method facilitates the segmentation of images into semantically meaningful regions.

Quick Start

The following snippet demonstrates how to perform discrete segmentation using the [NcutoffPredictor](#):

```
from ncut import predictor
import NcutoffPredictor

# Initialize the predictor with a specific model configuration
predictor = NcutoffPredictor(model="clip", device="cuda:0")

# Load an image
img = Image.open("Image/clip/00000000000000000000000000000000.jpg")

# Create a colored visualization of the segmentation (with borders)
aligned_t = predictor.refAffinity(img_t)

# Save the result
aligned_t.save("aligned.jpg")
```

Understanding K-way NCut Segmentation

The visualizations below illustrate the results of applying a very NCut to features extracted from a DINO v2 (left) and SAM (right). The layout presents the original image, the discrete NCut segmentation, and the clusters.

The choice of K (the number of clusters) significantly impacts the segmentation granularity:

- Large K :** Results in few segmentation, capturing both but potentially excluding many overlapping objects.
- Smaller K :** Produces many segments, merging distinct areas into broader regions.
- Postural Decoding:** You may observe background segmentation patterns; these are often artifacts of the DINO architecture's postural decoding.

Use the slider below to observe how the segmentation evolves with different values of K .



The Role of K

As demonstrated above, selecting an appropriate K is a trade-off between detail and interpretability. A very small K leads to a few large clusters, which may capture semantically coherent regions or objects, avoiding both the over-segmentation of textures and the under-segmentation of distinct object details.

Intermediate Outputs and Implementation Details

For a deeper understanding of the process, we can examine the intermediate outputs, specifically the transition from continuous eigenvectors to discrete clusters.

[View code for implementation code](#)

Visualization Before vs. After NCut

The panels below compare the raw eigenvectors from the standard NCut algorithm with the [axis-aligned projection](#) obtained after K -way NCut.

Axis-aligned projection (original): The eigenvectors show smooth, continuous variations. Each eigenvector highlights specific textures or structures (often noisy), while other ones capture higher frequency details.

After NCut: The axis-aligned projection after K -way NCut transforms these vectors into more discrete, unimodal representations. Each channel tends to highlight a specific cluster (e.g., a face or a background region), making the results significantly sharper and easier to visualize.

[View code](#)

[After NCut](#)

Before NCut (original eigenvectors)



The first row is thematically near-constant, despite more noise/higher frequency information.



2 Nyström NCut – How to do visual debugging

0.Assumption

The easier Ncut can separate the clusters, it will be easier for the downstream models to learn

[NCut_for_Visual_Debugging/README.md at main · NZ-Liam-Zhong/NCut_for_Visual_Debugging](#)

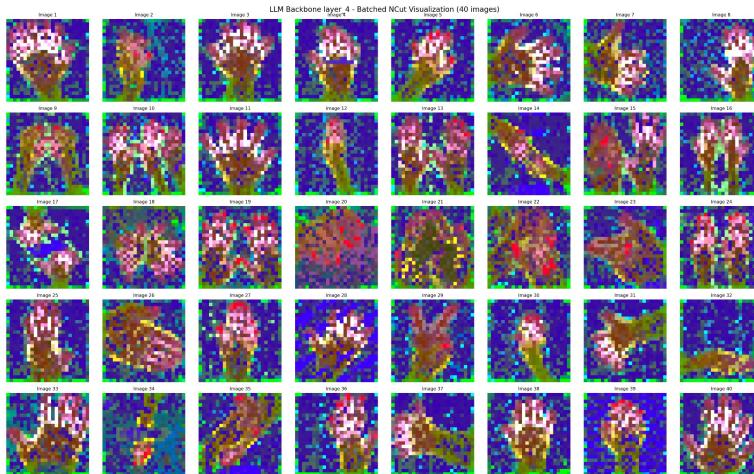
2 Nyström NCut – How to do visual debugging

1. Curate a batch of similar photos that expose the suspected failure mode, then run batched Nyström NCut on that batch. For example, if LLaVA miscounts fingers, collect hands with different finger counts and segment them together; inspect how cluster colors change under these controlled variations.



2 Nyström NCut – How to do visual debugging

2. Analyze per layer with the entire batch side by side. If a small perturbation in one photo causes color shifts only in that case, the model is highly sensitive to that factor. If colors stay stable across the batch despite perturbations, the model is insensitive to that semantic cue. First, we can visualize the results of layer 4 of vision-language backbone of LLaVA (the text tokens can get information from the image tokens in every single layer of vision-language backbone, not necessarily only the last feature).



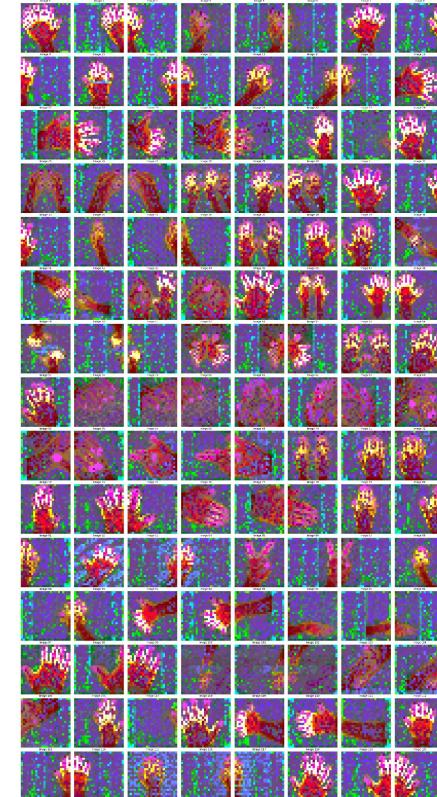
Row \ Col	1	2	3	4	5	6	7	8
01-08	5 Fingers	4 Extended	5 Fingers	5 Fingers	5 Fingers	5 Fingers	5 Fingers	5 Fingers
09-16	Unknown	16 Fingers	5 Fingers	4 Upwards	10 (2 Hands)	Unknown	5 Fingers	10 Total
17-24	10 Total	12 Visible	10 Fingers	10 Fingers	4 Visible	10 Visible	4 Extended	10 (2 Hands)
25-32	5 Fingers	5 Fingers	5 Fingers	5 Fingers	4 Fingers	5 Fingers	5 Fingers	4 Extended
33-40	5 Fingers	4 Fingers	10 (Clasped)	5 Fingers	5 Fingers	5 Fingers	5 Fingers	5 Fingers

2 Nyström NCut – How to do visual debugging

3. Trace across layers (and submodules when the model has multiple parts). For multi-branch systems like LLaVA, separately inspect the vision tower and the vision-language backbone to localize which component fails to preserve the relevant structure.

Finger numbers predicted by AI

Row \ Col	1	2	3	4	5	6	7	8
01-08	5 Fingers	4 Extended	5 Fingers	5 Fingers	5 Fingers	5 Fingers	5 Fingers	5 Fingers
09-16	Unknown	16 Fingers	5 Fingers	4 Upwards	10 (2 Hands)	Unknown	5 Fingers	10 Total
17-24	10 Total	12 Visible	10 Fingers	10 Fingers	4 Visible	10 Visible	4 Extended	10 (2 Hands)
25-32	5 Fingers	5 Fingers	5 Fingers	5 Fingers	4 Fingers	5 Fingers	5 Fingers	4 Extended
33-40	5 Fingers	4 Fingers	10 (Clasped)	5 Fingers	5 Fingers	5 Fingers	5 Fingers	5 Fingers

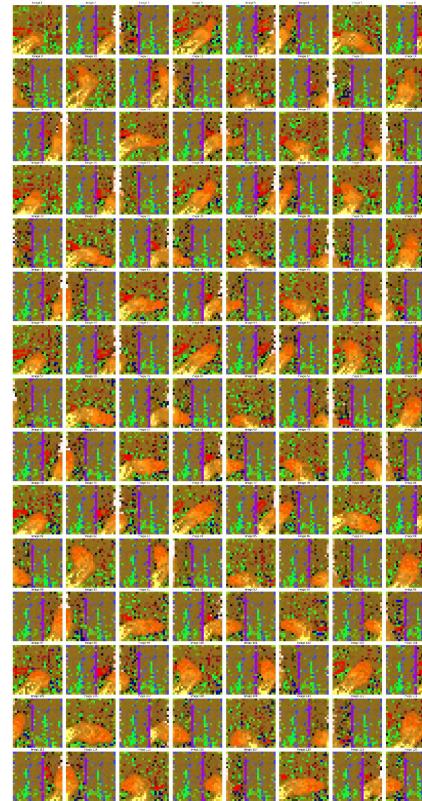


2 Nyström NCut – How to do visual debugging

Other Examples:

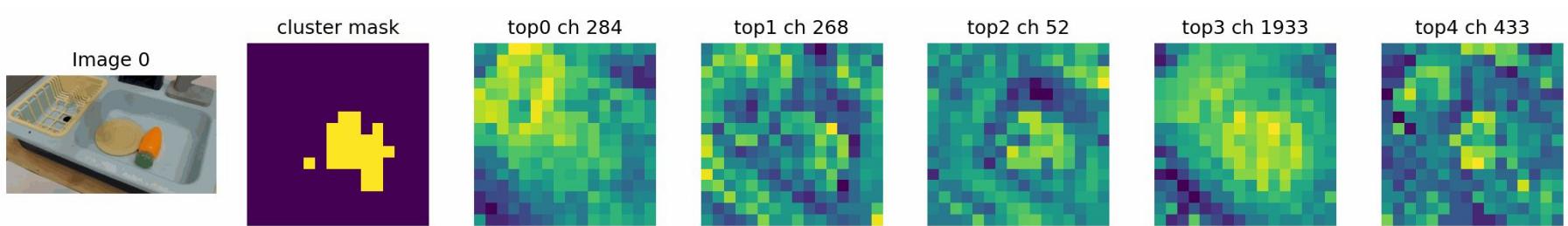
LLAVA prediction

1: 3	2: 4	3: 4	4: 4	5: 4	6: 4	7: 1	8: 4
9: 3	10: 3	11: 3	12: 3	13: 1	14: 3	15: 3	16: 4
17: 3	18: 4	19: 3	20: 3	21: 3	22: 4	23: 3	24: 3
25: 4	26: 4	27: 4	28: 4	29: 4	30: 4	31: 3	32: 3
33: 4	34: 4	35: 4	36: 4	37: 1	38: 4	39: 4	40: 3



2 Nyström NCut – How to do visual debugging

What's more: See what channels contribute to the cluster



3 VLA Encoder Extraction

- Vision backbone produces patch/grid tokens; language stack produces text tokens; cross-attention fuses modalities before decoding text
- Encoder Sourcing: HuggingFace AutoProcessor and Local Cache (spatial)
- Images + Prompt → Processor (pixel_values, input_ids, attention_mask) → Model Forward → capture vision backbone tokens + language layer of Encoder → concatenate across batch → NCut → t-SNE to RGB → render (NCut grids, colored tokens, videos/HTML)
- Ncut with language token alignment
 - Gather all language token embeddings from the final language layer across the batch into one matrix. Run NCut to partition that graph; each token gets an eigenvector-based coordinate. Map those NCut-derived coordinates back to each sample's token list
- Tsne to RGB
 - High-D NCut Eigen → t-SNE down to 3-D → Treat 3-D as RGB values
 - Each token's NCut position is now a color

4 Dataset Preparation

(quick side-note)

Many real-world robot-action datasets exist, including compilation meta-datasets (e.g. Open X-Embodiment).

Nevertheless, there are major shortcomings of open-source datasets of robot actions and operations, even in Pick n' Place:

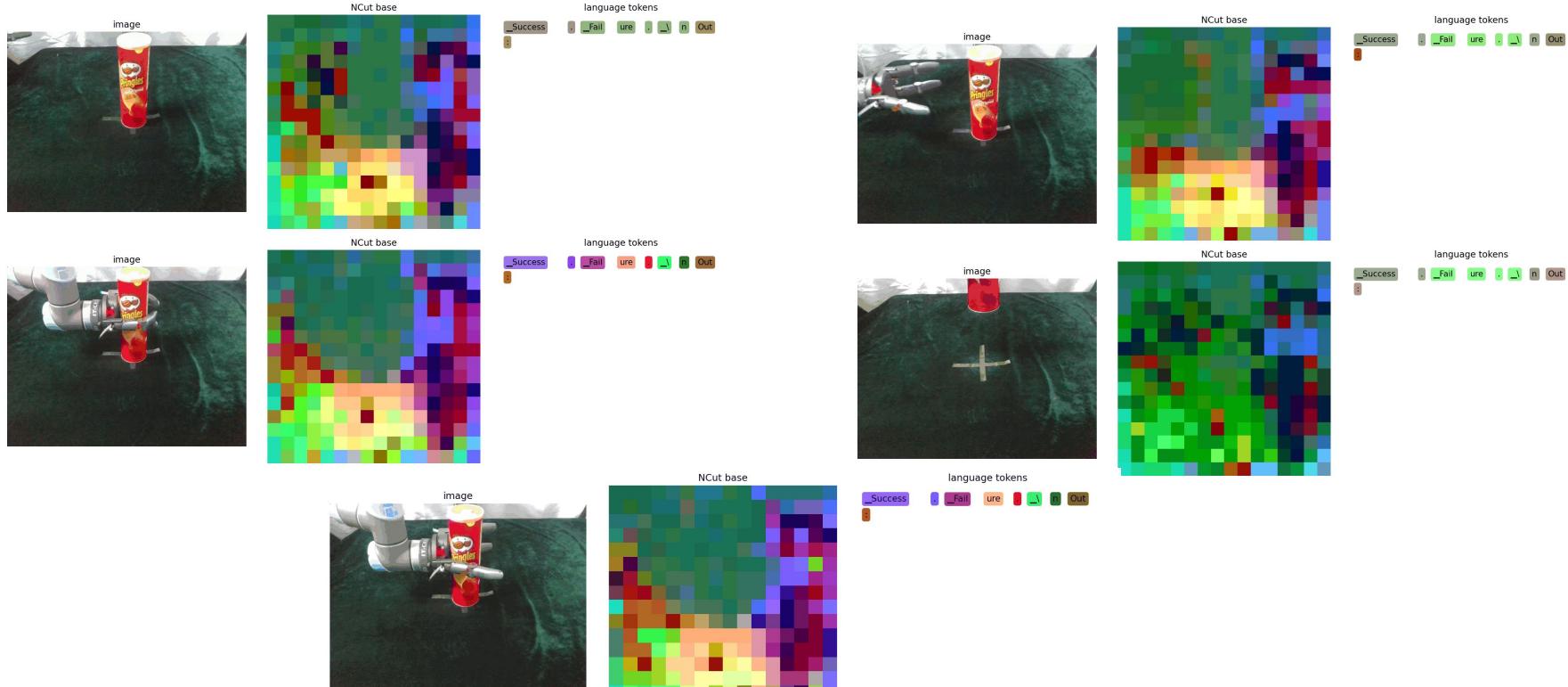
- Significant lack of structured datasets including fail cases - most often completely removed during curation
- Fail case explorations mostly explored in simulations, often offering niche use cases and implementation code only
- Datasets which do offer fail cases typically either:
 - Lacked size
 - Lacked action diversity
 - Lacked thorough labelling
 - Focused on fail cases specific to one application only
- Researchers often have to resort to self-curated datasets
 - Limited formality in curation – setup, data collection, inappropriate data, erroneous data – problematic for certain applications
 - Large 'gaps' in the datasets' generality, operation breadth, and edge cases
 - Limited information as to strategies for replicating dataset generation

We made use of a dataset, which while informal and sometimes containing erroneous data, offered the Pick n' Place tasks we wanted to focus on. For our application, irregularities and drastic permutations served as an advantage in exploring task generalisation [7]:

1. Manual labelling of task success
2. Removal of unnecessary/incomplete data
3. Frame extraction for N-cut processing

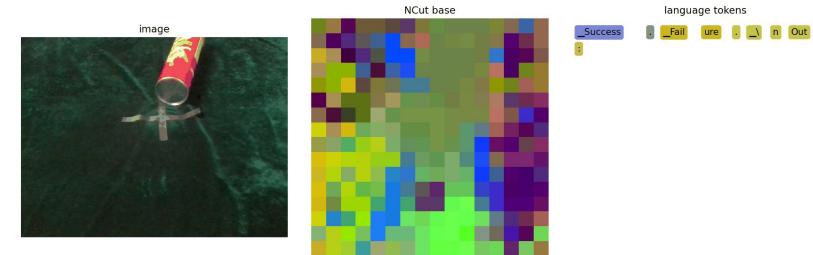
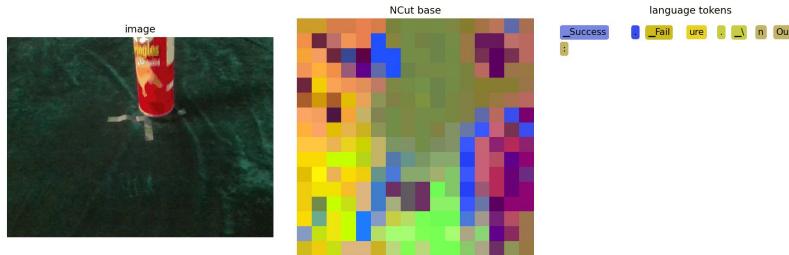
5 Findings – Text-image Semantics

Recognition of key objects for robot interactions:

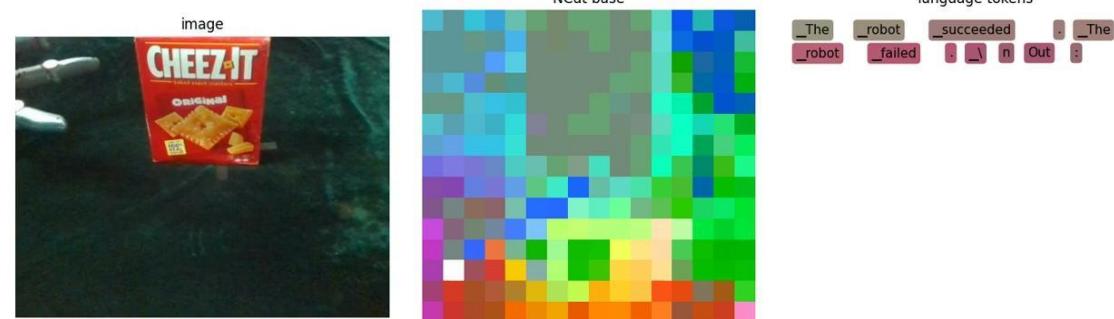


5 Findings – Text-image Semantics

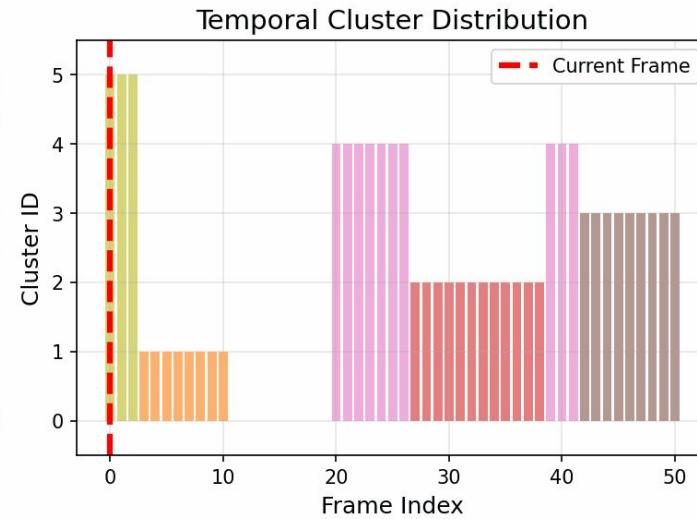
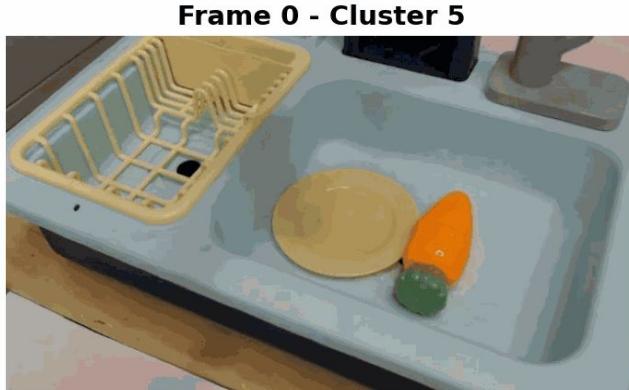
Semantic sensitivity to indicators of typical robotics failures:



Significant semantic shifts at key points of robot interactions:



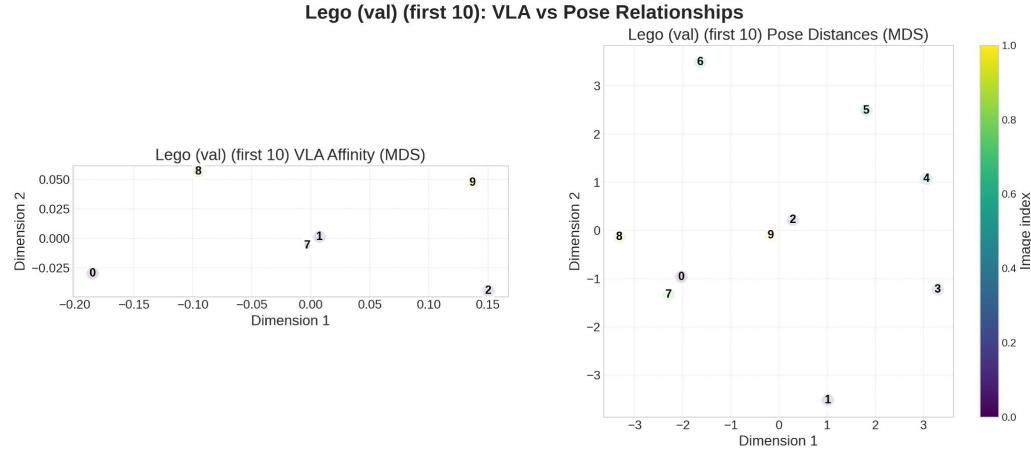
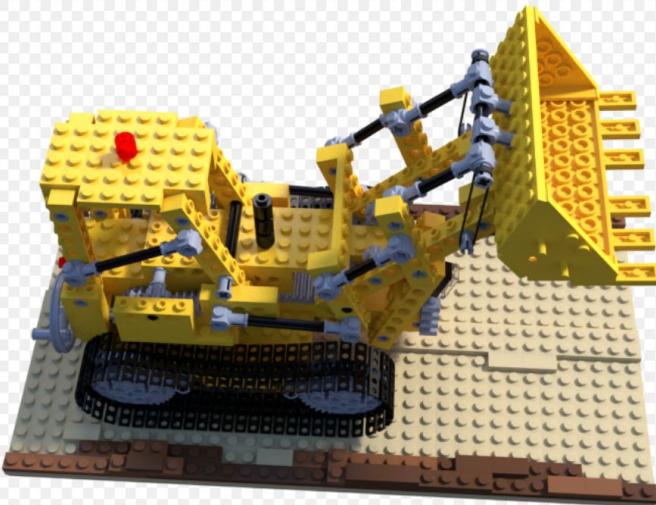
5 Findings – Temporal NCut



5 Findings – Input Sensitivity

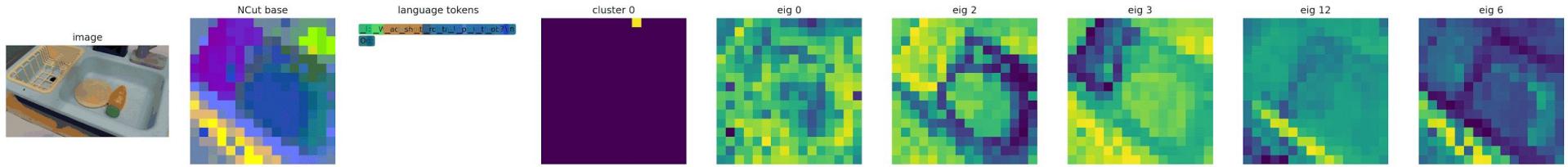
1. I have split the text in coco datasets and cluster them. For example, there are 30 images containing the word "push" in the image datasets. I took the features from those 30 images and averaged them as templates. And calculate the cosine similarities with other pics to calculate the MAP of the classification results.
 - Nouns Raking:
 - room(≈ 0.363)、street(≈ 0.291)、field(≈ 0.272)、grass(≈ 0.255)、plate(≈ 0.200)、building(≈ 0.186)、group(≈ 0.154)
 - :picture(≈ 0.085)、shirt(≈ 0.071)、water(≈ 0.115)、area(≈ 0.123)
 - Verbs Ranking:
 - park(≈ 0.451)、sit(≈ 0.375)
 - walk(≈ 0.122)、ride(≈ 0.107)、play(≈ 0.097)
 - look(≈ 0.098)、wear(≈ 0.075)、fill(≈ 0.068)、show(≈ 0.051)
2. I have changed verbs ,nouns and images. And see the change of feature space. And I see the changes in images > 10 times change in verbs > change in nouns. The output of VLAs doesn't really care about the text but only cares about the images

5 Findings – Spatial/orientation Comprehension

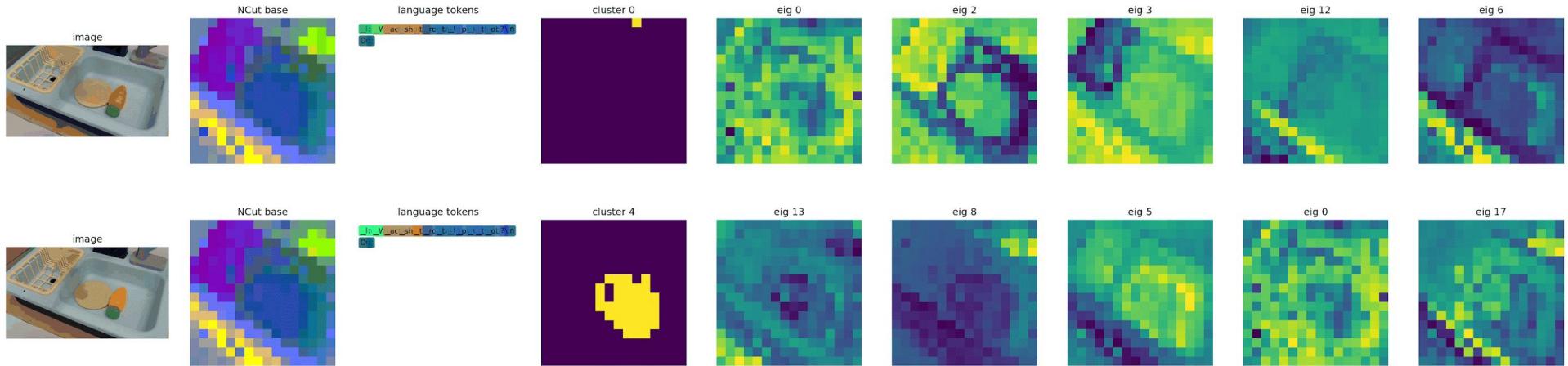


I use mlps to fit nerf datasets (features->poses). In the train set, the loss is 0.003 and the test is 0.998. It seems that the feature <->poses pair are one to one mapping but lacks generalizations in terms of spatial intelligence. It seems it had learned nothing about spatial information.

5 Findings – Back Propagation



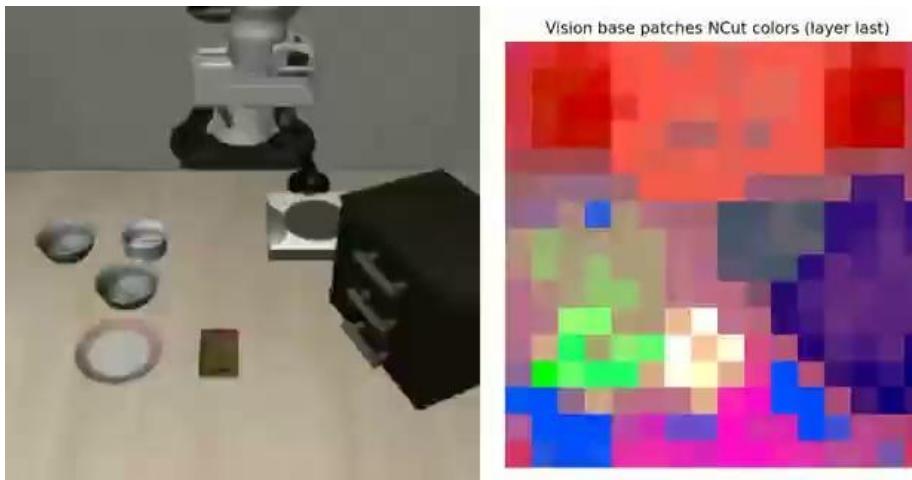
5 Findings – Back Propagation



5 Findings – Simulation



5 Findings – Simulation



5 Findings – Simulation

- ~70.83%



~80%



6 Future Work

- Apply to OpenPi 0.5
- Deeper semantic information?
 - Does temporal Ncut coincide with prompt/instruction text?
 - Can the model predict future or missing tasks given limited context?
- Making OpenVLA Fail

References

- [1] Y. Dai, J. Lee, N. Fazeli, and J. Chai, “RACER: Rich Language-Guided Failure Recovery Policies for Imitation Learning,” arXiv preprint arXiv:2409.14674, 2024.
- [2] J. Duan, W. Pumacay, N. Kumar, Y. R. Wang, S. Tian, W. Yuan, R. Krishna, D. Fox, A. Mandlekar, and Y. Guo, “AHA: A Vision-Language-Model for Detecting and Reasoning Over Failures in Robotic Manipulation,” arXiv preprint arXiv:2410.00371, 2024.
- [3] M. J. Kim et al., “OpenVLA: An Open-Source Vision-Language-Action Model,” arXiv preprint arXiv:2406.09246, 2024.
- [4] Physical Intelligence et al., “pi0.5: a Vision-Language-Action Model with Open-World Generalization,” arXiv preprint arXiv:2504.16054, 2025.
- [5] J. Shi and J. Malik, “Normalized cuts and image segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 888–905, 2000, doi: 10.1109/34.868688.
- [6] “Nyström Normalized Cuts PyTorch,” ncut-pytorch Documentation, [Online]. Available: <https://ncut-pytorch.readthedocs.io/en/latest/> [Accessed: Dec. 10, 2025].
- [7] Paper: Wang, T., Yang, C., Kirchner, F., Du, P., Sun, F., & Fang, B. (2019). Multimodal grasp data set: A novel visual–tactile data set for robotic manipulation. International Journal of Advanced Robotic Systems. <https://doi.org/10.1177/1729881418821571>