

NoMaD: Navigation with Goal-Masked Diffusion

Sehaj Ganjoo, Shobhnik Kriplani,
Abhishek Kumar Jha, Namashivayaa V

IISc Bengaluru
BTech. Mathematics and Computing

April 2025

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

NoMaD = {EfficientNet + Vision Transformer} \leftarrow 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.

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.

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.

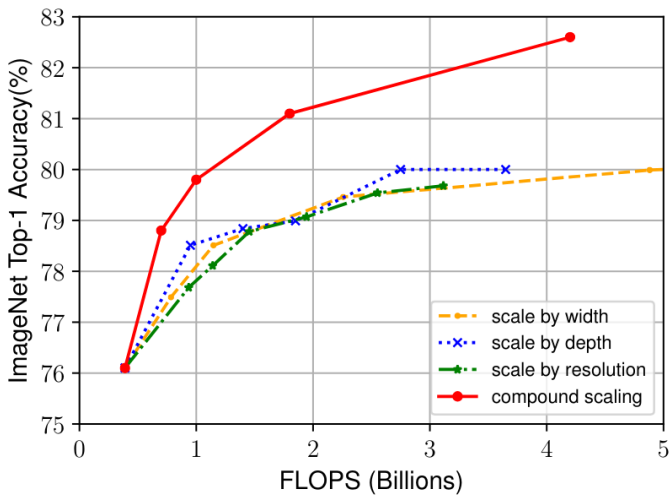


Figure: Compound Scaling

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

Compound Scaling

EfficientNet introduces a principled way to scale up CNNs using a single compound coefficient ϕ .

- Simultaneously scales:
 - Network depth d
 - Width w
 - Input resolution r
- Scaling formulas:

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

- Constants α , β , and γ 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.

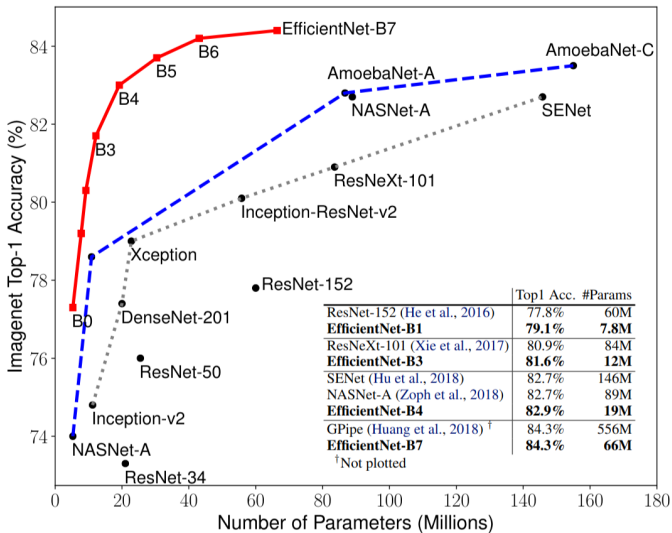


Figure: Accuracy on imagenet

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.

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

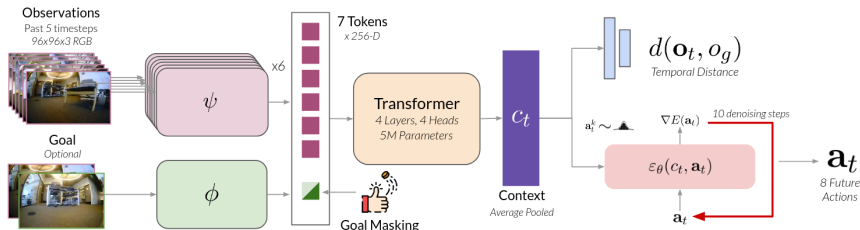
NoMaD = {EfficientNet + Vision Transformer} \leftarrow ViNT
+ Diffusion Policies

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} \leftarrow 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
- α_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:

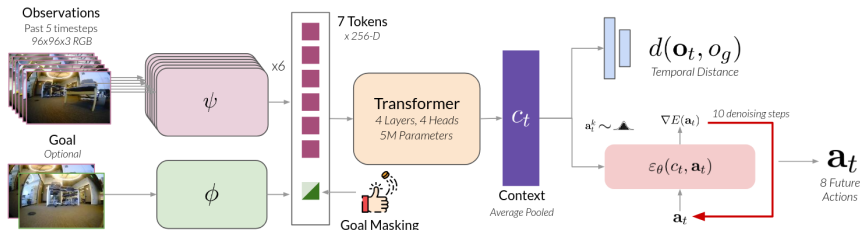
- Here, ϵ_θ is the noise prediction network conditioned on the context c_t , which may or may not include the goal depending on m .
 - It is a 1D conditional U-Net with 15 CNN layers.
 - Input: Noisy action a_t^k , Context vector c_t , and the diffusion step k .
 - the predicted noise vector $\hat{\epsilon}_k$,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



Architecture Overview

- **Perception:** EfficientNet-B0 backbone → Transformer-based temporal encoder.
- **Diffusion Decoder:** Conditional UNet1D for generating waypoint sequences.
- **Action Decoder:** Maps waypoints to low-level actions.

Training Procedure

- Dataset: SACSoN / RECON / GoStanford
- Batch size: 32, Epochs: 100
- Optimizer: AdamW, LR: 10^{-4}
- Scheduler: Cosine annealing
- Loss: MSE on predicted noise + temporal distance

$$\mathcal{L}_{NoMaD} = \text{MSE}(\epsilon, \hat{\epsilon}) + \lambda \cdot \text{MSE}(d(o_t, o_g), f_d(c_t))$$

Metrics:

- Diffusion Loss ≈ 1.11
- Distance Loss ≈ 128
- Cosine Similarity ≈ 0.47

Comparison with ViNT:

- Similar performance in goal-conditioned tasks
- No performance degradation when adding diffusion

Challenges Faced

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

Sehaj Ganjoo:

- Set up training pipeline and environment
- Integrated and debugged diffusion model
- Wrote training script and logging tools
- Conducted experiments and generated plots
- Created report and presentation

Conclusion and Future Work

- Successfully trained NOMAD using diffusion for visual navigation
- Showed compatibility with ViNT-based perception
- Future work:
 - Evaluate in simulation / real-world
 - Improve runtime performance
 - Try larger ViTs and alternate decoders

Thank you!

Questions are welcome.