NOMAD Notes

Authors Mathematics and Computing
Indian Institute of Technology

April 9, 2025

1 Introduction

> In this paper, we present a design for such a policy by combining a Transformer backbone for encoding the high- dimensional stream of visual observations with diffusion models for modeling a sequence of future actions and instan- tiate it for the particular problem of visual exploration and goal-seeking in novel environments

What is nomad?
NoMaD is a robot navigation system that can:

- Explore unknown places on its own (goal-agnostic behavior).
- Go to a specific place or object when given a goal image (goal-directed behavior).

The key idea: Instead of having two different systems for these tasks (like other methods do), NoMaD does both using one unified model.

Why is this important?

In the real world:
-A robot might need to search for a place (like exploring a building).
-Later, it may need to go back to that place (like retrieving an object from a room).
-Existing systems often split these into two steps — exploration and navigation — with different models or planning stages.
NoMaD combines both into one model, making the robot:
-Smarter.
-Faster.
-Less prone to mistakes (like collisions).

How does it work?

Input: An image of the environment, and optionally a goal image (like "go here").
Model: Uses a Transformer (like those in NLP/AI) to understand the visual world, and a

Diffusion Model to plan actions.
Goal Masking: If a goal image is given, it uses that to guide movement. If not, it goes into exploration mode.
This ”masking” lets the model ignore the goal if it doesn’t have one, which allows exploration to happen naturally.

Why Diffusion Models?
Diffusion models are great at generating complex, multi-modal outputs. That means:
The robot doesn’t just plan a single path.
It can consider many possibilities, which helps it handle real-world messiness (like obstacles or uncertainty).
We factorize the exploration problem into (i) learned control policies that can take diverse, short-horizon actions, and (ii) a high-level planner based on a topological graph that uses the policy for long-horizon goal-seeking

> Exploring a new environment is often framed as the problem of efficient mapping....

Robots exploring a new area are essentially trying to map it efficiently—this means covering as much area as possible, ideally without wasting time.
Some methods use local control strategies (e.g., make decisions based only on nearby information), and others use global strategies (like ”frontier exploration”, which targets unexplored boundaries).
However, making very accurate 3D maps is hard, especially if the robot doesn’t have good depth sensors.

- Some approaches use simulation data (training in virtual environments).
- Others learn from real-world data directly. These models may use:
  Intrinsic rewards: Encouraging the robot to explore new things.
  Semantic prediction: Going to interesting or informative places.
  Latent variable models: Abstract models of how actions affect the world.

Challenge!
Even the best learning-based models trained in simulation don’t transfer well to real-world environments, and many fail in complex spaces.

Enter NoMaD: A New Method
ViNT is a previous system that used a big model to suggest where the robot should go next by generating subgoal images.
NoMaD improves on this by:

- Not generating images.
- Directly predicting actions using diffusion models, which are typically used in image generation tasks but can model complex probabilities really well.

- This makes NoMaD more accurate and much lighter (needs 15x fewer parameters).
- It’s so efficient, it can run on low-power devices like a Jetson Orin (a type of small computer often used in robots).

In exploration, there can be many possible good actions at any moment (multi-modal distribution)
Diffusion models are powerful because they can learn complex distributions (e.g., ”what are all the good things I might do next?”) without needing to simulate the future.
Nomad adds goal-conditioning to diffusion-based action generation, meaning it can do both:
-Goal-directed exploration (e.g., ”go there”)
-Undirected exploration (e.g., ”explore as much as you can”)

Goal: A Visual Navigation Policy

The authors want to build a control policy (called π) that lets a robot navigate using only its camera (RGB images). The robot:

- Takes in past and current camera images ($o_t := o_{t-P:t}$)
- Outputs the distribution over future actions ((a_t):=$a - t : t + H$)
- Might also be given an image of a goal (o_g) to help guide it.

There are two possible cases:

- If a goal image is provided, π tries to move the robot toward the goal.
- If no goal is given (like in exploration tasks), π should still act safely (avoid obstacles, stay on paths) and explore effectively.

To manage longer-term planning, the robot also uses:

- A topological Memory $\mathcal{M}$ (a kind of virtual memory map)
- A high-level planner to help decide where to explore next.

Visual Goal-Conditioned Policies (ViNT as Backbone)

They use ViNT (Visual Navigation Transformer), which is based on the Transformer architecture (very popular in modern AI).
How ViNT works:

- Each image (current or past) is passed through an EfficientNet-B0 encoder to extract features.
- The goal image and current state are combined via a special goal fusion encoder.

- The resulting features (called tokens) are fed through Transformer attention layers, producing a context vector c_t.
- This vector is then used to predict future actions (a_t) and temporal distance to the goal ($d(o_t, o_g)$).

These predictions are trained using supervised learning (the model is shown what the correct actions and distances should be and learns from that).

But: ViNT only works if there is a goal image. It can't explore on its own

To enable the robot to operate in large, complex environments, it needs more than just immediate camera inputs.
So, they introduce: A topological memory $\mathcal{M}$

- A graph where each node is a past visual observation.
- Edges connect nodes that the robot can travel between, using ViNT's predicted distances to know which transitions are possible.

When a robot can't reach a goal directly using visual input, it uses this graph $\mathcal{M}$ to plan a path via intermediate goals (subgoals). Even without a specific goal, this graph helps the robot explore logically and not just randomly.

Frontier-Based Exploration

In this work, they test NoMaD's ability to propose diverse subgoals and explore unseen areas using a frontier-based method:

- Frontier = edge between explored and unexplored space.
- The robot is encouraged to go to frontiers, gradually covering new space.

They adopt the framework from ViKiNG, but replace the old policy with NoMaD, which can now do both:
Goal-seeking (when a goal is provided)
Exploration (when no goal is provided)

Methods

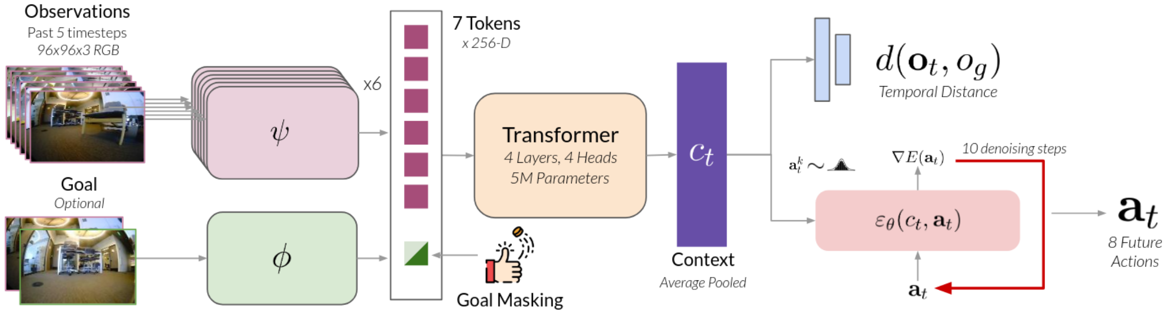

Figure 1: Overview of the NoMaD architecture. The model takes in RGB images and outputs a distribution over future actions. It can also use a goal image to guide its actions.

A. Goal Masking

The idea is based on the ViNT architecture, which uses Transformer layers to encode inputs.
New in NoMaD:
They introduce a binary variable m, the goal mask.

- m=0: Goal is visible → conditioned on goal image.
- m=1: Goal is hidden → acts without knowing any goal.

training:
sample m Bernoulli(p_m) where $p_m = 0.5$
– 50 % of the training time, goal provided.
– 50 % of the training time, no goal (exploratory behavior).
This balance allows shared weights to generalize across both behaviors.

Modified Transformer Input:

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

Where:

- c_t = context vector
- ψ and ϕ = visual encoders
- o_i = current image
- o_t, o_g = time-step and goal image

- m = goal mask
- f = Transformer function (ViNT style)

Masking is applied in the attention layers, so goal embeddings are ignored if m=1.

B. Diffusion Model

In general, diffusion models are generative models. They work by gradually adding noise to data (a forward process), and then learning to remove that noise (a reverse denoising process) to generate realistic samples.

In NoMaD, instead of generating images or audio, the diffusion model is used to generate robot actions — specifically, the next velocity command (like linear and angular speed) that's safe and goal-aligned.

Action spaces in navigation are multimodal. Need to avoid actions leading to collisions.
diffusion model setup:
They model the conditonal distrubution of the next action a_t as

$$p(a_t|c_t)$$

Here's how it works:
The Forward Diffusion Process (Add Noise)
1. Start with a real action a_t^0 from the dataset.
Then we add gaussian noise over multiple steps k = 1 to K.

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

where:

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

This is only used during training.
Reverse Denoising (Learned Model)
2. The model learns how to reverse this process:

- starting from pure noise a_t^k $\mathcal{N}(0, I)$, 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:

- ϵ_θ is a neural network (UNet) that predicts the noise.
- c_t is the context vector from the Transformer.(images,goal mask)
- k: current diffusion step.
- γ, α, σ are scheduler constants.

What is the Network ϵ_θ Like? It's a 1-D conditional UNet.
– 15 convulational layers.
– Takes in : the noisy action a_t^k, the context vector c_t, and the diffusion step k.
– Outputs the predicted noise vector $\hat{\epsilon}_k$
During training, it is compared to the true noise added earlier.
Training Objective
The loss is just Mean Squared Error (MSE) betweeen the predicted and the actual noise. For overall loss function, we add the MSE temporal distance loss to the objective function.

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

where:

- ψ,ϕ correspond to the visual encoders for the observation and goal images.
- f corresponds to the transformer layers,
- θ corresponds to diffusion process parameters,
- f_d corresponds to the temporal distance predictor.
- $\lambda = 10^{-4}$ is a hyperparameter that controls the relative weight of the temporal distance loss
- $p_m = 0.5$ is the goal mask probability.
- Square Cosine Noise Scheduler is used.
- The model is trained using AdamW optimizer with a learning rate of 1e-4.
- Model is trained for 30 epochs, with batch size of 256.

- We use cosine scheduling and warmup to stabilize the training process and follow other hyperparameters from ViNT
- For the ViNT observation encoder, we use EfficientNet-B0 to tokenize observations and goals into 256-dimensional embeddings, followed by a Transformer decoder with 4 layers and 4 heads.