NoMaD: **N**avigati**o**n with Goal-**Ma**sked **D**iffusion

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

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

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?

- Recieves: 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?

- Recieves: 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.

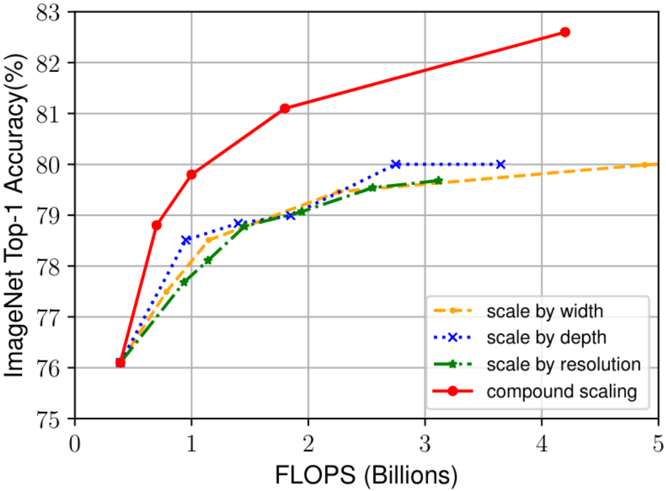


Figure: Compound Scaling

EfficientNet

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

Optimization Objective:

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

- $\mathrm{ACC}(m)$: accuracy of model m
- $\mathrm{FLOPS}(m)$: 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 ϕ.

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

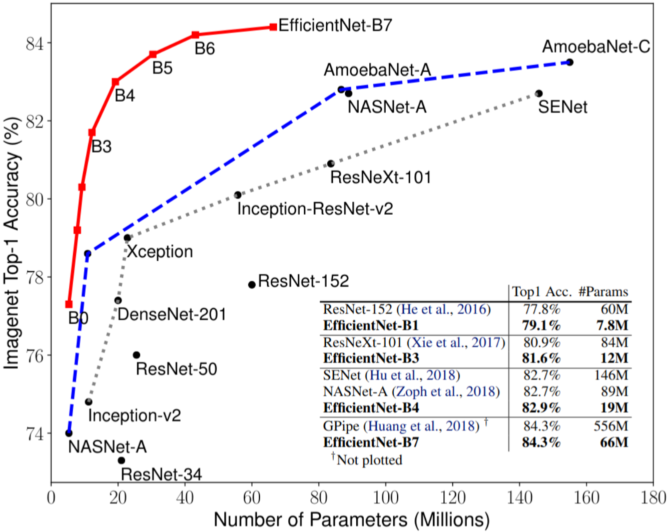
$$d = \alpha^{\phi}, \quad w = \beta^{\phi}, \quad r = \gamma^{\phi}$$

- 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.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Figure: Accuracy on imagenet

Overview of NoMaD Architecture

Visual Goal-Conditioned Navigation

Backbone: ViNT (Visual Navigation Transformer)
How does ViNT work?

- Recieves: 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?

- Recieves: 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

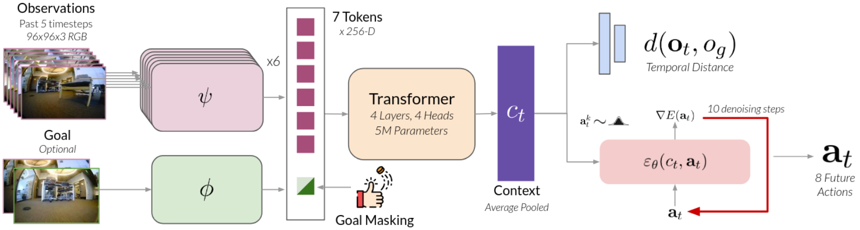
`NoMaD = {EfficientNet + Vision Transformer} ← ViNT`
`+ Diffusion Policies`

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 distri- bution 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 :

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

Where:

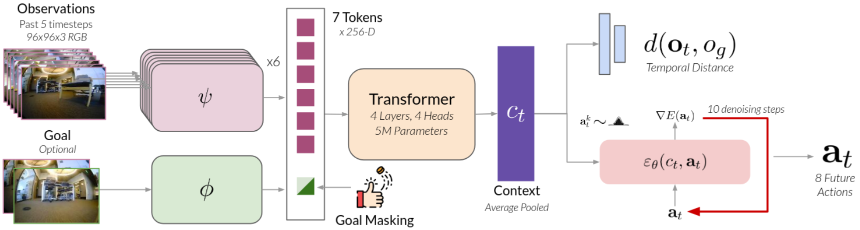
- Here, ϵ_θ is the noise prediction network conditioned on the context c_t, which may or may not include the goal depending on m.
  - It is a 1D conditional U-Net with 15 CNN layers.
  - Input:Noisy action a_t^k, Context vector c_t, and the diffusion step k.
  - the predicted noise vector $\hat{\epsilon}_k$,During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

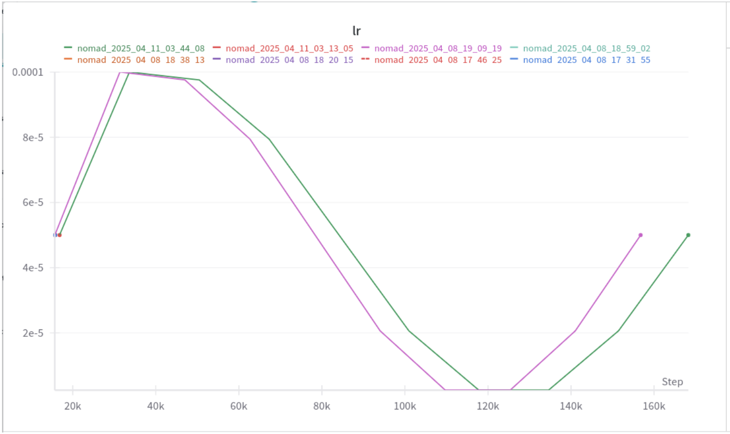
3. Action Decoder: The denoised action a_t^0 is then passed through a low-level action decoder to generate the final action a_t.

- 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.



Training Details and Experiments

- Datasets used: SACSoN , RECON , GoStanford and SCAND
- Batch size: 32, 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 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.

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

where:

- We set λ to 10^{-4}
- ψ, ϕ correspond to the visual encoders for the observation and goal images.
- f corresponds to the transformer layers, θ to diffusion parameters,
- f_d corresponds to the temporal distance predictor.

Experiments and Results

Metrics:

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

Comparision with ViNT

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

Team Contributions

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

Q&A

Thank you!
Questions are welcome.