

# NoMaD: Navigation with Goal-Masked Diffusion

Abhishek, Namashivayaa,  
Sehaj, Shobhnik

IISc Bengaluru  
BTech. Mathematics and Computing

April 2025

## **Robotic navigation in unfamiliar environments requires:**

- Task-oriented navigation — reaching specified goals
- Task-agnostic exploration — discovering and mapping new areas

# Motivation and Goal

## Robotic navigation in unfamiliar environments requires:

- Task-oriented navigation — reaching specified goals
- Task-agnostic exploration — discovering and mapping new areas

## The Challenge

These two objectives are typically handled by *separate systems*.

Exploration can be decomposed into:

- **Local Exploration:** Learning short-horizon control policies for diverse actions
- **Global Planning:** Using those policies to achieve long-horizon, goal-directed behavior

# Motivation and Goal

## Robotic navigation in unfamiliar environments requires:

- Task-oriented navigation — reaching specified goals
- Task-agnostic exploration — discovering and mapping new areas

## The Challenge

These two objectives are typically handled by *separate systems*.

Exploration can be decomposed into:

- **Local Exploration:** Learning short-horizon control policies for diverse actions
- **Global Planning:** Using those policies to achieve long-horizon, goal-directed behavior

## Key Question

Can a *single model* unify both tasks — exploration and navigation?

# What is NoMaD?

**NoMaD** is a transformer-based diffusion policy designed for long-horizon, memory-efficient navigation.

It supports both:

- **Goal-conditioned navigation** — moving towards a specified visual goal
- **Open-ended exploration** — learning diverse behaviors without explicit goals

# What is NoMaD?

**NoMaD** is a transformer-based diffusion policy designed for long-horizon, memory-efficient navigation.

It supports both:

- **Goal-conditioned navigation** — moving towards a specified visual goal
- **Open-ended exploration** — learning diverse behaviors without explicit goals

NoMaD = {EfficientNet + Vision Transformer} ← ViNT  
+ Diffusion Policies

# What is NoMaD?

**NoMaD** is a transformer-based diffusion policy designed for long-horizon, memory-efficient navigation.

It supports both:

- **Goal-conditioned navigation** — moving towards a specified visual goal
- **Open-ended exploration** — learning diverse behaviors without explicit goals

NoMaD = {EfficientNet + Vision Transformer} ← ViNT  
+ Diffusion Policies

It combines a transformer backbone to encode the high-dimensional visual stream, with diffusion models that predict a sequence of future actions in a generative manner.

# Overview of NoMaD Architecture

## Visual Goal-Conditioned Navigation

Backbone: ViNT (Visual Navigation Transformer)

How does ViNT work?

- Receives: A sequence of past and current observations  $o_t = o_{t-P:t}$
- **Visual Encoder:** Each observation is processed using an EfficientNet-B0 encoder to extract feature embeddings.

# Overview of NoMaD Architecture

## Visual Goal-Conditioned Navigation

Backbone: ViNT (Visual Navigation Transformer)

How does ViNT work?

- Receives: A sequence of past and current observations  $o_t = o_{t-P:t}$
- **Visual Encoder:** Each observation is processed using an EfficientNet-B0 encoder to extract feature embeddings.

EfficientNet?

- A new method of Scaling CNNs to improve accuracy and efficiency
- It uses a **compound scaling** to uniformly scale all dimensions of depth, width, and resolution.

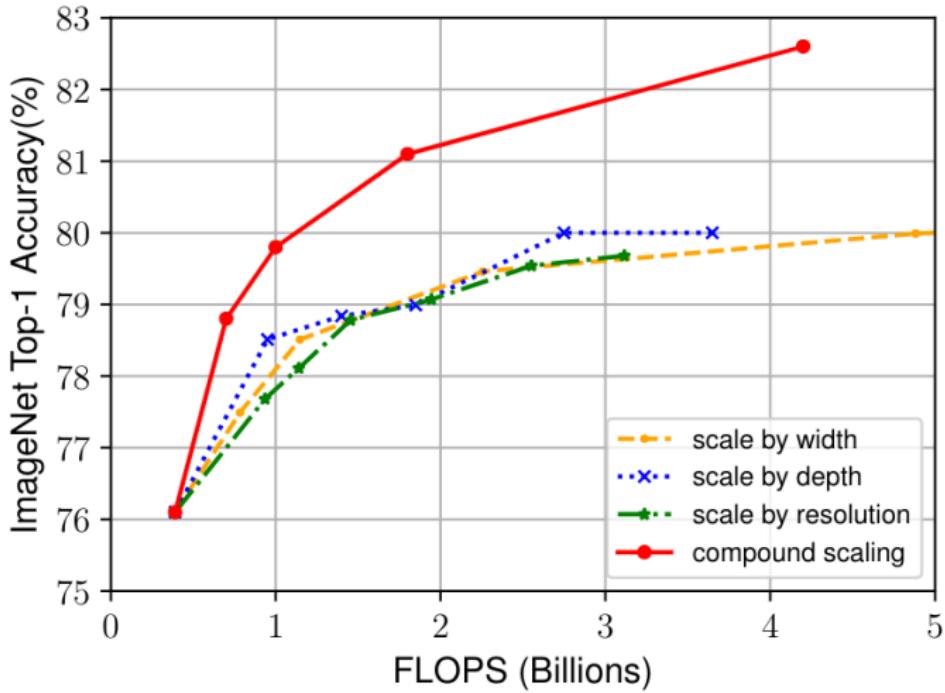


Figure: Compound Scaling

Use Network Architecture Search (NAS) to find the best baseline network (EfficientNet-B0)

## Optimization Objective:

$$\text{ACC}(m) \times \left[ \frac{\text{FLOPS}(m)}{T} \right]^w$$

- $\text{ACC}(m)$ : accuracy of model  $m$
- $\text{FLOPS}(m)$ : floating point operations
- $T$ : target FLOPS
- $w = -0.07$ : controls trade-off between accuracy and FLOPS

# EfficientNet Scaling

## Compound Scaling

EfficientNet introduces a principled way to scale up CNNs using a single compound coefficient  $\phi$ .

- Simultaneously scales:

- Network depth  $d$
- Width  $w$
- Input resolution  $r$

- Scaling formulas:

$$d = \alpha^\phi, \quad w = \beta^\phi, \quad r = \gamma^\phi$$

- Constants  $\alpha$ ,  $\beta$ , and  $\gamma$  are determined via grid search.

**Subject to constraint:**

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

Ensures that the model scales within a fixed computational budget.

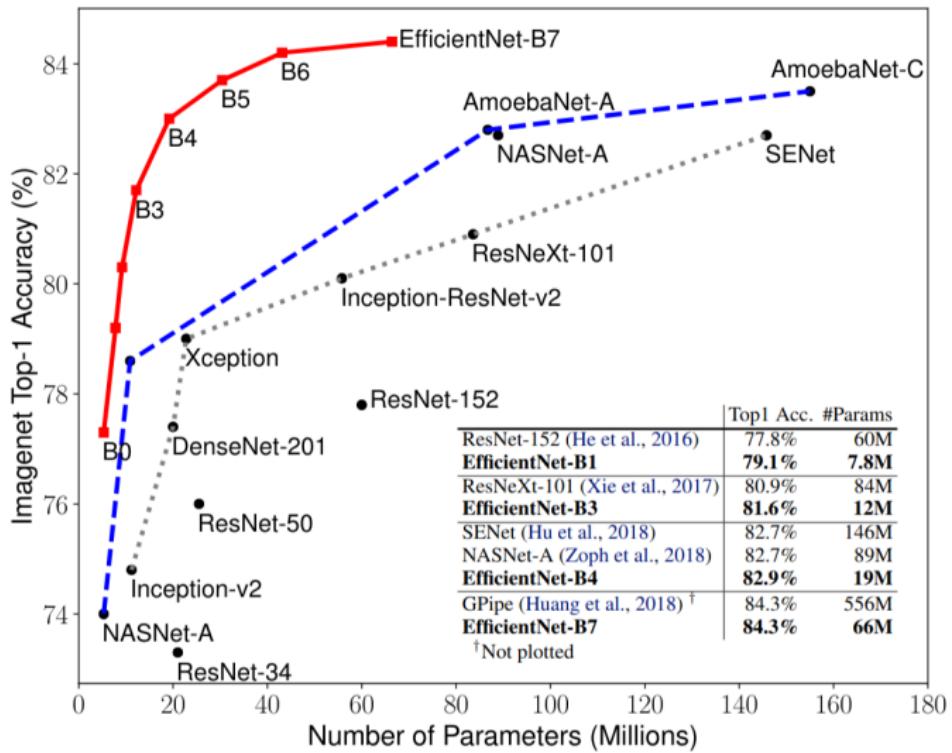


Figure: Accuracy on imagenet

# Overview of NoMaD Architecture

## Visual Goal-Conditioned Navigation

Backbone: ViNT (Visual Navigation Transformer)

How does ViNT work?

- Receives: A sequence of past and current observations  $o_t = o_{t-P:t}$
- **Visual Encoder:** Each observation is processed using an EfficientNet-B0 encoder to extract feature embeddings.

# Overview of NoMaD Architecture

## Visual Goal-Conditioned Navigation

Backbone: ViNT (Visual Navigation Transformer)

How does ViNT work?

- Receives: A sequence of past and current observations  $o_t = o_{t-P:t}$
- **Visual Encoder:** Each observation is processed using an EfficientNet-B0 encoder to extract feature embeddings.
- **Goal Fusion:** The current and goal images are combined using a goal-fusion encoder.
- **Transformer Attention:** These fused features (tokens) are passed through a Transformer model to generate a context vector  $c_t$ .
- **Predictions:** The context vector is used to predict:
  - A distribution over future actions:  $a_t = f_a(c_t)$
  - An estimate of temporal distance to the goal:  $d(o_t, o_g) = f_d(c_t)$

# Extending to Long-Horizon Planning with Topological Memory

However, ViNT is inherently goal-conditioned—it cannot operate in the absence of a goal image, limiting its ability to explore autonomously.

# Extending to Long-Horizon Planning with Topological Memory

However, ViNT is inherently goal-conditioned—it cannot operate in the absence of a goal image, limiting its ability to explore autonomously.

## Solution

To enable open-ended exploration, NoMaD incorporates a Topological Memory  $\mathcal{M}$ :

- ① Nodes represent previously encountered visual observations.
- ② Edges represent traversable paths, established using ViNT's predicted distances.

This enables:

- **Subgoal Planning:** The model can plan a sequence of subgoals to reach a target location.
- **Frontier Exploration:** The model can autonomously explore new areas by identifying frontiers in the topological map.

# Overview of NoMaD Architecture

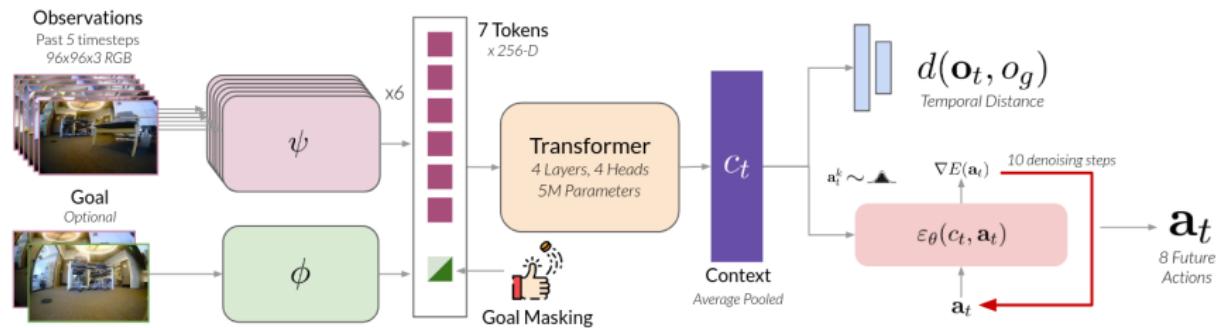
NoMaD = {EfficientNet + Vision Transformer} ← ViNT  
+ Diffusion Policies

Nomad builds upon ViNT by:

## Attention based Goal Masking:

Introduces a binary mask  $m$ , and modifies the context vector  $c_t$  as:

$$c_t = f(\psi(o_i), \phi(o_t, o_g), m)$$



# Overview of NoMaD Architecture

NoMaD = {EfficientNet + Vision Transformer} ← ViNT  
+ Diffusion Policies

## Diffusion Policies:

To model complex, multimodal action distributions, NoMaD employs a diffusion model to approximate the conditional distribution of the next action as:  $p(a_t|c_t)$ .

**1. Forward Process:** Start with a real action  $a_t^0$  and add gaussian noise to it over multiple steps.

$$a_t^k = \sqrt{\alpha_k} a_t^{k-1} + (\sqrt{1 - \alpha_k})\epsilon$$

where:

- $\epsilon \sim \mathcal{N}(0, I)$  is a random noise
- $\alpha_k$  is a noise scheduler (eg square cosine)
- By step K, the action is almost pure noise.

# Overview of NoMaD Architecture

NoMaD = {EfficientNet + Vision Transformer}  $\leftarrow$  ViNT  
+ Diffusion Policies

**2. Reverse Denoising:** starting from pure noise  $a_t^k \sim \mathcal{N}(0, I)$ , it denoises step by step to recover the final clean action  $a_t^0$ .

Each denoising step is :

$$a_t^{k-1} = \alpha(\alpha_t^k - \gamma_k \cdot \epsilon_\theta(c_t, a_t^k, k)) + \mathcal{N}(0, \sigma^2 \cdot I)$$

Where:

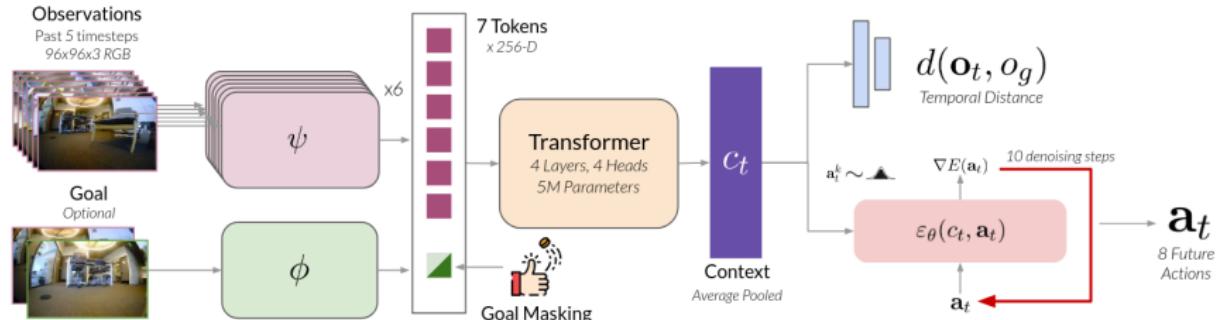
- Here,  $\epsilon_\theta$  is the noise prediction network conditioned on the context  $c_t$ , which may or may not include the goal depending on  $m$ .
  - It is a 1D conditional U-Net with 15 CNN layers.
  - Input: Noisy action  $a_t^k$ , Context vector  $c_t$ , and the diffusion step  $k$ .
  - the predicted noise vector  $\hat{\epsilon}_k$ , During training, it is compared to the true noise added earlier.
- $\gamma, \alpha, \sigma$  are scheduler constants.

# Overview of NoMaD Architecture

NoMaD = {EfficientNet + Vision Transformer}  $\leftarrow$  ViNT  
+ Diffusion Policies

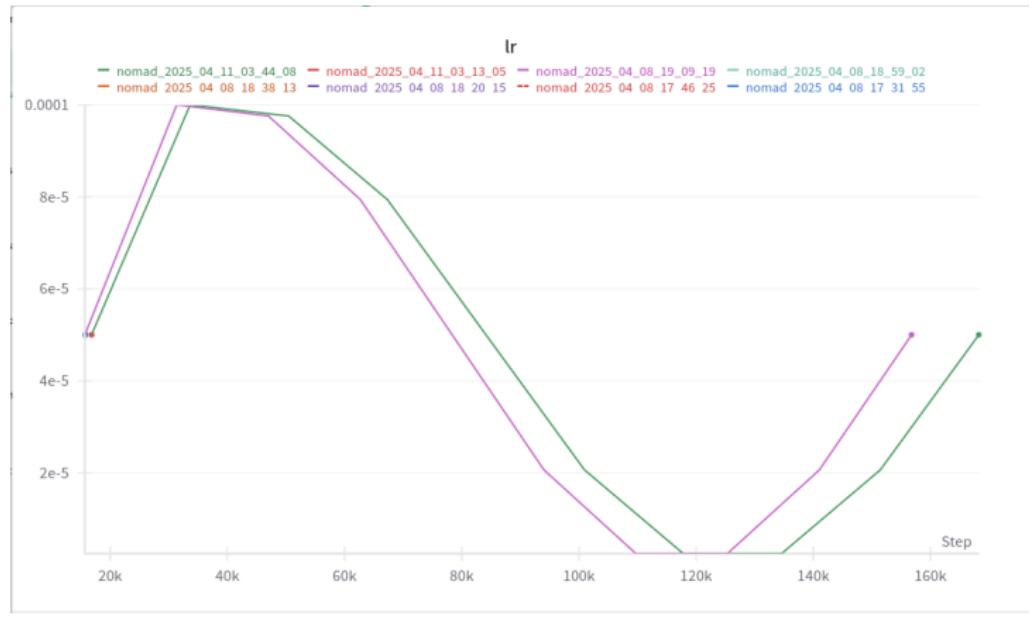
**3. Action Decoder:** The denoised action  $a_t^0$  is then passed through a low-level action decoder to generate the final action  $a_t$ .

- The decoder maps the denoised action to a low-level control command for the robot.
- It can be a simple feedforward network or a more complex recurrent network.



# Training Details and Experiments

- Datasets used: Sacson/HuRoN, RECON and SCAND
- Batch size: 47, Epochs: 10
- Optimizer: AdamW, Lr:  $10^{-4}$
- Scheduler: Cosine annealing



## Training Details and Experiments

- Goal Masking Probability:  $p_m = 0.5$
- Diffusion Steps: 10
- Noise Scheduler: Square Cosine

# Training Details and Experiments

- Goal Masking Probability:  $p_m = 0.5$
- Diffusion Steps: 10
- Noise Scheduler: Square Cosine

## Training Objective

- **Diffusion Loss:** Measures the difference between the predicted and true noise.
- **Distance Loss:** Measures the difference between the predicted and true distance to the goal.

$$\mathcal{L}_{NoMaD}(\phi, \psi, f, \theta, f_d) = MSE(\epsilon^k, \epsilon_\theta(c_t, a_t^0 + \epsilon^k, k)) + \lambda \cdot MSE(d(o_t, o_g), f_d(c_t))$$

where:

- We set  $\lambda$  to  $10^{-4}$
- $\psi, \phi$  correspond to the visual encoders for the observation and goal images.
- $f$  corresponds to the transformer layers,  $\theta$  to diffusion parameters,
- $f_d$  corresponds to the temporal distance predictor.

# Training Visualization with wandb

## Why Weights & Biases (wandb)?

We used wandb to log training progress, visualize losses, and monitor both model behavior and system resources (e.g., GPU/CPU utilization) throughout experimentation. Metrics such as **action loss**, **goal prediction error**, and the **learning rate schedule** were automatically tracked and visualized, which helped with debugging and plotting out results.



Figure: QR code to project dashboard

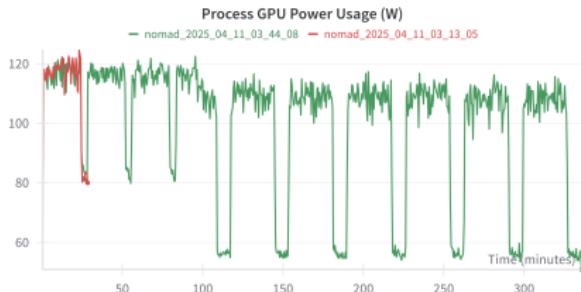
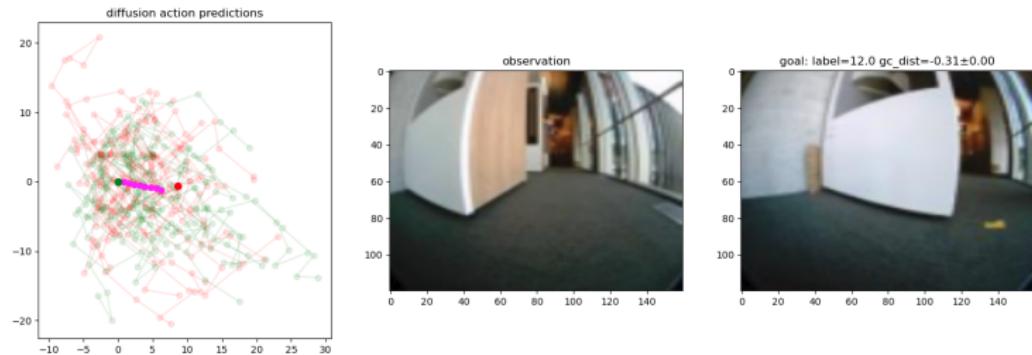


Figure: GPU power usage during training

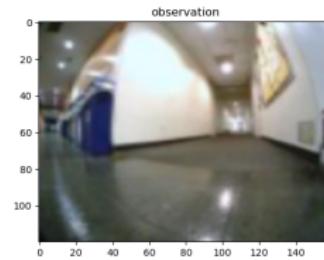
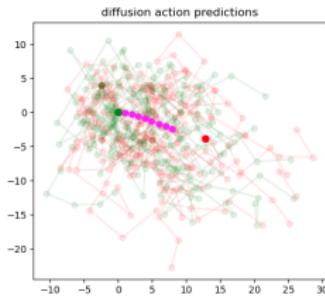
# Action Samples generated by NoMaD

## During Training



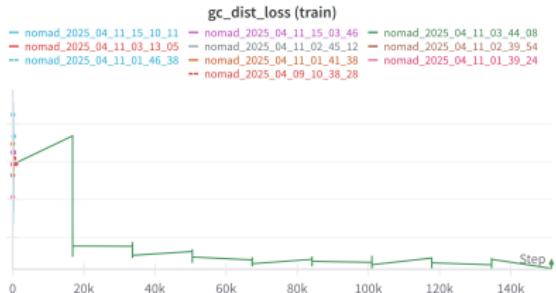
# Action Samples generated by NoMaD

## During Testing

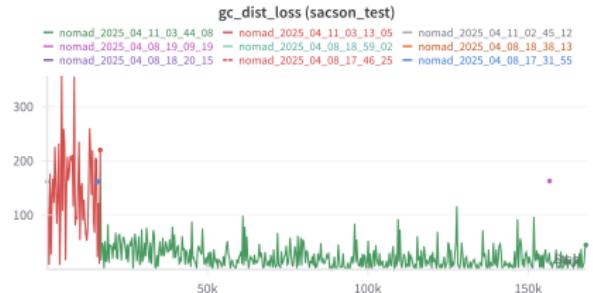


# Experiments and Results

## Distance Loss (Goal Conditioned)



(a) Distance Loss on Training Set

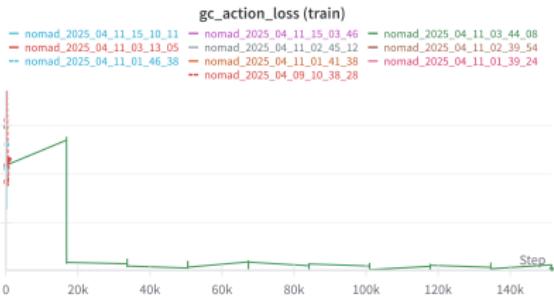


(b) Distance Loss on Validation Set

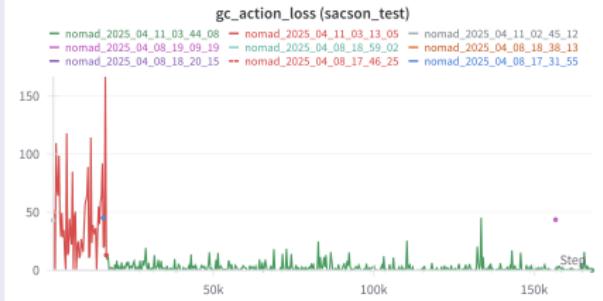
**Figure:** Distance loss comparison between training and validation sets under goal-conditioned evaluation.

# Experiments and Results

## Action Loss (Goal Conditioned)



(a) Action Loss on Training Set

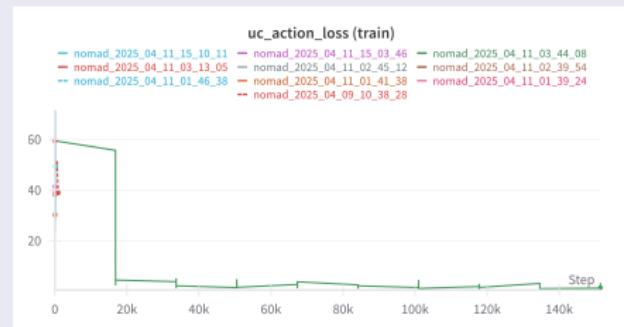


(b) Action Loss on Validation Set

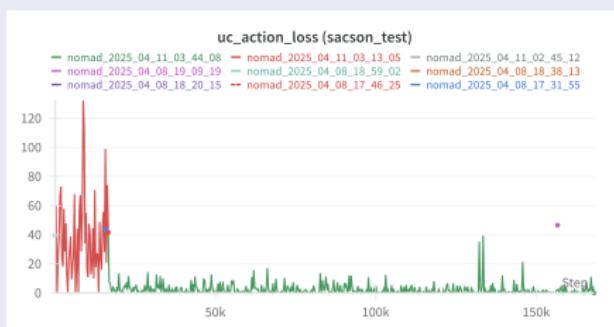
Figure: Action loss comparison between training and validation sets under goal-conditioned evaluation.

# Experiments and Results: Unconditioned Setting

## Unconditioned Action Loss Evaluation



Training Set

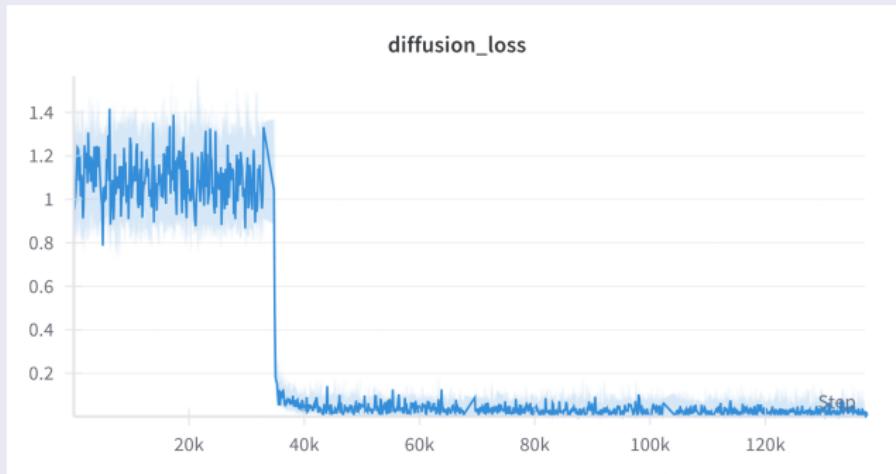


Validation Set

The unconditioned action loss measures model accuracy in open-loop, goal-agnostic settings. Lower loss indicates better generalization in exploratory behavior.

# Experiments and Results: Diffusion Loss

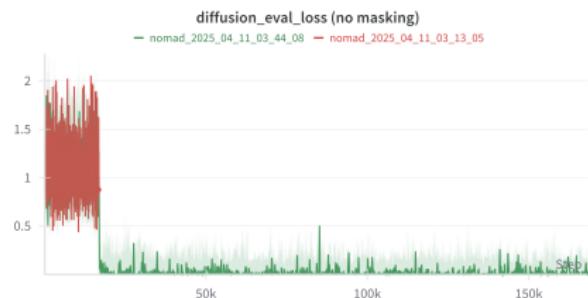
## Training Performance



**Figure:** Diffusion loss over training batches. This loss reflects how well the model learns to denoise the trajectory samples using the learned conditional distribution.

# Experiments and Results: Diffusion Loss on Validation Set

## Goal-Conditioned ( $m = 0$ )



**All tokens receive goal signal.**  
Evaluates the robot's ability to follow a target.

## Unconditioned ( $m = 1$ )



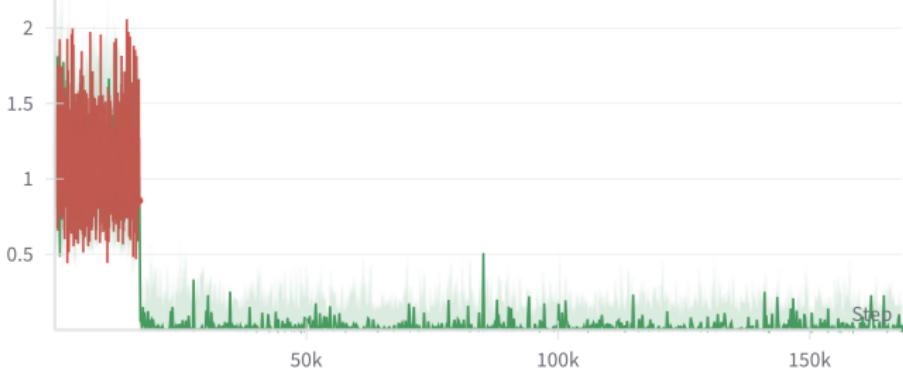
**No goal information.**  
Evaluates purely exploratory navigation behavior.

*We see that loss goes downstream in both cases.*

# Evaluation: Mixed Masking ( $m \sim \mathcal{B}(0.5)$ )

## Stochastic Goal Conditioning

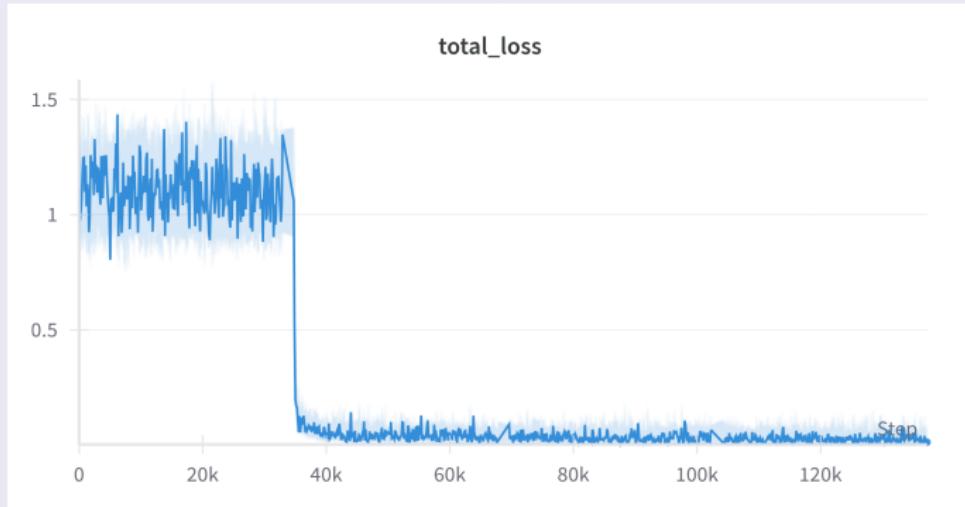
diffusion\_eval\_loss (goal masking)  
— nomad\_2025\_04\_11\_03\_44\_08 — nomad\_2025\_04\_11\_03\_13\_05



**Description:** Each token is randomly masked with probability 0.5. This encourages the model to balance between exploring and exploiting goal cues.

# Experiments and Results :Total Loss

## Total Loss



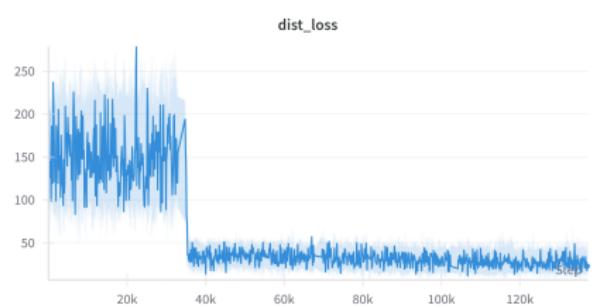
**Description:** The total loss combines both the diffusion and distance losses, providing a holistic measure of model performance.

# Comparision with ViNT: Temporal Distance Loss

## Motivation

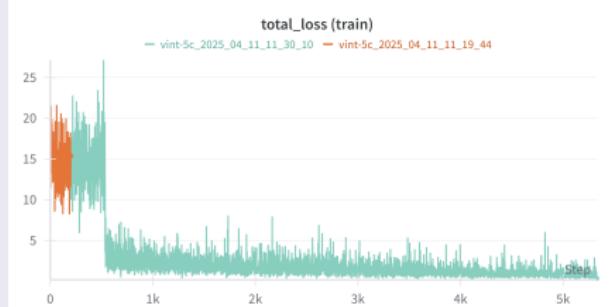
ViNT serves as a strong baseline for visual navigation with transformer-based context encoding. We compare its distance prediction ability against NoMaD.

### NoMaD Distance Loss



NoMaD achieves lower and more stable distance loss due to diffusion-based modeling and goal conditioning.

### ViNT Distance Loss



ViNT performs comparably in early epochs but struggles with long-horizon distance regression.

*Observation:* NoMaD's diffusion-based decoder improves distance supervision without sacrificing ViNT's transformer strengths.

# ViNT vs NoMaD: Action Loss

## Objective

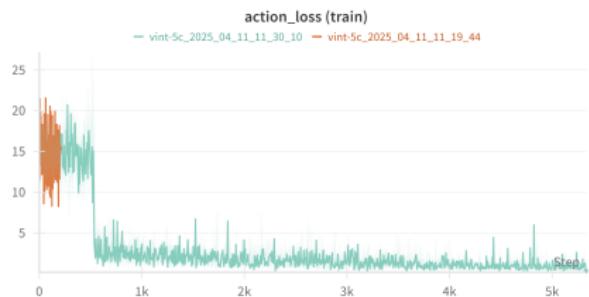
We compare the quality of predicted waypoints by measuring the Mean Squared Error (MSE) between predicted and ground-truth actions.

### NoMaD Action Loss



NoMaD consistently achieves lower action loss, aided by the stochastic denoising process and goal-conditioned context.

### ViNT Action Loss



ViNT shows slightly higher variance, and loss plateaus earlier due to MLP-based decoding.

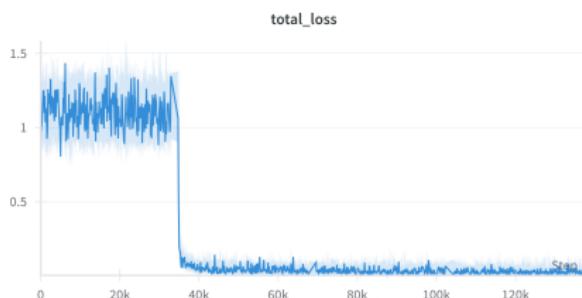
*Observation: NoMaD benefits from the expressiveness of diffusion decoding in predicting fine-grained waypoint actions.*

# Comparison with ViNT: Total Loss

## Motivation

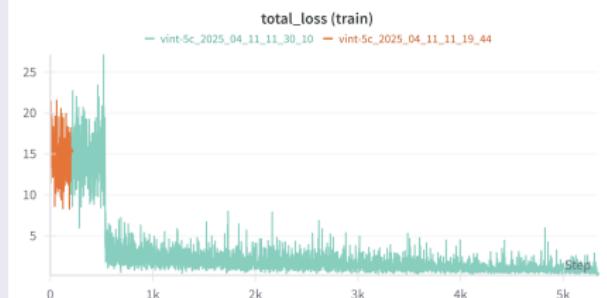
To assess the overall effectiveness of NoMaD vs. ViNT, we compare the combined loss: action prediction loss + temporal distance prediction loss.

### NoMaD Total Loss



NoMaD achieves better convergence, with lower final loss due to richer diffusion-based decoding.

### ViNT Total Loss



ViNT exhibits slower learning and higher loss, limited by direct MLP-based action prediction.

**Takeaway: NoMaD demonstrates improved learning dynamics and generalization.**

# Challenges Faced

- CUDA Out Of Memory errors on limited GPU
- Module import issues with nested folder structures
- Gradients not propagating due to detached variables

# Team Contributions

## **Abhishek Kumar Jha:**

- Helped in Understanding the diffusion model
- Helped in solving the errors while training the NoMaD model

## **Namashivayaa V:**

- Helped in Understanding the theory for the entire project
- Helped Extracting the data from the datasets
- Made the Appendices section of the Report

# Team Contributions

## **Sehaj Ganjoo:**

- Implemented the ViNT architecture
- Developed the goal fusion encoder
- Conducted experiments and analysis for comparing NoMaD and ViNT
- Made the Presentation and Report

## **Shobhnik Kriplani:**

- Implemented the Tokenizer, embeddings and positional encodings from scratch
- Implemented the NoMaD architecture
- Developed the training pipeline
- Conducted experiments and analysis

# Conclusion and Future Work

- Successfully trained NoMaD using diffusion for visual navigation
- Showed compatibility with ViNT-based perception
- Future work:
  - Deploy NoMaD on real robots and check performance on real world environments
  - Explore the use of NoMaD for other tasks like object detection and tracking
  - Explore how we can improve the current architecture.
  - Try larger ViTs and alternate decoders

# Thank You!

# Appendices

## Appendices

- Appendix A: Additional Results
- Appendix B: Implementation Details
- Appendix C: References

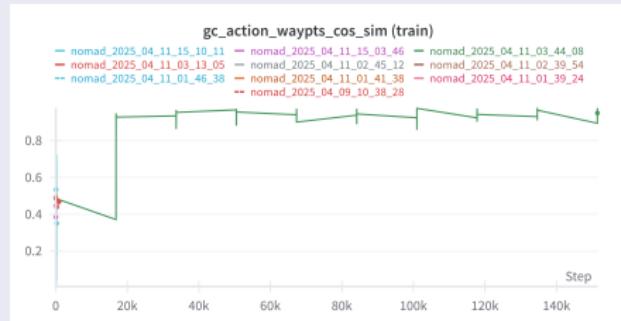
# Appendix A: Additional Results

## Additional Results

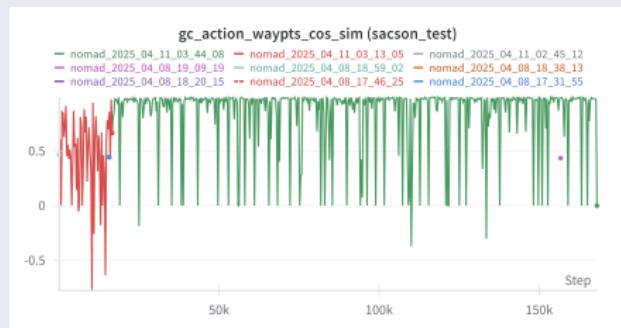
- Additional results and analysis of NoMaD's performance

# Experiments and Results: Cosine Similarity

## Goal-Conditioned Action Waypoints Cosine Similarity



Training Set

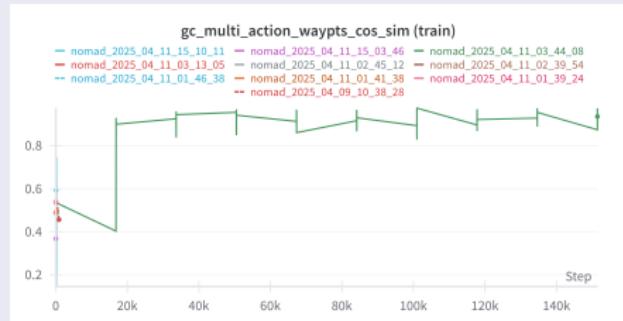


Validation Set

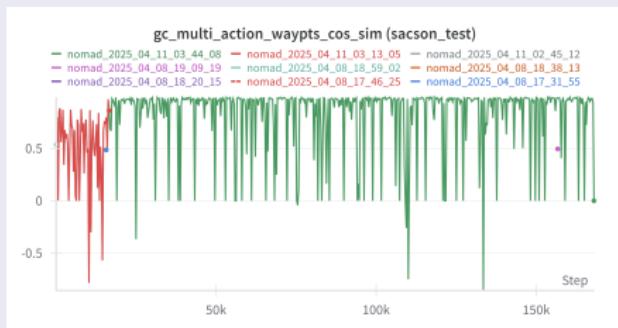
Cosine similarity evaluates directional alignment between predicted and ground-truth action waypoints. Higher values ( $\sim 1.0$ ) indicate better trajectory alignment under goal-conditioned settings.

# Experiments and Results: Multi-Action Cosine Similarity

## Goal-Conditioned Multi-Action Waypoints Cosine Similarity



Training Set



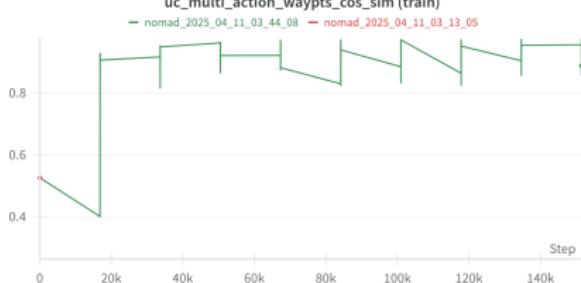
Validation Set

Multi-action cosine similarity compares the overall alignment of full predicted trajectory vectors with ground truth, rather than frame-by-frame. This provides a more holistic measure of long-horizon trajectory quality.

# Experiments and Results: UC Multi-Action Cosine Similarity

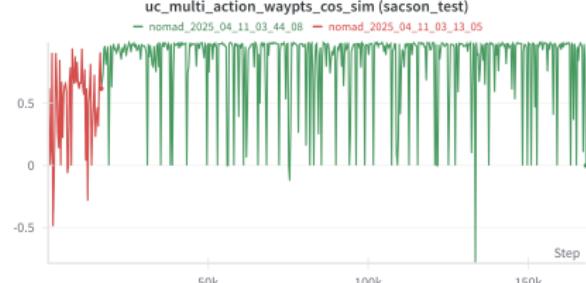
## Unconditioned Multi-Action Waypoints Cosine Similarity

uc\_multi\_action\_waypts\_cos\_sim (train)



**Training Set**

uc\_multi\_action\_waypts\_cos\_sim (sacson\_test)



**Validation Set**

Cosine similarity across the full predicted trajectory in unconditioned setting. Higher similarity indicates better alignment with ground-truth behavior.